Introduction

Overview

ActiveReports .NET is a reporting solution for designing and delivering reports using report designers and controls for .NET platform. The main purpose of any reporting tool is to be able to create and print digital documents. ActiveReports supports fully integrated Visual Studio components that combine user-friendly visual controls with the low-level control of code in Visual Studio .NET programming languages to provide reliable and intuitive report designers. The product also includes a standalone designer application that allows you to build different report types using a versatile set of feature-packed controls, including but not limited to:

- Barcode
- Charts and Sparklines
- RichTextBox
- Shapes
- Table
- Tablix
- TOC
- Image (with supported format such as SVG)

ActiveReports .NET supports unique reporting features, like:

- Lightweight report designers and viewers for powerful .NET Core reporting
- Extensive API for customizing and maintaining control
- A suite of three designers: End-user report designers for web/desktop applications, Visual Studio Report Designer and standalone designer application
- Provides an interactive look to your report at run time by supporting parameters, filters, drill-down, links, document map and sorting

Also, with ActiveReports, reports can be bound to:

- CSV Data Source
- JSON Data Source
- DataSet Data Source
- Object Data Source
- OData Data Source
- OleDb Data Source
- XML Data Source
- ADO.NET Data objects
- IList Binding using DataGridView control
- Using LINQ

In case of any question, do not hesitate to contact our Sales, Support, or write in our Forum.

Product Requirements

To install and use ActiveReports 18, you need the following hardware and software.

Hardware requirements (minimum)

- **Hard drive space**: 1 GB available

Development Environments

		.NET Framework 4.6.2 to 4.8.1	.NET 6, .NET 7, .NET 8
		Desktop (WinForms, WPF) ASP.NET (WebForms, MVC 5)	Console [1] Desktop (WinForms, WPF) ASP.NET Core (MVC, Blazor) Azure Function [2]
IDE	Visual Studio 2022 (17.8)	✓	✓
	Visual Studio 2019 (16.11)	✓	✗
	Visual Studio 2017 (15.9)	✓	✗
Operating System	Windows 11	✓	✓ [3]
	Windows 10	✓	✓ [3]
	Windows Server 2022	✓	✓ [3]
	Windows Server 2019	✓	✓ [3]
	Windows Server 2016	✓ [3]	✓ [3]

[1] Tutorial: Create a .NET console application using Visual Studio

[2] Quickstart: Create your first C# function in Azure using Visual Studio

[3] Install .NET on Windows

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

You are required to perform the following updates from the links provided for the proper working in Visual Studio.

- Developer Pack for .NET Framework 4.6.2 or above
- TypeScript Plugin for VS2017/VS2019/2022
- node.js and npm

Run Time Supported Environments

Console and Desktop

Controls	Framework		Operating System		
	.NET Framework 4.6.2 to 4.8.1	.NET 6 .NET 7 .NET 8	Windows 11 Windows 10	Windows Server 2022 Windows Server 2019 Windows Server 2016	macOS 10.15+ [1] Linux [2]
Console Application (Page/RDLX reports)	✓	✓	✓	✓	✓
Console Application (Section Reports)	✓	✓	✓	✓	✓
WinForms Controls (Viewer, Designer)	✓	✓	✓	✓	✗
WPF Controls (Viewer)	✓	✓	✓	✓	✗

1 .NET on macOS

2 .NET on Linux

Web

Server	Framework			Operating System (*1)	
	.NET Framework 4.6.2 - 4.8.1	.NET 6 .NET 7 .NET 8	Windows 11 Windows 10	Windows Server 2022 Windows Server 2019 Windows Server 2016	macOS 10.15+ (*2) Linux (*3)
ASP.NET WebForms (WebViewer)	✓	✗	✓	✓	✗
ASP.NET MVC 5 (JS Viewer, WebDesigner)	✓	✗	✓	✓	✗
ASP.NET Core MVC (JS Viewer, WebDesigner)	✗	✓	✓	✓	✓
ASP.NET Core Blazor (Viewer)	✗	✓	✓	✓	✓

Controls	Browser					
	Edge Cromium 109	Chrome 109	Firefox 115	Safari 15.6	Internet Explorer 11	Edge Legacy
ASP.NET WebForms (WebViewer)	✓	✓	✓	✓	✗	✗

JS Viewer, WebDesigner	✓	✓	✓	✓	✗	✗
ASP.NET Core Blazor (Viewer)	✓	✓	✓	✓	✗	✗

*1: requires IIS 10. Also read the following links:

- Web server implementations in ASP.NET Core
- The .NET Core Hosting Bundle

*2: Install .NET on macOS
*3: Install .NET on Linux

Cloud

	Framework	
	.NET Framework 4.6.2 to 4.8.1	.NET 6, .NET 7, .NET 8
Azure Functions (Linux) [3]	✗	✓ [2]
Azure Functions (Windows)	✓ [1]	✓ [2]
Azure App Service (Linux) [3]	✗	✓ [2]
Azure App Service (Windows)	✓ [1]	✓ [2]
Azure WMs (Linux)	✗	✓
Azure WMs (Windows)	✓	✓
Amazon ES2 (Linux)	✗	✓
Amazon ES2 (Windows)	✓	✓
Google Compute Engine (Linux)	✗	✓
Red Hat OpenShift (Linux) [3]	✗	✓

[1] Azure Functions and App Services support only .NET 4.8

[2] Azure Functions and App Services support only .NET 6

[3] Page/RDLX reports support only

Limitations of .NET Core support

- You need to specify encodings before using ActiveReports with ASP.NET Core MVC. See Troubleshooting for details.
- While working with Section Reports with scripts in WinForms Viewer, WPF Viewer, and Windows Designer components in .NET Core applications, it is mandatory to install 'System.Text.Encoding.CodePages' NuGet package and update the Program.cs file accordingly. Otherwise 'System.NotSupportedException' is thrown on previewing the report. See Troubleshooting for details.

What's New

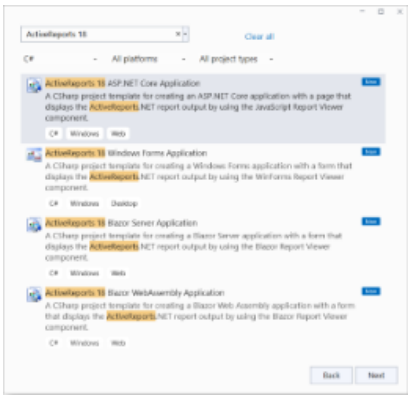
For Developers

.NET 8 Support in ActiveReports 18

With the release of ActiveReports 18, we are proud to announce full support for .NET 8, marking a significant step forward in ensuring our reporting tool remains at the forefront of technology. This advancement not only enhances the performance and security of your reports but also aligns ActiveReports 18 with the latest .NET framework, ensuring optimal compatibility and the ability to leverage the newest features and improvements in .NET 8.

Breaking Changes | Samples

Enhanced Visual Studio Project Templates



In our commitment to streamline your experience with ActiveReports, we've introduced new Visual Studio project templates. These templates are designed to replace previous versions, offering a more intuitive and efficient start to your reporting projects. The newly available templates include:

- ActiveReports Windows Forms Application: Tailored for desktop environments, this template sets the stage for rich, interactive reports on Windows Forms.
- ActiveReports ASP.NET Core Application: Perfect for web developers, this template integrates seamlessly with ASP.NET Core MVC for dynamic, server-side reporting.
- ActiveReports Blazor Server Application: Leverage the power of Blazor Server to create interactive and real-time reporting applications.
- ActiveReports Blazor WebAssembly Application: Build client-side reporting applications with the modern capabilities of Blazor WebAssembly.

Choosing any of these templates will automatically launch the new Report Wizard, providing you with options for RDLX, RDLX Dashboard, Page, and Section reports. This enhancement simplifies the initial setup process and ensures you have the flexibility to select the report type best suited to your project's needs.

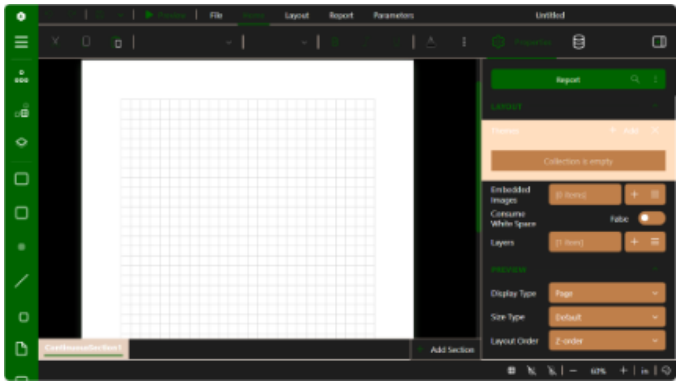
Quick Start

Improved ASP.NET Middleware for ActiveReports Web Integration

We've enhanced the ASP.NET middleware in ActiveReports 18, streamlining the integration of ActiveReports Web components for developers. This improvement simplifies the process of embedding reporting capabilities into web applications, ensuring a smoother, more efficient development experience.

WebDesigner Application | Js Viewer Application | Blazor Viewer Application

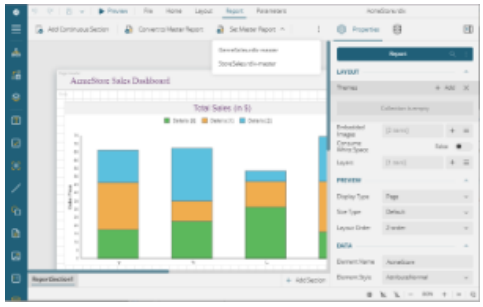
Customizable UI with Themes for JS Viewer and Web Designer



ActiveReports 18 introduces themes support for both JS Viewer and Web Designer components, allowing you to personalize the user interface according to your preferences. Choose from a selection of built-in UI themes, or craft a custom theme to tailor the look and feel of these components to match your application's aesthetic or branding requirements.

Apply Themes to WebDesigner and Js Viewer Components | WebDesigner API | JS Viewer API

Master Reports in Web Designer: Empowering End-Users with Pre-defined Templates

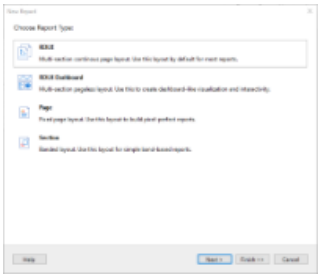


ActiveReports 18 enhances the Web Report Designer component by introducing support for Master Reports. This feature enables end-users of your application to kick-start their report creation process using pre-defined templates that provide a structured starting point, ensuring consistency and compliance with design standards while still offering flexibility in report customization.

Master Report (RDLX Report)

For Report Authors

Introducing the New Report Wizard



ActiveReports now features an innovative Report Wizard designed to streamline the creation of RDLX, Page, Dashboard reports. This brand-new wizard simplifies the report creation process, guiding you through each step with ease, from initial setup to data binding. Whether you are crafting detailed financial reports, informative data dashboards, or any other document-based presentation, our new wizard ensures a smooth, intuitive experience, making it easier than ever to transform your data into actionable insights.

Create RDLX Report

Expanded Data Source Support for Page/RDLX Reports

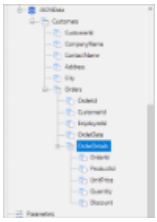
In our ongoing effort to enhance data connectivity and flexibility within ActiveReports, we are excited to announce the addition of support for several new built-in data sources. Report authors can now connect their RDLX, Page, and Dashboard reports directly to:

- MySQL: Tap into the power of one of the world's most popular open-source relational database management systems.
- PostgreSQL: Leverage this advanced, enterprise-class open-source database system known for its reliability and robustness.
- Excel: Directly import data from Excel files, allowing for seamless integration of spreadsheet data into your reports.

These enhancements significantly broaden the range of data that can be incorporated into your reports, ensuring that you have the tools needed to make informed, data-driven decisions.

MySQL | PostgreSQL | Excel

Enhanced Flexibility with JSON and XML Nested Dataset Support



ActiveReports 18 introduces nested dataset support for JSON and XML data sources, a significant enhancement designed to expand your data handling capabilities. This new feature allows for the utilization of hierarchical data structures directly within your reports, offering unprecedented choice and flexibility in how you design and present your data.

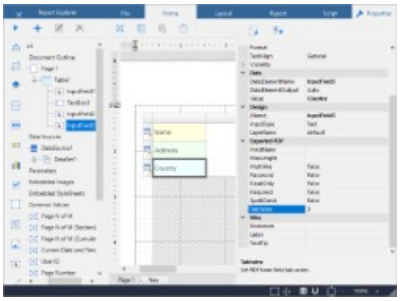
Nested Datasets (JSON and XML) | Nested Datasets

Enhanced Image Rendering with SVG Support in Section Reports

ActiveReports 18 now supports the SVG format within the Picture control of Section reports. To ensure superior image quality and scaling, we recommend utilizing SVGs, particularly in Cross-Platform compatibility mode. This update allows for sharper, more scalable graphics, enhancing the visual appeal of your reports across different platforms.

Picture

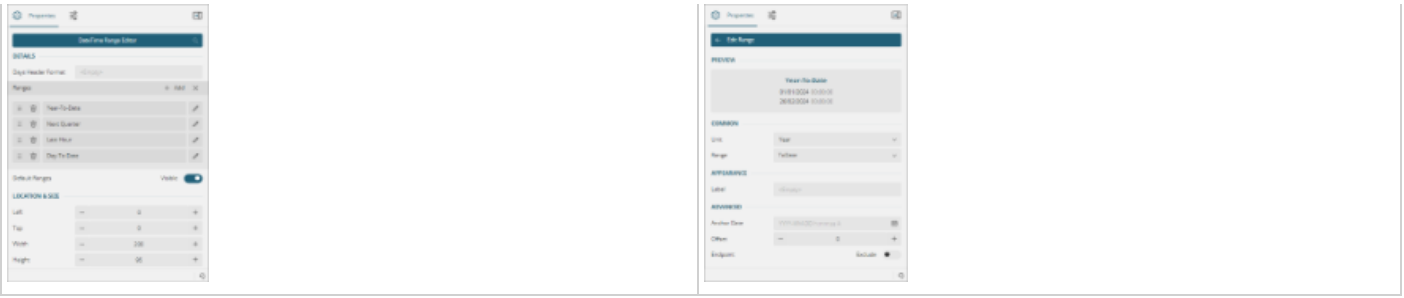
Enhanced Navigation with Tab Order for Editable Fields in PDF



ActiveReports 18 introduces the ability to precisely control the tab order of editable fields in PDF forms through a new TabIndex property on InputField controls. This addition allows for the customization of navigation, enabling users to set a specific sequence for moving from one field to another using the Tab key. This feature enhances form usability and improves the overall user experience in PDF documents.

InputField (Page/RDLX Report) | InputFieldText (Section Report) | InputFieldCheckBox (Section Report)

DateTime Range Control: Enhanced Precision and Predefined Ranges

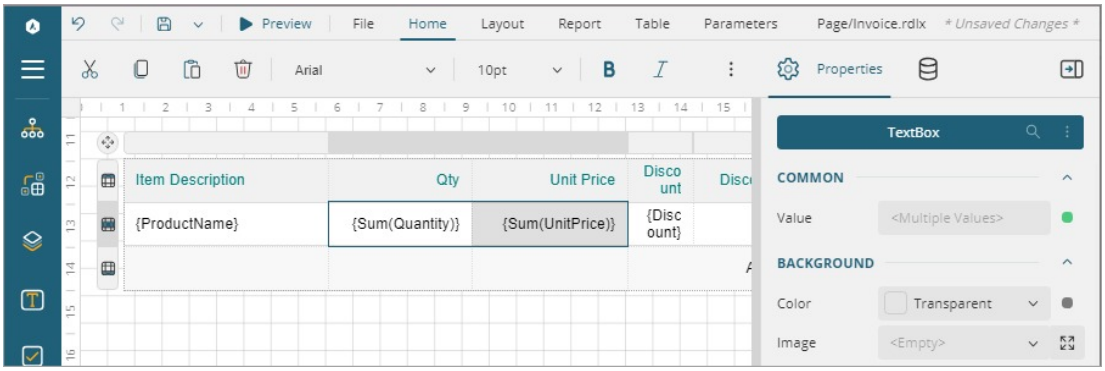


The DateTime Range control in the custom parameters panel of ActiveReports 18 has been enhanced to offer end-users greater flexibility and precision. Users now have the capability to specify the exact time for their reports and also select from a range of predefined values, such as "last year." This update provides a more nuanced approach to selecting time periods, catering to both specific and broad reporting needs with ease.

Custom Parameters View

For Users of Web Report Designer

Streamlined Textbox Operations



ActiveReports Web Report Designer now supports simultaneous updates to multiple TextBoxes through the "Current Textbox Value" option in the Expression Editor. This enhancement simplifies the process of applying uniform expressions, significantly improving efficiency and consistency in report design.

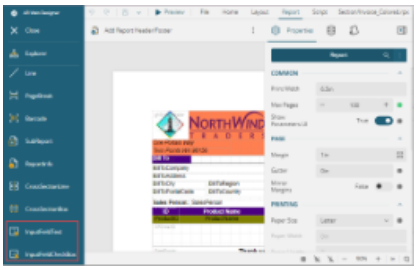
Expressions

Enhanced Scripting with Events and Objects Toolbar in Section Report

The scripting workspace for Section Reports in ActiveReports 18 has been enhanced with the introduction of the Events and Objects Toolbar. This new feature provides the option to easily generate event handlers for basic report objects, streamlining the scripting process.

Report Events

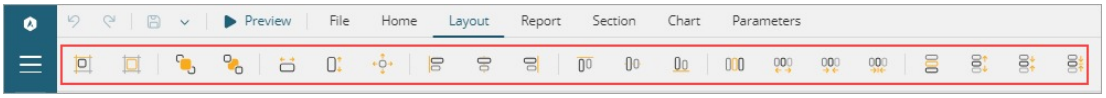
Section Report Toolbar Enhancements: New InputField Controls



The Section Report Toolbar in ActiveReports 18 Web Designer has been upgraded to include the InputFieldCheckBox and InputFieldText controls. These additions expand the range of interactive elements available in the Web Designer, allowing for the creation of more dynamic and user-interactive reports directly within the web environment.

InputFieldCheckBox | InputFieldText

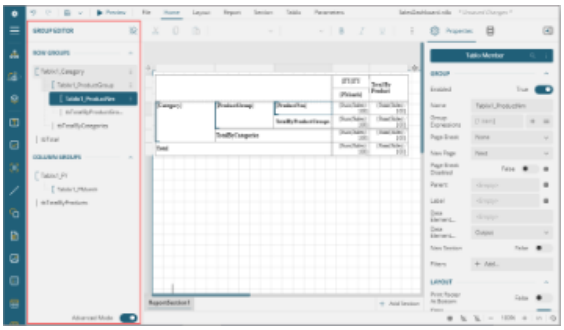
Improved Control Management with Enhanced Layout Ribbon Tab



The Layout Ribbon Tab in ActiveReports 18 has received significant enhancements, offering expanded capabilities for managing the layouts of multiple controls simultaneously. With these enhancements, you can effortlessly align multiple controls either vertically or horizontally, standardize the size of multiple controls to match each other, and more.

WebDesigner

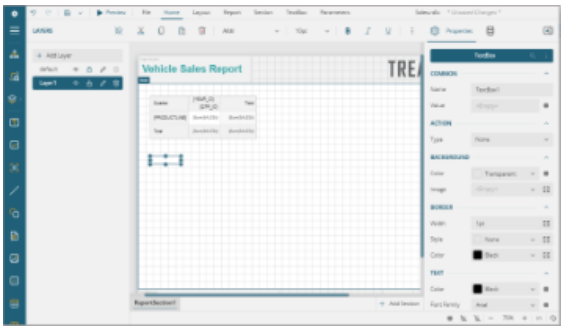
Enhanced Visualization with Advanced Tablix Group Editor



The Tablix Group Editor in ActiveReports 18 has been upgraded to an advanced version, which significantly enhances how users interact with and visualize the hierarchy of group members. Now, when selecting single or multiple rows, columns, or cells within the Tablix body, the editor automatically highlights the corresponding hierarchy, making it easier to understand and manage group relationships.

WebDesigner

Streamlined Layer Management with Active Layer Support



ActiveReports 18 introduces active layer support for RDLX, Page, and Dashboard reports, enhancing the way layers are managed during report design. With this feature, a selected layer automatically becomes active, meaning any new controls are added directly to this layer.

Layers

Get Started

The help file divides the ActiveReports.NET knowledge base based on the target audience - the Developers, the DevOps, the Report Authors, and the Report Readers. The primary target audience of ActiveReports is developers. If you are a developer, we recommend that you start with Quick Start.

For Report Authors, we have Quick Start and tutorials to create reports.

You should feel free to see our demos and samples (web samples) that provide quick demonstrations of core features and capabilities.

Report Readers: Viewer Components

Report viewers are the components that display a report generated by the reporting engine via report designer. You get complete .NET embedded reporting tool for web and desktop applications. Our viewers support technologies such as Blazor, ASP.NET Core, ASP.NET MVC, Angular, React, HTML5/JS, WPF, and WinForms.

This section introduces the available report viewing options for desktop and web applications, the Desktop Viewers and Web Viewers.

You can also preview the report at design time in Visual Studio Integrated Designer.

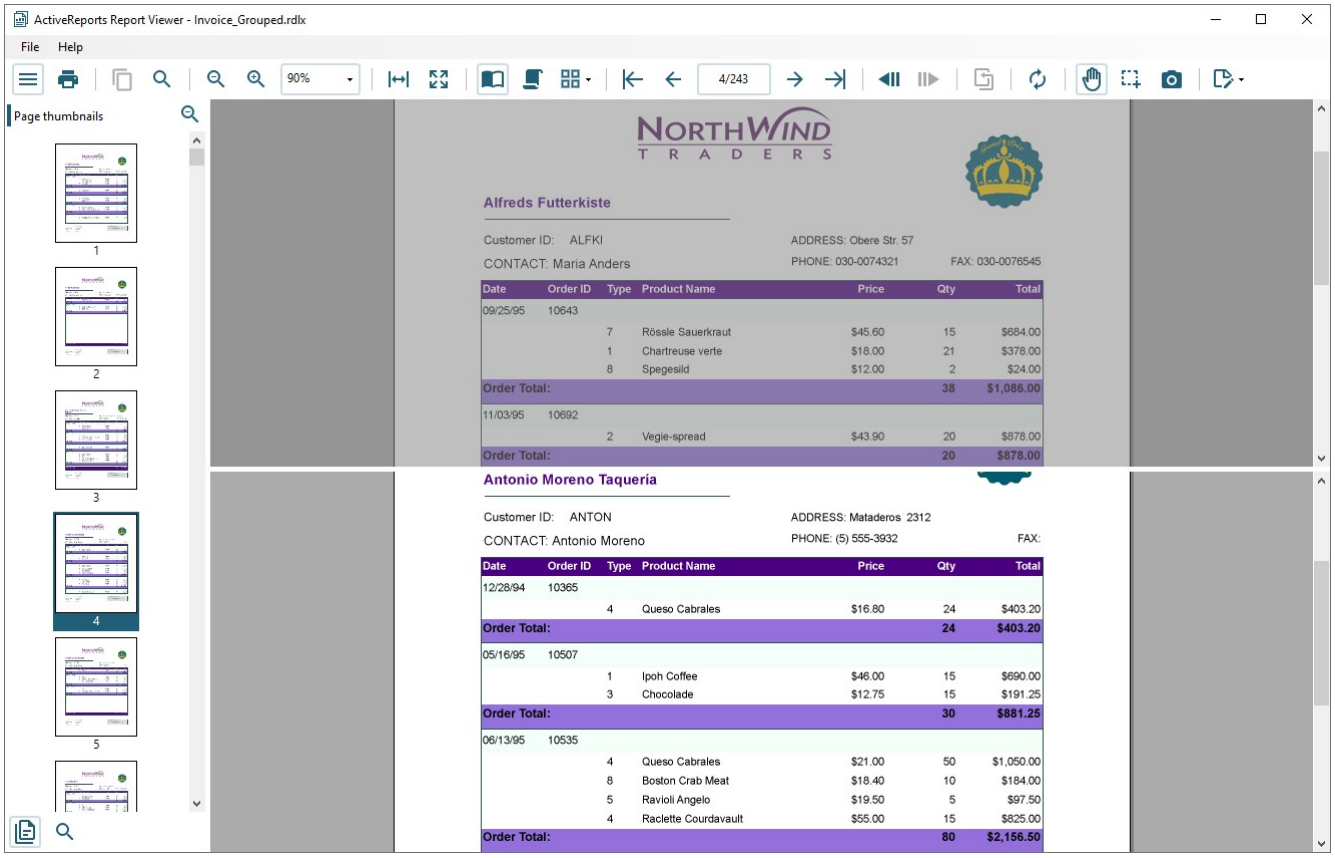
Desktop Viewers

As a report reader, you can view reports by running an application with the ActiveReports Viewer control integrated, and loading the report in the viewer. ActiveReports provides executable files for the Viewer controls (Windows and WPF) in the startup menu. You can use the samples available for WinForms and WPF platforms: WinViewer or the WpfViewer.

All the viewers provided by ActiveReports are feature-rich that allow you to easily view, print, and export reports. The viewers are highly interactive, giving you the ability to enter parameters, perform sort and drill down, navigate to other reports using drill-through links, and so on. The ActiveReports viewers toolbar and sidebar offer a handful of features. In addition, the viewers are customizable and localizable according to the personal needs.

Key Features

- Highly Interactive
- Touch-enabled
- Support Annotations
- Provide Advanced Print Options
- Provide Exportability to Different Outputs
- Customizable when used as a Control
- Desktop Viewers provide two views in one display, as shown:

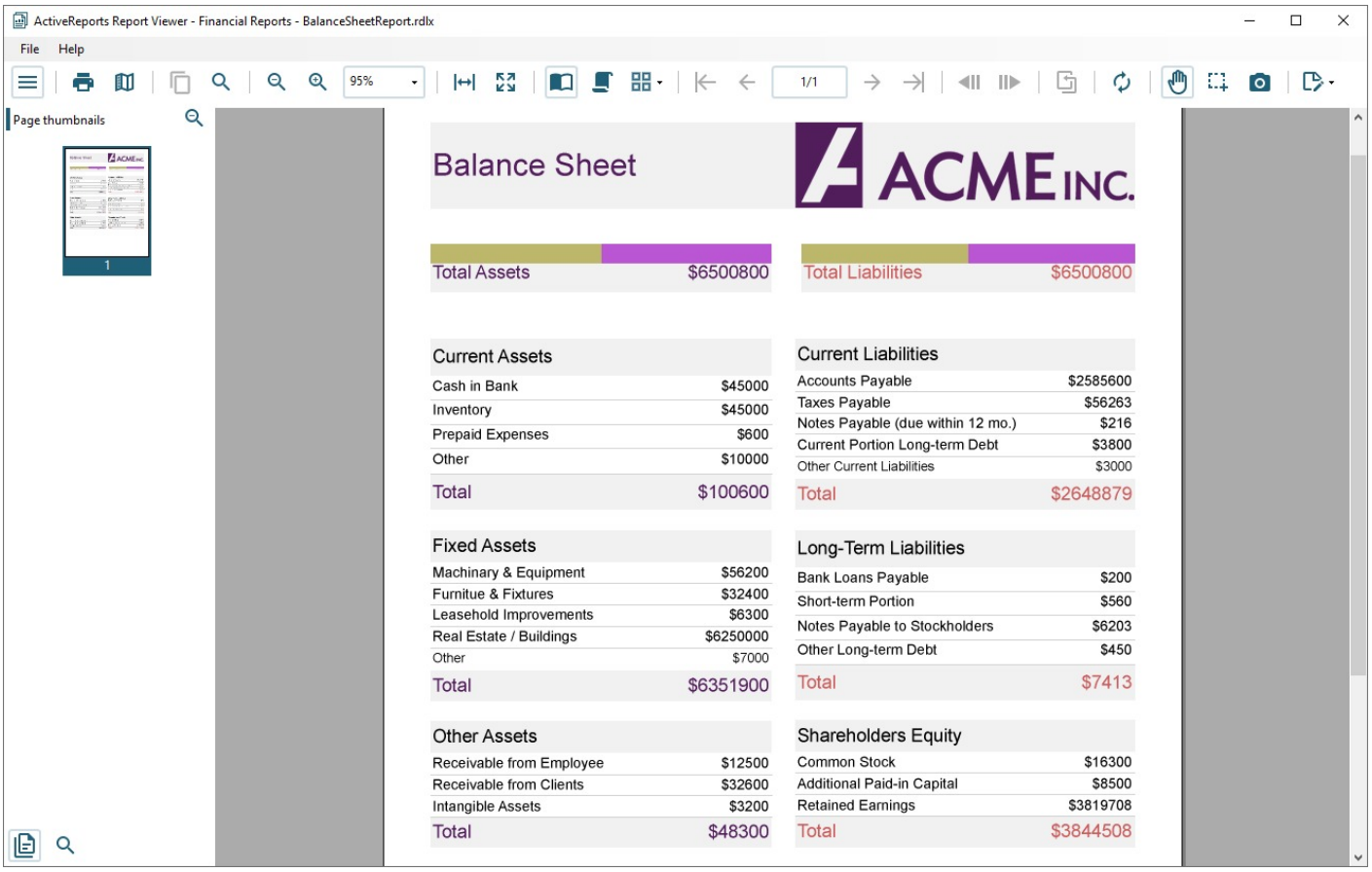


Windows Forms Viewer

The **ActiveReports** provides the Windows Forms Viewer application **ActiveReports.Viewer.exe**, bundled with the ActiveReports installer. The app can be launched by installing the ActiveReports installer package.

You can run the application by selecting the **ActiveReports 18 Viewer** from the Start menu, or by running the **ActiveReports.Viewer.exe** from the *C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools* location.

Report readers wanting to quickly preview a report in WinForms platform can use WinViewer sample. You just need to load the report you want to view in the viewer control. The Windows Forms Viewer with a report loaded looks like the following.



The Windows Forms Viewer is basically a Windows Forms application with an ActiveReports Viewer control in it. The default user interface of this application provides an ActiveReports Viewer control along with a menu bar.

You can open a .rdlx or .rpx report in the standalone viewer application, by going to the **File** menu > **Open** menu option and selecting a report to load in the viewer. Unlike the Viewer control, no code implementation is required to load the report in the standalone application.

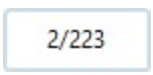
Please note that any additional features activated through code like the annotation toolbar, are not available in the standalone viewer application.

Interface Elements

Viewer Toolbar

The features available through the Viewer Toolbar are elaborated in this table.

Toolbar Element	Name	Description
	Toggle sidebar	Displays the sidebar that includes the Thumbnails, Parameters, Document map and Search results panes.
	Print	Displays the Print dialog where you can specify the printing options.

	Galley mode	Provides a viewer mode which removes automatic page breaks from a Report Definition Language (RDLX) and displays data in a single scrollable page. This mode maintains page breaks you create in the report and removes only automatic page breaks.
	Copy	Copies text that you select in the Selection mode to the clipboard.
	Find	Displays the Find dialog to find any text in the report.
	Zoom out	Decreases the magnification of your report.
	Current zoom	Displays the current zoom percentage, which can also be edited.
	Zoom in	Increases the magnification of your report.
	Fit width	Fits the width of the page according to viewer dimensions.
	Fit page	Fits the whole page within the current viewer dimensions.
	Single page view	Shows one page at a time in the viewer.
	Continuous view	Shows all preview pages one below the other.
	Multipage view	Offers you an option to select how many pages to preview in the viewer at one time.
	Navigate to First Page	Takes you to the first page of the report. This button is enabled when a page other than the first page is open.
	Navigate to Preceding Pages	Takes you to the page prior to the current page. This button is enabled when a page other than the first page is open.
	Navigate to Next pages	Takes you to the page following the current page. This button is disabled on reaching the last page of the report.
	Navigate to Last page	Takes you to the last page of the report. This button is disabled on reaching the last page of the report.
	Current page	Opens a specific page in the report. To view a specific page, type the page number and press the Enter key.
	Backward	Takes you to the last viewed page. This button is enabled when you move to any page from the initial report page. Clicking this button for the first time also enables the **Forward** button.
	Forward	Takes you to last viewed page before you clicked the Backward button. This button is enabled once you click the **Backward** button.
	Back to parent report	Returns you to the parent report in a drillthrough report.

	Default	Allows you to specify a default mouse pointer mode.
	Refresh	Refreshes the report.
	Pan mode	A hand symbol serves as the cursor that you can use to navigate.
	Selection mode	Allows you to select contents on the report. Click the Copy icon (see image and description below) to copy the selected content to the clipboard.
	Snapshot mode	Allows you to select content on the report that you can paste as an image into any application that accepts pasted images.
	Cancel	Cancels the report rendering.
	Touch Mode	Allows you to select touch mode for the viewer. **Note:** The **Touch Mode** button only appears on the toolbar while working on touch enabled devices.
	Annotations	Lets you add annotations. • **Text**: A rectangular box in which you can enter text. • **Circle**: A circle without text. You can change the shape to an oval. **Rectangle**: A rectangular box without text. • **Arrow**: A 2D arrow in which you can enter text. You can change the arrow direction. • **Balloon**: A balloon caption in which you can enter text. You can point the balloon's tail in any direction. • **Line**: A line with text above or below it. You can add arrow caps to one or both ends and select different dash styles. • **Image**: A rectangle with a background image and text. You can select an image and its position, and place text on the image.

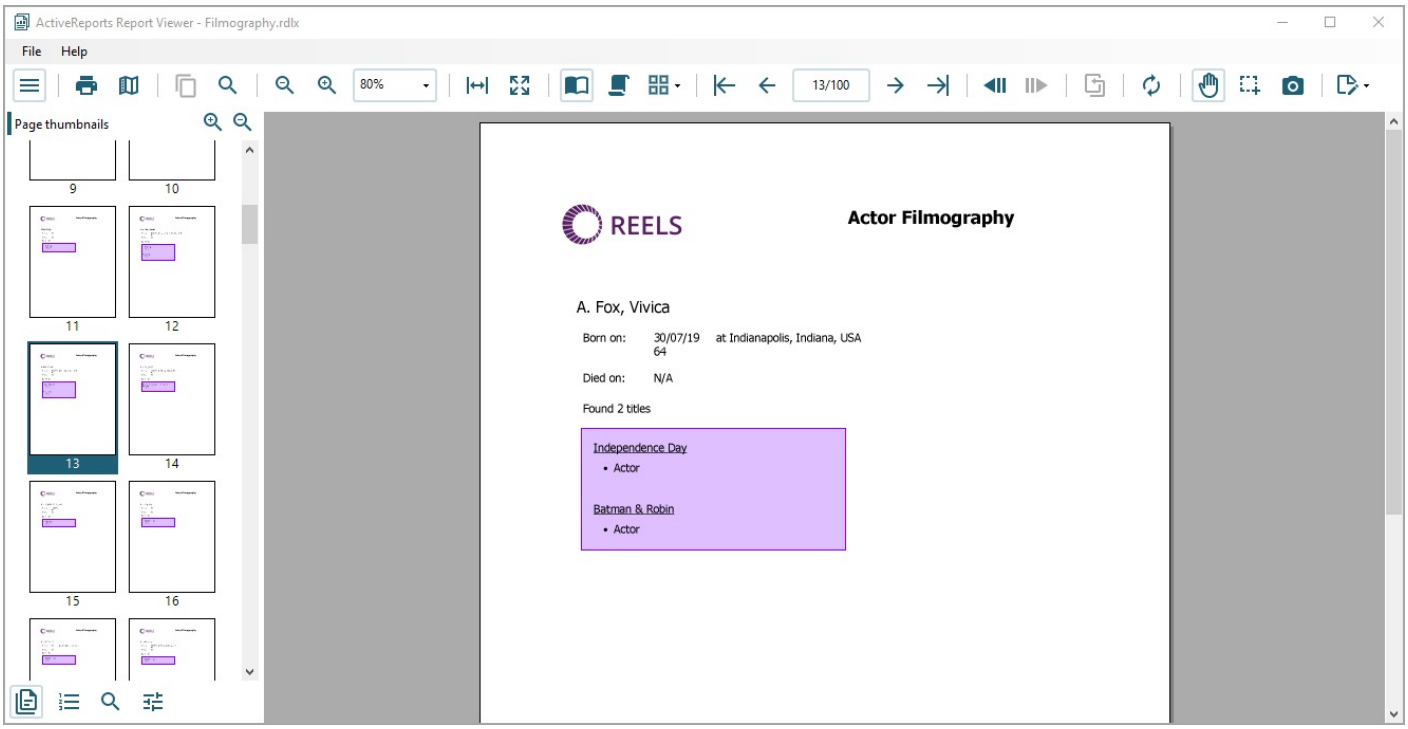
Viewer Sidebar

The Viewer sidebar appears on the left of the viewer when you click the **Toggle sidebar** button in the toolbar. By default, the sidebar shows the Thumbnails and Search Results panes. The additional Document map and Parameters also appear in this sidebar according to the interactive features added in the report. You can toggle between any of the viewer panes by clicking the buttons for each pane at the bottom of the sidebar.

Thumbnails pane

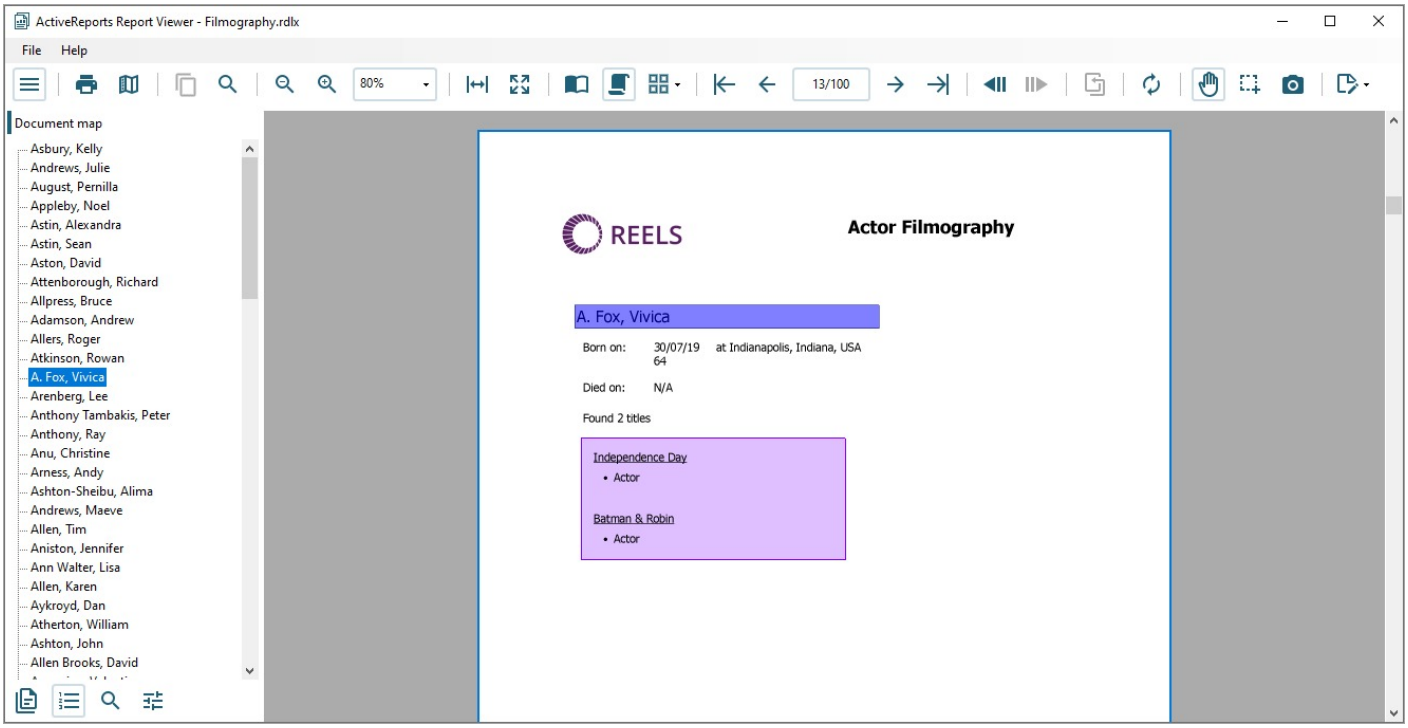
The **Thumbnails** pane appears by default in the sidebar when you click the **Toggle sidebar** button in the toolbar.

This pane is composed of a thumbnail view of all the pages in a report. Click any thumbnail to navigate directly to the selected report page. You can also zoom in and zoom out of the thumbnails using (+) or (-) button.



Document map pane

The Documents map pane is enabled for reports where the Label property or the Document map label is set. This pane displays each value for the text box, group, or sub report that you label, and you can click them to navigate to the corresponding area of the report in the Viewer.

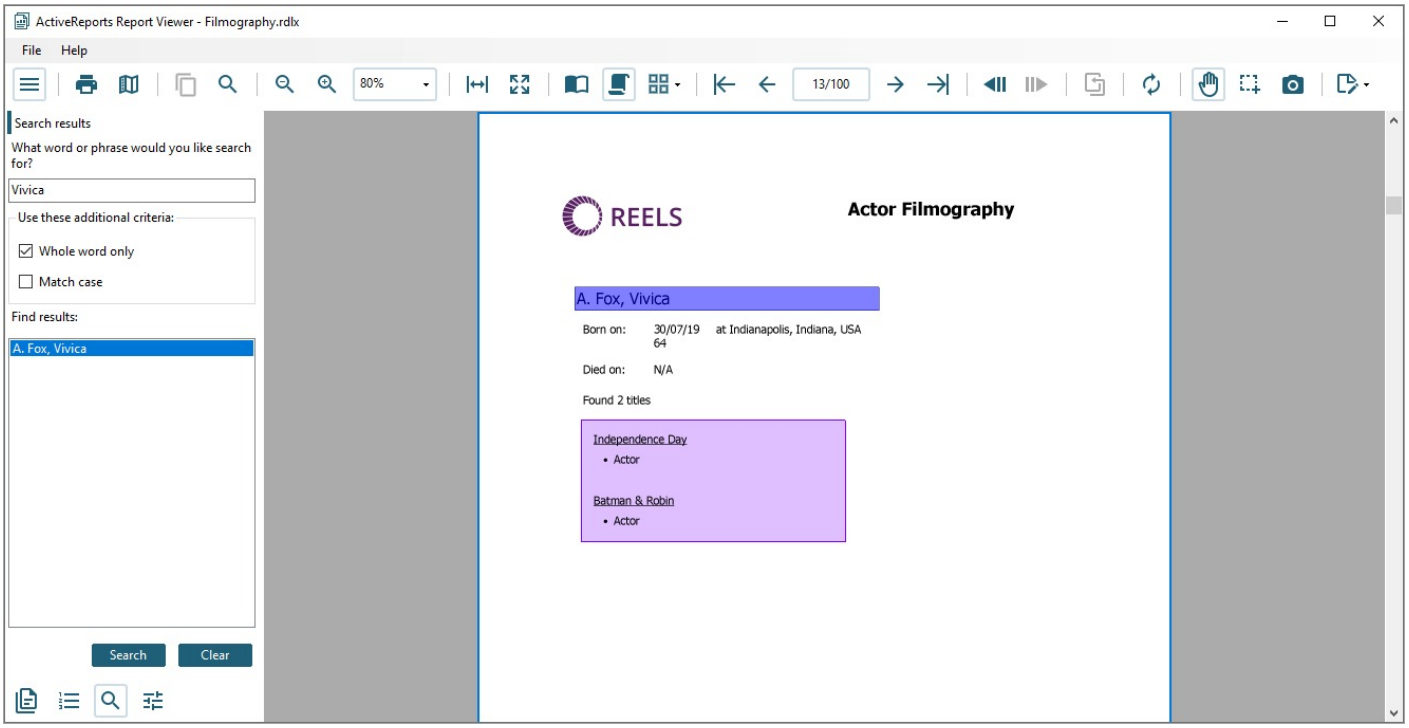


If a report does not have the Label property or Document map label set, the Documents map pane does not appear in

the sidebar.

Search results pane

The **Search** pane is the other default pane besides Thumbnails that appears in the sidebar when you click the **Toggle sidebar** button. This pane lets you enter a word or phrase to search within the report.



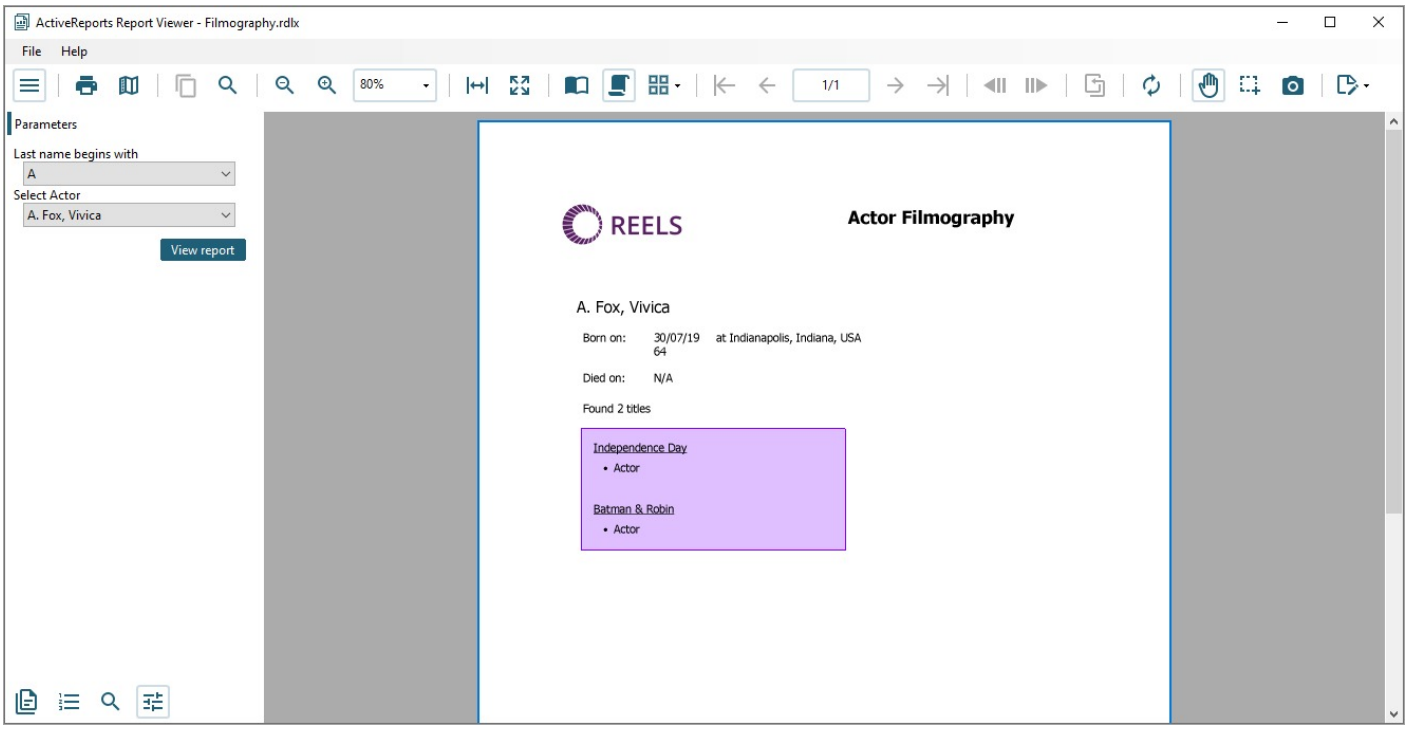
To search in a report:

1. Enter the word or phrase in the search field.
2. Under **Use these additional criteria**, you may optionally choose to search for the whole word or match the case of the search string while searching in the report.
3. Click the **Search** button to see the results appear in the **Find results** list.
4. Click an item in the list to jump to that item in the report.

To start a new search or clear the current search results, use the **Clear** button under the **Find results** list.

Parameters pane

The Viewer allows you to view reports with parameters. In the toolbar, click the **Toggle sidebar** button to open the Viewer sidebar and if your report contains parameters, the **Parameters** pane shows up automatically.



1. In the **Parameters** pane, you are prompted to enter a value by which to filter the data to display.
2. Enter a value or set of values and click **View report**, to filter the report data and display the report.

If a report does not have parameters, the Parameters pane does not appear in the sidebar.

Keyboard Shortcuts

The following shortcuts are available on the WinForms 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 view.
Ctrl + S	Turns off the continuous view.
Ctrl + I	Shows multiple pages.
Ctrl + G	Focuses on PageNumber area and selects content.
Shift + rotate mouse wheel	Scrolls horizontally.
Rotate mouse wheel	Scrolls vertically.
F5	Refreshes the report.
Home	Moves to the start of the current page.
End	Moves to the end of the current page.

Viewer's Thumbnails pane shortcut keys

You can use the following shortcut keys while using the thumbnails pane in the Viewer.

Keyboard Shortcut	Action
Up Arrow	Goes to the previous page.
Down Arrow	Goes to the next page.
Right Arrow	Goes to right page. If no thumbnail exist on the right, it goes to the next page.
Left Arrow	Goes to left page. If no thumbnail exist on the left, it goes to the previous page.
Page Down	Scrolls to the next thumbnail's view port. It also keeps the current selected page unchanged.
Page Up	Scrolls to the previous thumbnail's view port. It also keeps the current selected page unchanged.
Home	Goes to the first page.
End	Goes to last page.

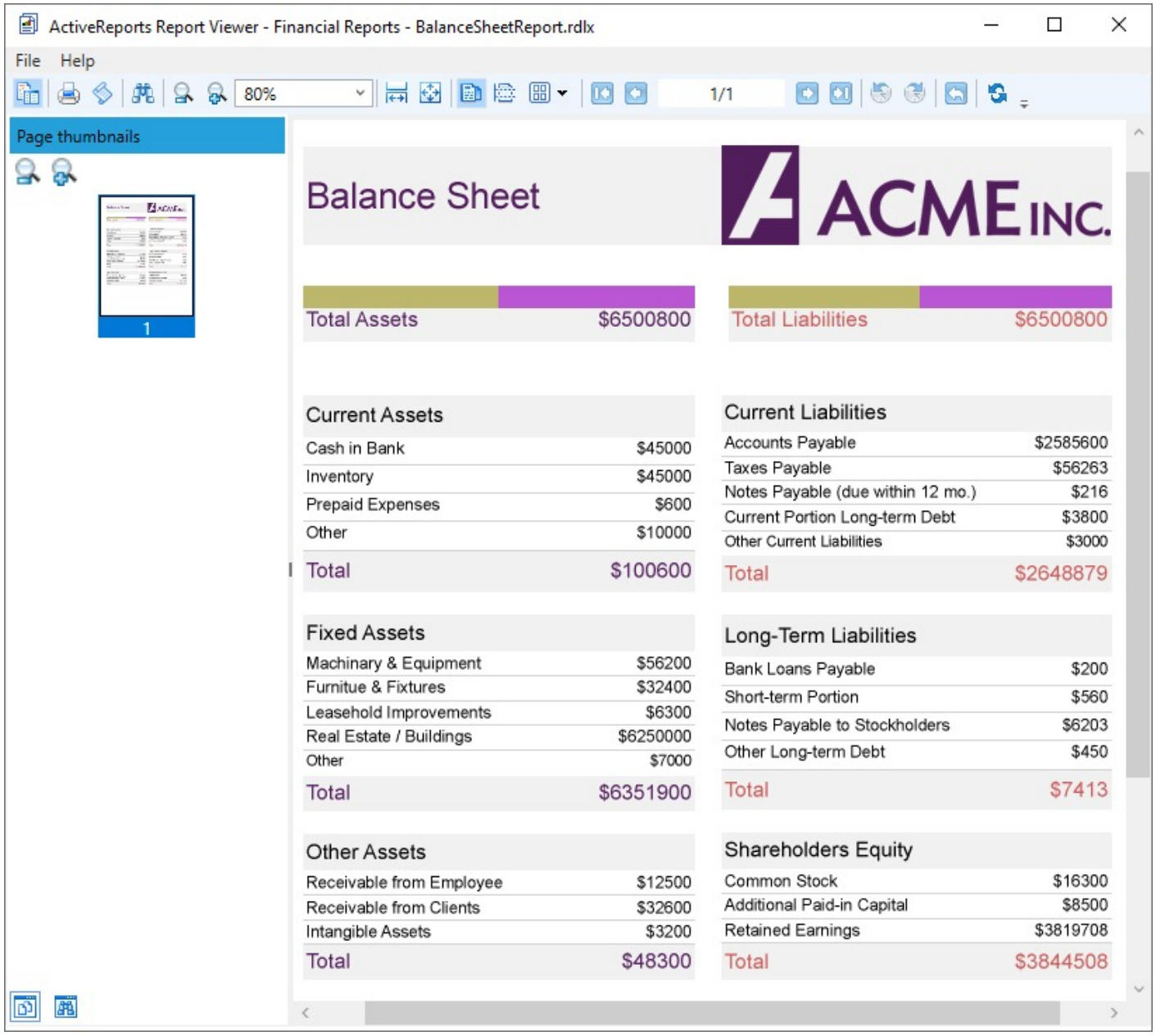
Windows Presentation Foundation Viewer

The **ActiveReports** provides the Windows Presentation Foundation Viewer application **ActiveReports.WpfViewer.exe**, which supports previewing all types of reports - Page, RDLX, and Section. The app can be launched by installing the ActiveReports installer package.

You can run the application by selecting the **ActiveReports 18 Viewer** from the Start menu, or by running the

ActiveReports.WpfViewer.exe from the *C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools* location.

Report readers wanting to quickly preview a report in WPF platform can use WpfViewer. You just need to load the report you want to view in the viewer control. The WPF Viewer with a report loaded looks like the following.



The WPF viewer is basically a WPF application with an ActiveReports WPF Viewer control in it. The default user interface of this application provides an ActiveReports along with a menu bar.

You can open an .rdlx or .rpx report in the standalone WPF Viewer application, by going to the **File** menu > **Open** menu option and selecting a report to load in the viewer. Unlike the Viewer control, no code implementation is required to load the report in the standalone application.

Note: The toolbar customization through code is not available in the standalone WPF Viewer application.

The WPF Viewer includes a toolbar and a sidebar with **Thumbnails**, **Search results**, **Document map** and **Parameters** panes.

The WPF Viewer has a limitation in the continuous mode. In this mode, a report is always rendered on top of the WPF Viewer because the WinForms content always appears on top of the WPF content.

Interface Elements

Viewer Toolbar

The following table lists the actions you can perform through the WPF Viewer toolbar. The toolbar allows you to perform the common or report-specific tasks on the report that is currently displayed.

Toolbar Element	Name	Description
	Toggle sidebar	Displays the sidebar that includes the Thumbnails, Parameters, Document map and Search results panes.
	Print	Displays the Print dialog where you can specify the printing options.
	Galley mode	Provides a viewer mode which removes automatic page breaks from a Report Definition Language (RDLX) and displays data in a single scrollable page. This mode maintains page breaks you create in the report.[1]
	Copy	Copies text that you select in the Selection mode to the clipboard.
	Find	Displays the Find dialog to find any text in the report.
	Zoom out	Decreases the magnification of your report.
100.00 %	Current zoom	Displays the current zoom percentage, which can also be edited.
	Zoom in	Increases the magnification of your report.
	Fit width	Fits the width of the page according to viewer dimensions.
	Fit page	Fits the whole page within the current viewer dimensions.
	Single page view	Shows one page at a time in the viewer.
	Continuous view	Shows all preview pages one below the other.
	Multipage view	Offers you an option to select how many pages to preview in the viewer at one time.
	Navigate to First Page	Takes you to the first page of the report. This button is enabled when a page other than the first page is open.
	Navigate to Preceding Pages	Takes you to the page prior to the current page. This button is enabled when a page other than the first page is open.

	Navigate to Next pages	Takes you to the page following the current page. This button is disabled on reaching the last page of the report.
	Navigate to Last page	Takes you to the last page of the report. This button is disabled on reaching the last page of the report.
	Current page	Opens a specific page in the report. To view a specific page, type the page number and press the Enter key.
	Backward	Takes you to the last viewed page. This button is enabled when you move to any page from the initial report page. Clicking this button for the first time also enables the **Forward** button.
	Forward	Takes you to last viewed page before you clicked the Backward button. This button is enabled once you click the **Backward** button.
	Back to parent report	Returns you to the parent report in a drillthrough report.
	Refresh	Refreshes the report.
	Cancel	Cancels the report rendering.

 [1]To view drillthrough reports in WPF Viewer, you need to turn off the Galley mode.

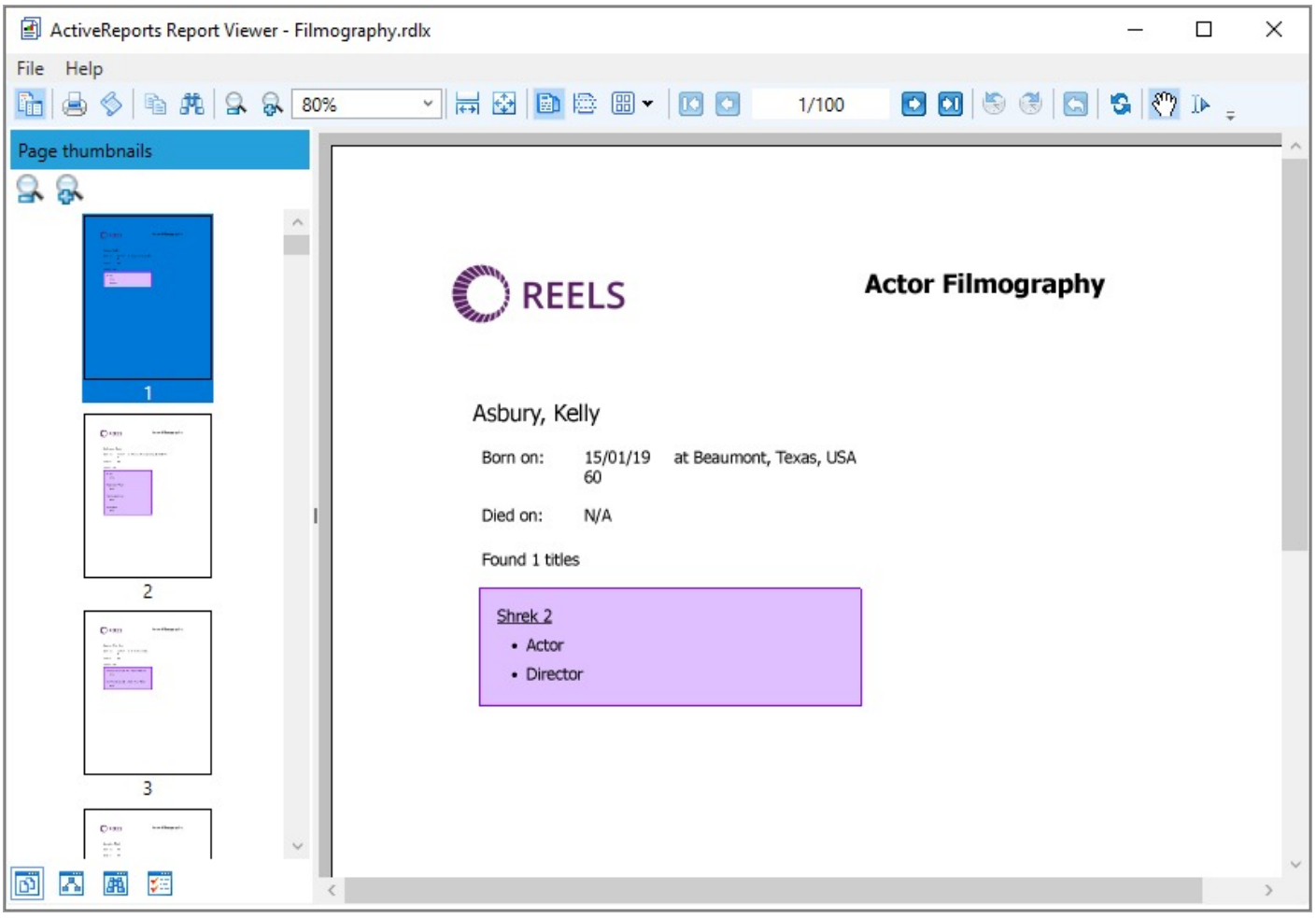
Viewer Sidebar

The Viewer sidebar appears on the left of the Viewer when you click the **Toggle sidebar** button in the toolbar. By default, the sidebar shows the Thumbnails and Search Results panes. The additional Document map and Parameters also appear in this sidebar. You can toggle between any of the viewer panes by clicking the buttons for each pane at the bottom of the sidebar.

Thumbnails pane

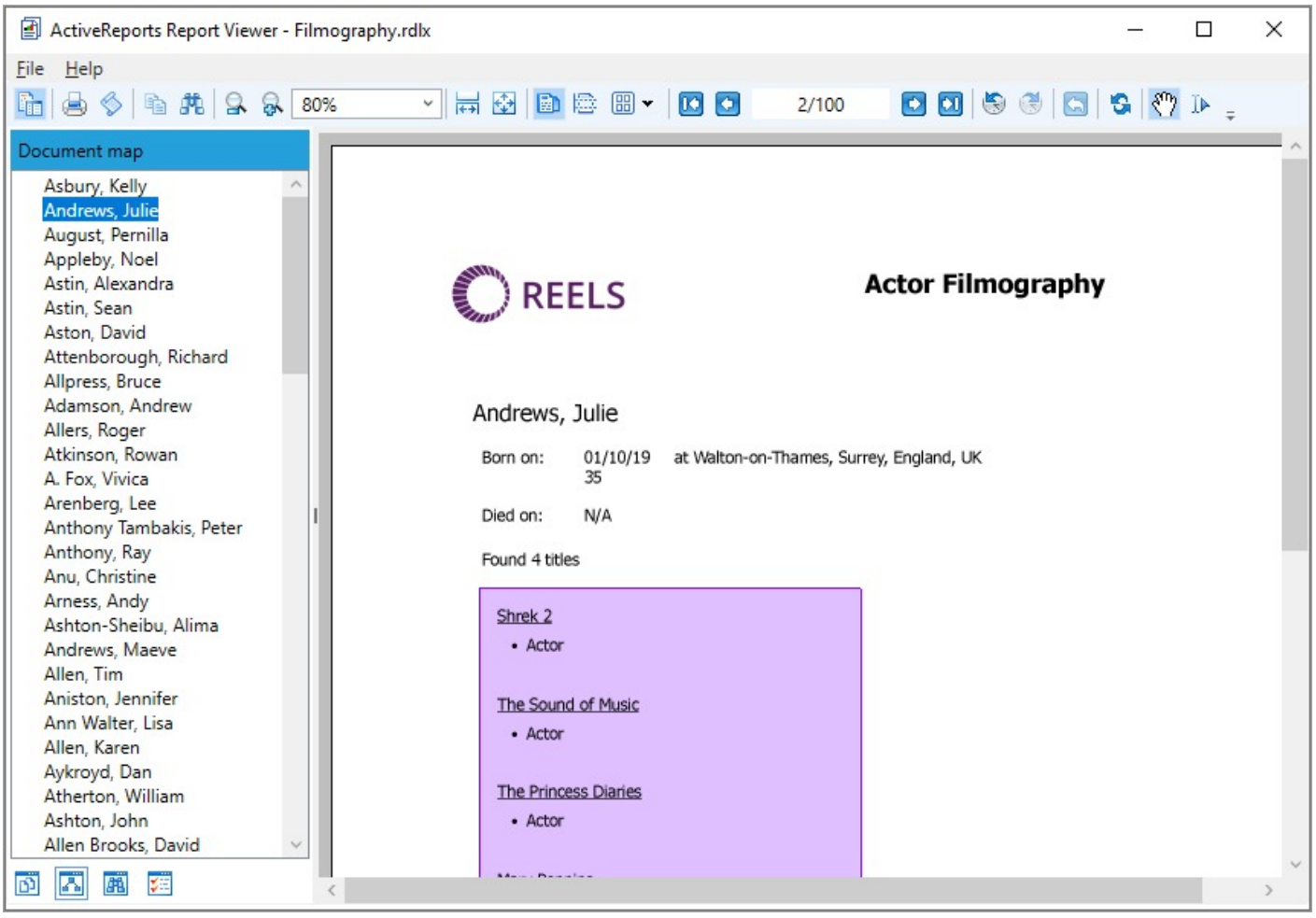
The **Thumbnails** pane appears by default in the sidebar when you click the **Toggle sidebar** button in the toolbar.

This pane is composed of a thumbnail view of all the pages in a report. Click any thumbnail to navigate directly to the selected report page. You can also modify the size of the thumbnail using (+) or (-) button to zoom in and zoom out.



Documents map pane

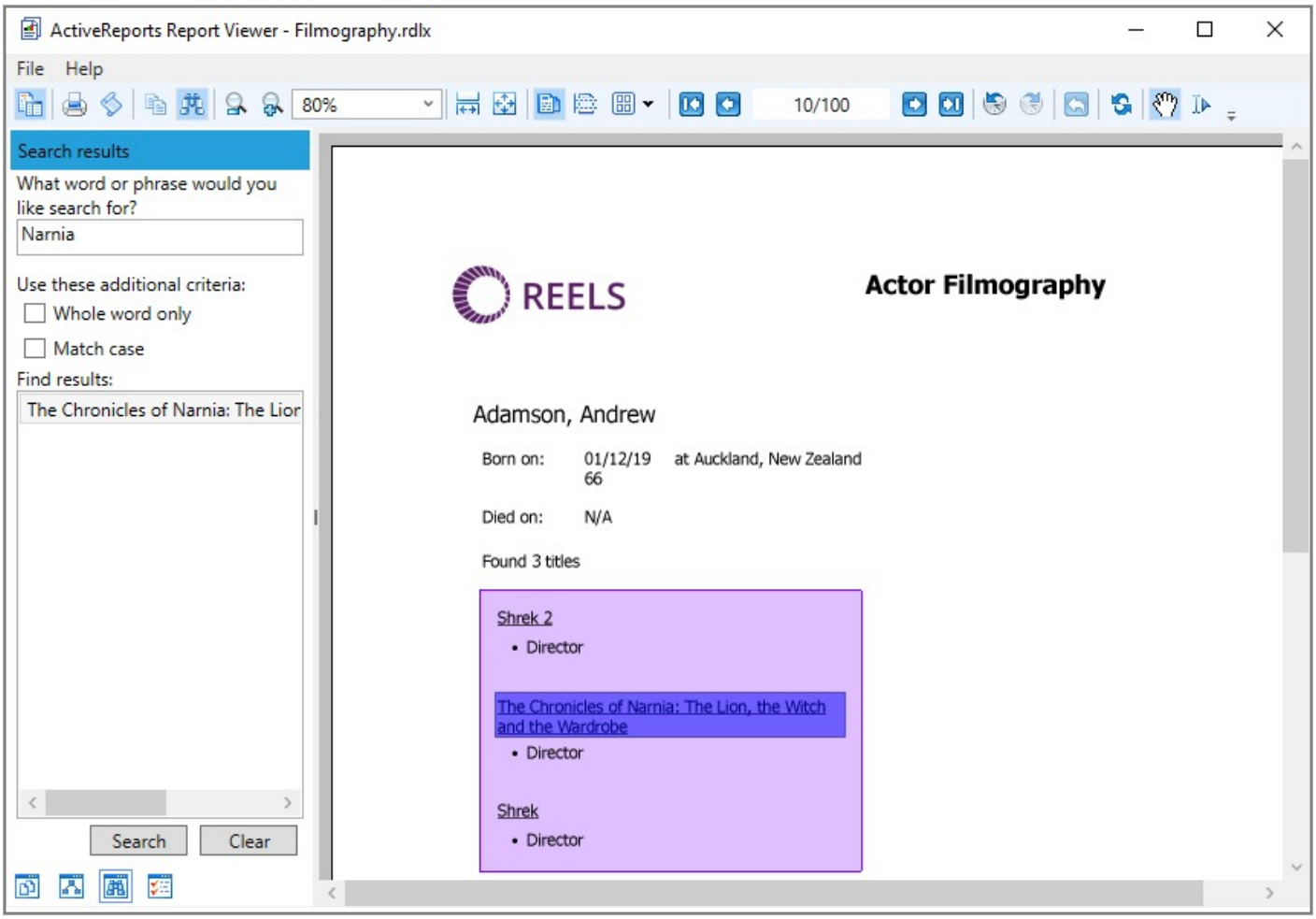
The Documents map pane is enabled for reports where the Label property or the Document map label is set. This pane displays each value for the text box, group, or sub report that you label, and you can click them to navigate to the corresponding area of the report in the Viewer.



If a report does not have the Label property or Document map label set, the Documents map panes does not appear in the sidebar.

Search results pane

The **Search** pane is the other default pane besides Thumbnails that appears in the sidebar when you click the **Toggle sidebar** button. This pane lets you enter a word or phrase from which to search within the report.



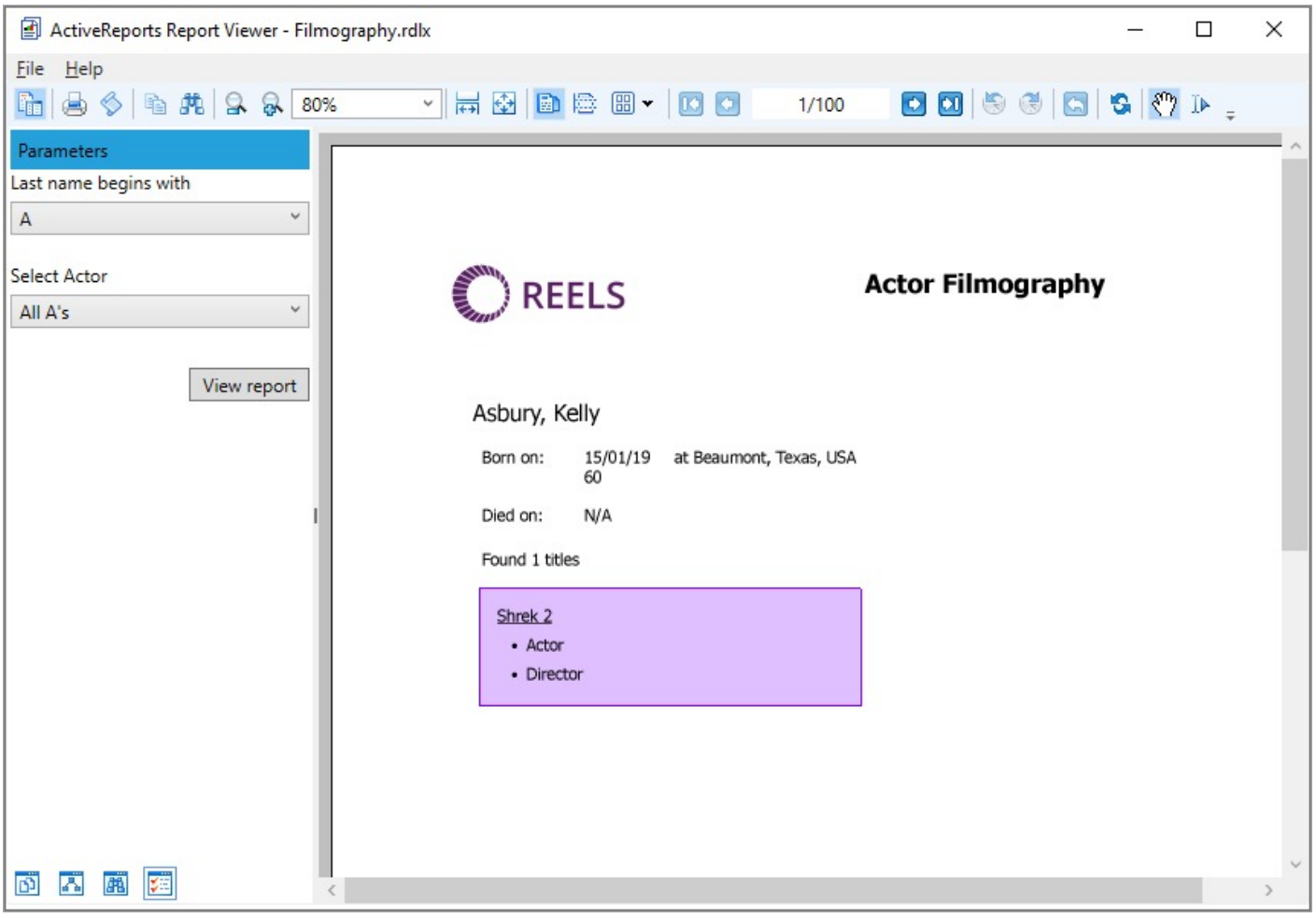
To search in a report:

1. Enter the word or phrase in the search field.
2. Under **Use these additional criteria**, you may optionally choose to search for the whole word or match the case of the search string while searching in the report.
3. Click the **Search** button to see the results appear in the **Find results** list.
4. Click an item in the list to jump to that item in the report and highlight it.

To start a new search or clear the current search results, click the **Clear** button under the **Find results** list.

Parameters pane

The Viewer allows you to view reports with parameters. In the toolbar, click the **Toggle sidebar** button to open the Viewer sidebar and if your report contains parameters, the **Parameters** pane shows up automatically.



1. In the **Parameters** pane, you are prompted to enter a value by which to filter the data to display.
2. Enter a value or set of values and click **View report**, to filter the report data and display the report.

If a report does not have parameters, the Parameters pane does not appear in the sidebar.

Keyboard Shortcuts

The following shortcuts are available on the WPF Viewer.

Keyboard Shortcut	Action
Ctrl + F	Shows the search pane.
Ctrl + P	Shows the print dialog.
Esc	Closes the print dialog.
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 + G	Moves the focus to current page toolbar option.
Shift + rotate mouse wheel	Scrolls horizontally.
Rotate mouse wheel	Scrolls vertically.
F5	Refreshes the report.

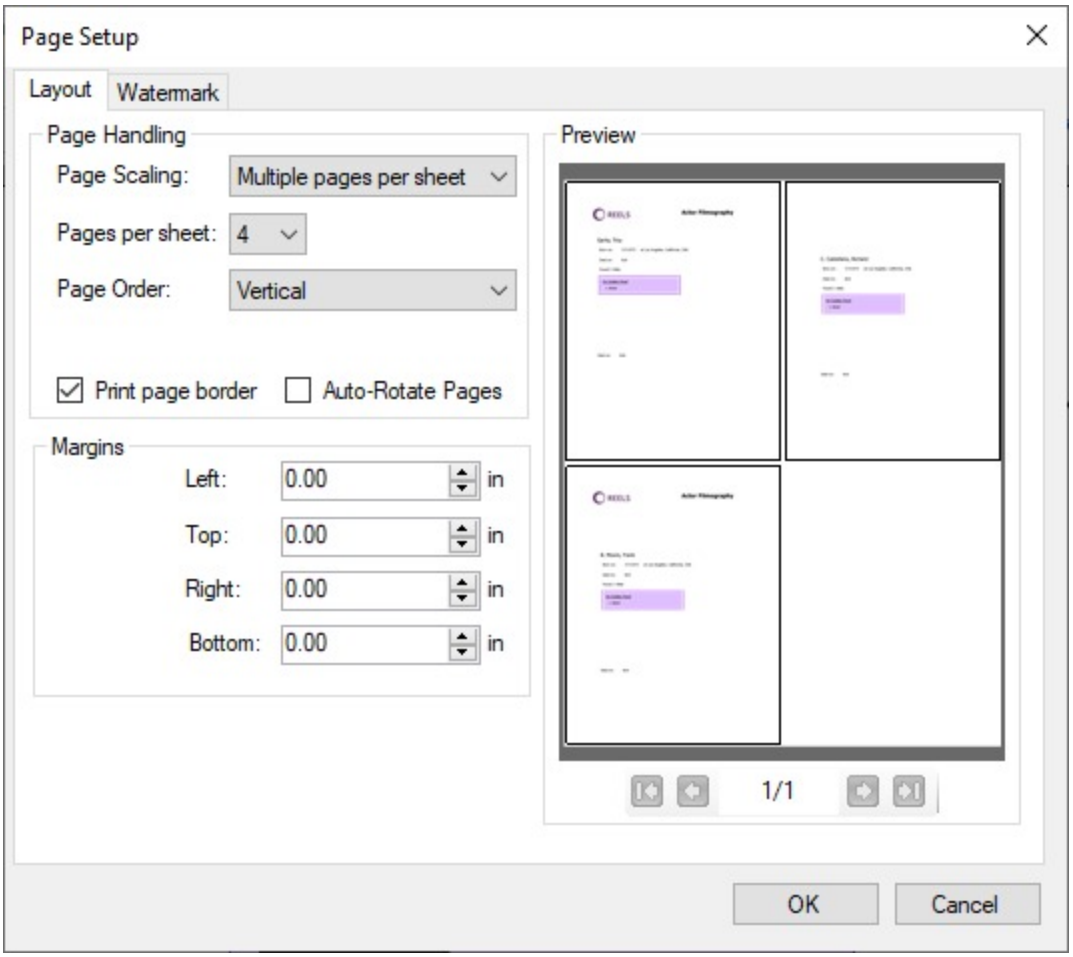
Print in Desktop Viewers

The advanced report printing options in ActiveReports Viewer allow end-users to specify a printer and set options like page scale, page margins, and watermark.

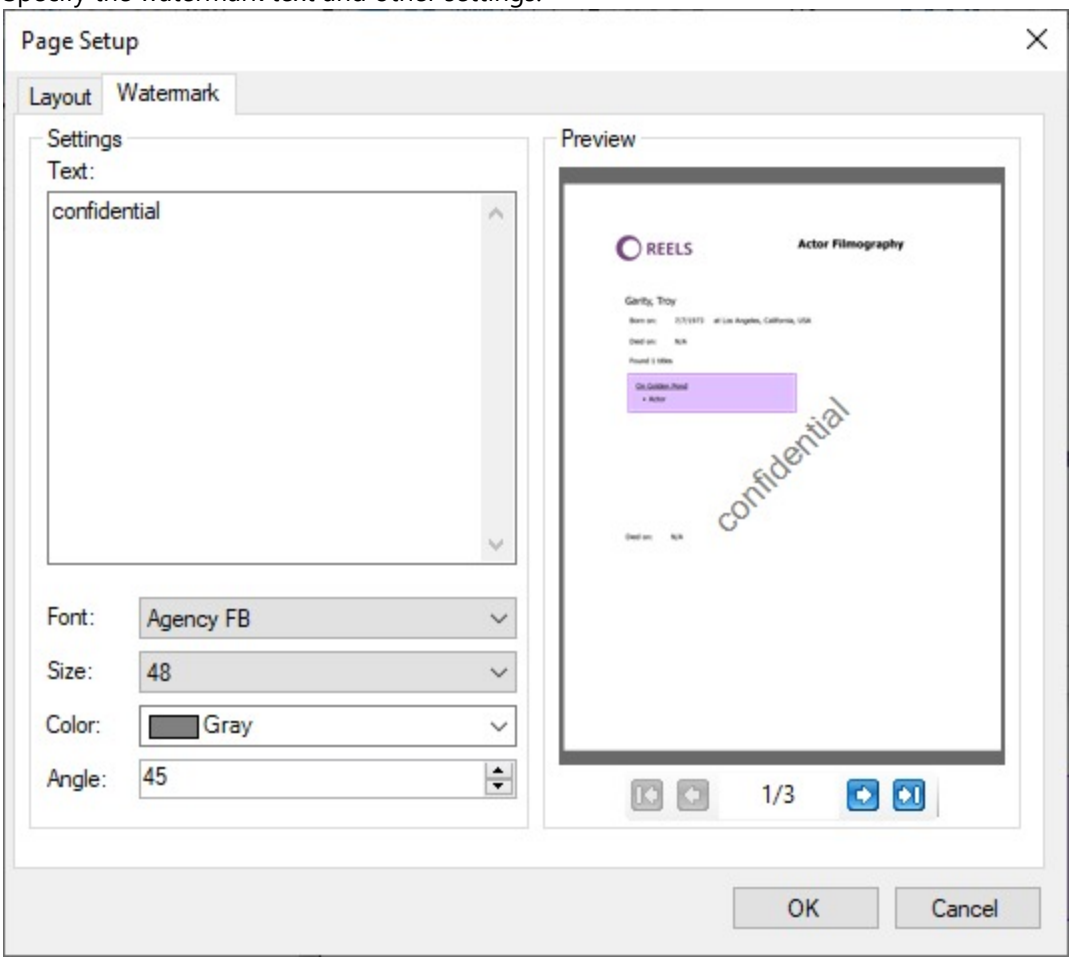
Set Advanced Print Options

1. In the Viewer toolbar, click the **Print** button. See Windows Forms Viewer for information on the Viewer toolbar.
2. In the Print dialog, click **Advanced**.
3. In the Page Setup dialog, go to the **Layout** tab to set page scaling and page margins, and **Watermark** tab to set watermark text.

Property	Description
Layout	Set page scaling and page margins.



Page Scaling	Choose from any of the following options to set the page scaling: • Fit to printable area • Shrink to printable area • Multiple pages per sheet • Booklet
Pages per sheet	Specify the pages per sheet in case you select 'Multiple pages per sheet' in 'Page Scaling'.
Page order	Specify the page order in case you select 'Multiple pages per sheet' in 'Page Scaling'.
Print page border	Select whether to print the page border from: • Horizontal • Horizontal Reversed • Vertical • Vertical Reversed
Auto-Rotate Pages	Select the check-box if you want to set the page orientation based on the content and the page size.
Margins	Enter the values for the Left, Top, Right, and Bottom margins for the page.

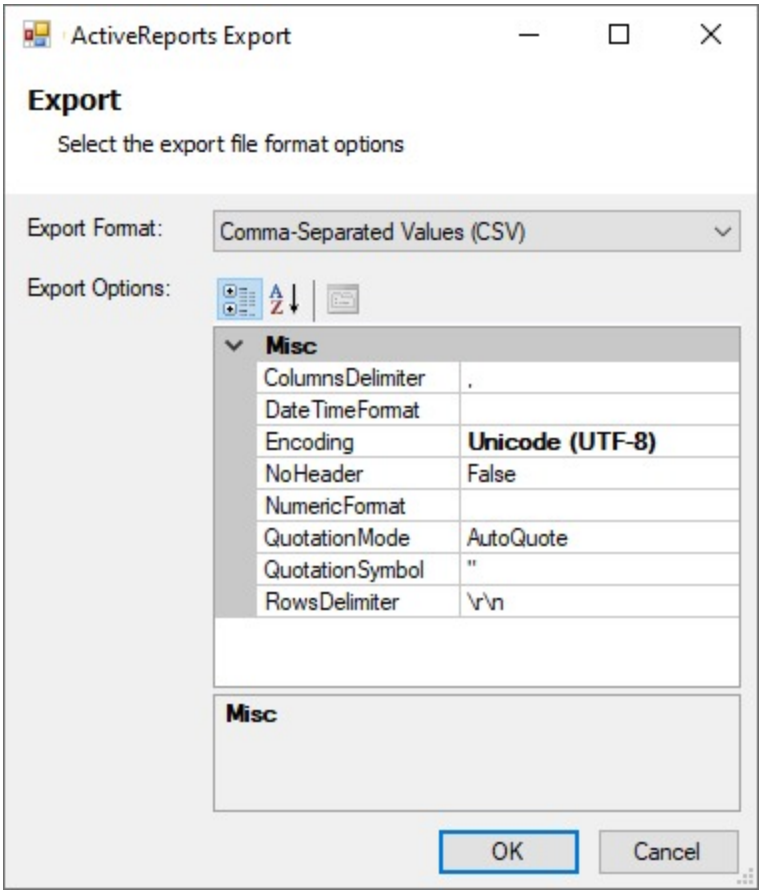
Watermark	Specify the watermark text and other settings. 
Text	Specify the watermark text.
Font	Specify the font style of the text.
Size	Specify the font size of the text.
Color	Set a color for the watermark text.
Angle	Enter a numeric value between 0 and 360. A value of 0 renders straight left-to-right text. A value of 180 renders inverted text.

Export in Desktop Viewers

Your application with the ActiveReports Viewer can contain any exports with any available dialogs. You can export your reports from ActiveReports Viewers to various formats, depending on the report type. This topic discusses all possible export options.

Note: As export options are customizable, the **Export** dialog may display a different list of export options.

To export a report in the Viewer, go to **File** menu and select **Export** (Ctrl+E).



Below you can find full information on properties of each export format option.

Page/RDLX reports

For **Page and RDLX reports**, the following export formats are available.

- Portable Document Format (PDF)
- Microsoft Excel WorkSheet (XLS, XLSX)
- Microsoft Excel WorkSheet - Data (CSV, XLSX)
- Microsoft Word Document (DOC, DOCX)
- Comma-separated values (CSV)
- JavaScript Object Notation (JSON)
- Text Print (TXT)
- Image Format (BMP, GIF, JPEG, TIFF, PNG)
- Hypertext Markup Language (HTML)
- MIME Hypertext Markup Language (MHT)
- Rich Text Format (RTF)
- eXtensible Markup Language (XML)

Portable Document Format (PDF)

Portable Document Format (PDF) is a format recommended for printing and for preserving formatting.

PDF is considered as the best format for printing and it also supports interactive features like Document Map, Bookmarks and Hyperlinks. However, in case you have any data hidden (like in a drill-down report) at the time of rendering, it does not show up in the output.

PDF Rendering Properties

Property	Description
Application	Set the value that appears for application in the Document Properties dialog of the PDF viewer application.
Author	Enter the name of the author to appear in the Document Properties dialog of the PDF viewer application.
CenterWindow	Set to True to position the document's window in the center of the screen.
DisplayMode	Specifies how the document is displayed when opened. FullScreen mode displays the document with no menu bar, window controls, or any other window visible.
DisplayTitle	Set to True to display text you enter in the Title property. When set to False it displays the name of the PDF file.
DpiX	Set the horizontal resolution of the rendered PDF file.
DpiY	Set the vertical resolution of the rendered PDF file.
EmbedFonts	Select how the fonts used in the report should be embedded in the PDF document. **Note:** By default, all fonts get embedded in the exported PDF document.
Encrypt	Determines whether the document is encrypted or not. **Note**: If Encrypt is set to False, permissions and passwords have no effect.
EndPage	The last page of the report to render. The default value is the value for StartPage, that is, 0.
FallbackFonts	Gets or sets a comma-delimited string of font families to locate missing glyphs from the original font.
FitWindow	True to resize the document’s window to fit the size of the first displayed page. Default value: false.
HideMenubar	True to hide the viewer application’s menu bar when the document is active. Default value: false.
HideToolbar	True to hide the viewer application’s toolbars when the document is active. Default value: false.
HideWindowUI	True to hide user interface elements in the document’s window (such as scroll bars and navigation controls), leaving only the document’s contents displayed. Default value: false.
ImageInterpolation	Interpolation value of images. Allows to enable/disable image interpolation, when exporting the file to PDF.
Keywords	Keywords associated with the document.
OwnerPassword	The owner password that can be entered in the reader that permits full access to the document regardless of the specified user permissions.

PageHeight	The page height value, in inches, to set for the report. You must include an integer or decimal value followed by "in" (for example, 1in). This value overrides the report's original settings.
PageWidth	The page width value, in inches, to set for the report. You must include an integer or decimal value followed by "in" (for example, 1in). This value overrides the report's original settings.
Permissions	Specifies the user permissions for the document. Permissions can be combined using a comma between values. In order to use **AllowFillin**, **AllowAccessibleReaders**, and **AllowAssembly** permissions, you must set the **Use128Bit** property to **True**.
PrintLayoutMode	Specifies layout mode to be used for PDF document.
PrintOnOpen	Gets or sets the value indicating whether the document should be printed after open.
PrintPresets	Gets or sets the PDF print preset dialog.
SizeToFit	Determines whether PDF pages are fit to the selected paper size or not.
StartPage	The first page of the report to render. A value of 0 indicates that all pages are rendered.
Subject	The subject of the document.
Title	The title of the document.
Use128Bit	True to use 128 bit encryption with full permissions capability. False to use 40 bit encryption with limited permissions
UserPassword	The user password that can be entered in the reader. If this value is left empty, the user will not be prompted for a password, however the user will be restricted by the specified permissions.
Version	Set the PDF version. The supported versions are: PDF-1.2 | PDF-1.3 | PDF-1.4(default) | PDF-1.5 | PDF-1.6 | PDF-1.7 | PDF-2.0 | PDF/A-1a | PDF/A-1b | PDF/A-2a | PDF/A-2b | PDF/A-2u | PDF/A-3a | PDF/A-3b | PDF/A-3u | PDF/UA-1
WatermarkAngle	Specify the degree of angle for the watermark text on the PDF document. Valid values range from 0 to 359, where 0 is horizontal, left to right.
WatermarkColor	Select a color for the watermark text on the PDF document. The default value for the watermark color is gray, but you can select any Web, System, or Custom color.
WatermarkFont	Set the font to use for the watermark on the PDF document.
WatermarkFontSize	Set the font size to use for the watermark on the PDF document.
WatermarkFontStyle	Set the font style to use for the watermark on the PDF document.
WatermarkTitle	Enter text (i.e. CONFIDENTIAL) to use as the watermark on the PDF document.

Microsoft Excel WorkSheet (XLS, XLSX)

You can export excel files in two formats: Xls and Xlsx. XLSX 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 XLSX export filter has a number of useful properties that allow you to control your output.

Reports rendered in Excel support a number of interactive features like Bookmarks and Hyperlinks. However, in case you have any data hidden at the time of rendering (like in a drill-down report), it does not show up in the output. It is recommended that you expand all toggle items prior to rendering.

Excel Rendering Properties

Property	Description
PageSettings	Initializes Excel file print settings: PageOrientation and PaperSize.
Pagination	Forces pagination or galley report layout mode.
RightToLeft	Shows direction of sheets from right to left.
Security	Initializes the document security.
UseDefaultPalette	Indicates whether to export the document with the default Excel palette.
FileFormat	Specifies the output format of the Excel document, i.e. Xls or Xlsx.
OpenXmlStandard	Specifies the level of Open XML document conformance on exporting in Xlsx file format. You can choose from the following values: • Strict: The default value. • Transitional: The Excel file generated by scheduled task execution using Strict (the default value of OpenXMLStandard) cannot be viewed on IOS devices.
MultiSheet	Indicates whether to generate a single-sheet or multi-sheet Excel document.
EnableToggles	Allows to export collapsible rows in the detail and row groups of the **Table** control of an RDLX report. This property gets displayed in the Export menu when the **Pagination** property is set to **False**

Microsoft Excel WorkSheet - Data (CSV, XLSX)

You can export excel files in two formats, Xlsx and Csv.

Excel Data exports only data from Tablix, Table, and Matrix data regions, preserving the data region structure and ignoring layout-related features (page break, cumulative total, etc). Other controls and data regions of the original report are ignored at this export.

> **Note**: If a report does not contain any data region (Table or Tablix), a Csv file is not generated.

For the Xlsx format, when a report has multiple data regions, each data region is exported to a separate excel sheet.

For the Csv format, a separate CSV file is created for each data region, available in the report.

Excel Data Rendering Properties

Property	Description
Csv	Csv related properties. See **Csv Rendering Properties** below.
FileFormat	Indicates whether to use Csv or OpenXml format for the output file.
Xlsx	OpenXml related properties. See **Xlsx Rendering Properties** below.

Csv Rendering Properties

Property	Description
ColumnsDelimiter	Sets or returns the text inserted between columns.
Encoding	Specifies the encoding schema for output.
NoHeader	Specifies whether to omit the CSV Header.
QuotationSymbol	Sets or returns the qualifier character to put around results.
RowsDelimiter	Sets or returns the text inserted between rows.

Xlsx Rendering Properties

Property	Description
AllowImages	Indicates whether to allow images or just plain data content.
AutoRowsHeight	Indicates whether to export rows height or specify auto height.
OpenXmlStandard	Specifies the level of Open XML document conformance on exporting in Xlsx file format. You can choose from the following values: • Strict: The default value. • Transitional: The Excel file generated by scheduled task execution using Strict (the default value of OpenXMLStandard) cannot be viewed on IOS devices.
RightToLeft	Shows direction of sheets from right to left.
Security	Initializes the document security.
UseCompression	Indicates whether to use compression on exporting to an Xlsx file.

Microsoft Word Document (DOC, DOCX)

You can export Page reports and RDLX reports to Microsoft Office Open XML (OOXML) format (.Docx) or Word HTML format (.Doc) using the **FileFormat** property.

The Word HTML format (.Doc) provides greater layout accuracy for Page and RDLX Reports in Microsoft Word, while the OOXML format (.Docx) provides excellent editing experience for the exported reports.

> **Note**: OOXML and Word HTML formats have a great difference: while Word HTML is almost WYSIWYG, OOXML allows editing and has a number of limitations. To learn about the limitations for both Word HTML and OOXML, see Word HTML/OOXML Limitations.

The OOXML format (.Docx) is recommended in the following scenarios:

- **Open exported reports in a wide range of applications**: You can open and modify the exported Word document in any of the following applications.
  - Microsoft Office 2013+
  - Microsoft Office for Mac 2016+
  - iWork and Pages for OS X (all supported versions)
  - LibreOffice
  - Google Quickoffice for Android

    - Documents Free (Mobile Office Suite) by SavySoda for iOS

- **Customize reports after exporting:** Positioning and arrangement of report elements in the exported document is implemented using the OOXML format (.Docx) which provides a natural document flow for editing the exported documents.
- **Use Word automation features:** With support for automation features in the OOXML format (.Docx), tasks that previously required manual adjustments in the exported Word document are now handled automatically. Report elements such as page header and footer, expressions, heading levels, and table of contents are automatically transformed to the OOXML format (.Docx).
- **Set compatibility mode:** You can render a report as a Word document that is compatible with Microsoft Word 2007, 2010, or 2013 using the **DocumentCompatibleVersion** property from the export settings.

Word Rendering Properties

Common properties

Property	Description
Author	Sets the name of the author that appears in the Author field of the Properties dialog in the rendered Word document.
FileFormat	Sets the output file format to HTML (.Doc) or OOXML (.Docx). By default the file format is set to HTML format.
Title	Sets the title for a document that appears in the Title field of properties dialog in the rendered Word document.

HTML format

Property	Description
BaseUrl	Sets the base URL for any relative hyperlinks that appear in the Hyperlink base field of the Properties dialog in the rendered Word document.
Generator	Sets the identity of the document generator in the rendered Word document.
PageHeight	Sets the height of the report pages in inches for the rendered Word document. The value in this property overrides the original settings in the report.
PageWidth	Sets the width of the report pages in inches for the rendered Word document. The value in this property overrides the original settings in the report.

OOXML format

Property	Description
CompanyName	Sets the name of the organization or company that appears in the Company field of Properties dialog in the rendered Word document.
DocumentCompatibilityVersion	Sets the compatibility mode of the document to previous versions (Microsoft Word 2007 - 2013) of Word. By default the compatibility version is set to Word2013.
DpiX	Sets the horizontal resolution of the images in the rendered Word document. By

	default DpiX is set to 96.
DpiY	Sets the vertical resolution of the images in the rendered Word document. By default DpiY is set to 96.
PageOrientation	Sets a value that specifies whether the document pages should be printed in portrait or landscape in the rendered Word document.
PaperSize	Sets the paper size for the page.
Password	Sets a password that must be provided to open the rendered Word document.
ReadOnlyRecommended	Sets a value that indicates whether Microsoft Office Word displays a message whenever a user opens the document, suggesting that the document is read-only.
WritePassword	Sets the write password that is required for saving changes in the rendered Word document.
TOCAutoUpdate	Automatically updates the TableOfContents control while opening the Word document. By default TOCAutoUpdate is set to False.

Comma-separated values (CSV)

Comma-Separated Values (CSV) is a form of structured data in a plain text. The text in a CSV file is saved as series of values separated by comma.

CSV Rendering Properties

Property	Description
ColumnsDelimiter	Sets or returns the text inserted between columns.
DateTimeFormat	Specifies the default format for date values, for example, 'yyyy-MM-dd'.
Encoding	Specifies the encoding schema for output.
NoHeader	Specifies whether to omit the CSV Header.
NumericFormat	Specifies the format for numeric values, for example, '0.####'.
QuotationMode	Specifies whether to add double quotes to the exported data. • **AutoQuote** – Simple values are exported without quotes. The quotes are added only when the data contains column or row delimiters. This is the default export behavior. • **AlwaysQuote** – Exported values are always quoted.
QuotationSymbol	Sets or returns the qualifier character to put around results.
RowsDelimiter	Sets or returns the text inserted between rows.

JavaScript Object Notation (JSON)

JavaScript Object Notation (JSON) is a text-based data format in which the data is stored in the hierarchical form.

JSON Rendering Properties

Property	Description
Formatted	Specifies whether to format the file with tabs and spaces for readability.
QuotePropertyNames	Specifies whether to enclose property names in quotation marks.

Image Format (BMP, GIF, JPEG, TIFF, PNG)

Image Format option converts your report to an image file. Make sure that you select the **ImageType** property to any of the image formats available: BMP, GIF, JPEG, TIFF, and PNG.

By default, a separate file is created for each page in a report and an index to each corresponding file name is added (for example, image001.PNG, image002.PNG, etc).

Reports rendered as images do not support any of the interactive features of ActiveReports reports. Any data hidden at the time of export is hidden in the image.

> **Note:** To render the entire report as a single image, set the **Pagination** property to False.

Image Rendering Properties

Property	Description
Compression	Sets or returns a value which specifies the compression to be used when exporting.
Dither	Specifies whether the image should be dithered when saving to a black and white output format, like CCITT3 or Rle. This property has no effect if the CompressionScheme property is set to Lzw or None(represents color output).
DpiX	Adjust the horizontal resolution of rendered images. The default value is 96.
DpiY	Adjust the vertical resolution of rendered images.
ImageType	Select the type of image to which you want to render the report. Supported types are BMP, GIF, JPEG, TIFF, and PNG.
Pagination	By default, each page of a report is rendered as a separate image. Set this value to **False** to render the entire report as a single image.
Quality	Gets or sets the quality of the report to be rendered as an image.

Hypertext Markup Language (HTML)

HTML, or hypertext markup language, is a format that opens in a Web browser. You can export your reports to HTML or MHT formats. It is a good format for delivering content because virtually all users have an HTML browser.

Reports rendered in HTML support a number of interactive features. Hyperlinks, Bookmarks and Drill through links can be rendered to HTML. However, Document Maps are not available in this format. For a drill down report, make sure that the data you want to display is expanded before rendering, otherwise it renders in the hidden state.

> **Note**: HTML is not the best format for printing. Use the PDF rendering extension instead.

HTML Rendering Properties

Property	Description
MhtOutput	Gets or sets whether or not the output should be in Mht format. True indicates the output should be in Mht format; otherwise false. The default is false.
RenderingEngine	The RenderingEngine property is set to **Mixed** by default for improved quality output. The choices are Html or Mixed, where Mixed uses SVG to render charts.
StyleStream	Set the StyleStream to True to create an external .css file containing style information from your report controls' style properties. If you prefer to have style information embedded in the HTML file, set the StyleStream property to False.
LinkTarget	Specify a link target to control whether drill down reports and other links open in a new window or reuse the current window. By default, no value is set and links open in the same window. A value of _blank opens the link in a new window, or you can specify a window using window_name. By default this value is not set.
Mode	Galley mode renders the report in one HTML stream. Select Paginated mode to render each page as a section inside the HTML document.
OutputTOC	Indicates whether the report's existing TOC should be added in the output.

MIME Hypertext Markup Language (MHT)

The MHT file extension refers to MIME HTML documents, a format for archiving web pages saved by web browsers, including Internet Explorer and Firefox. MHT files can contain resources like images, flash, java, audio, external links and html code. At that, MHT files contain all elements within a single file.

MHT Rendering Properties

Property	Description
Fragment	Set to True to return only the contents inside the body tags (to embed it within a Web page). Set to False to return the full HTML text.
Link Target	Enter a value of a target for hyperlinks contained inside a report. A value of _blank opens a new window; _self opens in the same window.
Mode	Select **Paginated** to render each page as a section inside the HTML document, with page headers and footers. The **Galley** mode renders one long page with a single page header and footer.
Output TOC	Indicates whether to include a table of contents, if available, in the exported report.

Rich Text Format (RTF)

> **Note:** The RTF export, otherwise supported only in Section reports, is possible in Page/RDLX reports too since this export uses the section report document (RDF format) internally.

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.

RTF Rendering Properties

Property	Description
EnableShapes	Indicates whether to export the Shapes and Lines to the RTF format if set to True. Microsoft Word is required to view it correctly.
Pagination	Indicates whether to use pagination for the output RTF document.

eXtensible Markup Language (XML)

XML is a useful format for delivering data to other applications as the resulting XML file opens in an internet browser.

XML format does not support interactive features except that when rendering a report to XML, complete drill-down data is shown regardless of whether the data is rendered in expanded state or not.

Xml Rendering Properties

Property	Description
Encoding	Select the encoding schema to use in the XML transformation.
XslStylesheet	Select the existing XSL Stylesheet file to use to transform the resulting XML file. **Note**: When using the XslStylesheet option, be sure to save the file in the correct file format, such as HTML.

Section Reports

For **Section reports**, you can use these export format options.

- Portable Document Format (PDF)
- Microsoft Excel Workbook (XLS, XLSX)
- Hypertext Markup Language (HTML)
- MIME Hypertext Markup Language (MHT)
- Rich Text Format (RTF)
- Tagged Image Format (TIFF)
- Plain Text (TXT)

PDF Document Format (PDF) (Section report)

PDF Rendering Properties

Property	Description
ConvertMetaToPng	Sets or returns a value indicating whether Windows metafiles are converted to PNG files in the exported PDF document.
ExportBookmarks	Sets or returns a value indicating whether bookmarks are exported to the PDF document.
FontFallback	Gets or sets a comma-delimited string of font families that will be used to lookup glyphs missing in the original font.
ImageInterpolation	Specifies the images interpolation value. Allows to enable/disable image interpolation, when exporting the file to PDF.

ImageQuality	Specifies the quality used for any images that are converted by ActiveReports. Note that if a JPG image is used in the report, it is written directly to PDF without any conversion. Other image formats may incur a conversion, which this value will effect.
ImageResolution	Sets or returns the resolution of images converted from metafiles.
NeverEmbedFonts	Sets or returns a semicolon-delimited string of values indicating fonts that should not be embedded in a PDF document. **Note**: When you add fonts to NeverEmbedFonts property, 2 bytes characters may be distorted in PDF output since the export filter in such cases uses the glyphs from the default system font.
Options	Returns an object allowing you to specify viewer preferences and document information options for the exported PDF document.
Application	Set the value that appears for application in the Document Properties dialog of the PDF viewer application.
Author	Enter the name of the author to appear in the Document Properties dialog of the PDF viewer application.
CenterWindow	Set to True to position the document's window in the center of the screen.
DisplayMode	Specifies how the document is displayed when opened. FullScreen mode displays the document with no menu bar, window controls, or any other window visible.
DisplayTitle	Set to True to display text you enter in the Title property. When set to False it displays the name of the PDF file.
FitWindow	True to resize the document’s window to fit the size of the first displayed page. Default value: false.
HideMenubar	True to hide the viewer application’s menu bar when the document is active. Default value: false.
HideToolbar	True to hide the viewer application’s toolbars when the document is active. Default value: false.
HideWindowUI	True to hide user interface elements in the document’s window (such as scroll bars and navigation controls), leaving only the document’s contents displayed. Default value: false.
Keywords	Keywords associated with the document.
OnlyForPrint	Indicates whether the PDF is only for print.
Subject	The subject of the document.
Title	The title of the document.
PrintPresets	Returns an object allowing you to specify the print presets.
Security	Returns the PdfSecurity object for initializing document encryption and security.
Encrypt	Determines whether the document is encrypted or not. **Note**: If Encrypt is set to False, permissions and passwords have no effect.
OwnerPassword	The owner password that can be entered in the reader that permits full access to the document regardless of the specified user permissions.
Permissions	Specifies the user permissions for the document. Permissions can be combined using a comma

	between values. In order to use **AllowFillin**, **AllowAccessibleReaders**, and **AllowAssembly** permissions, you must set the **Use128Bit** property to **True**.
Use128Bit	True to use 128 bit encryption with full permissions capability. False to use 40 bit encryption with limited permissions
UserPassword	The user password that can be entered in the reader. If this value is left empty, the user will not be prompted for a password, however the user will be restricted by the specified permissions.
Version	Sets or returns the version of the PDF format the exported document is saved in.
Pagination	Gets or sets the value that indicates whether to use pagination in the exported PDF document. This property is only useful for Page/RDLX reports exports through RDF and does not affect a section report.

Microsoft Excel Workbook (XLS, XLSX) (Section report)

Excel Rendering Properties

Property	Description
PageSettings	Initializes Excel file print settings: PageOrientation and PaperSize.
Pagination	Forces pagination or galley report layout mode. This property is only useful for Page/RDLX reports exports through RDF and does not affect a section report.
RightToLeft	Shows direction of sheets from right to left.
Security	Initializes the document security.
UseDefaultPalette	Indicates whether to export the document with the default Excel palette.
FileFormat	Specifies the output format of the Excel document, i.e. Xls or Xlsx.
OpenXmlStandard	Specifies the level of Open XML document conformance on exporting in Xlsx file format. You can choose from the following values: • Strict: The default value. • Transitional: The Excel file generated by scheduled task execution using Strict (the default value of OpenXMLStandard) cannot be viewed on IOS devices.
MultiSheet	Indicates whether to generate a single-sheet or multi-sheet Excel document.
EnableToggles	Allows to export collapsible rows in the detail and row groups of the **Table** control of an RDLX report. This property gets displayed in the Export menu when the **Pagination** property is set to **False**

Hypertext Markup Language (Section report)

HTML Rendering Properties

Property	Description

BookmarkStyle	Sets or returns a value indicating whether to create a page of bookmarks if the ActiveReports document contains bookmarks.
CharacterSet	Sets or returns the character set encoding that will be used in the exported HTML pages.
CreateFramesetPage	Sets or returns a value indicating whether the HTML pages appear in a frame set. If set to True, any bookmark entries appear on the left, and the report document contents appear on the right. The resulting file will use the specified filename with the extension ".frame.html".
IncludeHtmlHeader	Sets or returns a value indicating whether the exported HTML files will include normal HTML page headers such as the HTML, HEAD, and BODY elements.
IncludePageMargins	Sets or returns a value indicating whether the page's margins are included in the output.
MultiPage	Sets or returns a value indicating whether multiple HTML pages are generated for the document.
Output Type	Gets or sets a value indicating whether the document will be exported as DHTML or HTML.
Pagination	Gets or sets the value, that indicates whether to use pagination for the resultant html document. This property is only useful for Page/RDLX reports exports through RDF and does not affect a section report.
RemoveVerticalSpace	Sets or returns a value indicating whether to completely remove empty vertical spacing from the output.
Title	Sets or returns the Title used in the header of HTML pages.

MIME Hypertext Markup Language (MHT) (Section report)

The MHT file extension refers to MIME HTML documents, a format for archiving web pages saved by web browsers, including Internet Explorer and Firefox. MHT files can contain resources like images, flash, java, audio, external links and html code. At that, MHT files contain all elements within a single file.

MHT Rendering Properties

Property	Description
BookmarkStyle	Sets or returns a value indicating whether to create a page of bookmarks if the ActiveReports document contains bookmarks.
CharacterSet	Sets or returns the character set encoding that will be used in the exported HTML pages.
CreateFramesetPage	Sets or returns a value indicating whether the HTML pages appear in a frame set. If set to True, any bookmark entries appear on the left, and the report document contents appear on the right. The resulting file will use the specified filename with the extension ".frame.html".
IncludeHtmlHeader	Sets or returns a value indicating whether the exported HTML files will include normal HTML page headers such as the HTML, HEAD, and BODY elements.
IncludePageMargins	Sets or returns a value indicating whether the page's margins are included in the output.
MultiPage	Sets or returns a value indicating whether multiple HTML pages are generated for the document.
Output Type	Gets or sets a value indicating whether the document will be exported as DHTML or HTML.

Pagination	Gets or sets the value, that indicates whether to use pagination for the resultant html document. This property is only useful for Page/RDLX reports exports through RDF and does not affect a section report.
RemoveVerticalSpace	Sets or returns a value indicating whether to completely remove empty vertical spacing from the output.
Title	Sets or returns the Title used in the header of HTML pages.

Rich Text Format (RTF) (Section report)

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.

RTF Rendering Properties

Property	Description
EnableShapes	Indicates whether to export the Shapes and Lines to the RTF format if set to True. Microsoft Word is required to view it correctly.
Pagination	Indicates whether to use pagination for the output RTF document. This property is only useful for Page/RDLX reports exports through RDF and does not affect a section report.

Tagged Image Format (TIFF) (Section report)

TIFF Rendering Properties

Property	Description
CompressionScheme	Specifies the compression scheme to be used when exporting a TIFF file.
Dither	Specifies whether the image should be dithered when saving to a black and white output format, like CCITT3 or Rle. This property has no effect if the CompressionScheme property is set to Lzw or None(represents color output).
DpiX	Adjust the horizontal resolution of rendered images. The default value is 96.
DpiY	Adjust the vertical resolution of rendered images.
Pagination	By default, each page of a report is rendered as a separate image. Set this value to **False** to render the entire report as a single image. This property is only useful for Page/RDLX reports exports through RDF and does not affect a section report.

Plain Text (TXT) (Section report)

Export your ActiveReports documents to plain text.

TXT Rendering Properties

Property	Description

Encoding	Gets or sets the character encoding used for the exported text.
PageDelimiter	Gets or sets the character or sequence of characters that marks the beginning or end of a page.
QuotationSymbol	Gets or sets the quotation symbol.
SuppressEmptyLines	Gets or sets a value which determines whether empty lines will be inserted for layout purposes.
TextDelimiter	Gets or sets the character or sequence of characters that marks the beginning or end of a text field.

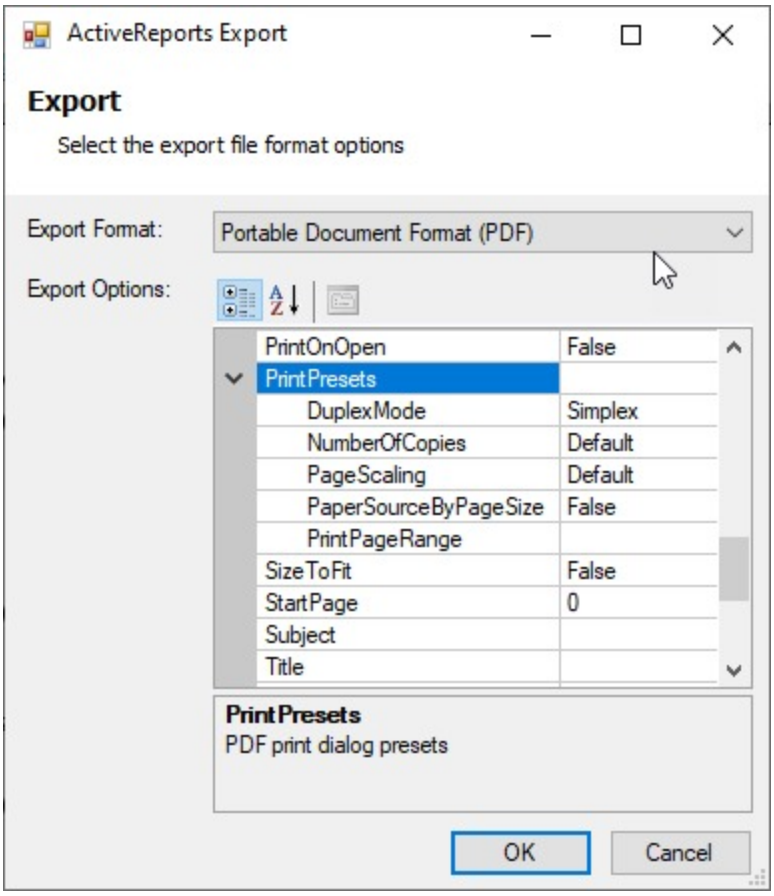
Print Presets

To economize your efforts each time a PDF document is printed, you can preset basic print options when exporting a report to a PDF format.

> **Note:** The print preset properties are only available with the Professional Edition license. An evaluation message is displayed when used with the Standard Edition license.

Set PDF Print Presets using Export Dialog

1. Open the **Export** dialog.
2. In the **Export Format** field of the **Export** dialog, select Portable Document Format (PDF).
3. Expand **PrintPresets** options and set the required properties for print presets.



Note: These properties are available in PDF version 1.7 or higher.

Property	Description
DuplexMode	Specify the printer duplex mode.
NumberOfCopies	Specify the number of print copies.
PageScaling	Specify the page scale. The PageScaling property is supported in PDF version 1.6.
PaperSourceByPageSize	Set it to 'True' to set the page size according to the PDF page size rather than the page specified in the page setup option.
PrintPageRange	Specify the page range to print.

Export Limitations - Word HTML/OOXML

HTML format

- Although background colors for controls export to Word documents, background colors for sections such as Body and Page Header or Footer do not.
- The BackgroundImage is not supported when used in TextBox and CheckBox controls embedded in data regions such as Table and Tablix.
- The BackgroundImage is not supported for reports, and for List, Container, Shape, FormattedText, Table, and

Tablix report controls.
- KeepTogether property of Table/Tablix is not supported.
- Some FormattedText tags, for example bi and <s>es</s>, are not exported to Word.
- Image alignments other than the defaults (HorizontalAlignment: Left and VerticalAlignment: Top) are not supported.
- Checkbox color does not affect the color of the square.

OOXML format

Report properties

- The LineSpacing property of a report's style sheet, StartPageNumber property of the report, PrintOnLastPage property of the PageHeader and PageFooter are not supported.
- For Page reports, some of the NumberingStyle property (DocumentMap settings) options are not supported. The supported NumberingStyle options are Decimal, DecimalZero, LowerLetter, UpperLetter, LowerRoman, UpperRoman.
- BackgroundImage is not supported for report item except Shape.
- The BackgroundRepeat property of the BackgroundImage is not supported in Page (Page reports) and Body (RDLX reports).
- For RDLX reports, Background and Border properties of Page Header or Page Footer are not supported.
- Microsoft Word calculates the columns width by the document width. an RDLX report calculates the columns width based on the body width, therefore the width of columns in an exported RDLX report may differ from an original RDLX report.
- In Microsoft Word, the maximum supported page size is 22 inches (55.87 cm) wide and 22 inches (55.87 cm) high. If an exported report exceeds the maximum size, some data may be lost during export.
- In Microsoft Word, a table can have a maximum of 63 columns. If an exported report table has more than 63 columns, then the table is split and therefore an exported document may differ from an original report.
- A repeated Table Footer (the RepeatOnNewPage property of Table Footer) or multiple repeated headers on a single page are not supported.
- The OverflowPlaceHolder control is not supported.
- If the PrintOnFirstPage property of PageHeader or PageFooter is set to False, then both PageHeader and PageFooter will not be available on the first page of the exported document.
- The report data gets rendered only in first theme if a Page report containing multiple themes is exported to Docx format.

Report controls

- The Inset, Outset, and window-inset border styles (in BorderStyle property) are not supported.
- The Map, Chart, Image, Barcode, SparkLine, Bullet, and CustomControl report controls are exported as an image. If a report control uses the BorderColor, BorderStyle or BorderWidth properties, a report is exported as a table.
- The BorderWidth property of report controls is not exported as is and may differ from the original BorderWidth value.
- PageBreaks are not fully supported. The report contents exported to the Word's table or cell items do not support the page breaks.
- For Shape report control, if the BorderStyle property is set to Double, it is exported as Solid.
- For Line control, if LineStyle property is set to Double/Transparent, it is exported as Solid.
- For BandedList data region, only the BandedList Header is repeated on each page. The BandedList Footer, GroupHeaders and GroupFooters are not supported.
- Tablix data region is exported as a single table without horizontal split.
- For Image control, Border properties and Padding properties are not supported if the Sizing property is set to

Clip.

- For Container report control, rounding corners (the RoundingRadius property) are not supported.
- RepeatToFill property for Table and Tablix is not supported.
- Images embedded in a Table data region are not properly supported.
- Pagination is not supported due to difference in the layouts of ActiveReports and Word.
- Page Number in Section (Page N of M(Section)) is not supported.
- KeepTogether property of Table/Tablix is not supported.
- Horizontal aligned images may overlap in iWord.
- The properties set to the 'Body' region of a Subreport are not exported.
- If a report contains overlapped report controls, these controls appear side by side in an exported Word document and not overlapped as in the Designer's preview.
- **FormattedText** limitations.
    - FormattedText is exported as it is. It does not support all HTML and CSS features.
    - The <a> tag without a href attribute, <abbr> and <q> tags are exported as simple text.
    - For Border Styles - Inset and Outset, the <ul> tags are not exported.
    - The BackgroundColor, BackgroundImage, BorderColor, BorderStyle and BorderWidth properties are not supported.
    - Anchors with an href attribute are exported as hyperlinks.
    - Headers like h1, h2, etc. are exported as corresponding Microsoft Word built-in header styles.
- **TableOfContents** limitations.
    - The TextAlign, DisplayPageNumber, TextIndent, and Overline TextDecoration properties are not supported.
    - The Source property of the Document Map settings is not fully supported. Only the Headings Only option of the Source property is supported.
    - If a report uses more than one TableOfContents controls, the properties of the first TableOfContents are applied to the other TableOfContents controls in the exported document.
    - The FillCharacter property is exported as dots.
    - The background of TOC control appears black on opening exported file in LibreOffice.
- **TextBox/CheckBox** limitations.
    - If the Format property is set to Numeric or Date, the exported TextBox has a right alignment. Other Format values are exported with the left alignment.
    - The Transparent color for text is exported as white.
    - The Underline for numbered lists, Right-To-Left (RTL) option of the Direction, Angle, ShrinkToFit and Overline TextDecoration properties are not supported.
    - The action Jump to report is not supported.
    - The tb-rl (vertical text) option of the WritingMode property is not supported. TextBoxes with the WritingMode property set to tb-rl are exported as lr-tb.
    - The NoWrap option of the WrapMode property is exported as WordWrap.
    - The LineSpacing property of an exported document will differ from the original report. This is because in Microsoft Word, the line spacing is calculated by the font size value of a report control plus the line spacing value of a report control.
    - For CheckBox control, the CheckAlignment property is exported as MiddleRight for TopRight, MiddleRight, and BottomRight options. Other CheckAlignment options are exported as MiddleLeft.
    - Paddings exceeding 31 inches is exported as border spaces.
    - Right-To-Left text direction does not work in the LibreOffice.
    - On exporting to Word 2013, when the background (shading) and padding are applied on a paragraph, the padding is also applied to the background, so a gap between the border and the background appears on the left side of the paragraph.
    - CharWrap property is not supported.
    - The fields in a TextBox control are evaluated as follows:

- PageNumber and TotalPages expressions are exported as special fields, evaluated by a text editor (Word or other).
- The fields placed in Header or Footer are automatically evaluated.
- The fields placed in the Body should be re-evaluated manually by clicking 'Update field' from context menu.

Annotations

Annotations are floating text bars or images to call attention to specific items or values or to add notes and special instructions directly to the reports. Annotations added via the viewer's toolbar are temporary and are destroyed when the report closes.

These annotations are accessible through the **Annotation** button present on the Viewer toolbar which is hidden by default. You can make the Annotations toolbar visible by setting the **AnnotationDropDownVisible ('AnnotationDropDownVisible Property' in the on-line documentation)** property to True in the viewer's properties window.

Available Annotations

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:

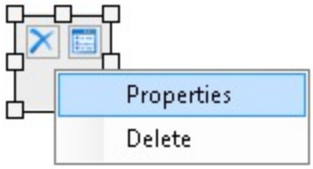
Annotation Name	Description
AnnotationText 	A rectangular box to enter text using the **Text** property.
AnnotationCircle 	A circle without text. You can change the shape to an oval by dragging its corners.
AnnotationRectangle 	A rectangular box without text. You can change the shape to a square by dragging its corners.
AnnotationArrow 	A 2D arrow to enter text using the **Text** property. You can also change the arrow direction using the **ArrowDirection** property.
AnnotationBalloon 	A balloon caption to enter text using its **Text** property. You can also point the balloon's tail in any direction using its **Quadrant** property.
AnnotationLine 	A line to enter text above or below it using its **Text** and **LineLocation** properties. You can also add arrow caps to one or both ends and select different dash styles using **DashCap**, **DashStyle**, and **ShowArrowCaps** properties.
AnnotationImage 	A rectangle with a background image and text. You can select an any image inside it using the **BackgroundImage** property. You can also place text on the image using the **Text** property.

Add annotations in the Viewer

These steps assume that you have already placed the Viewer control onto a Windows Form and loaded a report in it. See Windows Forms Viewer for more information.

1. In your Visual Studio project, on the Form where the Viewer control is placed, select the Viewer control an right-click to choose Properties.
2. In the Properties Panel that appears, set the AnnotationDropDownVisible property to **True** to get an additional toolbar in the viewer control.

3. Run the report application and select the annotation you want use from the Annotation toolbar on the Viewer.
4. Drag the annotation to the desired location on the report design surface. The annotation appears with a **Delete** and a **Properties** button on the top left corner.

5. Inside the annotation, click the Properties button to view its properties in the Properties Panel and use those properties to enter text, change color or transparency, set border or font, alignment etc.
6. Close the Properties Panel to apply changes to the annotation.
7. Drag the corners to resize the annotation as needed. You can also select entire annotation to move it to another location on the report.
8. Right-click the annotation to display the annotation context menu. The context menu includes the **Properties** and **Delete** commands.


Note: You can print a Page, RDLX or Section Report that contains annotations. In a Section Report, you can also save a report with annotations in RDF format. See Add and Save Annotations for further details.

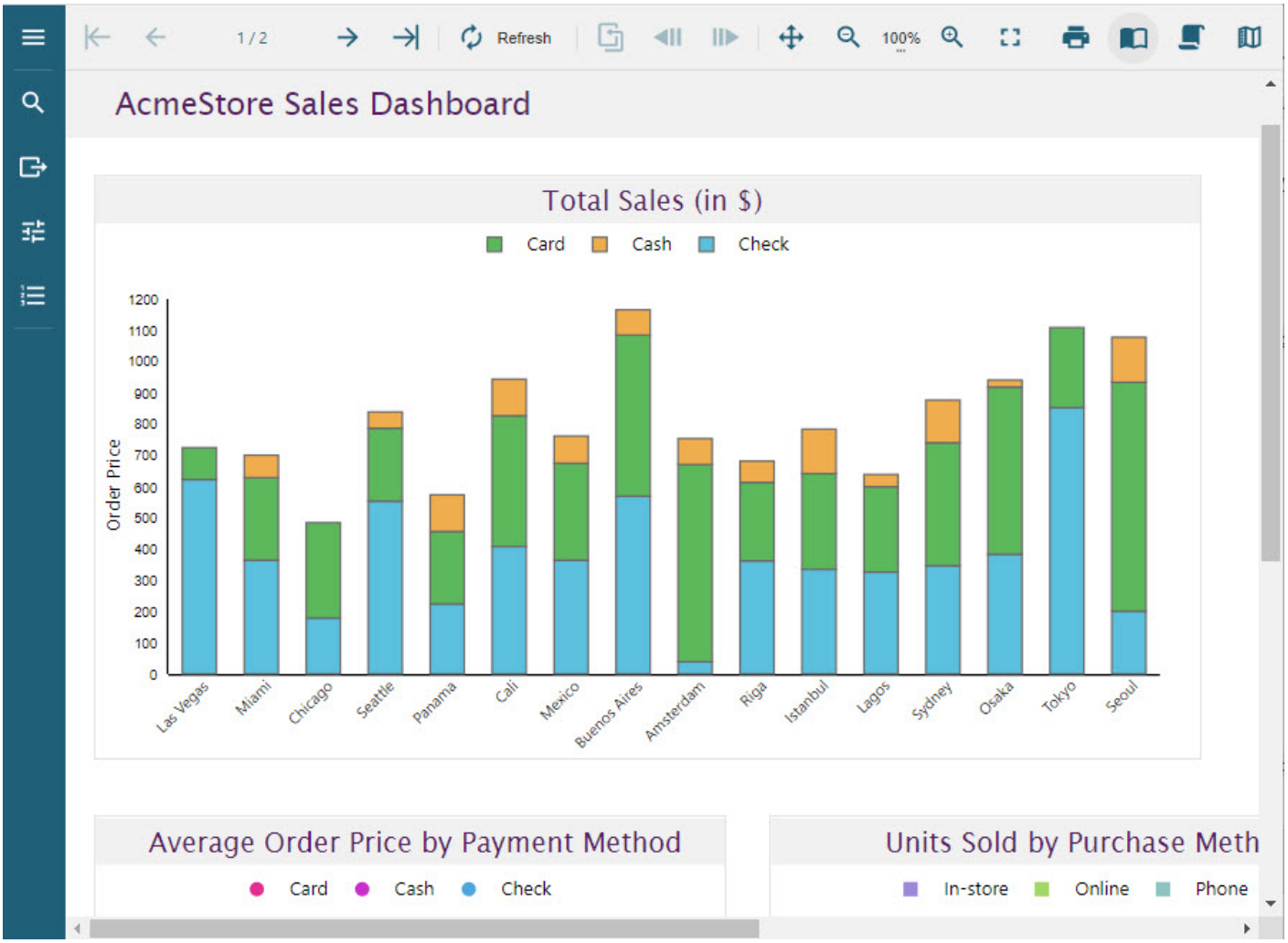
Web Viewers

With Web Viewers, you can view reports in Blazor, ASP.NET Core, ASP.NET MVC, Angular, React, HTML5/JS applications. These viewers are lightweight with built-in support for client-side printing. Like Desktop Viewers, the Web Viewers are feature-rich that allow you to easily view, print, and export reports. The viewers are highly interactive, giving you the ability to enter parameters, perform sort and drill down, navigate to other reports using drill-through links, and so on. Multiple View Modes let you display your reports as Paginated, Continuous, or in Galley view. In addition, the viewers are customizable and localizable according to your personal needs.

The viewer API lets you customize the viewers and blend them into your web page. You can add or hide features or buttons on viewers toolbar and sidebar, you can customize the parameters panel, and so on. For this, you must have hands-on experience in coding, see this set of articles for details.

- JS Viewer Application
- Blazor Viewer Application
- ASP.NET WebViewer Application

As Report Readers, let us focus on the UI of Web Viewers.



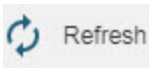
Viewer UI

Viewer Sidebar

Sidebar Element	Name	Description
	Display Sidebar	Displays the sidebar that includes the Search, Export, and Parameters panes.
	Search	Displays the Search pane.
	Export	Displays the Export pane where you can select an export format and options for the report you are previewing.

	Parameters	Displays the Parameters pane. If a report does not have parameters, the Parameters button is not displayed.
	Table of Contents	Displays the Table of Contents pane. If a report does not have the Label property or Document map label set, the Table of Contents or Documents map pane does not appear.

Viewer Top Toolbar

Toolbar Element	Name	Description
	Go to First/Last page	Jumps to the first or last page of a report.
	Go to Previous/Next page	Navigates through a report page by page.
	Current page	Displays the current page number and page total. Enter the page number to view a specific page.
	Refresh	Refreshes the report.
	History: Back to Parent	Returns to the parent report in a drill-down Page Report or RDLX report.
	History: Go Back	Navigates to a previous page in a parent report in a drill-down Page Report or RDLX report.
	History: Go Forward	Navigates to a next page in a parent report in a drill-down Page Report or RDLX report.
	Move Tool	A move tool that you can use to navigate the report.
	Zoom Out	Decreases the magnification of your report.
	Zoom mode	Displays the current zoom percentage which can also be edited. Allows you to select from the available zoom options - **50%**, **100%**, **150%**, **200%,** 300%, **Fit to Page**, and **Fit to Width**.
	Zoom In	Increases the magnification of your report.
	Toggle Fullscreen	Switches to a Fullsreen mode.
	Print	Displays the Print screen to specify printing options.

	Single Page View	Shows one page at a time in the viewer.
	Continuous View	Shows all preview pages one below the other.
	Galley mode	Provides a viewer mode which removes automatic page breaks from an RDLX report and displays data in a single scrollable page. This mode maintains page breaks you create in the report.

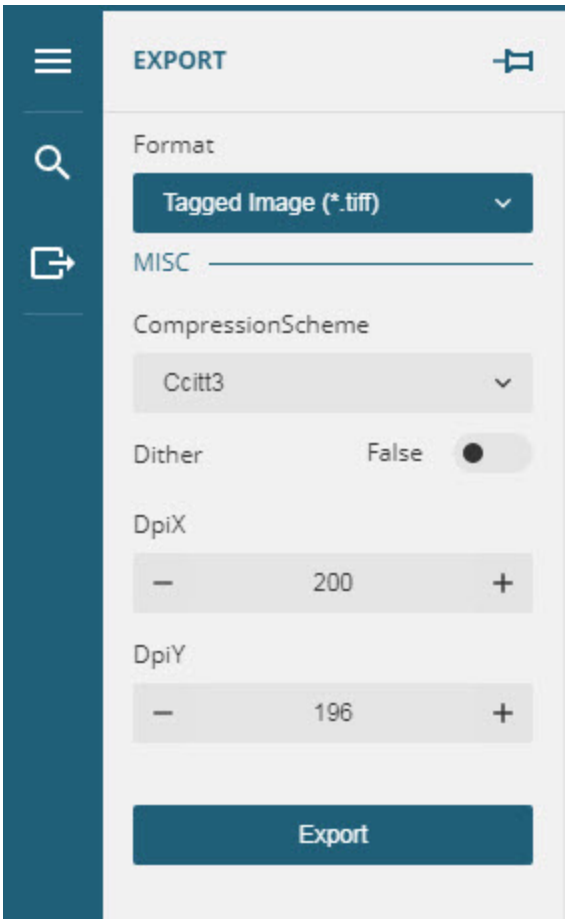
Viewer Search

The **Search** pane lets you enter a word or phrase for which to search within the report. Under **Use these additional criteria**, you may optionally select additional criteria. When you click **Search**, any results appear in the **Find results** list. Click an item in the list to jump to the item you selected and highlight it.

Viewer Export

To display the Export pane, click **Export** in the sidebar. The **Export** pane lets you enter parameters for exporting a report that you are previewing. The available format options for Page/RDLX reports are Excel 2003, Excel, Word 2003, Word, PDF, CSV, JSON, XML, Tagged Image, Web Archive, and Excel Data (hidden by default).

The available format options for Section Reports are Rich Text Format, Excel 2003, Excel, PDF, Tagged Image, Plain Text, and Web Archive.

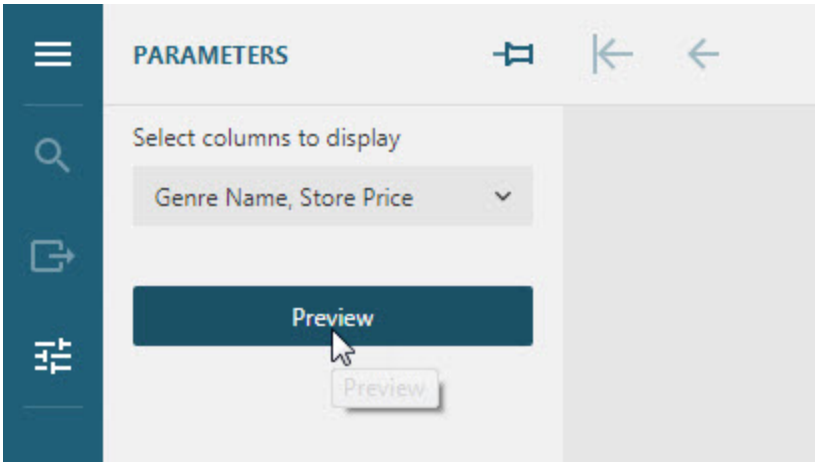


After you set all necessary export properties, click **Export**.

Viewer Parameters Pane

The Parameters pane appears when you click the **Parameters** button in the sidebar. In the **Parameters** pane, enter a value to filter the data to be displayed and click **Preview**.

If a report does not have parameters, the Parameters pane is disabled.

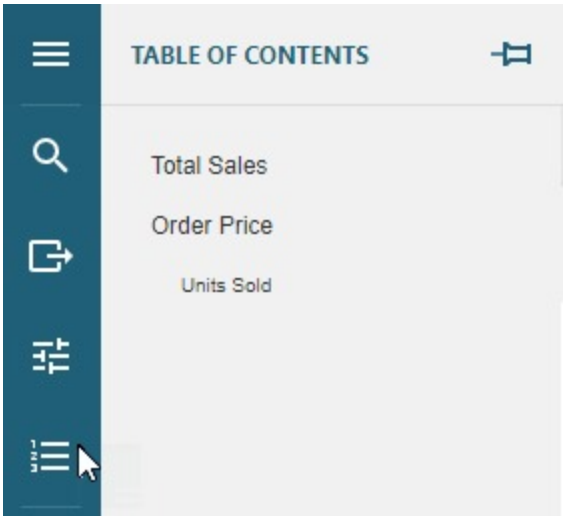


JSViewer supports previewing a custom parameter panel, designed in ActiveReports WebDesigner.

Viewer Table of Contents Pane

The Table of Contents pane appears when you click the **Table of Contents** button in the sidebar. Click any TOC item to navigate to the corresponding section of the report in the Viewer.

Note that the Table of Contents is only available for reports with Bookmarks. The Table of Contents displays each value for the text box, group, or subreport that you bookmark, and you can click them to navigate to the corresponding section of the report in the Viewer.



Export

Use the **Export** button to export any report type to a desired format, except RDLX Dashboard report:

- Excel 2003 (.xls)
- Excel (.xlsx)
- Word (.docx)
- PDF (.pdf)
- CSV (.csv)
- JSON (.json)
- XML (.xml)
- Tagged Image (.tiff)
- Web Archive (.mht)

The following exports are supported in RDLX Dashboard reports:

- CSV (.csv)
- JSON (.json)
- XML (.xml)
- Excel Data (.xlsx)
- Text Print (.txt)

The Export dialog enables you to export the report preview results to a file with several options to fill. Please see Predefined Export Settings for more details on the export settings and how a developer can predefine export settings via code.

Report Authors: Designer Components

A report author prepares a complete report—designs the layout of the report, defines data sources and datasets, uses the data regions and report items/controls to represent data.

ActiveReports provides report authors with a compiled tool—a standalone designer application for desktop applications. The predefined templates and simple drag-and-drop operations in the designer make report designing easy and quick, without using any code.

Another designer that allows end-users to create ad hoc reports in all major browsers is WebDesigner, available for the ASP.NET Core MVC application. The rich API enables UI customization based on the functionality you want the users to experience.

Developers may be report authors, who can in addition use the Visual Studio Integrated Designer to create report layouts in Visual Studio and edit them at design time, visually and through code, script, or expressions, and prepare applications based on the Win or Web samples; for example, see the FlatEndUserDesigner sample.

The Report Authors section assumes that you are using the standalone designer application as the primary way to create your report.

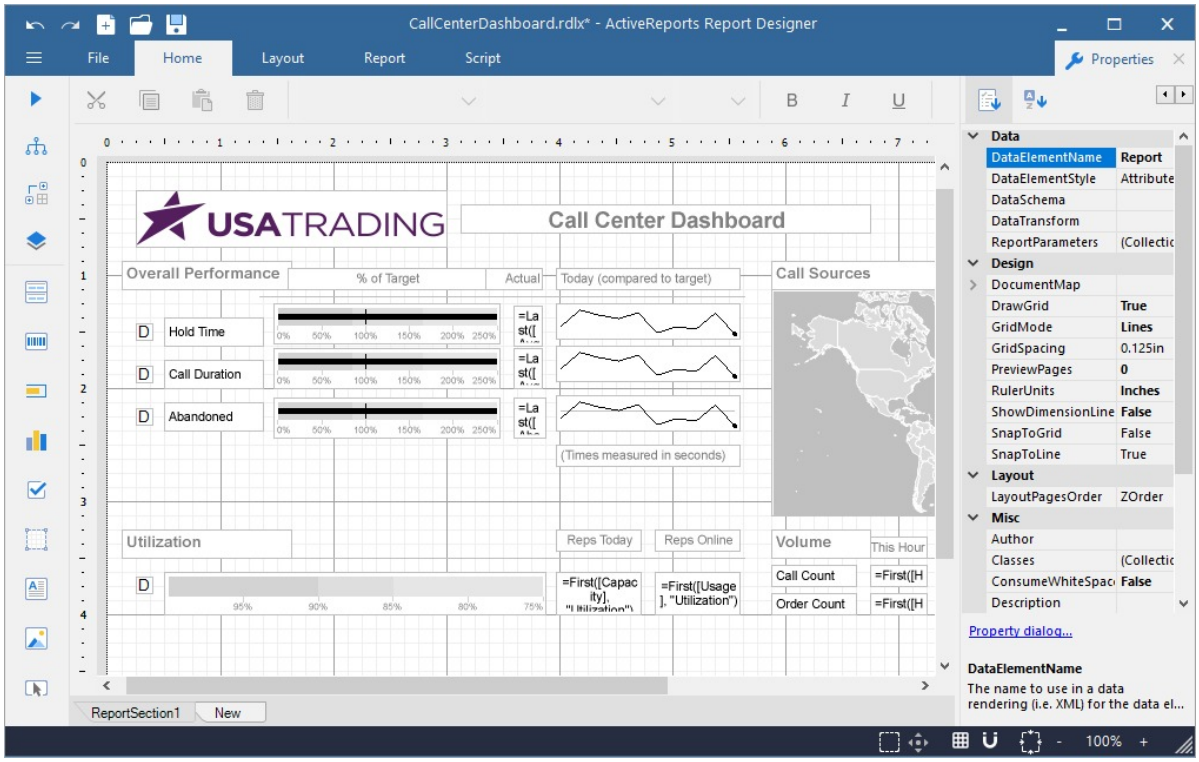
Standalone Designer

ActiveReports provides the standalone designer application **ActiveReports.Designer.exe**, which supports all types of reports - Page, RDLX, RDLX Dashboard, and Section. The application can be launched by installing the ActiveReports installer package. The standalone designer application appears with an RDLX layout by default.

You can run the standalone report designer application by selecting the **ActiveReports 18 Designer** from the Start menu, or by running the **ActiveReports.Designer.exe** from the *C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools* location.

The standalone designer application can be used to create a report layout, save it in .rpx or .rdlx format and then load it in the WinForms Viewer application to view the report.

Design area



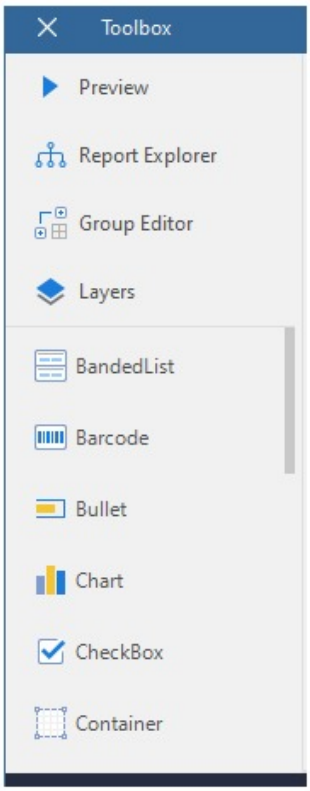
The UI of ActiveReports designer application consists of the following regions:

Quick Access Toolbar

The Quick Access Toolbar contains a set of quickly accessible commands such as Undo, Redo, New, Open, and Save. It is located on the top-left corner of the app.

Toolbox

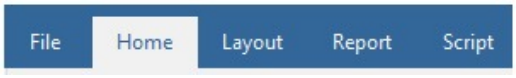
The Toolbox contains a set of elements including the Report controls.



The key elements of the Toolbox are listed below:

- **Preview**: Shows report preview.
- **Report Explorer**: Provides an overview of the hierarchy of added report items and allows managing data sources, parameters, embedded images, embedded stylesheets, etc.
- **Group Editor**: Shows Column and Row group hierarchies of Tablix members for currently selected Tablix data region. You can also switch between Horizontal and Vertical mode.
- **Layers**: Provides options to add or remove layers and send the Layer back or bring it to the front. You can edit or customize any element only in the layer in which it was added.
- **Report Controls**: Report controls to be used while creating a report, such as BandedList, Barcode, Bullet, Chart, CheckBox, Container, FormattedText, Image, InputField, Line, List, Map, Shape, Sparkline, Subreport, Table, TableOfContents, Tablix and TextBox.

Ribbon

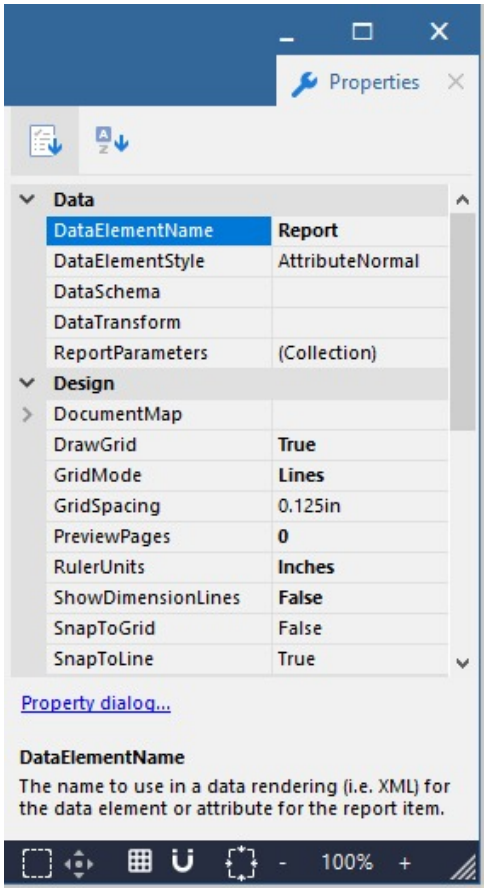


The Ribbon contains the following tabs:

- **File**: Contains options to create, open, save reports or exit the designer. It also contains the version information in the About option.
- **Home**: Consists of report editing options such as cut, copy, paste, and delete. It also provides shortcuts for text formatting such as font, font size, font color, and horizontal and vertical text alignments.
- **Layout**: Contains options to align to grid, size to grid, bring to front, send to back and other sizing and spacing options.
- **Report**: Contains options to define report parameters, embedded images, report properties, and add or remove header and footer (RDLX report), and change report stylesheets. It also contains option to create and manage report parts in case of RDLX report.
- **Script**: Allows you to embed Visual Basic .NET or C# script in reports.

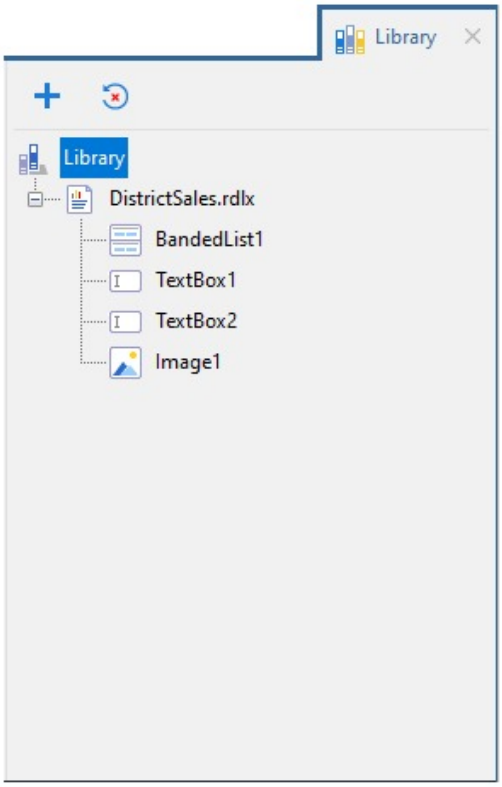
Properties panel

The Properties tab is located on the right-hand side of the app. Click the tab to view the Properties panel, which displays the properties of the selected report element. If more than one element is selected, only their common properties are shown.



Library panel

The Library panel, located on the right-hand side of the app, is disabled by default. See Enable Report Library for details. Click the tab to view the Library panel, which allows adding reports to the designer along with its data source, dataset, parameter, etc. The elements of the reports added in the library can be used in creating a report.



Status bar

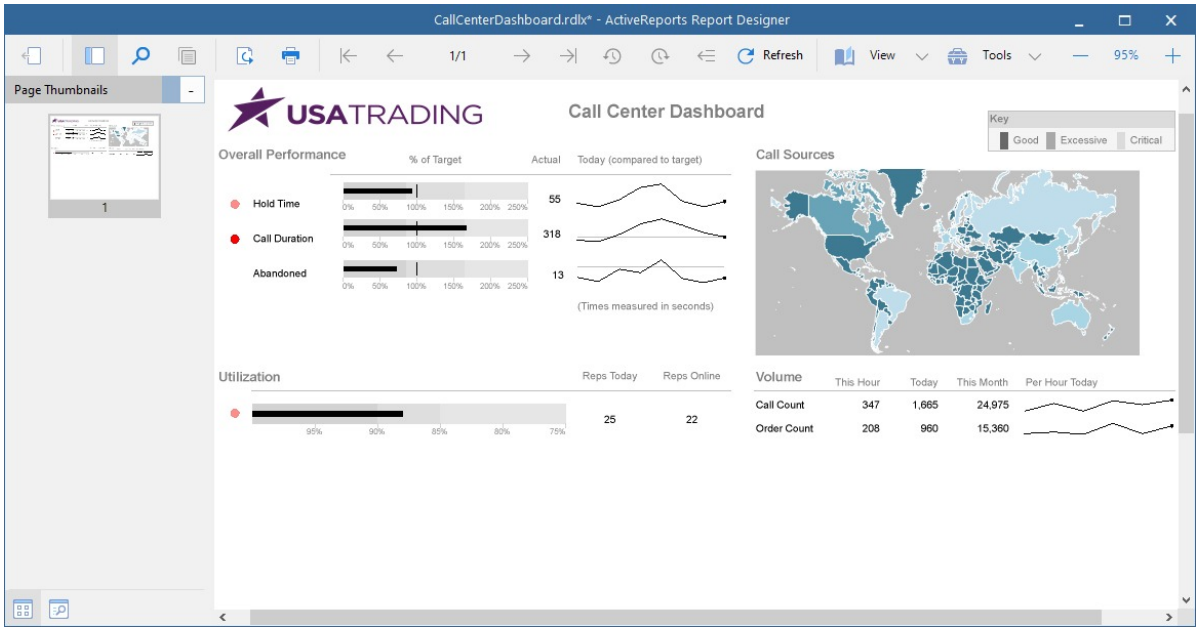
The Status bar appears like a horizontal bar at the bottom of the designer app.

The key elements of the status bar are listed below:

- **Grid Mode**: Shows or hides the grid. Grids help in accurate placements of controls.
- **Zoom Support**: Changes the zoom level of the design area by using zoom in (+) and zoom out (-) buttons, or by using shortcuts [Ctrl] + [+] to zoom in and [Ctrl] + [-] to zoom out.
- **Grid Settings**: Includes the following settings:
  - **Grid Size** - Changes the size of the grid. The value should be between 0.025in and 2in.
  - **Snap to Grid**- Allows the selected control to snap to the grid at set locations.
  - **Snap to Lines** - Allows the selected control to snap to the vertical or horizontal lines relative to the position of other controls.
  - **Dimension Lines** - Displays the dimensions of the element when it is being resized.
- **Actual Size**: Restores the actual size of the report.
- **Pan Mode**: Easy report navigation by dragging it up or down.
- **Select Mode**: Selects all the elements in the selected area.

Preview

When the user clicks the **Preview** button, a Preview window opens up as shown below:



The Preview toolbar consists of the following elements:

- **Back**: Back to the designer.
- **Side Panel**: Includes the following elements:
  - **Page Thumbnails** - Displays thumbnails of all the report pages in the side panel.
  - **Search Results** - Searches any word or phrase.
- **Find**: Opens **Find** window to find any word or phrase.
- **Copy**: Copies the selected text on clipboard.
- **Export**: Exports report to various formats like csv, json, jpeg, etc.
- **Print**: Prints the report.
- **First Page**: Navigates to the first page of the report.
- **Previous Page**: Navigates to previous page. Page Number: Navigates to the specific page of the report.
- **Next Page**: Navigates to the next page.
- **Last Page**: Navigates to the last page of the report.
- **Backward**: Navigates to the page you accessed before the current page.
- **Forward**: Navigates to the page from where you accessed the current page.
- **Refresh**: Refreshes the report.
- **View**: You can select the following view modes from the dropdown list:
  - **Single Page** - Shows one page of a report at a time.
  - **Continuous** - Shows all pages of the report one below the other.
  - **Galley** - Shows RDLX reports by removing automatic page breaks and displaying data in a single scrollable page.
  - **Multipage** - Shows multiple pages at one glance in a tabular format.
- **Tools**: Includes the following elements:
  - **Pan** - Easy report navigation by dragging it up or down.
  - **Selection** - Select report element(s).
  - **Snapshot** - Captures a snapshot and saves it on clipboard.
- **Zoom Support**: Changes the zoom level of the design area by using zoom in (+) and zoom out (-) buttons, or by using shortcuts [Ctrl] + [+] to zoom in and [Ctrl] + [-] to zoom out.

Keyboard Shortcuts

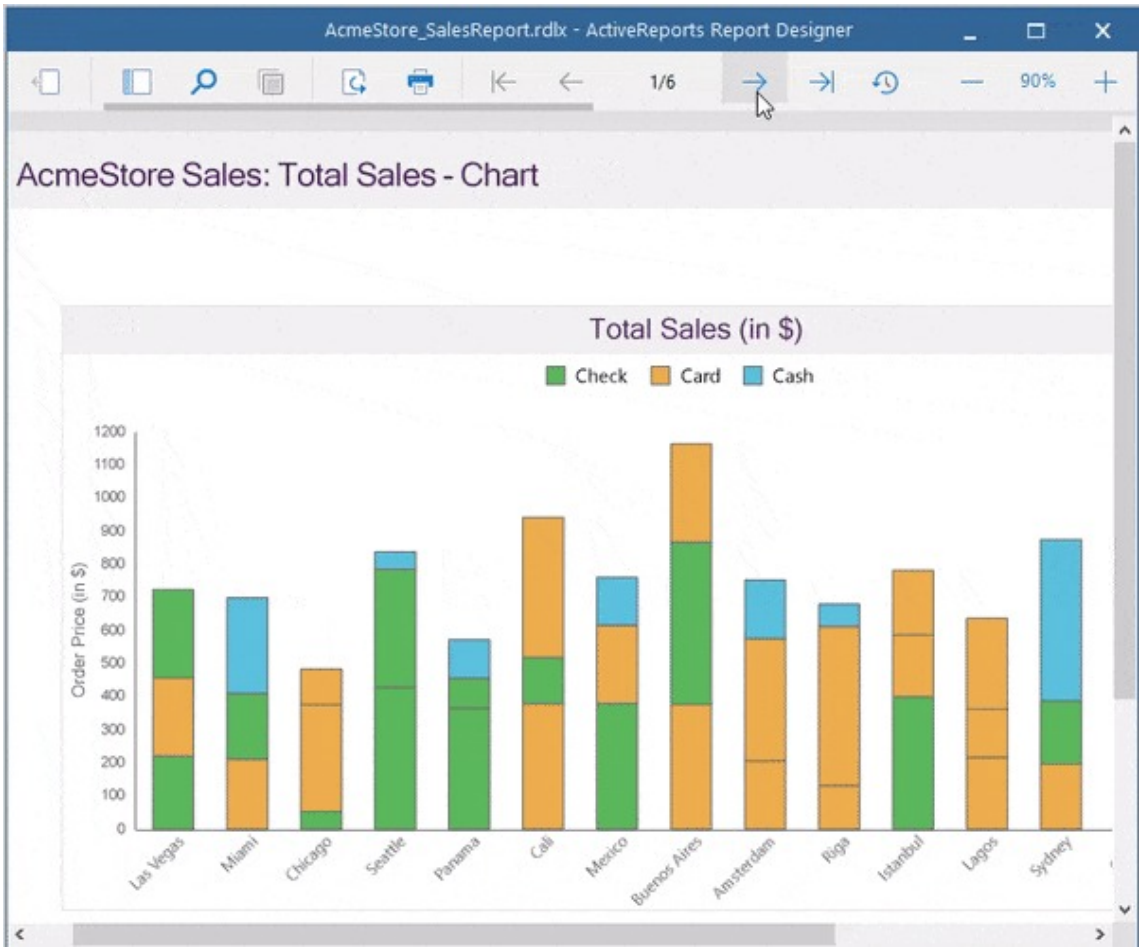
The following shortcuts are available in the Standalone Designer.

	Keyboard Shortcut	Action
Designing	Ctrl + A	Selects all cells in the Table and Tablix controls. In the List, Body and Container controls, selects all controls in the current container.
	Ctrl + O	Opens the **Open** report dialog.

	Ctrl + S	Opens the **Save** report dialog.
	Ctrl + Z	Undoes the last action.
	Ctrl + Y	Redoes the last action.
	Ctrl + X	Cuts text and controls.
	Ctrl + C	Copies text and controls.
	Ctrl + V	Pastes text and controls.
	Del	Deletes text and controls.
	Left, Right, Up, Down arrow keys	Moves the visible area of the page in the corresponding direction. In the Table, navigates between the cells. When controls inside List and Container controls and in the Body of the report are selected, arrow keys allow moving controls by grid-size. In the Chart Control, arrow keys move data-fields and category-fields.
	Tab	Navigates in the forward direction between the cells in the Table and Tablix controls. When controls inside List and Container controls and in the Body of the report are selected, Tab key switches between controls in the forward direction.
	Shift + Tab	Navigates in the backward direction between the cells in the Table and Tablix controls. When controls inside List and Container controls and in the Body of the report are selected, Shift + Tab switches between controls in the backward direction.
	Shift + rotate mouse wheel	Scrolls horizontally.
	Rotate mouse wheel	Scrolls vertically.
Formatting	Ctrl + B	Makes the text bold.
	Ctrl + I	Makes the text italic.
	Ctrl + U	Underlines the text.
	Ctrl + L	Aligns text to the left.
	Ctrl + E	Aligns text to the center.
	Ctrl + R	Aligns text to the right.
	Ctrl + J	Aligns text justified.
	Ctrl + T	Aligns text to the top.
	Ctrl + M	Aligns text to the middle.
	Ctrl + H	Aligns text to the bottom.
Previewing (F5)	Ctrl + F	Finds a text in the report.
	Ctrl + E	Exports the report.
	Ctrl + P	Prints the report.
	Ctrl + S	Switches view mode to Single page.
	Ctrl + M	Switches view mode to Continuous.
	Ctrl + I	Switches view mode to Multiple page.
	Shift + rotate mouse wheel	Scrolls horizontally.
	Rotate mouse wheel	Scrolls vertically.

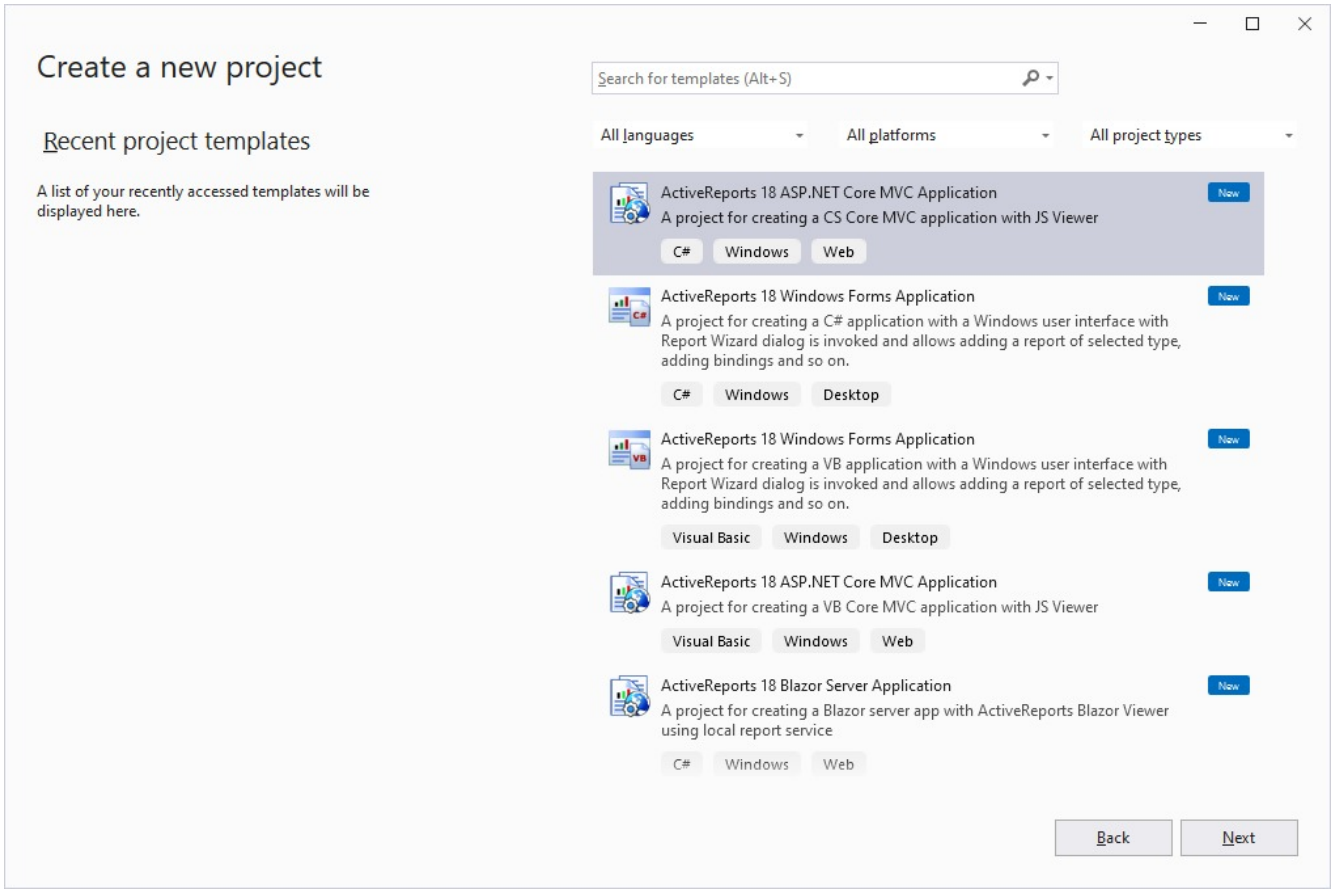
Quick Start

Let us create a report to represent sales data using tables and charts.



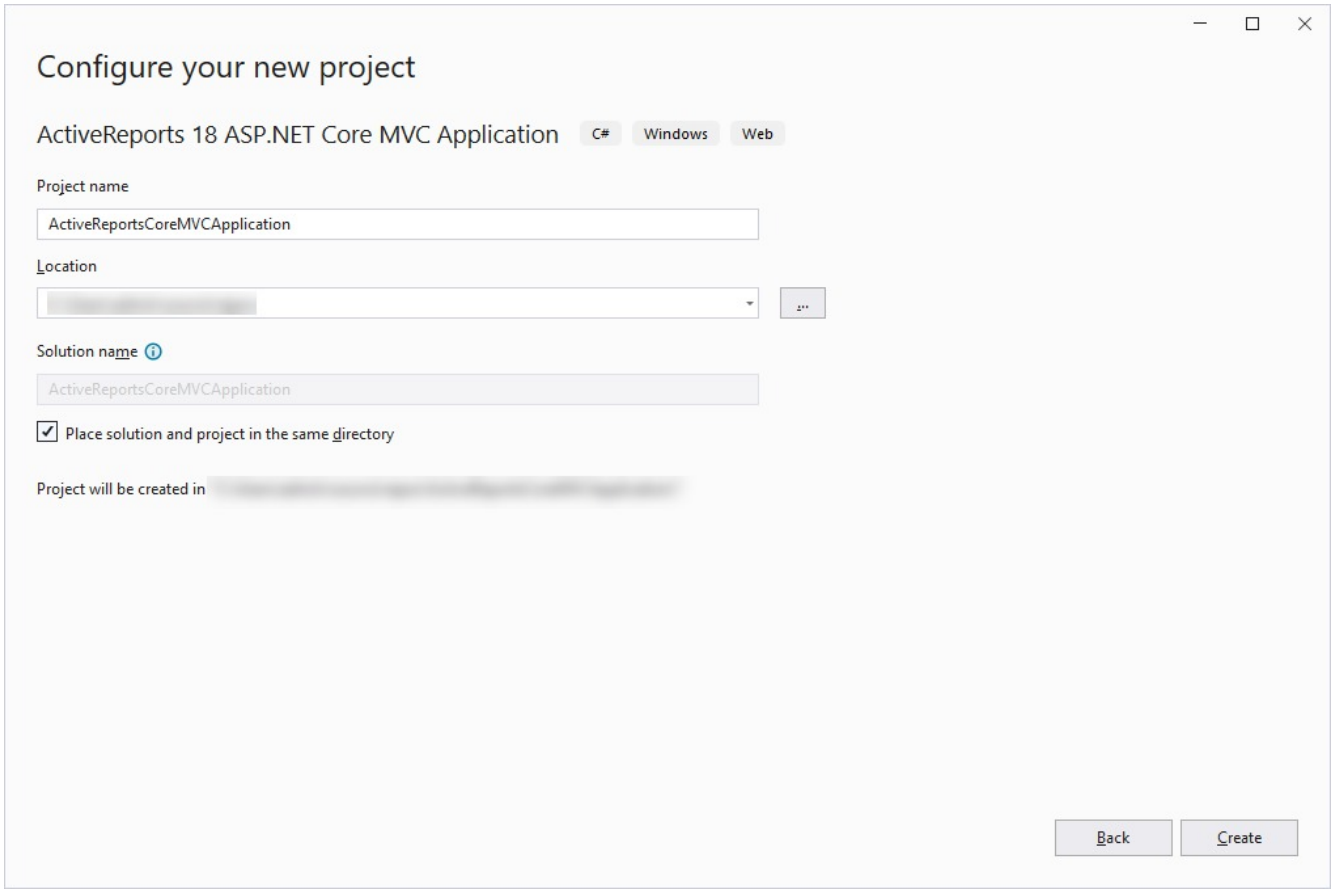
Create Report in Visual Studio Integrated Designer

1. Install ActiveReports.
2. In **Microsoft Visual Studio 2022** (version 17.0 or above), select **ActiveReports 18 ASP.NET Core MVC Application** template and click **Next**.

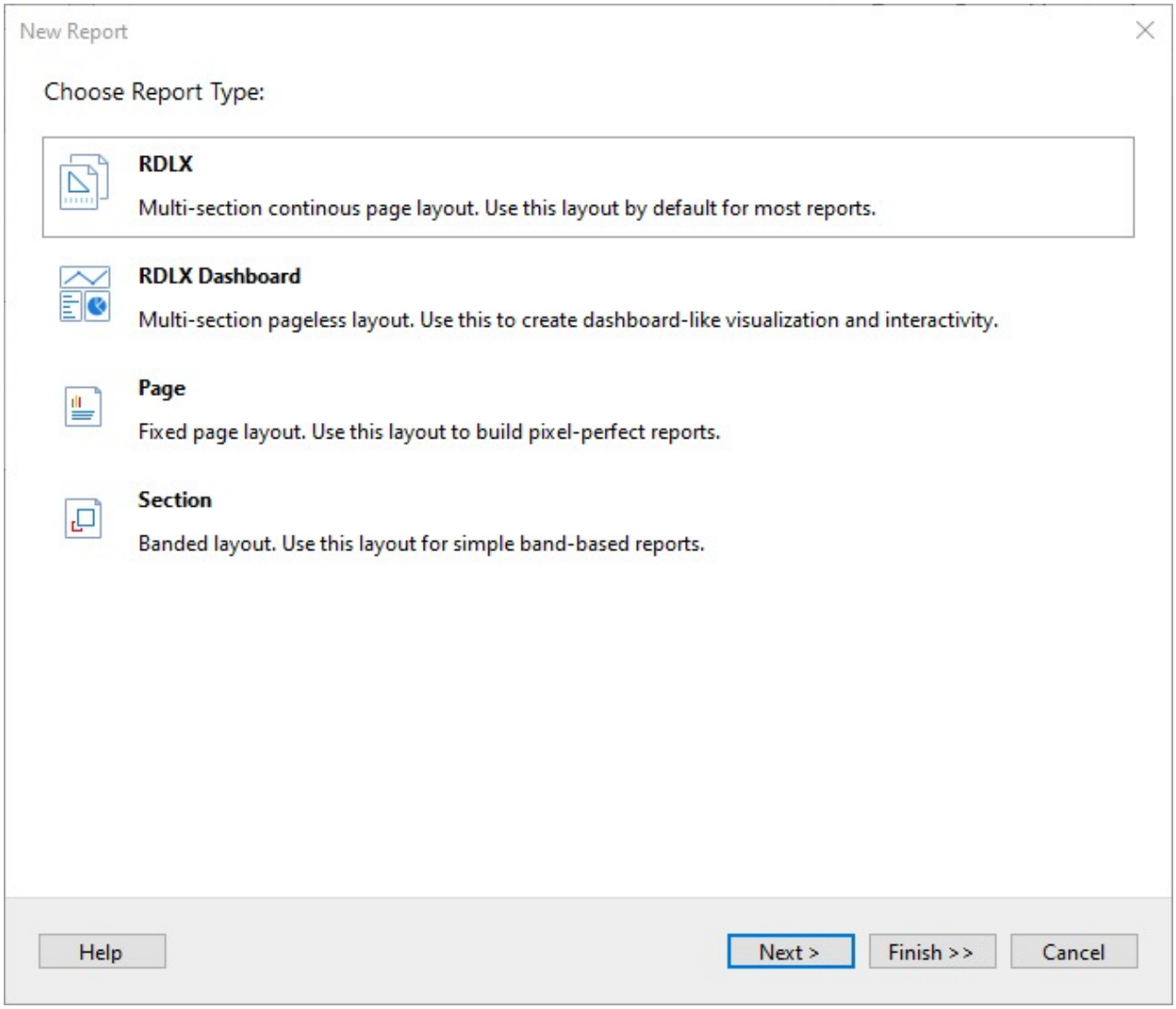


For complete list of built-in templates, see the Project Templates.

3. Type a name for your project and click **Create**.

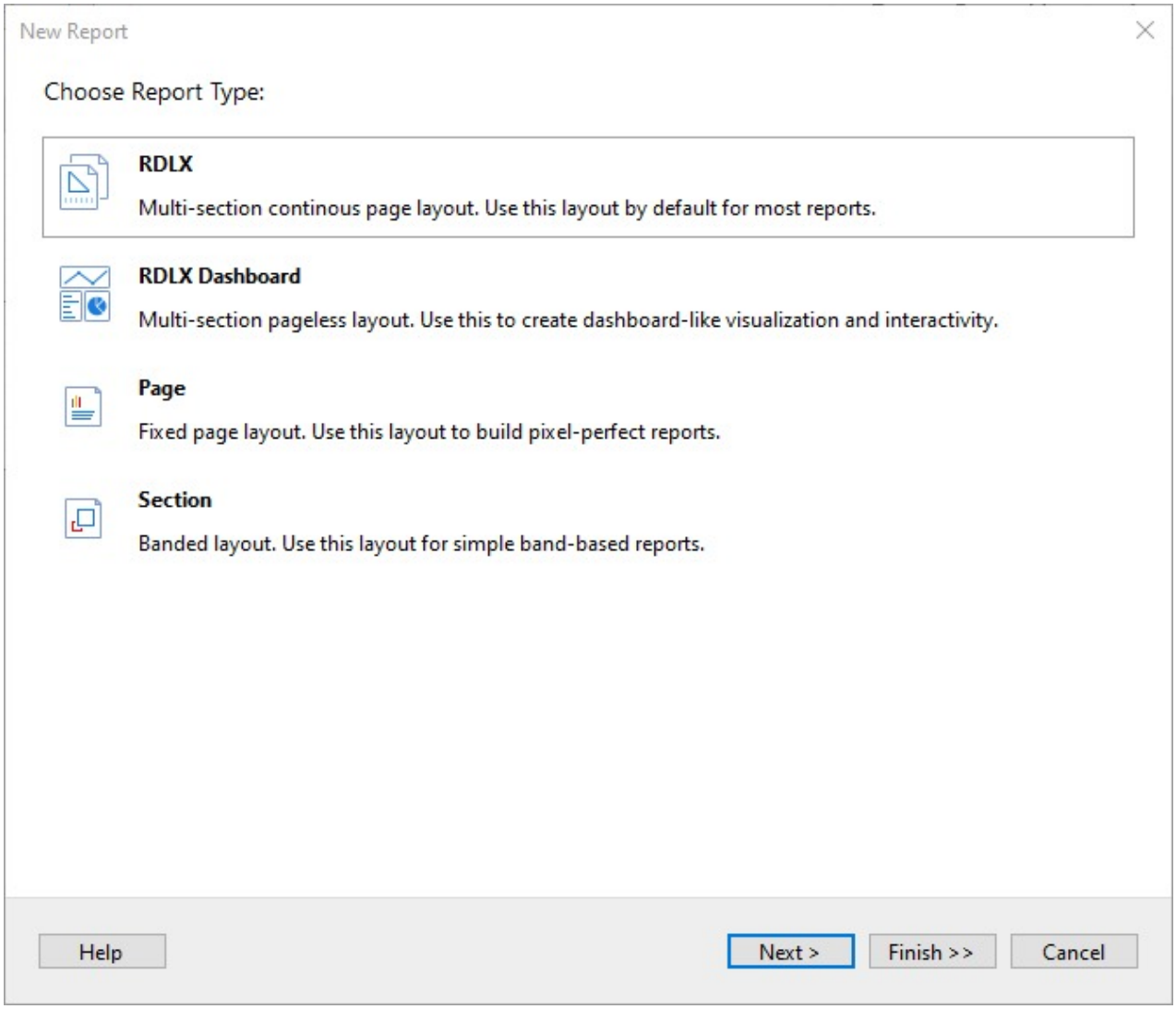


4. In the **New Report** wizard, choose the Report Type as **RDLX** and click **Next**.



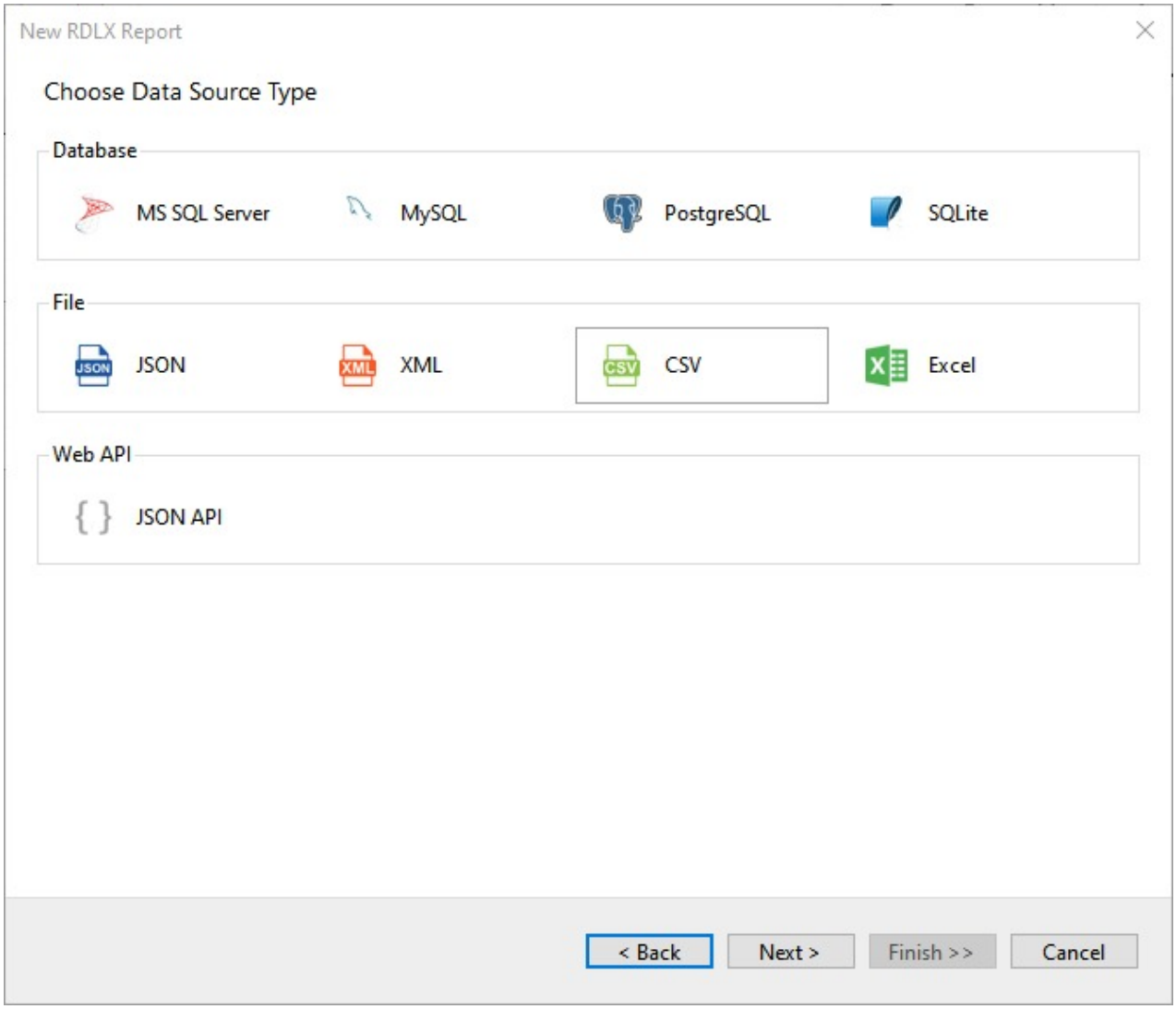
Create Report in Standalone Designer

1. Run the ActiveReports Designer.
2. Go to **File** and click **New** to create a new report.
3. In the **New Report** wizard, choose the Report Type as **RDLX** and click **Next**.

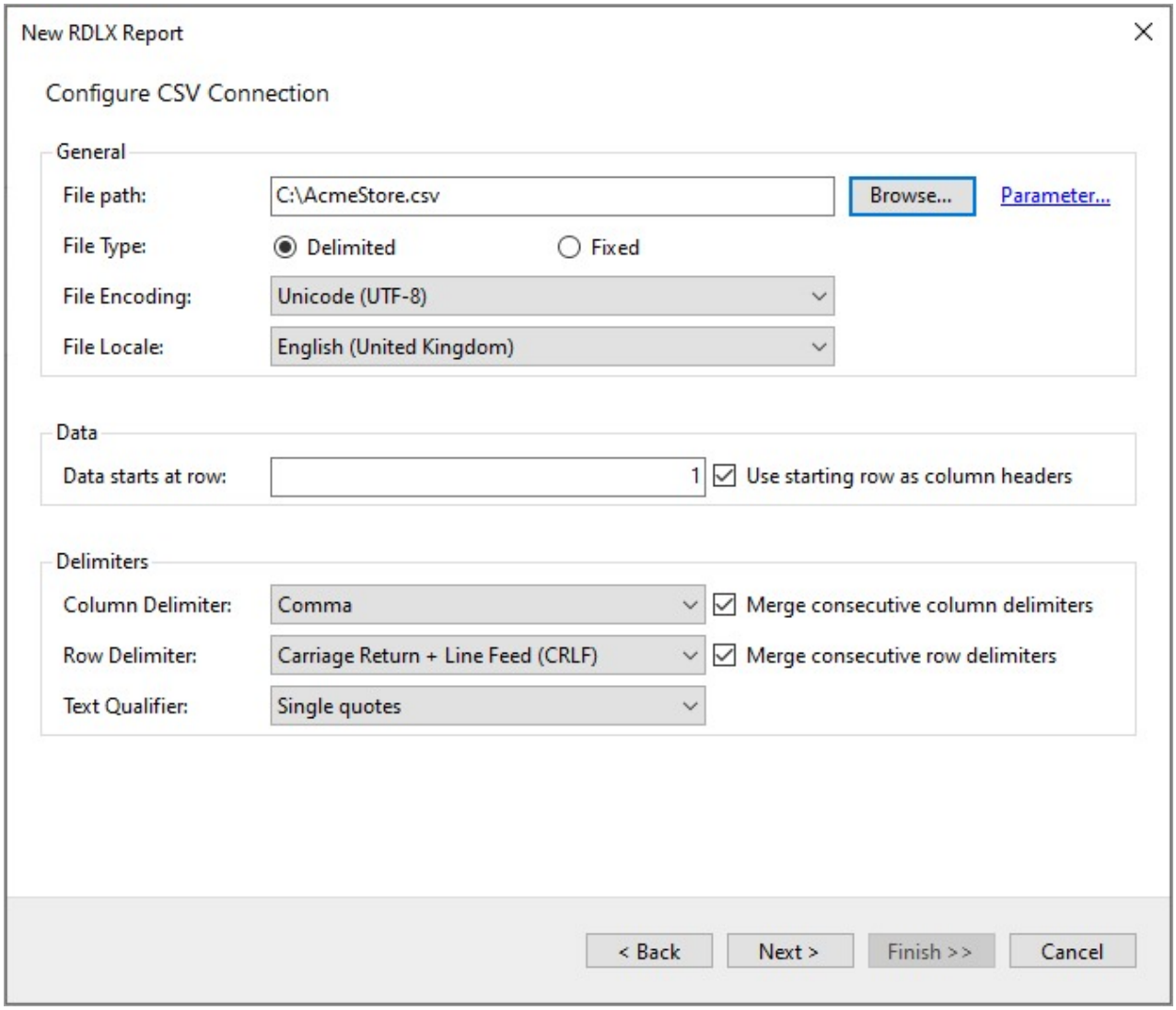


Bind Report to Data

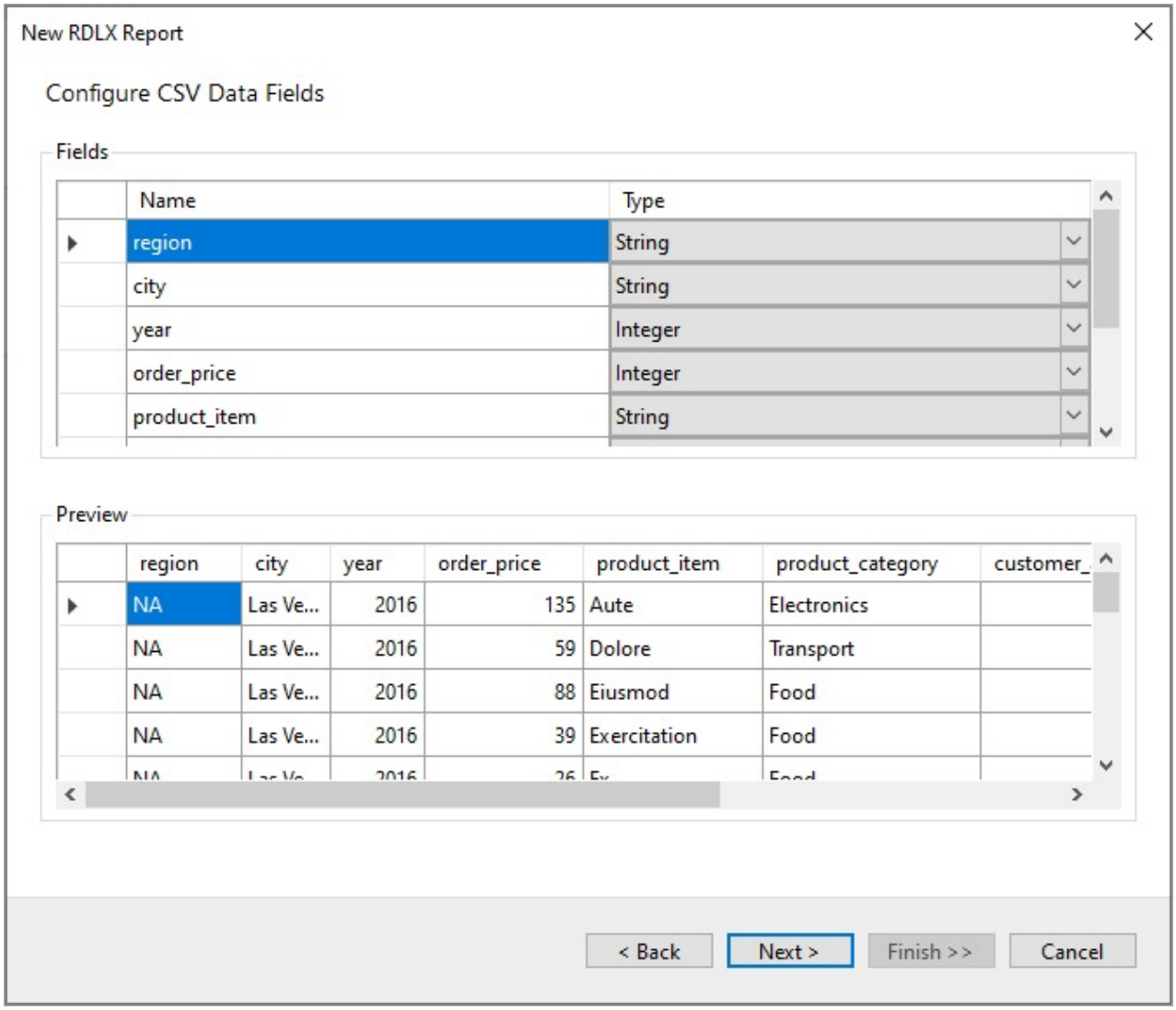
1. On the **Choose Data Source Type** screen of the wizard, select **CSV** and click **Next**.



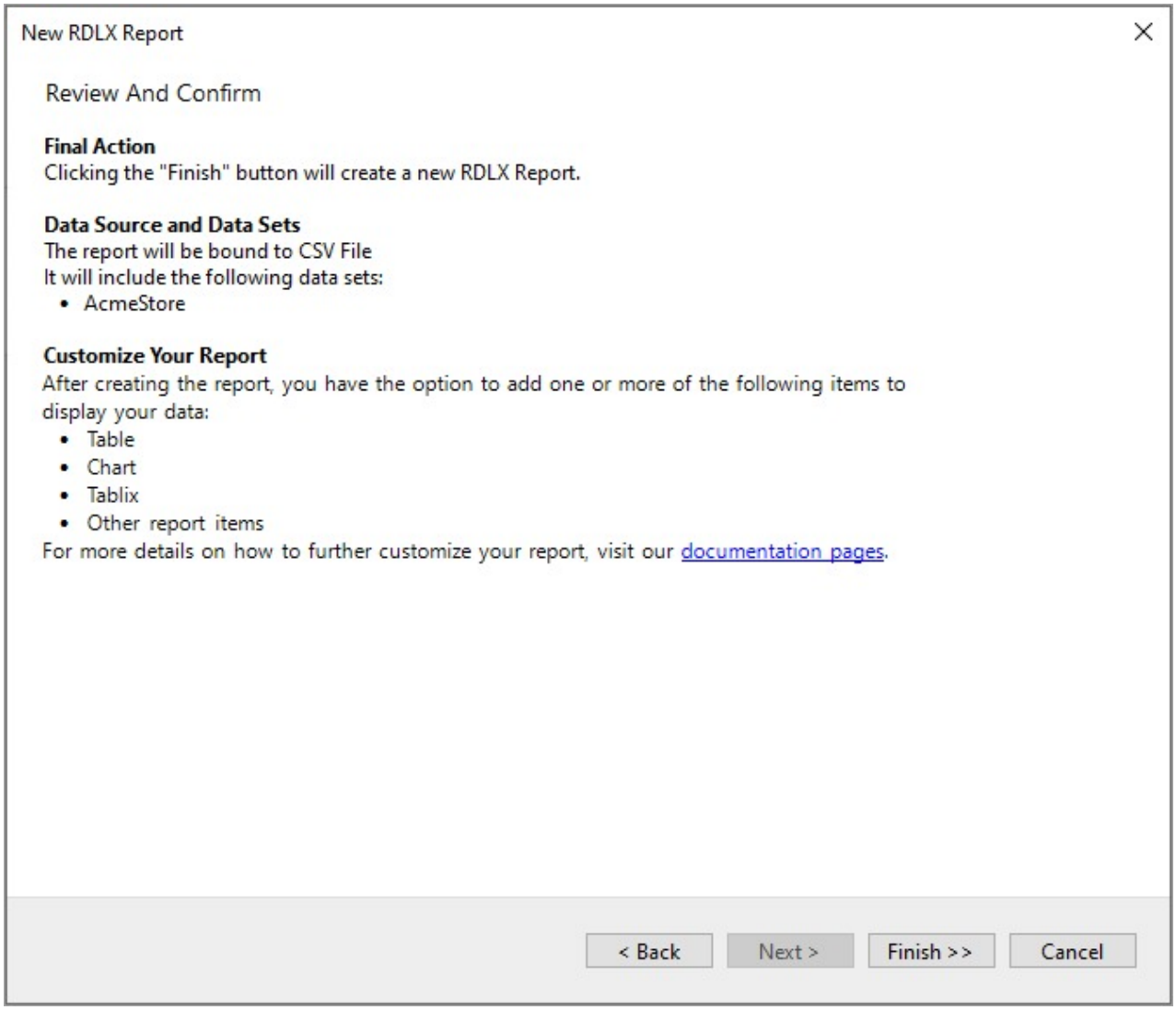
2. On the **Configure CSV Connection** screen, specify the **File Path** by clicking the **Browse** button and navigating to the desired folder on your system. Let us bind data to the **AcmeStore.csv (on-line documentation)** file. See CSV topic for more information.



3. Click **Next** to proceed to the step of configuring CSV data fields.



4. Click **Next** to proceed to the final screen of the Report Wizard.

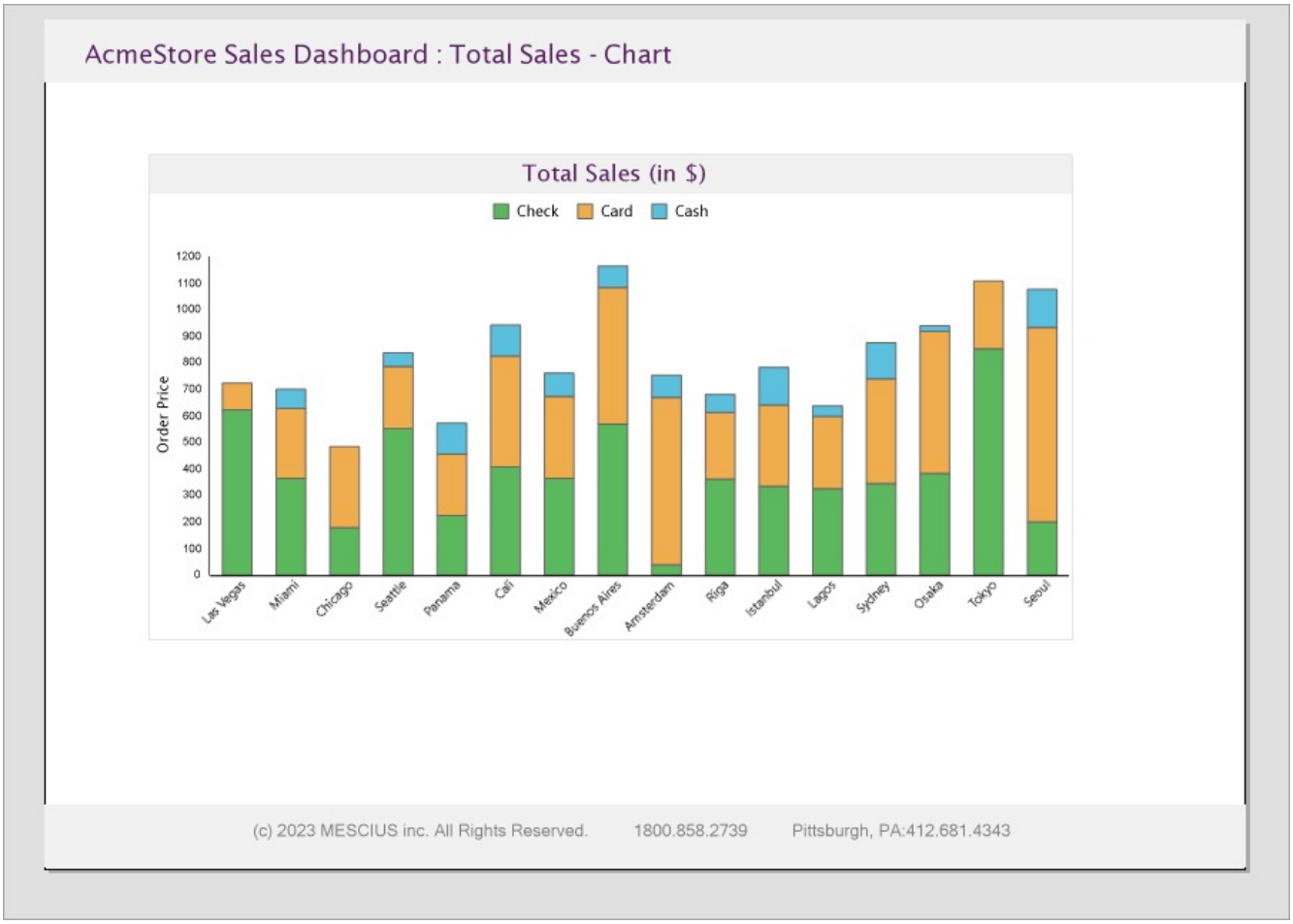


5. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the CSV data source.

Design Report Layout

ReportSection1

1. In ReportSection1, which is open by default in the design area, drag-drop a **Chart** data region from the Toolbox to the design area.
2. Design a chart that shows the total sales in different cities. See Create Stacked Column Chart tutorial to visualize the data in a stacked column chart.
3. Click the **ReportSection1** tab and from the Properties panel, set **PaperOrientation** to 'Landscape'.



ReportSection2

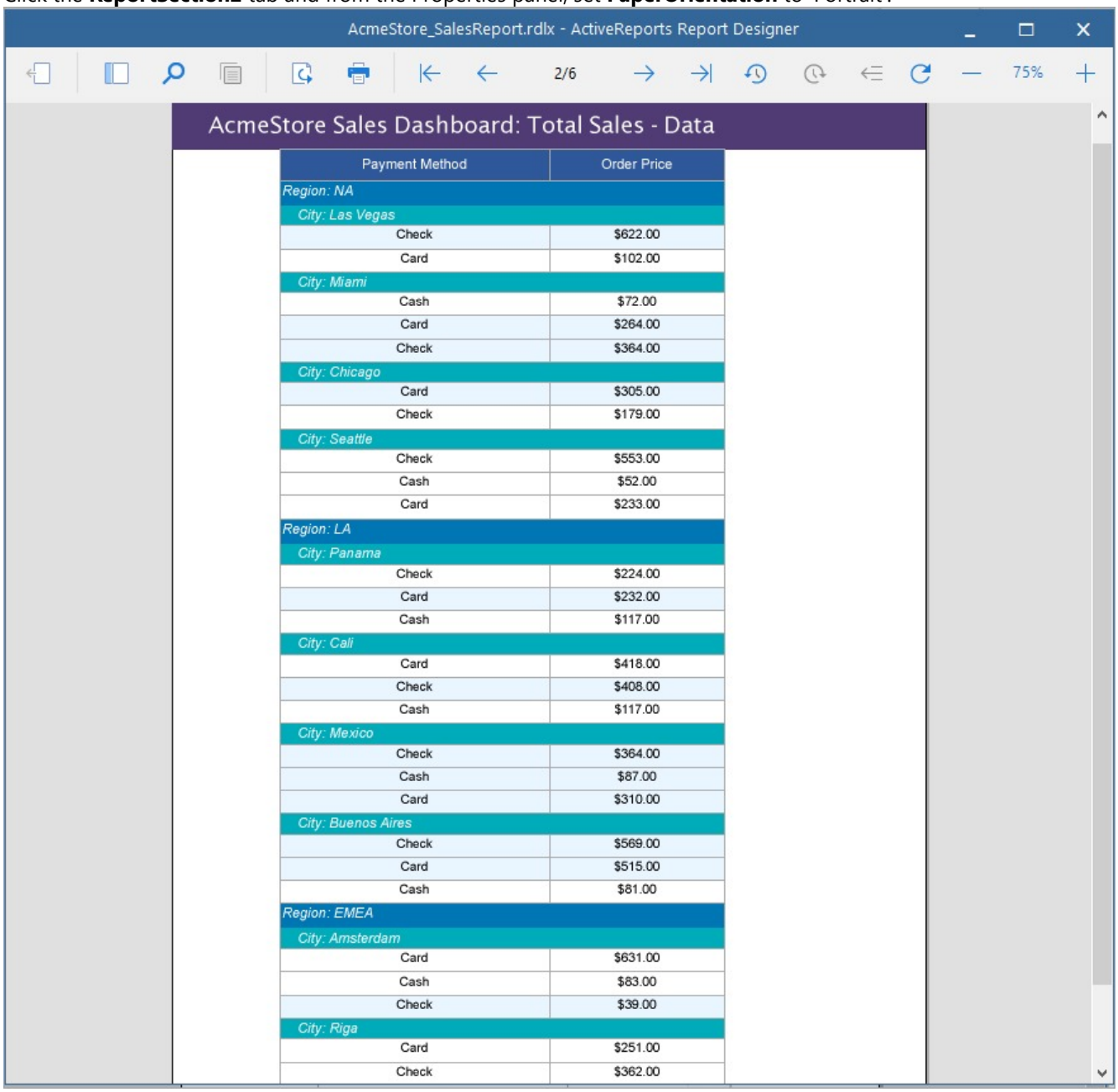
1. From the **New** tab that appears below the design area, add ReportSection2.
2. In this section of the report, drag-drop the **Table** data region.
3. Delete the third column of the table since we want to display payment methods and the order prices as the sales data. This data will be grouped by payment method, city, and region.
4. Select the **Details** row, right-click, and select **Insert Group** from the context menu. We will be adding three groups - payment method, city, and region.
5. In the **Table - Groups** dialog, add the groups and set the **Name** and **Group on** > **Expression** in the following sequence as follows:

S.no.	Name	Expression
1.	GroupByRegion	`=[region]`
2.	GroupByCity	`=[city]`
3.	GroupByPaymentMethod	`=[payment_method]`

6. Click **OK** to close the dialog.
7. Delete the **Details** row of the table.
8. Populate the data in the Table data region as follows:
    1. Merge the first table group and in the **Value** property of the merged cell, enter the expression `="Region: "Fields!region.Value`

2. Merge the second table group and in the **Value** property of the merged cell, enter the expression `="City: "&Fields!city.Value`
3. In the third table group, enter the following expression in the first and second cell, respectively:
    1. `=Fields!payment_method.Value`
    2. `=Sum(Fields!order_price.Value)`

9. Click the **ReportSection2** tab and from the Properties panel, set **PaperOrientation** to 'Portrait'.

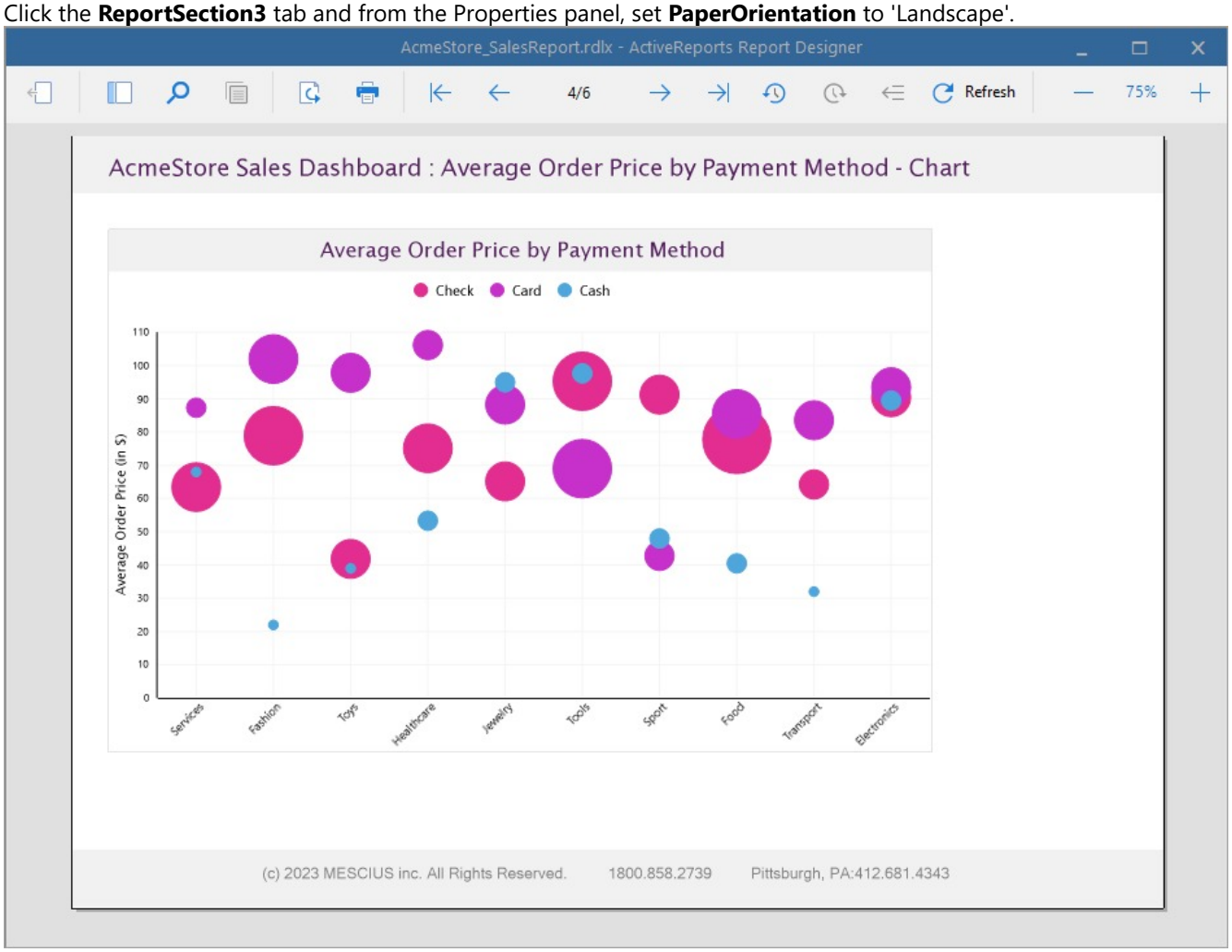


Payment Method	Order Price
Region: NA	
City: Las Vegas	
Check	$622.00
Card	$102.00
City: Miami	
Cash	$72.00
Card	$264.00
Check	$364.00
City: Chicago	
Card	$305.00
Check	$179.00
City: Seattle	
Check	$553.00
Cash	$52.00
Card	$233.00
Region: LA	
City: Panama	
Check	$224.00
Card	$232.00
Cash	$117.00
City: Cali	
Card	$418.00
Check	$408.00
Cash	$117.00
City: Mexico	
Check	$364.00
Cash	$87.00
Card	$310.00
City: Buenos Aires	
Check	$569.00
Card	$515.00
Cash	$81.00
Region: EMEA	
City: Amsterdam	
Card	$631.00
Cash	$83.00
Check	$39.00
City: Riga	
Card	$251.00
Check	$362.00

ReportSection3

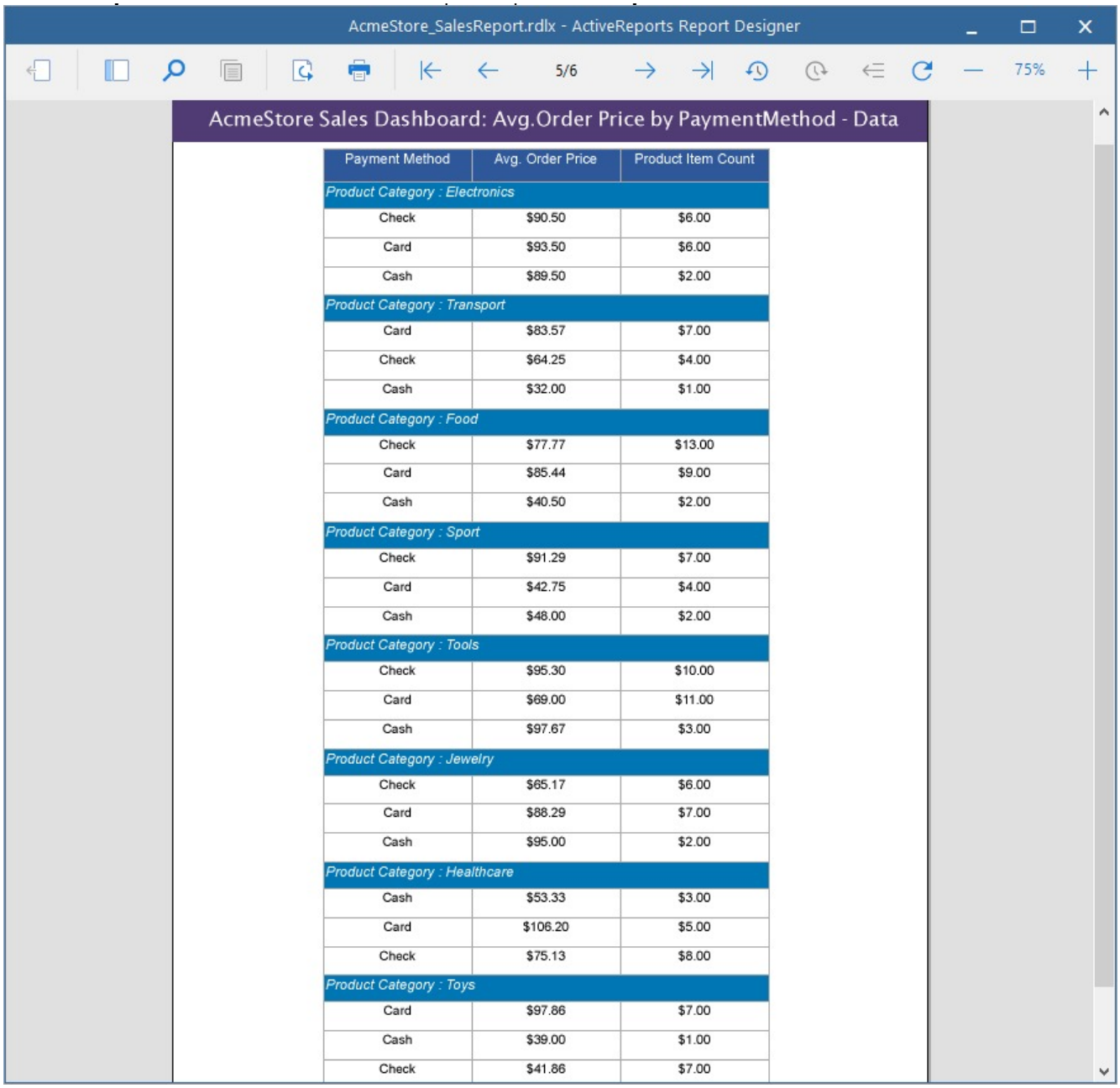
1. From the **New** tab that appears below the design area, add ReportSection3.
2. In this section, drag-drop a **Chart** data region to plot the average order price by payment method using a Bubble chart. See Create Bubble Chart tutorial to visualize the data in a bubble chart.

3. Click the **ReportSection3** tab and from the Properties panel, set **PaperOrientation** to 'Landscape'.



ReportSection4

1. From the **New** tab that appears below the design area, add ReportSection4.
2. In this section, drag-drop a Table data region and drop the fields onto the table cells to display the average order price and payment method and the product item count.
3. Click the **ReportSection4** tab and from the Properties panel, set **PaperOrientation** to 'Portrait'.



Payment Method	Avg. Order Price	Product Item Count
Product Category : Electronics		
Check	$90.50	$6.00
Card	$93.50	$6.00
Cash	$89.50	$2.00
Product Category : Transport		
Card	$83.57	$7.00
Check	$64.25	$4.00
Cash	$32.00	$1.00
Product Category : Food		
Check	$77.77	$13.00
Card	$85.44	$9.00
Cash	$40.50	$2.00
Product Category : Sport		
Check	$91.29	$7.00
Card	$42.75	$4.00
Cash	$48.00	$2.00
Product Category : Tools		
Check	$95.30	$10.00
Card	$69.00	$11.00
Cash	$97.67	$3.00
Product Category : Jewelry		
Check	$65.17	$6.00
Card	$88.29	$7.00
Cash	$95.00	$2.00
Product Category : Healthcare		
Cash	$53.33	$3.00
Card	$106.20	$5.00
Check	$75.13	$8.00
Product Category : Toys		
Card	$97.86	$7.00
Cash	$39.00	$1.00
Check	$41.86	$7.00

Preview Report

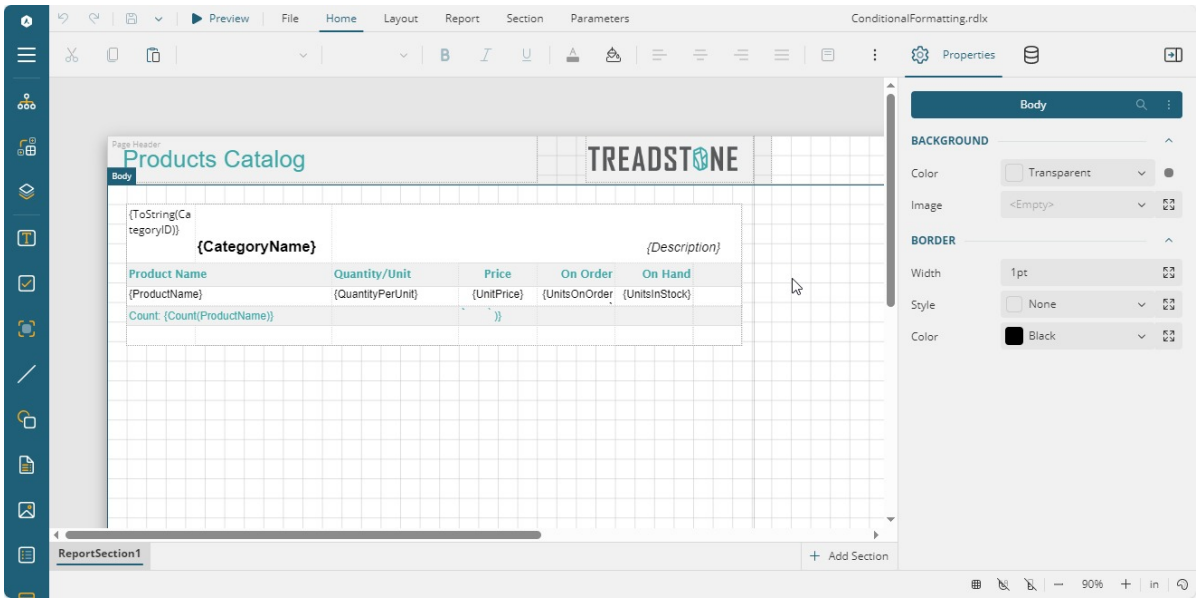
The final report is shown at the beginning of this page.

WebDesigner

ActiveReports WebDesigner is based on the HTML5/JS technology stack. The designer is an integrated reporting application that comes packed with a powerful yet intuitive user interface that facilitates users to create summarized and structured reports anywhere on any device. The designer supports all types of reports that a desktop designer supports - RDLX, RDLX Dashboard, Page, and Section.

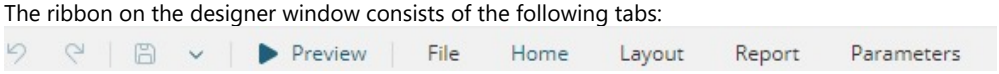
WebDesigner Interface

The UI of WebDesigner Interface consists of the following regions:



Ribbon Tabs

The ribbon on the designer window consists of the following tabs:



- **Undo/Redo**: Undo or redo actions on the designer.
- **Save/Save as:** Lets you save the report.
- **File**: Contains options to create, open, or save all types of reports - Page report, RDLX report (+Master report), and Section report.
- **Preview**: Shows the report preview.
- **Home**: Consists of report editing options such as cut, copy, paste, and delete. It also provides shortcuts for text formatting such as font, font size, Text color, Background color and horizontal and vertical text alignments.
- **Layout**: Contains options to align to grid, size to grid, bring to front, send to back and other alignment, sizing, and spacing options.
- **Report**: Contains options to add continuous section, convert to master report, set master report (in RDLX report), switch themes, add, delete, hide, or move pages (in Page report), and add or remove header and footer (in Section report).
- **Section**: Enables only in the RDLX report. It contains the options to add header or footer and add or move section.
- **Script**: Enables you to use a VB.NET or C# script to port your custom logic to report layouts. This permits layout saved to report XML (.rpx) files to serve as standalone reports.

  > **Note:** Following points must be noted in WebDesigner as compared to ActiveReports Designer control (see link):
  >
  > - Script tab is not available for any report except Section report. Users can still work with scripts, but cannot add or edit them directly.
  > - The built-in code editor is currently a simple text area without any additional functionality like syntax highlighting.

- **Dashboard**: Contains options to add, move, hide, duplicate sections and switch theme. This tab is available for the RDLX Dashboard Report only.
- **Parameters**: Contains the design area for designing a custom parameter panel. The controls making the panel have default properties set, like name, label, default value, etc.

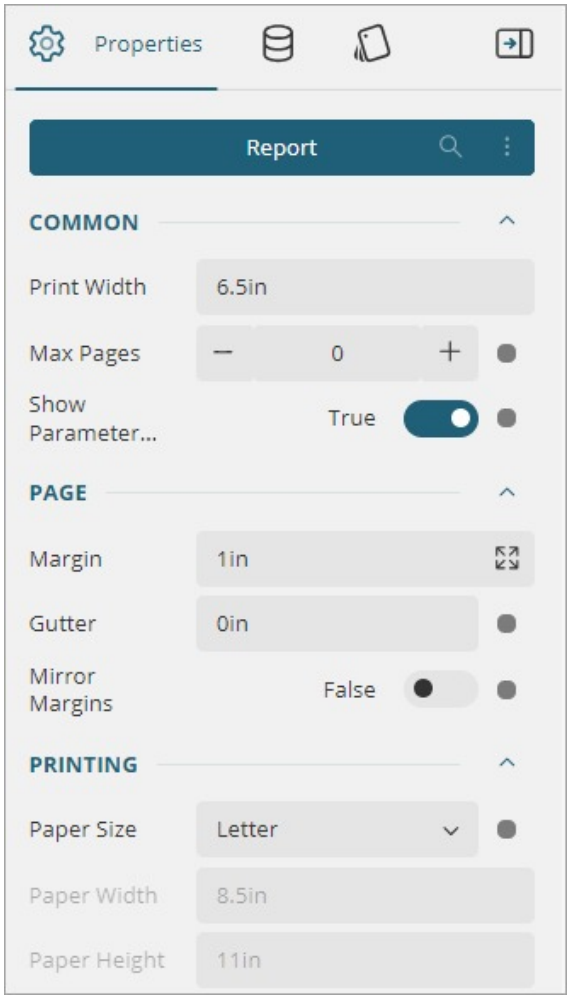
Menu Button

The Menu on the designer window contains the explorer, group editor, layers and report controls that assist in designing reports. You can preview the report by clicking on the **Preview** button.

- **Explorer**: Provides an overview of the hierarchy of added report items. It displays the current selection and allows for the selection of other report items. The section names in Section report can also be viewed in the Explorer. To resize any section, use the Properties panel.
- **Group Editor**: Shows column and row hierarchies of Tablix members for the currently selected Tablix or Table data region.
- **Layers:** Opens the Layers List window that displays a list of layers in the report along with their possibility and lock options. See Layers for details.
- **Report Controls**: Report controls to be used while creating a report. For an RDLX report, the report controls available are: TextBox, CheckBox, Container, Line, Shape, Table Of Contents, Image, List, Table, Tablix, Chart, Bullet, Barcode, Formatted Text, Sparkline, Subreport, Banded list, and Input Field.

Properties tab

The Properties tab is located on the right-hand side of the app. Click the tab to view the Properties panel, which displays the properties of the designer or the selected report element. If more than one element is selected, only their common properties are shown.



Data tab

Data tab is located next to the properties tab. It manages the data sources, data sets, and parameters. It also displays common values such as the current date and time, page number, total pages, and more.

Style Sheet tab

It contains styles, which are a set of properties that you can apply to selected controls in your reports to quickly change their appearance. In EUD you must open report's properties dialog to edit the style sheet. In WebDesigner we have a separate tab for that.

> **Note**: Style Sheet tab is available only for the Section report.

Designer Buttons

- **Designer Buttons**: Designer buttons consist of options to change the appearance and behavior of the design area:
  - **Grid**: Click this option to show or hide the grid. Grids help in accurate placements of controls.
  - **Grid Size, Snap to Grid, and Snap to Guides**: Grid Size allows you to specify the size of the grid. When the **Snap to Grid** option is enabled, the selected control snaps to the grid at set locations. When the **Snap to Guides** option is enabled, the selected control snaps to the vertical or horizontal lines relative to the position of other controls. This is helpful for consistent spacing and alignment of controls.

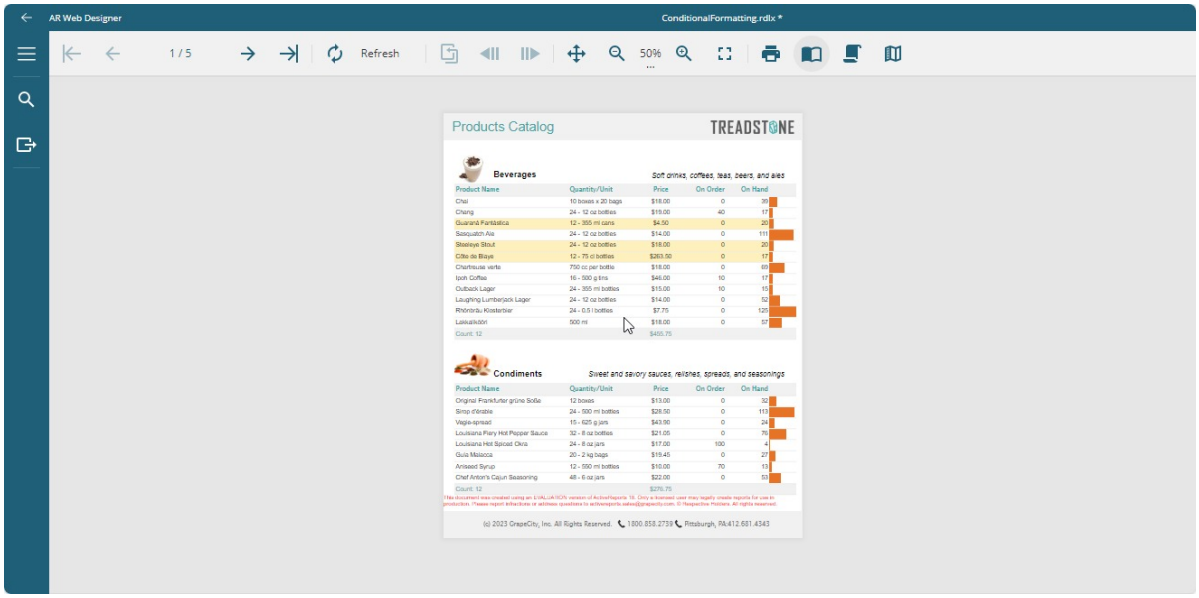
> **Note**: **Snap to Guides** option is not supported in Section report's design area.

- **Ruler** (disabled by default): When the ruler is enabled, you can drag the ruler's markers to change the report margins.

- **Zoom support**: Using the zoom in (+) and zoom out (-) buttons, you can change the zoom level of the design area. You can also use shortcuts **[Ctrl] + [+]** to zoom in and **[Ctrl] + [-]** to zoom out.
- **Ruler Units**: Lets you change the ruler measurements for a report to Centimeters (cm) or Inches (in) using ruler units.
- **Theme Picker**: Lets you change the theme of the designer from the 'System Theme' and the themes defined via API.

Preview

When the user clicks the **Preview** button, a Preview window opens as shown below:



The Preview toolbar consists of the following elements:

- **Go To First**: Navigates to the first page of the report.
- **Go to Previous**: Navigates to the previous page.
- **Go To Next**: Navigates to the next page.
- **Go To Last**: Navigates to the last page of the report.
- **Refresh**: Refreshes the report.
- **History**: **Back To Parent**: Navigates to the parent report (only available for drill through report)
- **History: Go Back**: Navigates to the previous page in the drill through report.
- **History: Go to Forward**: Navigates to the next page in the drill through report.
- **Move Tool**: Allows you to change the visible part of a page if it does not fit on the screen.
- **Zoom support**: Changes the zoom level of the design area by using zoom in (+) and zoom out (-) buttons, or by using shortcuts **[Ctrl] + [+]** to zoom in and **[Ctrl] + [-]** to zoom out.
- **Toggle Fullscreen**: Toggles the full screen.
- **Print**: Prints the report.
- **Single Page View**: Shows one page of the report at a time.
- **Continuous View**: Shows all pages of the report one below the other.
- **Galley Mode**: Shows RDLX reports by removing automatic page break and displaying data in a single scrollable page.

Keyboard Shortcuts

The following shortcuts are available in the WebDesigner.

Keyboard Shortcut	Action
Ctrl + A	Selects all cells in the Table and Tablix controls.

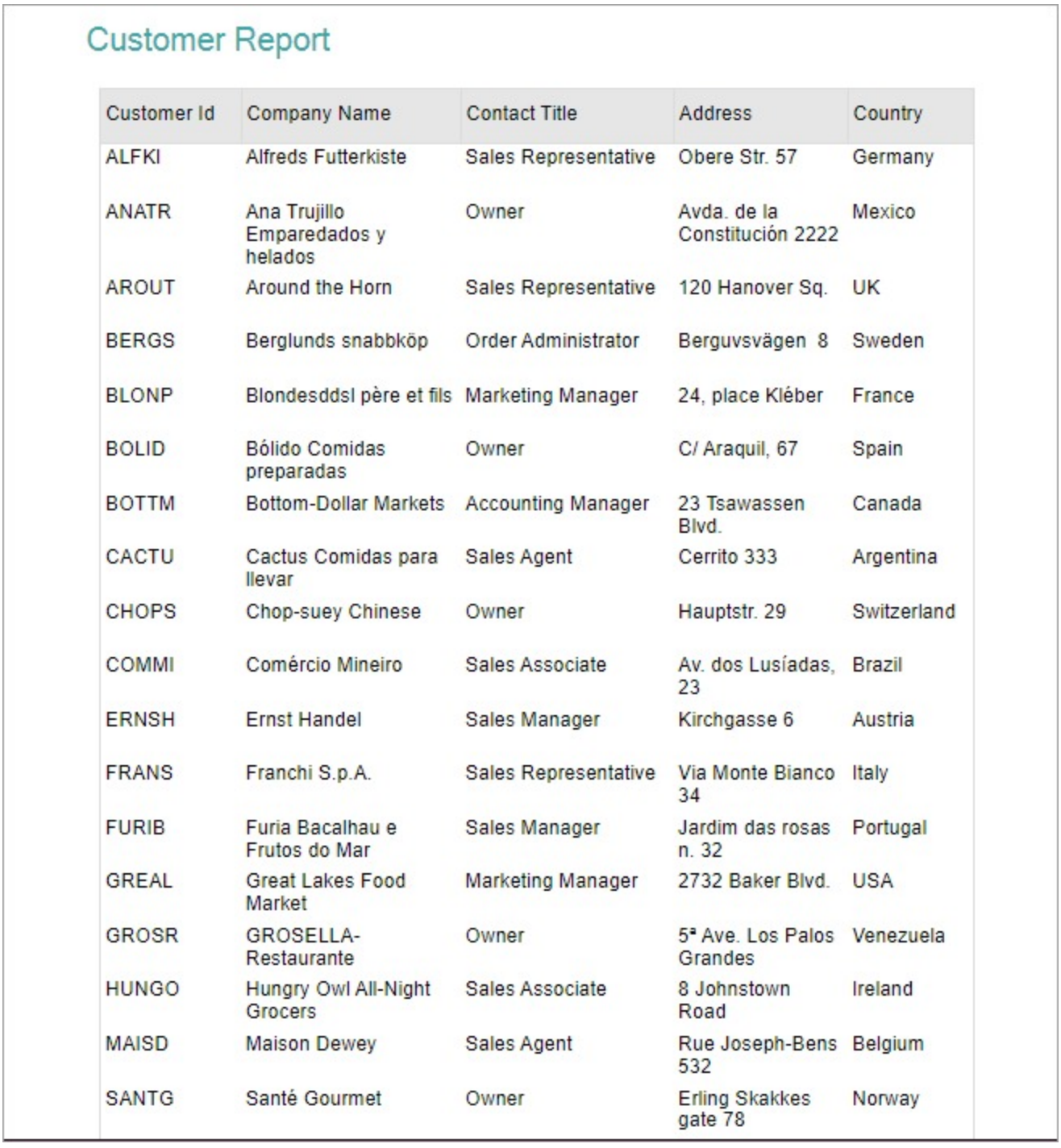
	In the List, Body and Container controls, selects all controls in the current container.
Ctrl + E	Opens the **New Report** page.
Ctrl + O	Opens the **Open Report** page.
Ctrl + S	Opens the **Save Report** page.
Ctrl + Z	Undoes the last action.
Ctrl + Y	Redoes the last action.
Ctrl + X	Cuts text and controls.
Ctrl + C	Copies text and controls.
Ctrl + V	Pastes text and controls.
Del	Deletes text and controls.
Left, Right, Up, Down arrow keys	Moves the visible area of the page in the corresponding direction. In the Table and Tablix controls, navigates between the cells. When controls inside List and Container controls and in the Body of the report are selected, arrow keys allow moving controls by grid-size. In the Chart Control, arrow keys move data-fields and category-fields.
Tab	Navigates in the forward direction between the cells in the Table and Tablix controls. When controls inside List and Container controls and in the Body of the report are selected, Tab key switches between controls in the forward direction.
Shift + Tab	Navigates in the backward direction between the cells in the Table and Tablix controls. When controls inside List and Container controls and in the Body of the report are selected, Shift + Tab switches between controls in the backward direction.

Quick Start

In tabular reports, the data is organized along the rows and columns of a Table data region. The data in the form of fields and entities is fetched from the data set added to the report.

This tutorial guides you through the steps to create a tabular report in ActiveReports WebDesigner.

After you complete this tutorial, you will have a report that looks similar to the following.



Customer Report

Customer Id	Company Name	Contact Title	Address	Country
ALFKI	Alfreds Futterkiste	Sales Representative	Obere Str. 57	Germany
ANATR	Ana Trujillo Emparedados y helados	Owner	Avda. de la Constitución 2222	Mexico
AROUT	Around the Horn	Sales Representative	120 Hanover Sq.	UK
BERGS	Berglunds snabbköp	Order Administrator	Berguvsvägen 8	Sweden
BLONP	Blondesddsl père et fils	Marketing Manager	24, place Kléber	France
BOLID	Bólido Comidas preparadas	Owner	C/ Araquil, 67	Spain
BOTTM	Bottom-Dollar Markets	Accounting Manager	23 Tsawassen Blvd.	Canada
CACTU	Cactus Comidas para llevar	Sales Agent	Cerrito 333	Argentina
CHOPS	Chop-suey Chinese	Owner	Hauptstr. 29	Switzerland
COMMI	Comércio Mineiro	Sales Associate	Av. dos Lusíadas, 23	Brazil
ERNSH	Ernst Handel	Sales Manager	Kirchgasse 6	Austria
FRANS	Franchi S.p.A.	Sales Representative	Via Monte Bianco 34	Italy
FURIB	Furia Bacalhau e Frutos do Mar	Sales Manager	Jardim das rosas n. 32	Portugal
GREAL	Great Lakes Food Market	Marketing Manager	2732 Baker Blvd.	USA
GROSR	GROSELLA-Restaurante	Owner	5ª Ave. Los Palos Grandes	Venezuela
HUNGO	Hungry Owl All-Night Grocers	Sales Associate	8 Johnstown Road	Ireland
MAISD	Maison Dewey	Sales Agent	Rue Joseph-Bens 532	Belgium
SANTG	Santé Gourmet	Owner	Erling Skakkes gate 78	Norway

Access the ActiveReports WebDesigner

Run the WebDesigner sample (WebDesigner_MVC(Core)) that you can download from the following link:

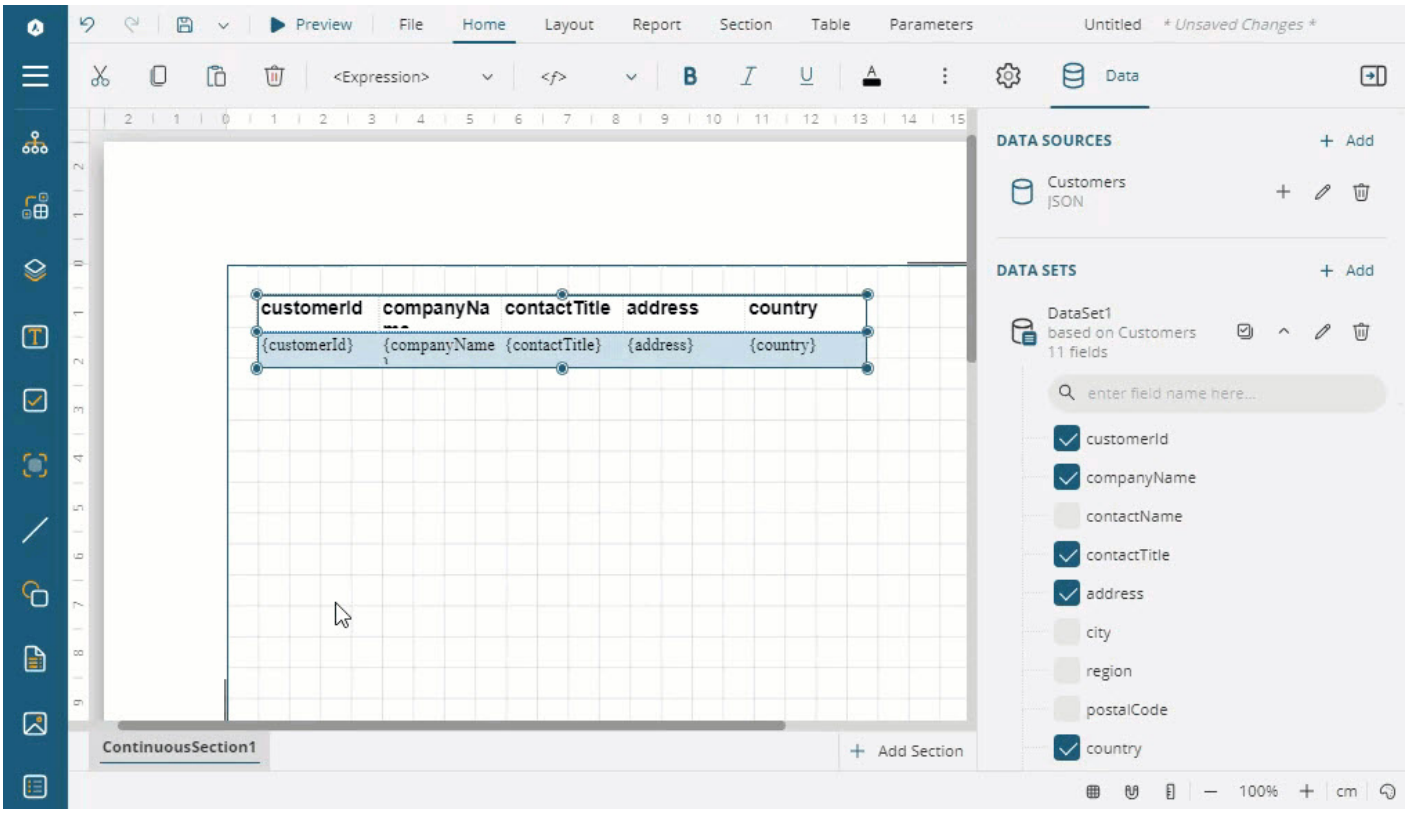
https://github.com/activereports/WebSamples18/tree/main/

The WebDesigner is opened in the browser and is ready to create your reports. By default, you have a blank RDLX Report.

Bind Data to Report

1. Connect to a data source.
    1. Go to **Data** tab and click **Add** next to **Data Sources**.
    2. In the **Data Source Editor** dialog, fill-in the **Name** field as 'Customers'.
    3. Select **Provider** as 'JSON Provider'.
    4. Select **Type** as 'External'.
    5. Enter the following URL in **Path**: https://demodata.mescius.io/northwind/api/v1/Customers
    6. Test the data source connection and click **Add**.
2. Add a dataset.
    1. Click **Add Data Set** next to the added data source.
    2. In the **New Data Set** dialog, fill-in a dataset name and enter the **Query**: `$.[*]`
    3. **Validate** the query to obtain the bound fields and select **OK** to add the dataset.

Create Data-bound Table



1. Expand the dataset fields and enable **Select Fields...**
2. Select the required fields and drag and drop them onto the design area.

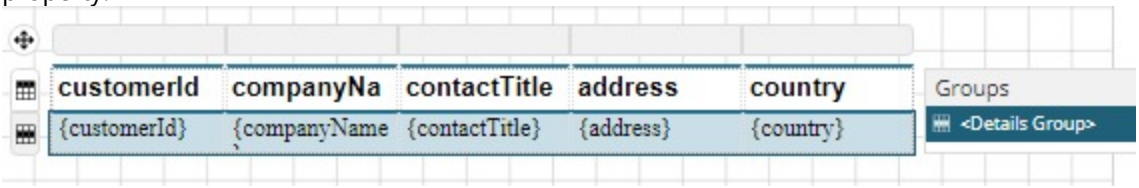
A table with its columns bound to the fields is created. The Header row above the Details row is automatically filled with labels.

Alternatively, you can first drag and drop the **Table** data region onto the design area of the report and then drag and drop fields onto the details row.

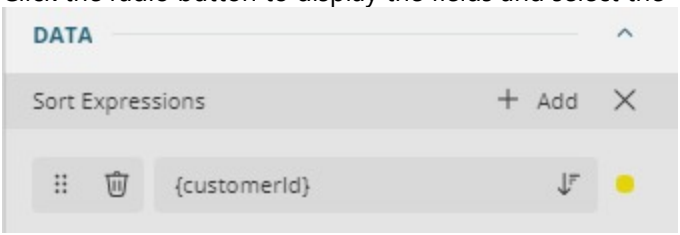
Sort Table Data

Apply sort on any field in the table data in ascending or descending order. Let us sort our table data by the **customerId** field in descending order.

1. From the adorner on the right side of the table, select <Details Group> and go to the **Sort Expressions** property.



2. Select **Add Item**.
3. Click the radio button to display the fields and select the 'customerId' field.



> **Note:** You can also enter a sort expression in the **Expression Editor: Data - Sort Expressions** dialog by selecting **Expression...**

4. Click descending icon to sort the 'customerId field' in descending order. You can change the sort order to ascending by clicking the ascending icon (default).

Group Table Data

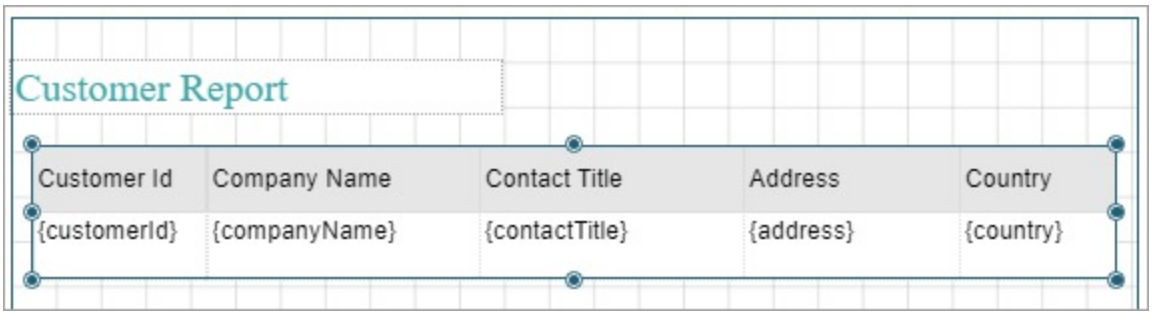
You can organize the table data in groups to add more meaning to the table. Let us group the table with respect to the country field, so that our table displays unique country values.

1. From the adorner on the right side of the Table data region, select <Details Group>, and go to the **Properties** pane.
2. Next to the **Group Expressions** property, click **Add Item**.
3. Click the radio button to display the fields and select the 'country' field.

Add Report Title

1. Drag and drop a TextBox control above Table data region.
2. Click inside the text box and enter the text 'Customer Report'.

Customize Appearance of Tabular Report



Note: You may need to resize and reposition the controls on the report to accommodate data, and for a cleaner look.

1. To customize the Page Header, set the **BACKGROUND - Color** property to '#f1f1f1'.
2. Select the text box containing the text 'Customer Report' and set the following properties.

Property	Value
TEXT - Color	#3da7a8
TEXT - Font Size	16pt
TEXT - Text Align	Left
TEXT - Vertical Align	Middle

3. To customize Header row of the table, set the following properties for all text boxes in the row.

Property	Value
BORDER - Width	0.25pt
BORDER - Style	Solid
BORDER - Color	Gainsboro
TEXT - Text Align	Left
TEXT - Vertical Align	Middle

4. Similarly, customize the text boxes in the Details row of the table.

Preview and Save Report

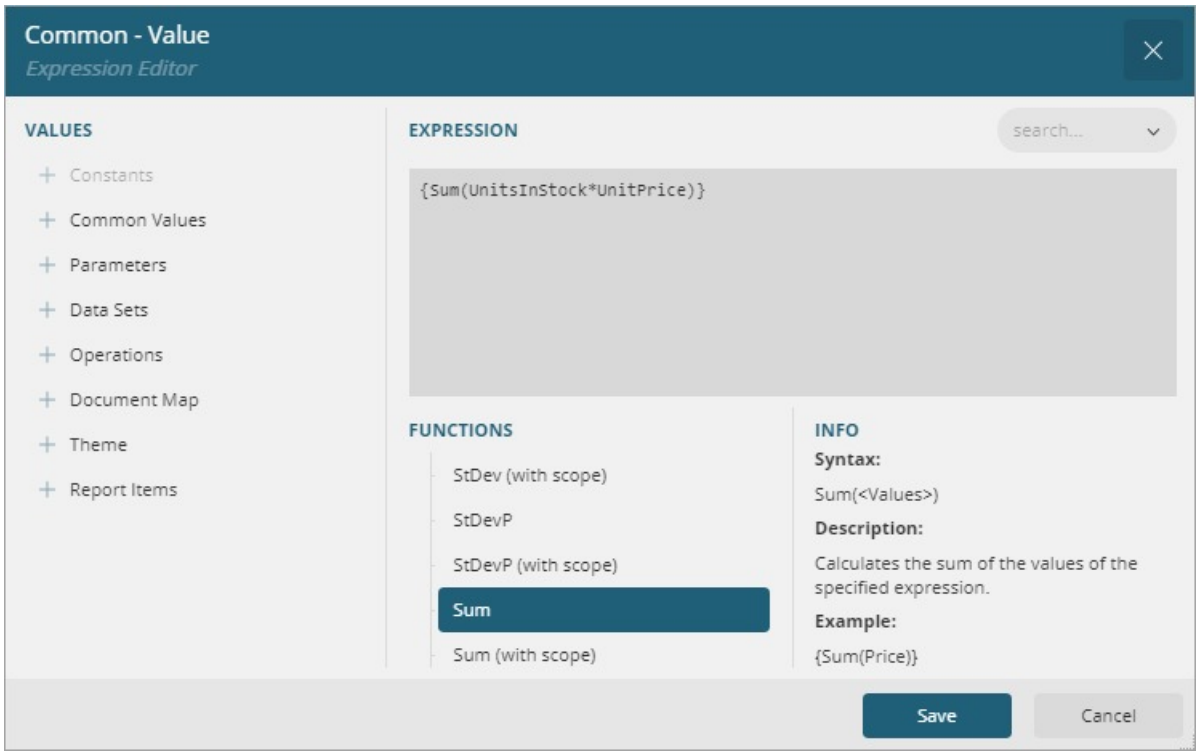
1. Click **Preview** to view the final output of your report.
2. Exit the preview mode by clicking **Back** on the left side of the designer.
3. Click **Save** to open the Save dialog box. Enter the report name and click **Save Report**.

Expressions

You can use an expression to set the value of a control in the report or set conditions under which certain styles apply. You can set expressions through the Expression Editor dialog while setting values in the properties window.

The editor allows you to choose from a number of fields available to the report as well as to a particular property. You can access the Expression Editor by selecting nearly any property of a control and choosing ***f* Expression** from the drop-down list. All expressions are enclosed within curly braces '{}'. Even the expression for a field value for a TextBox is set as follows: `{LastName}`.

While building an expression, you can directly add the entire expression or part of it in the Expression pane of the Expression Editor. Then use the Insert or Append buttons to create a complete expression.



Concatenating Fields and Strings

You can concatenate fields with strings and with other fields. For e.g., use the following expression to get a result like "Customer Name: Bossert, Lewis":

```
Customer Name: {LastName} , {FirstName}
```

Conditional Formatting

You can use expressions in properties like Color, Font, Border, etc. on specific field values based on a condition, to highlight a part of data. The formula for conditional formatting is:

```
{IIF(<Condition>, <TruePart>, <FalsePart>)}
```

For e.g., if you enter the following expression in the Font > FontWeight property of a text box that displays names of people, you get the name "Denise" in bold.

```
{IIF(FirstName = "Denise", "Bold", "Normal")}
```

Similarly, if you enter the following expression in the Background >Color property of a text box in a table, then you get alternating 'Transparent' and 'LightGray' colored text boxes in the rows of the table.

```
{IIF(RowNumber(Nothing) mod 2, "Transparent", "LightGray")}
```

Functions

You can use a number of aggregate and other functions in your expressions. ActiveReports includes a range of functions, including running value, population standard variance, standard deviation, count, minimum and maximum. For e.g., use the following expression to get a count of employees.

```
{Count(EmployeeID, Nothing)}
```

Summary and Total Value Calculations

You can set the following calculations for Summary and Total Value Calculations.

Calculation Option	Result
Default	`current value`
% Grand Total	`current value ÷ grand total of the tablix`
% Row Group Total	`current value ÷ current row group grand total`

% Parent Row Group Total	`current value ÷ parent row group subtotal in the current column group`
	For subtotals,
	`subtotal ÷ grand total of the current column group`

% Column Group Total	`current value ÷ current column group grand total`

% Parent Column Group Total	`current value ÷ parent column group subtotal in the current row group`
	For subtotals,
	`subtotal ÷ grand total of the current row group`

Multiple TextBoxes Operations

It is now possible to set the common expression for multiple TextBoxes (including TextBoxes in Table and Tablix) for all the properties.

To set the common expression for multiple TextBoxes (having the same or different data fields), first select the TextBoxes. In the Properties Window, select the property for which you want to set the expression, and then select **Expression** to open the Expression Editor. Under the **Values**, expand the **Common Values** node, select **Current Textbox Value**, enter a valid expression with the **($$$)** variable, and click **Save**.

Let us see how this works by looking at two samples, applied to the 'Page/Invoice.rdlx' report. This report is available in the 'resources' folder of WebDesigner_MVC_Core sample.

Example 1: Add the **Sum** expression to multiple data fields in a Table data region.

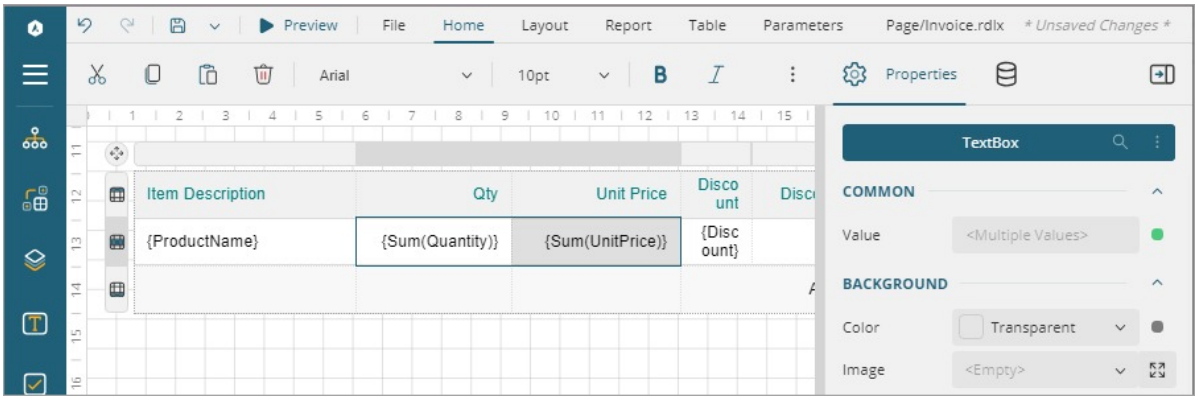
Steps:

1. From the WebDesigner_MVCcore sample, open the 'Page/Invoice.rdlx' report.
2. Select the `{Quantity}` and `{UnitPrice}` fields in the **Details** row of the Table data region.
3. Go to **Common > Value** property and enter the following expression in the **Expression Editor**:

```
{Sum($$$)}
```

4. Save the expression and close the Expression Editor.
5. Check that the 'Value' expression is updated for each of the selected fields to the following values:

    - `{Sum(Quantity)}`
    - `{Sum(UnitPrice)}`



Example 2: Add a background color to multiple data fields in a Table data region.

Steps:

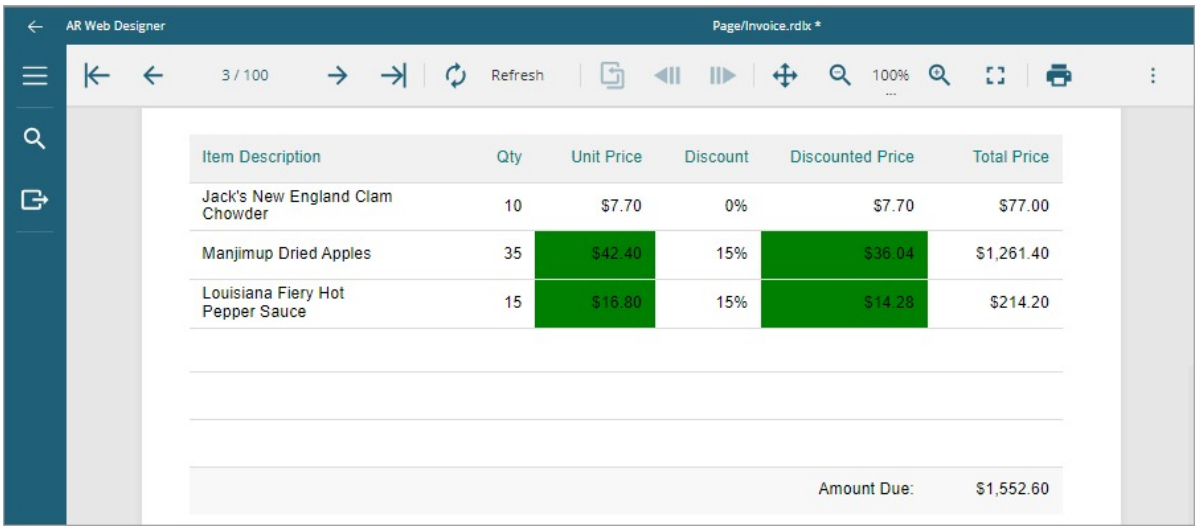
1. From the WebDesigner_MVCcore sample, open 'Page/Invoice.rdlx' report.
2. Select the `{UnitPrice}` and `{UnitPrice - UnitPrice * Discount}` fields in the **Details** row of the Table data region.
3. Go to **Background > Color** property and enter the following expression in the **Expression Editor**:

```
{iif($$$ > 14, "Green")}
```

4. Save the expression and close the Expression Editor.
5. Check that the 'Background Color' expression is updated for both selected fields to the following values:

    - `{iif(UnitPrice > 14, "Green")}`
    - `{iif(UnitPrice - UnitPrice * Discount > 14, "Green")}`

    This expression evaluates the background color to Green color if the selected values are greater than '14'.

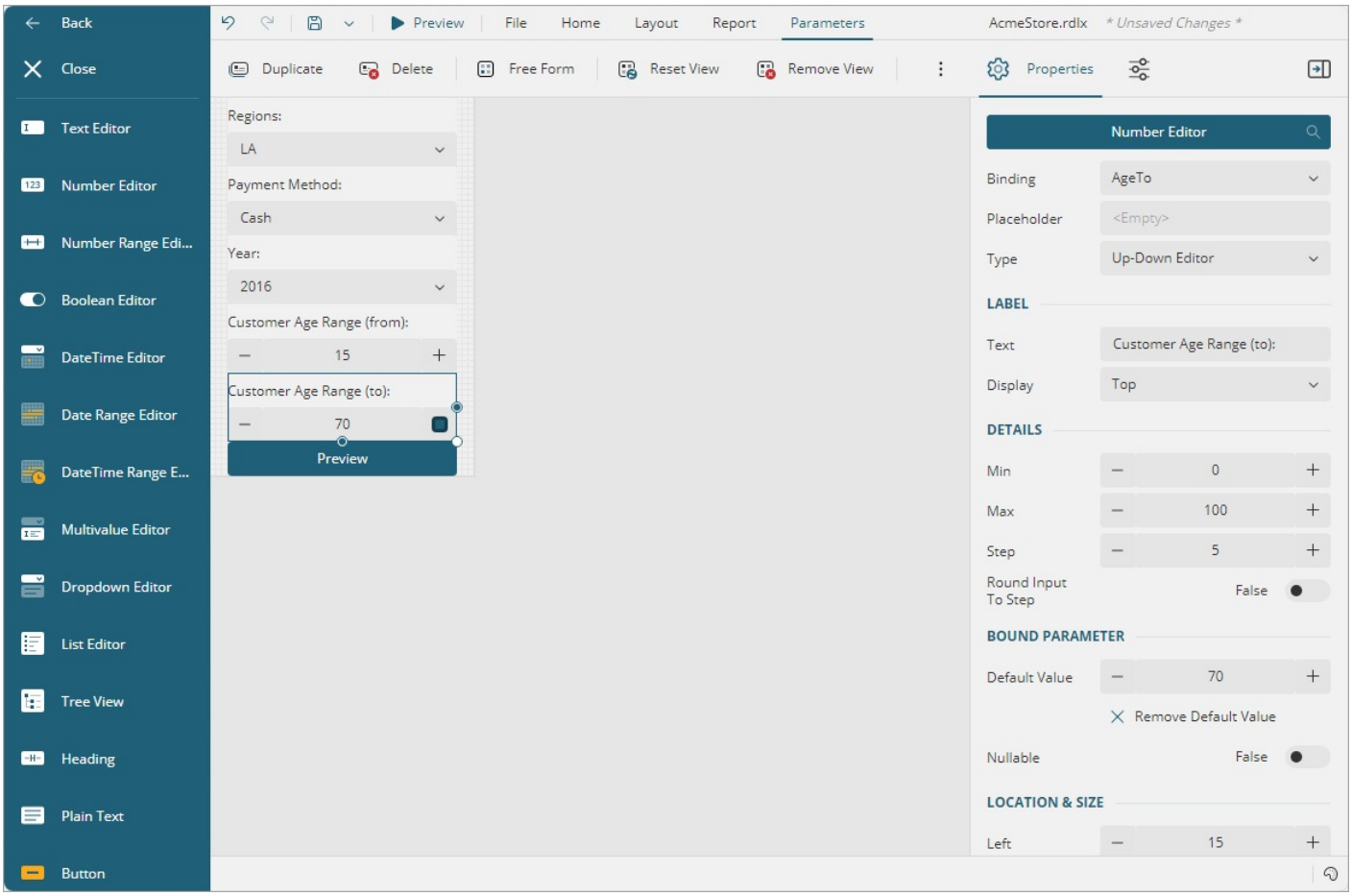


Custom Parameters View

The Parameters View/Panel can be designed in Report Designer using the exclusive set of controls to create and define parameters and prompts quickly. The controls include input controls like text editor, number editor, heading, and plain text; range editors like date range and date-time range; and list and dropdown editors.

On previewing the report, the customized parameters view/panel is shown in the Parameter Panel of the viewer on preview.

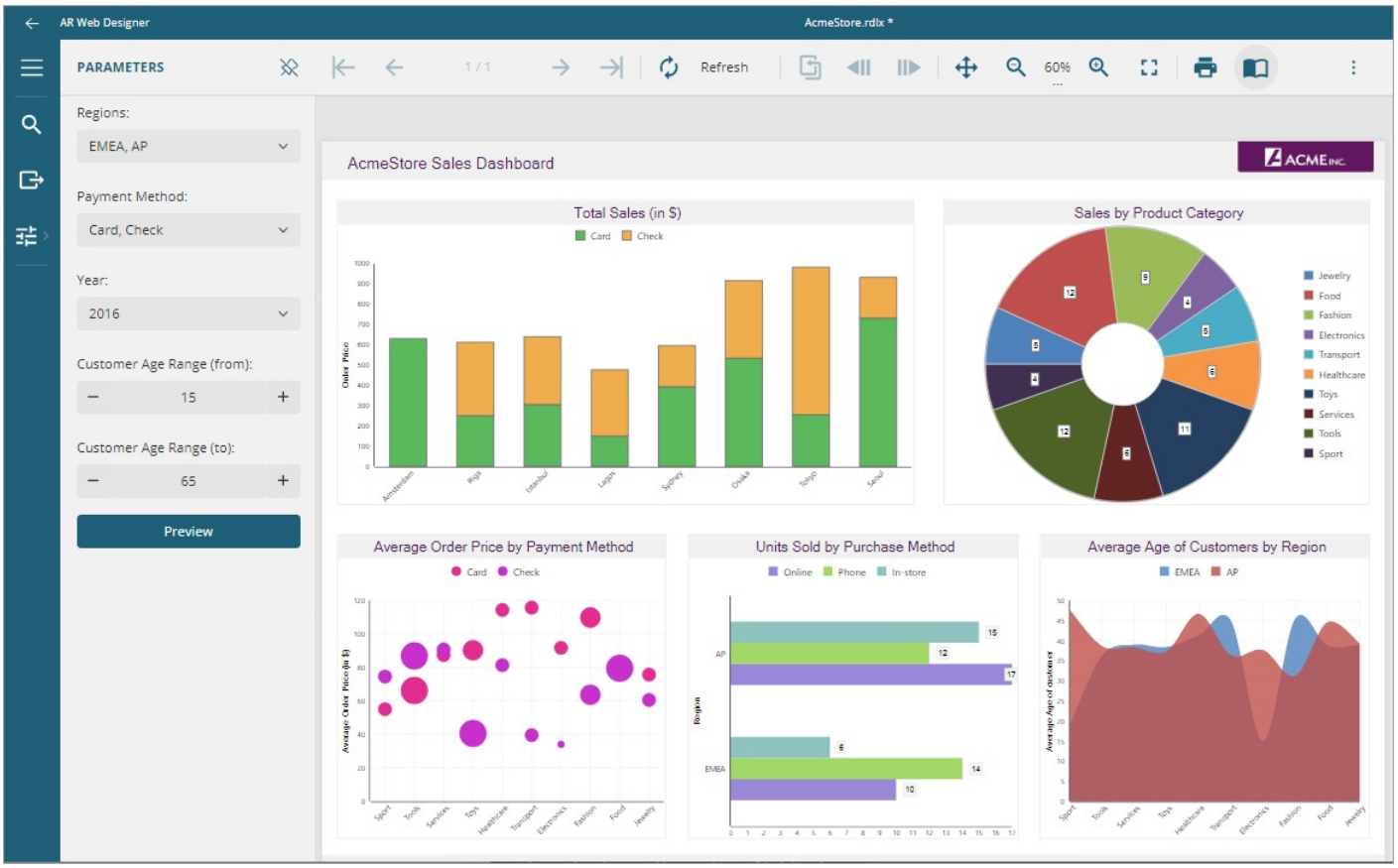
Design a Parameters View



The reference to AcmeStore.rdlx will be taken.

1. Go to the **Parameters** tab at the top.
2. Create the custom parameters view for the report by clicking the **Generate View** button in the toolbar or clicking the **Generate** button in the center of the designer
   If the report already has pre-defined report parameters, a default parameters layout is generated in the design area from these parameters. You can then build the parameters view from the default view or simply create a fresh parameter view using the controls available in the parameters designer.
3. Drag-drop the controls from the toolbox that will be used for inputting parameters.
4. Set the properties of each of the control such as Binding, Type, depending on the function each of the control.
5. Add buttons to preview the report based on selected parameters, reset the parameter selection to default, and clear the selection.

Preview the Report with Custom Parameters View



The controls provide an intuitive user experience with customization possibilities. Following is the list of controls available for designing a parameters view.

- **Text Editor**: For single-line parameter of the String type.
- **Number Editor**: For parameters of integer or float type. Set the control's type as **Up-down Editor** for increasing or decreasing at specified steps, or as **Slider** to be able to slide the value of the parameter.
- **Number Range Editor**: For a parameter range of integer or float type. You need to specify the range in **From** and **To** fields in **Binding**.
- **Boolean Editor**: For parameters with the Boolean value. The Boolean editor can be a Toggle, Checkbox, or a Radio button. As you change the type of the Boolean editor, the options accordingly can be filled in.
- **DateTime Editor**: For parameters with date and date-time values.
- **Date Range Editor**: For parameters with a date field. Specify the start date and end date in the **From** and **To** fields in **Binding**.
  You can also add custom ranges manually using the **Ranges** property (Add > Edit) or select ranges from the pre-defined options in a dropdown menu (Current, Last, Next, ToDate, and LastToDate) for intervals (specified via the **Unit** property: Year, Quarter, Month, Week, etc.) Using the **View Mode** property, you can set the calendar's initial view to Default, Days, Months, or Years.
- **DateTime Range Editor**: For parameters with a range of dates with an explicit starting and ending time.
  You can also add custom ranges manually using the **Ranges** property (Add > Edit) or select ranges from the pre-defined options in a dropdown menu (Current, Last, Next, ToDate, and LastToDate) for intervals (specified via the **Unit** property: Year, Quarter, Month, Week, etc.)
- **Multivalue Editor**: For multi-value parameters whose values are required to be entered manually. It appears with a list of multiple values in the drop-down with the search box.
- **Dropdown Editor**: A list of fields in the parameter appears in the drop-down with a search box. Single or

multiple values can be selected depending on whether the Multi Value property is enabled or not.
- **List Editor:** Fields bound to parameter appears as a list, with checkboxes to select the field.
- **Tree View**: For hierarchical parameter. Select the Tree View type as **List** or **Dropdown**.
- **Heading**: For static text, e.g., entering a heading with a choice of colors for error or warning.
- **Plain Text:** For static text, eg., entering single or multiline text.
- **Button**: To perform actions on the preview screen, select **Preview** to preview report based on the selected parameters, **Reset** to reset the parameter selection to default, and **Clear** to clear the selection, on preview window.

Use **Free Form/Stack** switch to see how parameters will look in the preview window.

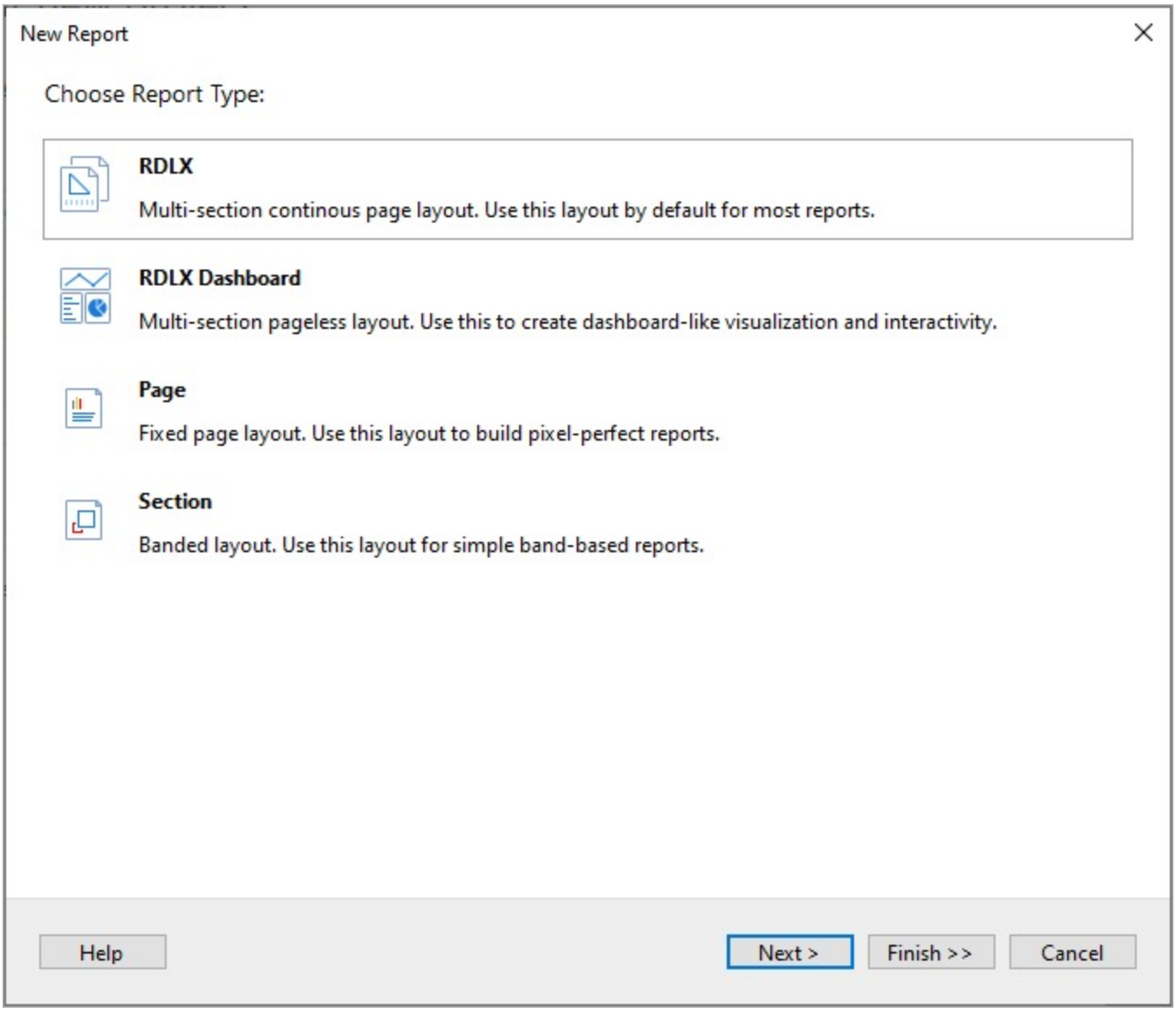
When the 'Free Form' layout type is on, the view matches the current parameter panel on the viewer's sidebar. The order of parameters can be changed by dragging the controls. You can also change the location and size of the controls.

When the 'Stack' layout is on, the view arranges the controls vertically. The order of parameters can be changed by dragging the controls. Also, you can select the 'Highlight Required' option to highlight any errors while designing the parameter panel.

For information on the WebDesigner API and its integration into applications, see WebDesigner Application.

Report Types

As Report Authors, you can design a report using a number of pre-defined templates, where each report type has its advantages and purpose.



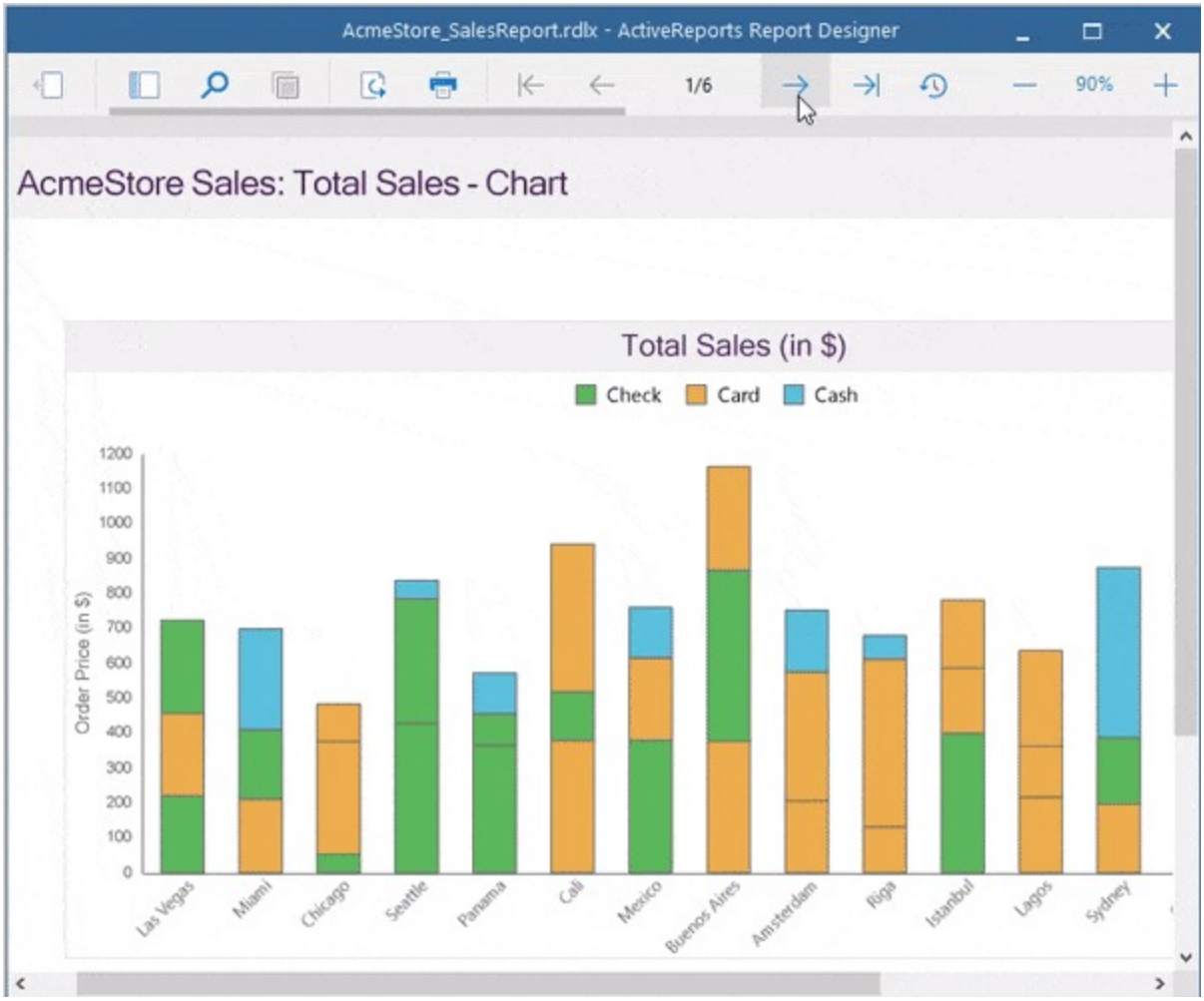
When choosing what report type best meets your needs, you should pay attention to the key benefit of each report type.

- **RDLX**
  A multi-section continuous page layout that combines multiple continuous page layouts together.
- **RDLX Dashboard**
  A multi-section pageless layout for creating dashboard-like reports.
- **Page**
  A fixed page layout that builds pixel-perfect reports.
- **Section**
  A banded layout that consists bands for special layouts in reports.

The guidelines for choosing a report type for DevOps and Developers are elaborated here.

The topics in this section provide you with more details on each report type.

RDLX Report



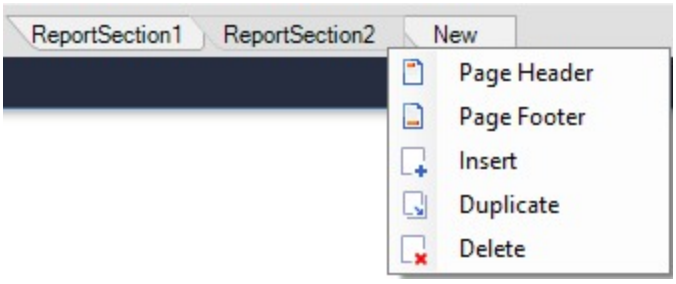
The RDLX report, provides you the ability to define multiple layouts in a report for the flexible presentation of your report data. In this report, you can place data regions that require different viewing layouts, on separate sections, and set margins, page size, and page orientation for pages in each section to accommodate the data regions.

An RDLX report has the following additional features.

Key Features

Manage Report Sections

You can add new sections from the **New** tab that appears below the design area. You can also add new sections by right-clicking the **ReportSection** tab and selecting the **Insert** option from the context menu. To duplicate the report section, select the **Duplicate** option from the context menu. You can drag the section tabs to reorder report sections. You can also change the report section's name using the **Name** property. Just right-click on the section's name (for example, ReportSection2) to quickly add a Page Header or Page Footer to the section or Delete the section. A report section can have its own page header or footer.



To control the visibility of the initial state of the section, use the **Hidden** property from the Properties panel and enter a Boolean value or an expression that evaluates to a Boolean value. See the topic to Hide or Show Sections in RDLX and RDLX Dashboard Reports.

Also see Quick Start topic that adds two report sections to create the report.

Set Layout of each Section

The layout of pages in each report section can be independently controlled using the following properties:

- Margins - Left, Right, Top, Bottom
- PageSize
- PaperOrientation - Landscape, Portrait

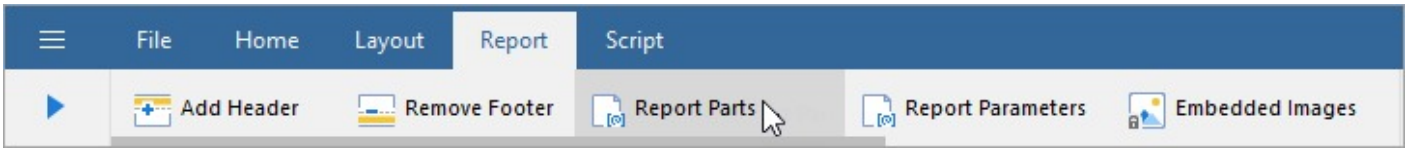
Merge Reports with RDLX Report

You always had the ability to combine multiple report types into a single report using **ReportCombiner.Mode ('Mode Property' in the on-line documentation)** property in code.

While **SubReports** mode combines multiple reports as subreports, the new **ReportSections** mode converts the reports being combined to a report with multiple report sections, that is, the output is an RDLX report. See Merge Multiple Reports for more information.

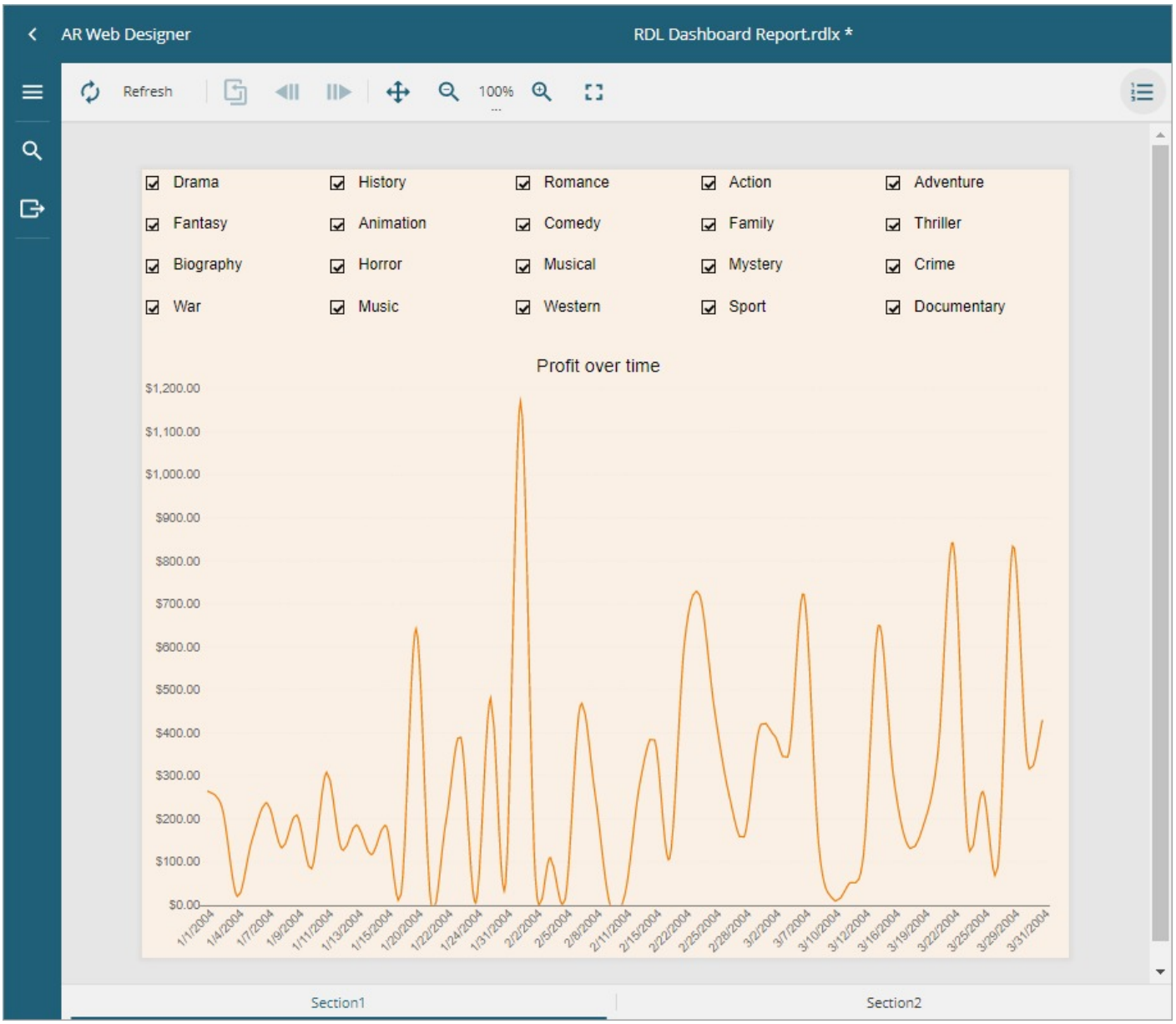
Create Report Parts for use in WebDesigner

The report lets you create report parts and the collection of these, called the report library, can be reused in WebDesigner. Use the **Report** menu > **Report Parts** option in the toolbar or from the **Properties** panel > **ReportParts** collection.



See creating Report Parts for more information.

RDLX Dashboard Report



The RDLX Dashboard report is the latest addition to the varied reports that you can design in ActiveReports. The RDLX Dashboard reports are best viewed in the JS Viewer.

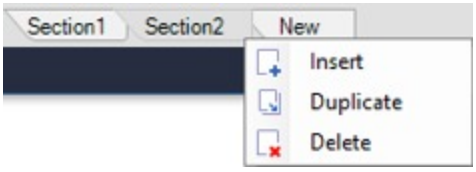
Key Features

Manage Sections

RDLX Dashboard Reports can be multi-page like RDLX Report. With multiple pages or sections, you can make the dashboard report more informative with details spread across the pages. These pages of the report can be navigated using the tabs.

You can add new sections to the report from the **New** tab that appears below the design area. You can also add new sections by right-clicking the **Section** tab and selecting the **Insert** option from the context menu. To duplicate the

report section, select the **Duplicate** option from the context menu. You can also delete the section by selecting the **Delete** option.



You can use a slider that allows to scroll the tabs horizontally. This slider appears if a report has many sections, which tabs do not fit into the page.

To control the visibility of the initial state of the section, use the **Hidden** property from the Properties panel and enter a Boolean value or an expression that evaluates to a Boolean value. See Hide or Show Sections in RDLX and RDLX Dashboard Reports for more information.

Scrollable Container Component

The Container control in the RDLX Dashboard Report is scrollable. You can manage the scroll behavior by using the **Overflow** property. See Container Control in RDLX Dashboard Report for more information.

Actionable Items Provide Extensive Interactivity

The **Action** property in report controls and data regions makes the dashboards highly interactive and helps slice and dice the report metrics. You can click on an action and see the action. The action works across dashboard pages too.

If in addition, there is action(s) set on control(s) inside a Table cell, List, or BandedList, that action will be given priority over the action set on the Table's row, List data region, and BandedList's band.

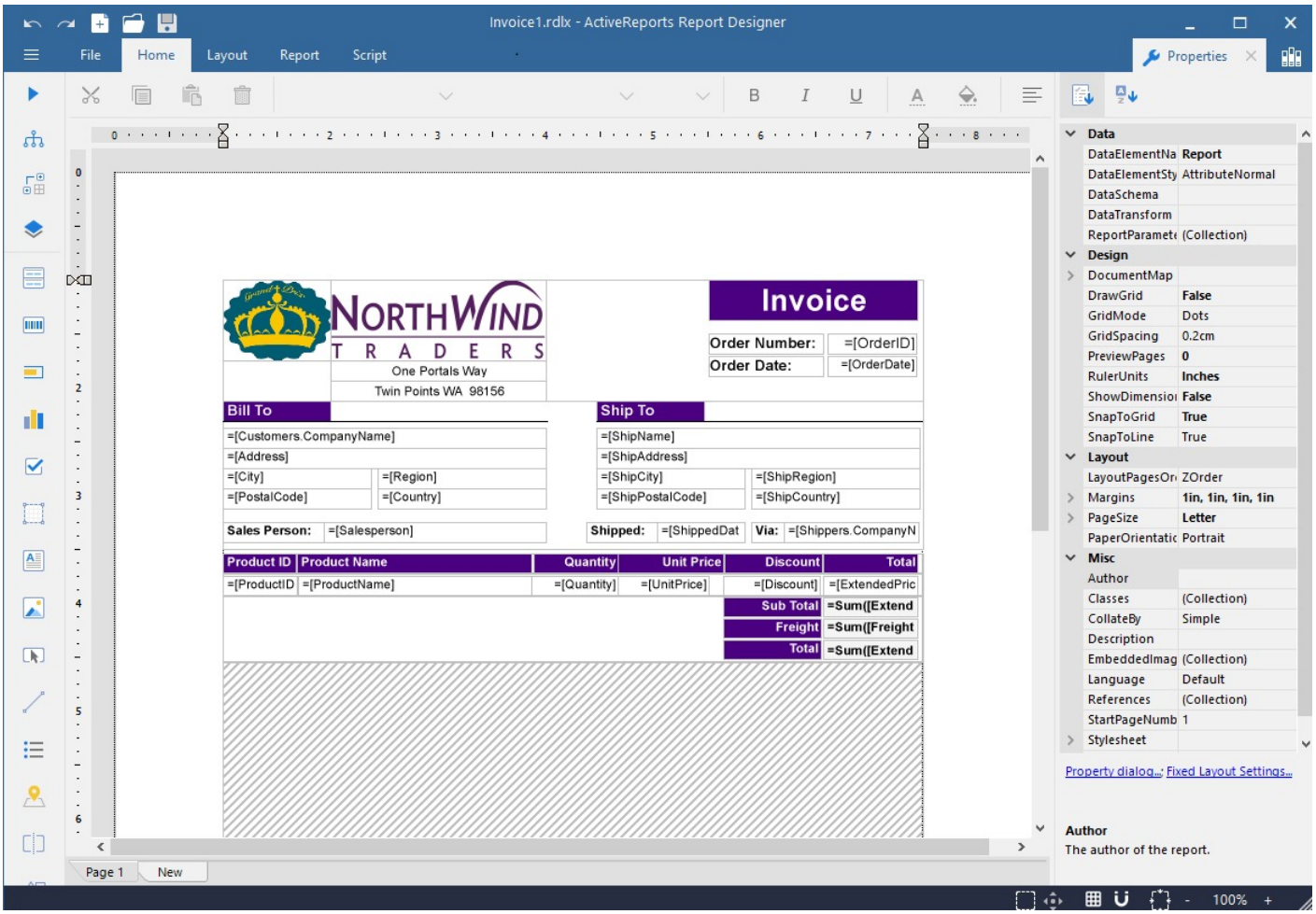
> **Note**: Since RDLX Dashboard reports have a page-less view, the page-related properties or expressions such as Page Totals, Page Numbers, and Page Breaks, are not supported.

Preview Limitations for Standalone and Visual Studio Integrated Designers

It is suggested to use JSViewer or WebDesigner preview for viewing RDLX Dashboard reports due to the following limitations at the moment:

- It is not possible to switch between report sections (sections are rendered as separate pages).
- Scrollable regions are not supported at preview.

Page Report



In a Page Layout, you can create very specific styles of reports that are very difficult, if not impossible, in other .NET reporting tools. You design this type of report on a page where none of the report controls can grow or shrink at run time, making it ideal for duplicating legacy paper forms. This report type is great for billing statements, mail merge, catalogs, forms, and other reports with layout constrictions.

Page report controls do not change in size based on the data, but you can use an Overflow Place Holder to handle any extra data.

With Page reports, there is no need to use code or add measurements to make sure that everything fits. Unlike the RDLX report, the controls remain fixed at run time, so you can drop a table on the report, set a property to size it exactly how you want it, and have something very close to a WYSIWYG report at design time.

Key Features

Grouping

You can set page level grouping to render one row of data on each page. This is ideal for something like a tax form that you want to print for every client or every employee, or an invoice that you want to print for every customer.

OverflowPlaceholder

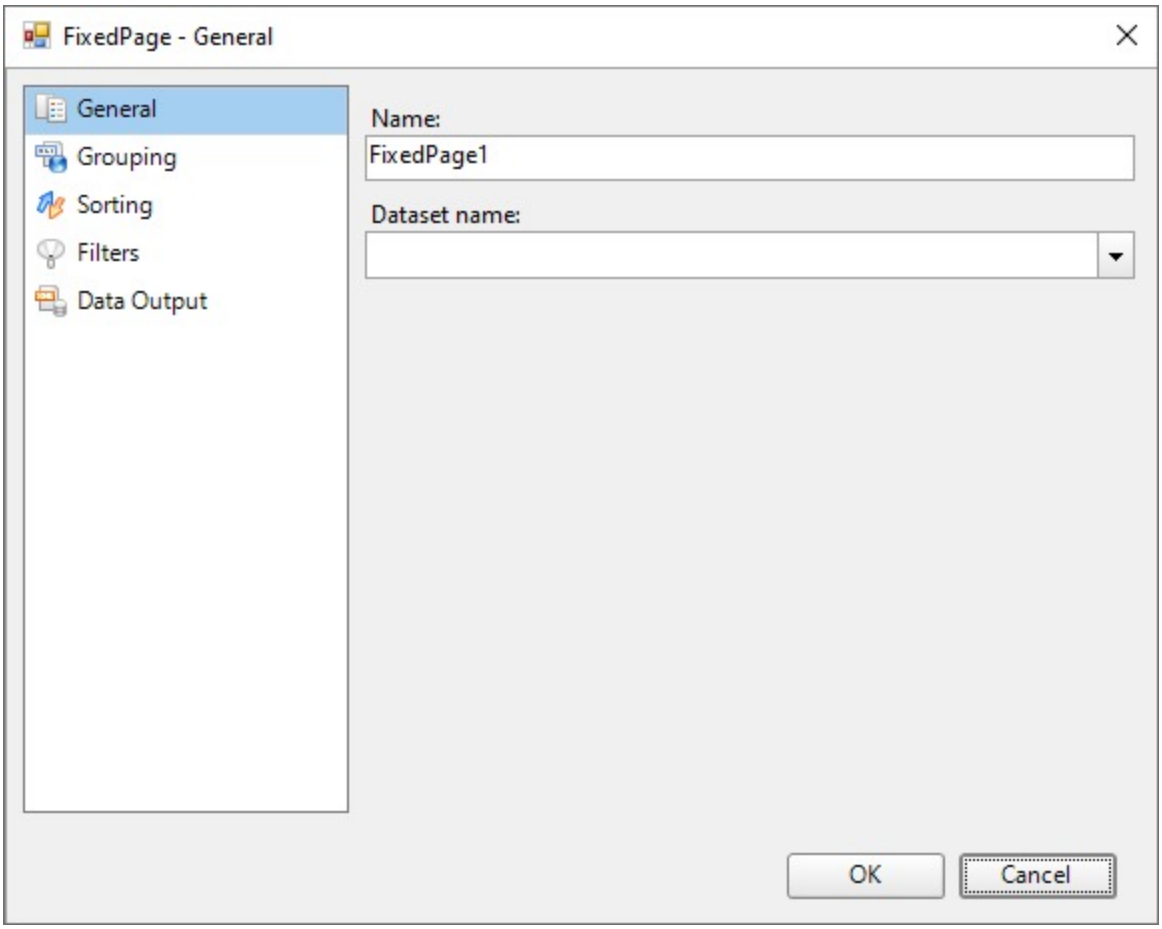
If there is data that does not fit within the space allocated for the data region at design time, you can assign it to flow

into an OverflowPlaceholder control. This can go on the same page in a different area, for example, in the form of columns, or it can go on a separate page.

Themes

You can add multiple themes to a report. In this case, the report renders a combination of multiple outputs for each theme. For example, if a report has two themes, then the report output includes a combination of the first and the second themes, applied to each report page. You can control the combination rules of the report output in the **CollateBy ('CollateBy Property' in the on-line documentation)** property.

Fixed Page Dialog



In a $$Page Report$, you can set the basic properties for your page from the **FixedPage** dialog. You can access the FixedPage dialog by doing one of the following:

- Right-click the gray area outside the design surface and from the context menu that appears, select **Fixed Layout Settings**.
- Click the gray area outside the $$design surface$$ to select the Report and in the commands section at the bottom of the Properties Panel, click the **Fixed Layout Settings** command.
- Click the $$design surface$$ to select the Page and in the Commands section at the bottom of the Properties Panel, click the **Property dialog** command.
- In the Report Explorer, select the Report node or the page node and in the commands section at the bottom of the Properties Panel, click the **Fixed Layout Settings** or **Property dialog** command respectively.

General

The General page of the FixedPage dialog allows you to control the following properties:

- **Name**: Enter a name for the Page Report. This name is used to call the page in code so it should be unique within the report.
- **Dataset name**: Select a dataset to associate with the Page Report. The dropdown list is automatically populated with all the datasets in the report's dataset collection.

Grouping

The Grouping page is useful when you want to show data grouped on a field or an expression on report's pages. See Group Data for more information.

The Grouping page contains following tabs which provide access to various grouping properties:

General

- **Name**: Enter a name for the Fixed Layout group that is unique within the report. This property cannot be set until after a Group on expression is supplied. A name is created automatically if you do not enter one.
- **Group On**: Enter an expression to use for grouping the data.
- **Document map label**: Enter an expression to use as a label to represent this item in the table of contents (document map).

Filters

The Filters tab allows you to add and control the Filter collection for the Fixed Layout Group. Use the + button to add a filter and the X button to delete a filter. You need to provide three values to add a new filter to the collection:

- **Expression**: Enter the expression to use for evaluating whether data should be included in the group.
- **Operator**: Select from the following operators to decide how to compare the expression to the left with the value to the right:
  - **Equal** Only choose data for which the value on the left is equal to the value on the right.
  - **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
  - **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
  - **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
  - **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
  - **LessThan** Only choose data for which the value on the left is less than the value on the right.
  - **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
  - **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
  - **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
  - **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
  - **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
  - **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
  - **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.
- **Value**: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

- **Values**: When you choose the **In** operator, you can enter as many values as you need in this list.

Data Output

The Data Output tab allows you to control the following properties when you export to XML:

- **Element name**: Enter a name to be used in the XML output for this group.
- **Collection**: Enter a name to be used in the XML output for the collection of all instances of this group.
- **Output**: Choose Yes or No to decide whether to include this group in the XML output.

Layout

Has own page numbering: Check this box to enable section page numbering like Page N of M (Section). See Add Page Numbering for further information on setting page numbering in Page Report.

Sorting

The Sorting page of the FixedPage dialog allows you to enter sort expressions for sorting data alphabetically or numerically.

Expression: Enter an expression by which to sort the data.

Direction: Select whether you want to sort the data in an Ascending or Descending order.

Filters

The Filters page of the FixedPage dialog allows you to control the Filter collection for the Fixed Layout. Use the + button to add filters and X buttons to delete them. You need to provide three values to add a new filter to the collection:

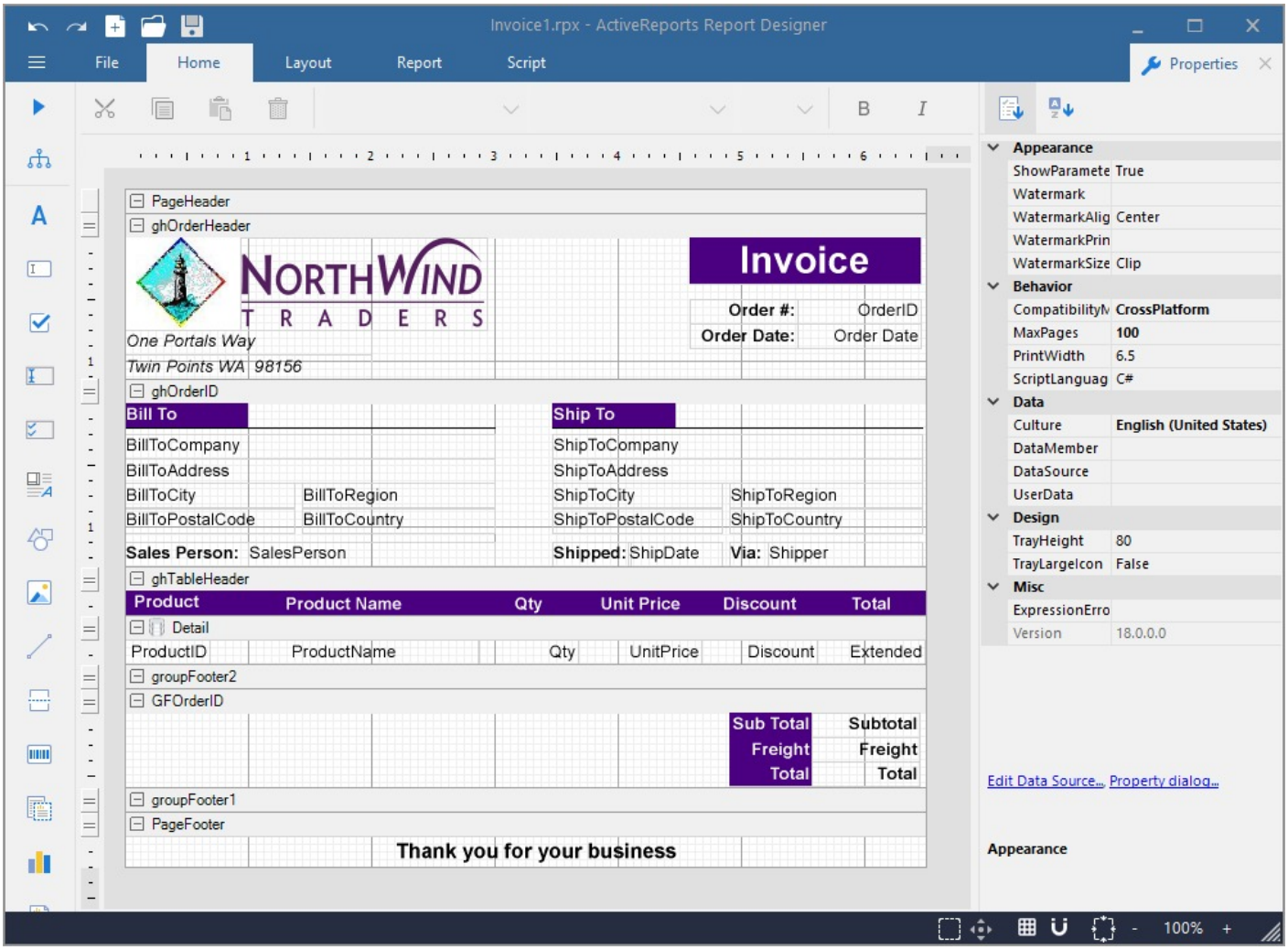
- **Expression**: Enter the expression to use for evaluating whether data should be included in the Fixed Layout.
- **Operator**: Select from the following operators to decide how to compare the expression to the left with the value to the right:
  - **Equal** Only choose data for which the value on the left is equal to the value on the right.
  - **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
  - **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
  - **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
  - **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
  - **LessThan** Only choose data for which the value on the left is less than the value on the right.
  - **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
  - **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
  - **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
  - **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
  - **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
  - **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
  - **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.
- **Value**: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.
- **Values**: When you choose the **In** operator, you can enter as many values as you need in this list.

Data Output

The Data Output page of the FixedPage dialog allows you to control the following properties when you export to XML:

- **Element name**: Enter a name to be used in the XML output for the Page Report.
- **Output**: Choose **Auto**, **Yes**, or **No** to decide whether to include the Page Report in the XML output. Choosing **Auto** exports the contents of the fixed layout.

Section Report



In a Section Layout, you design reports in banded sections. A PageHeader, Detail and PageFooter section appear by default, and you can remove any but the detail section. Right-click the report and select Insert to add other section pairs like ReportHeader and ReportFooter, or GroupHeader and GroupFooter.

You can hide any section that you do not want shown by setting the **Visible** property of the section to False.

Code-based Section Report

When you add an ActiveReports 18 Section Report (code-based) to your Visual Studio project, report layouts are saved as C# or Visual Basic files within the project in which they are created. These files are compiled into the application when you build it.

> **Note:** In Visual Studio 2022, only .NET Framework 4.7.2 and 4.8 are supported for code-based section reports.

This type of report is the most flexible in terms of what a .NET developer can achieve using code. It has an extensive API and is event-based, which allows you to control all aspects of the report and how it is generated. If you like, you can even build a report completely in code.

Xml-based Section Report

When you add an ActiveReports 18 Section Report (xml-based) report to your Visual Studio project, the layout is saved as a standalone Report XML (RPX) file. Since these files are not compiled into your application, they are a good option for solutions in which you need to update or add reports frequently.

The RPX format cannot contain Visual Basic.NET or C# code. Instead, you can add VB.NET or C# script in the **Script** view of the report.

> **Note**: You can combine multiple Section reports by combining the RDF document. For details, see Work with RDF Document.

Report Controls

ActiveReports provides an extensive collection of report controls (and data regions) that you can use when creating your reports. Just drag these report controls from the toolbox and drop them onto the design area to begin as the first step in designing the reports. The reports controls may be unbound like Label, Line, Shape, and Container that do not need data source to show data, or bound like Table, Tablix, Chart, etc. that need data source to show data.

Report Control/Data Region	Page Report	RDLX Report, RDLX Dashboard Report	Section Report
BandedList	✓	✓	✗
Barcode	✓	✓	✓
Bullet	✓	✓	✗
Chart	✓	✓	✓[1, 2]
CheckBox	✓	✓	✓
Container	✓	✓	✗
FormattedText	✓	✓	✗
Image	✓	✓	✗
InputField	✓	✓	✗
Line	✓	✓	✓
List	✓	✓	✗
Map	✓[3]	✓[3]	✗
OverflowPlaceHolder	✓	✗	✗
Shape	✓	✓	✓
Sparkline	✓	✓	✗
Subreport	✓	✓	✓
Table	✓	✓	✗
TableofContents	✓	✓	✗

Tablix	✓	✓	✗
TextBox	✓	✓	✓
Label	✗	✗	✓
InputFieldText	✗	✗	✓
InputFieldCheckBox	✗	✗	✓
RichTextBox	✗	✗	✓
Picture	✗	✗	✓
PageBreak	✗	✗	✓
ReportInfo	✗	✗	✓
CrossSectionLine	✗	✗	✓
CrossSectionBox	✗	✗	✓

Following are some notes for report controls in WebDesigner,

[1] In Section report, Chart data region is not supported in design-time.

[2] Classic Charts (charts in older versions) are converted to new charts showing notification for same when reports with classic charts are loaded in ActiveReports 15 WebDesigner and later. Unsupported classic charts such as Renko and Kagi are converted to column chart.

[3] In Page/RDLX reports, Map control is not fully supported at design time. However, it is correctly displayed at preview.

Page/RDLX Report

The Page or RDLX report (including RDLX Dashboard reports) toolbox group offers a number of report controls and data regions that you can use when creating a report. You can drag these from the toolbox and drop them onto your reports. These tools are different than those in the Section Report Toolbox.

Banded List

The BandedList is a data region with freeform bands in which you can place report controls. With a detail band that repeats data for every row in the dataset, this data region resembles the Section report design surface.

Barcode

The BarCode report control renders scannable barcodes in any of 39 popular symbologies. You can bind it to data, control the bar width, rotation, quiet zones, caption locations etc.

Bullet

The Bullet report control is an easy-to-read linear gauge that is a good alternative to using a dashboard for data visualization. You can bind it to data and set best, worst, and satisfactory values as well as labels and ranges.

Chart

The new Chart is a graphic data region similar to Classic Chart, which is based on representing the data using encodings. It is the default chart available in the designer.

Classic Chart (hidden by default)

Classic charts are legacy RDLX-like charts. The Classic Chart is a graphic data region which allows you to display data in a variety of chart styles with 3D effects and colors, and provides many options for customization. To enable the Classic charts, see Configure ActiveReports.

> **Note:** A classic chart is hidden by default and not supported in the WebDesigner.

CheckBox

The CheckBox report control can display Boolean data, or you can set its Checked property. You can also enter static text to display.

Container

The Container report control is a graphical element that is used as a container for other items. The Container report control has no data associated with it.

ContentPlaceHolder (RDLX Master Report)

The ControlPlaceHolder control is used to show the edit area when a Master report is available.

Formatted Text

The FormattedText report control displays data, and allows you to format selected areas of text within the control in different ways. This report control accepts XHTML input, and allows you to set up mail merge.

Image

The Image report control allows you to specify any image file to display from an external source, a database or an embedded image.

InputField

The InputField report control provides support for editable fields in an exported PDF report.

Line

The Line report control, a graphical element that has no data associated with it, visually marks boundaries or highlights specific areas of a report. You can use lines of various weight, color, and style to highlight regions of your reports and to add style and polish.

List

The List is a freeform data region in which you can place other report controls. It repeats any report control it contains for every record in the dataset.

Map

The Map data region shows your business data against a geographical background. You can select different types of map, depending on the type of information you want to communicate in your report.

Matrix (hidden by default)

The Matrix is an old-style tabular control, which has been replaced with a more powerful Tablix data region.

> **Note**: The Matrix control is hidden by default and not supported in the WebDesigner.

Overflow Place Holder

The Overflow Placeholder report control is only available with page reports. It is a simple rectangle that you link to a List, BandedList, Tablix, or Table data region to display data that extends beyond one page.

Shape

The Shape report control, a graphical element that has no data associated with it, allows you to mark visual boundaries or highlight specific areas of a report with rectangles, rounded rectangles, or elliptical shapes. Unlike the Container report control, it cannot contain other controls.

Sparkline

The Sparkline report control displays a data trend over time in a graph small enough to be used inline, with a height similar to the surrounding text. You can select from line, area, stacked bar, column, and whisker sparkline types.

Subreport

The Subreport control displays data from a separate report that you specify. You can pass a parameter to the subreport from the main report to filter data displayed in a subreport.

Table

The Table is a data region that shows data in rows. By default, it has three columns and three rows. Once set at design time, the columns are static, while the rows repeat for each row of data. The default rows are the header, detail, and footer.

TableOfContents

The TableOfContents (ToC) report control is used to display the document map, an organized hierarchy of the report bookmarks and labels along with their page numbers, in the body of a report. The TableOfContents control allows you to quickly understand and navigate the data inside a report in all viewers that are supported in ActiveReports.

Tablix

Tablix data region displays data in cells that are arranged in rows and columns. Tablix is mainly a combination of two data regions- table and a matrix.

TextBox

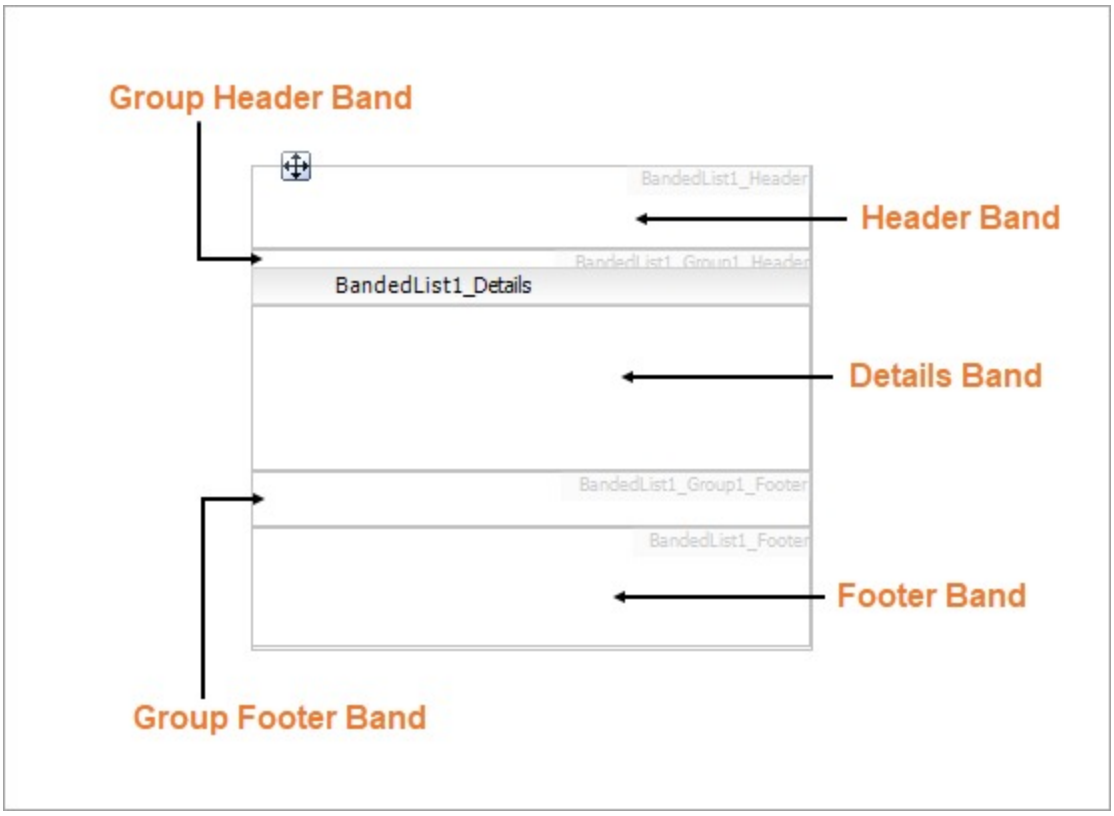
The TextBox displays data, and is the default data region that appears in each cell of a Table or Tablix data region. It is also the data region that is created automatically when you drag a field from the Data Explorer onto your report. You can use expressions to modify the data that appears in a TextBox.

BandedList

You can use the BandedList data region in your Page and RDLX reports to visualize bound data as free-form bands where data may be grouped by data field or an aggregate function like summary and total value.

The BandedList data region is a collection of bands. By default, it is composed of three bands: a header, a footer and a details band. The BandedList may also have group header and group footer bands, which are created when you add a data group to a banded list. Report controls in these bands repeat once for each group in a banded list.

Structure



Header Band

The Header band appears at the beginning of a banded list. Also, if its RepeatOnNewPage property is on, it prints on every page taken by the banded list content. You could use the header band to display the title or the logo on top of the report.

Group Header Band

The Group Header band appears at the beginning of a group when a group is added in the Details band. Also, if its RepeatOnNewPage property is on, it prints on every page taken by the banded list content. You could use the group header band to display the group's field value or summary value.

Details Band

The Details band repeats for each bound data set record that passed through the data set filters and data region filters. For example, the details band may display the name, title, email, phone and photo of each employee.

If a banded list has groups, then the details bands appear between the header and footer of the enclosing group instance.

Group Footer Band

The Group Footer band appears at the end of a group when a group is added in the Details band. Also, if its RepeatOnNewPage property is on, it prints on every page taken by the banded list content. You could use the group footer band to display summary values.

Footer Band

The Footer band appears at the end of a banded list. Also, if its RepeatOnNewPage property is on, it prints on every page taken by the banded list content. You could use the footer band to display grand totals.

Important Properties

Band Properties

Clicking inside each band reveals its properties in the Properties window.

Property	Description
BreakLocation (Details only)	Indicates where the report is broken on a new page of the band. Select from None, Start, End, StartAndEnd, and Between.
CanGrow	Change to True to allow the data region to grow vertically to accommodate data.
CanShrink	Change to True to allow the data region to shrink if there is not enough data to fill it.
KeepTogether	Change to True to have ActiveReports attempt to keep all of the data in the band together on one page.
RepeatOnNewPage	With header and footer bands, repeats the band on every page when the related details span multiple pages.
PageBreakAtStart (all bands except Details)	Indicates whether the report breaks to a new page at the start of the band.
PageBreakAtEnd (all bands except Details)	Indicates whether the header or footer bands are displayed on each page that includes the banded list.

BandedList Properties

Clicking the four-way arrow selects the entire data region and reveals its properties.

Property	Description
DataSetName	Select the dataset to use in the data region.
DataSetParameters	Specify the parameters for the data set.
KeepTogether (RDLX)	Change to True to have ActiveReports attempt to keep all of the data in the data region together on one page.
NewSection	Change to True to render the data region in a new section.
OverflowName (Page)	Select the name of the OverflowPlaceHolder control in which to render data that exceeds the allowed space for the data region on the first page of the report.

Banded List Dialog Properties

You can set the BandedList properties in the Banded List dialog. To open it, with the BandedList selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the banded list that is unique within the report. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Dataset name: Select a dataset to associate with the banded list. The combo box is populated with all of the datasets in the report's dataset collection.

Has own page numbering: Select to indicate whether this banded list is in its own section with regards to pagination.

Consume all white space during report rendering (RDLX): Select to have all white space consumed at report rendering.

Page Breaks (RDLX): Select any of the following options to apply to each instance of the banded list.

- Insert a page break before this banded list
- Insert a page break after this banded list
- Fit banded list on a single page if possible

NewPage (RDLX): Indicates on which page the content to start after the page break.

- **Next:** A default value that makes a new group start from the immediate next page of the report.
- **Odd:** A new group starts from the next odd page of the report.
- **Even:** A new group starts from the next even page of the report.

Header and Footer: Select any of the following options.

- Repeat header band on each page
- Repeat footer band on each page
- Prevent orphaned header on the page
- Prevent orphaned footer on next page
- Print footer at the bottom of the page

Visibility

By default, the banded list is visible when the report runs, but you can hide it when certain conditions are met, or toggle its visibility with another report control. The same applies to banded list groups.

Initial visibility

- **Visible**: The banded list is visible when the report runs.
- **Hidden**: The banded list is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the BandedList is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the BandedList. The user can click the toggle item to show or hide this BandedList.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this BandedList. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Groups

Click the plus sign button to add a new group to the banded list, and delete them using the X button. Once you add one or more groups, you can reorder them using the arrow buttons, and set up information for each group on the following tabs.

General

Name: Enter a name for the group that is unique within the report. This property cannot be set until after a Group on expression is supplied.

Group on: Enter an expression to use for grouping the data.

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

> ⚠ **Caution**: You cannot sort the detail data in a BandedList, so any sorting of this type must be done at the query level.

Expression: Enter an expression by which to sort the data in the group.

Direction: Select Ascending or Descending.

Visibility

By default, the group is visible when the report runs, but you can hide a group when certain conditions are met, or toggle its visibility with another report item.

Initial visibility

- **Visible**: The group is visible when the report runs.
- **Hidden**: The group is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the group is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. The user can click the toggle item to show or hide this band group. This enables the drop-down list where you can select the report control that users can click to show or hide this group.

Data Output

Element Name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose **Yes** or **No** to decide whether to include this group in the XML output.

Layout

BreakLocation: Select from these options to decide where to insert a page break in relation to the group.

- **None**: Inserts no page break.
- **Start**: Inserts a page break before the group.
- **End**: Inserts a page break after the group.
- **StartAndEnd**: Inserts a page break before and after the group.
- **Between**: Inserts a page break between groups (at the end of a current group and the beginning of a next group).

Include group header: Adds a group header band (selected by default).

Include group footer: Adds a group footer band (selected by default).

Repeat group header: Repeats the group header band on each page.

Repeat group footer: Repeats the group footer band on each page.

Has own page numbering: Used in conjunction with the "Page Number in Section" and "Total Pages in Section" properties, tells the report that the group constitutes a new page numbering section.

Keep together on one page if possible: Indicates that the group is kept together on one page if possible.

Prevent orphaned header: Indicates whether the orphaned header is displayed for the group.

Prevent orphaned footer: Indicates whether the orphaned footer is displayed for the group

Print footer at the bottom of the page: Indicates whether to print the group footer at the bottom of the page.

NewPage: Indicates on which page the content to start after the page break.

- **Next:** A default value that makes a new group start from the immediate next page of the report.
- **Odd:** A new group starts from the next odd page of the report.

- **Even:** A new group starts from the next even page of the report.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Data Output

The Data Output page of the Banded List dialog allows you to control the following properties when you export to XML.

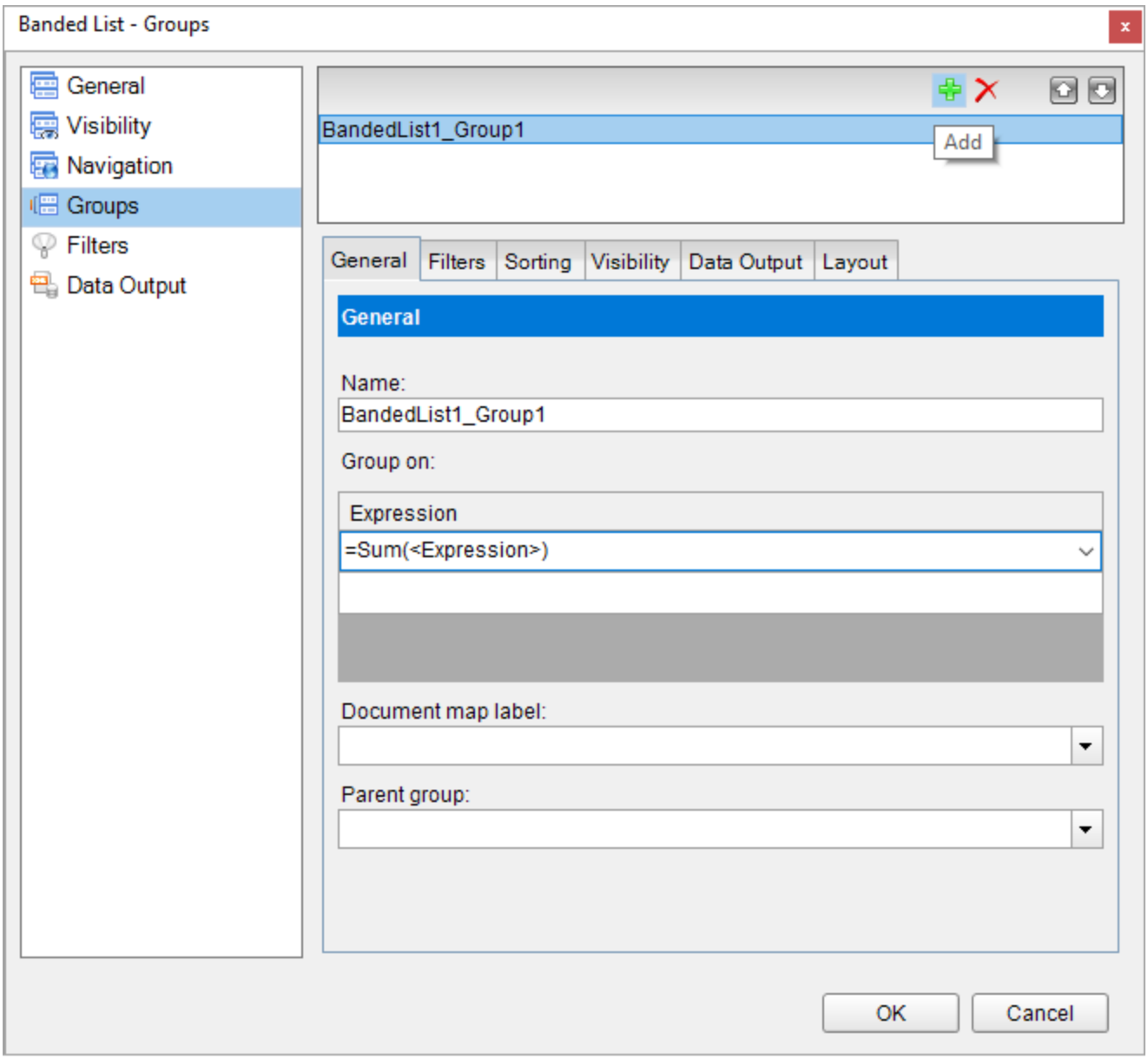
- **Element Name**: Enter a name to be used in the XML output for this BandedList.
- **Output**: Choose **Auto**, **Yes**, **No**, or **Contents only** to decide whether to include this BandedList in the XML output. Choosing **Auto** exports the contents of the BandedList.

BandedList Features

Adding Data Group

You can group data in the Details band by one or multiple criteria, for example, group sales data in the sales list by region, district and year.

1. In the Designer, select a banded list control and right-click it.
2. In the context menu that appears, select **Insert Group**.
3. In the **Banded List - Groups** dialog that opens, enter a grouping expression for each group in the **Group on** property. You can also nest groups, plus, in RDLX reports, you can nest other data regions in any header or footer band.
4. Click the plus sign button to add a new group to the BandedList, and delete them using the X button. Once you add one or more groups, you can reorder them using the arrow buttons, and set up information for each group on the tabs



In the example of the report with the banded list data region, you can see three (3) data groups - data group #1 (=[RegionID]), data group #2 (=[DistrictID]), and data group #3 (=[SaleYear]).

Adding Aggregates

You can use aggregate functions to group data in the BandedList header, footer and group header/footer bands. See Common Functions for the complete list of aggregate functions you can use.

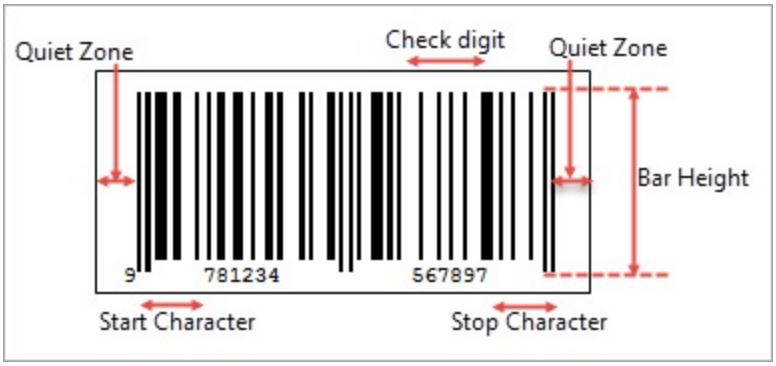
1. Select the banded list and click the **Property dialog** link at the bottom of the Properties window.
2. In the Banded List dialog that appears, go to the **Groups** tab.
3. In the **Expression** field under **Group on**, select <Expression...> to open the **Expression Editor**.
4. In the Expression Editor, select an aggregate function in the list of fields under Common Functions and click **OK**.
5. In the Banded List dialog, click OK.

For example, the **Sum** aggregate function calculates the sum of the values returned by the expression: `=Sum(Fields!TotalSales.Value)`. If you use this aggregate in the group header of the banded list, it will display the total sales value for a data group - in the example below, this is the district sales total value. The same aggregate function, `=Sum(Fields!TotalSales.Value)`, placed to the group footer of the banded list, displays the sum of the values for the District data group.

Barcode

The **Barcode** report control displays data in a machine-readable format to enable accurate scanning of sensitive information quickly and efficiently. This saves you the time and expense of finding and integrating a separate component.

Structure



Quiet zone is the left and right ends of the barcode. Both of the ends must be at least 10 times as wide as the minimum element width for proper scanning of the barcode data.

Start character indicates the start of the barcode data.

Stop character indicates the end of the barcode data.

Check digit is a numeric value used to check for read errors. It is located right after the barcode data.

Bar height is recommended to be greater than 15% of the barcode length. If the bar height is not high enough, it may lead to unstable readings of the barcode by the laser.

> **Note**: If you create reports programmatically, the Page report/RDLX report barcode is treated as a CustomReportItem.

Barcode Types

ActiveReports supports all of the most popular symbologies.

Symbology Name	Example	Description
Ansi39		ANSI 3 of 9 (Code 39) uses uses upper case alphabets (A to Z), numerals (0 to 9), - , * $ / + %. This is the default barcode style.
Ansi39x		ANSI Extended 3 of 9 (Extended Code 39) uses the complete ASCII character set.
Aztec		Aztec is a two-dimensional barcode symbology that supports all the ASCII characters from 0 to 255.
BC412		Data BC412 uses 35 characters, numerals (0 to 9), and upper case alphabets (A to Z). It is designed for semiconductor wafer identification.
Codabar		Data that can be encoded are numerals (0 to 9) and symbols ("-", "$", ":", "/", "+" and ";"). For the Start Character and the Stop Character, A, B, C or D can be selected.
Code_11		Encodes the numerals (0 to 9), the hyphen (-), and start/stop characters. It is primarily used in labeling telecommunications equipment.
Code_128_A		Code 128 A uses control characters, numerals (0 to 9), punctuation, and upper case alphabets (A to Z).
Code_128_B		Code 128 B uses punctuation, numerals (0 to 9), upper case, and lower case alphabets.
Code_128_C		Code 128 C uses only numerals (0 to 9).
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_2_of_5		Code 2 of 5 uses only numerals (0 to 9).
Code_93		Code 93 uses upper case alphabets (A to Z), % $ * / , + -, and numerals (0 to 9).
Code25intlv		Interleaved 2 of 5 uses only numerals (0 to 9).
Code39		Code 39 uses numerals (0 to 9), % * $ /. , - +, and upper case alphabets (A to Z).
Code39x		Extended Code 39 uses the complete ASCII character set.
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.
Code93x		Extended Code 93 uses the complete ASCII character set.
DataMatrix		Data Matrix is a high-density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern.
EAN_13		EAN-13 uses only numerals (12 numbers and a check digit). It takes only 12 numbers as a string to calculate a check digit (CheckSum) and add it to the thirteenth position. The check digit is an additional digit used to verify that a bar code has been scanned correctly. The check digit is added automatically when the CheckSum property is set to True.

EAN_8		EAN-8 uses only numerals (7 numbers and a check digit).
EAN128FNC1*		EAN-128FNC1 is an alphanumeric one-dimensional representation of Application Identifier (AI) data for marking containers in the shipping industry.
GS1QRCode		GS1QRCode is a subset of the QR Code. The GS1 QR Code is a 2D symbol that denotes the Extended Packaging URL for a trade item. It is processed to obtain one URL address associated with the trade item identified by the Global Trade Item Number (GTIN). GS1 QR Code requires the mandatory association of the GTIN and Extended Packaging URL. GS1 QR Code allows encoding GS1 System Application Identifiers (AI) into QR Code 2D barcodes. **Limitation:** Kanji, CN, JP and Korean characters.
HIBCCode128		HIBCCode128 barcode uses the Code128 symbology. It encodes 'Primary Data' and 'Secondary Data' using slash (/) as a delimiter. It is used in the health care products industry for identification purposes.
HIBCCode39		HIBCCode39 barcode uses the Code39 symbology, with the Code39OptionalCheckDigit property set to True. It encodes Primary Data and Secondary Data using slash (/) as a delimiter. It is used in the health care products industry for identification purposes.
IATA_2_of_5		IATA_2_of_5 is a variant of Code_2_of_5 and uses only numerals with a check digit.
IntelligentMail		Intelligent Mail, formerly known as the 4-State Customer Barcode, is a 65-bar code used for domestic mail in the U.S.

IntelligentMailPackage		IntelligentMailPackage is more efficient in terms of processing and tracking mails than Intelligent Mail barcode.
ISBN		International Standard Book Number barcode is a special form of the EAN-13 code and is used as a unique 9-digit commercial book identifier.
ISMN		Internationally Standard Music Number barcode is a special form of the EAN-13 code. It is used for marking printed musical publications.
ISSN		International Standard Serial Number barcode is a special form of the EAN-13 code. It is used to identify serial publications, publications that are issued in numerical order, such as the volumes of a magazine.
ITF14		Interleaved Two of Five code is used to mark cartons that contain goods with an EAN-13 code. One digit is added in front of the EAN-13 code to mark the packing variant.
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.
Matrix_2_of_5		Matrix 2 of 5 is a higher density barcode consisting of 3 black bars and 2 white bars.
MaxiCode		MaxiCode is a special polar barcode that uses 256 characters. It is used to encode a specific amount of data.

MicroPDF417		MicroPDF417 is two-dimensional (2D), multi-row symbology, derived from PDF417. Micro-PDF417 is designed for applications that need to encode data in a two-dimensional (2D) symbol (up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits) with the minimal symbol size. MicroPDF417 allows you to insert an FNC1 character as a field separator for variable-length Application Identifiers (AIs). To insert an FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at run time.
MicroQRCode		MicroQRCode is a two-dimensional (2D) barcode that is designed for applications that use a small amount of data. It can handle numeric and alphanumeric data as well as Japanese kanji and kana characters. This symbology can encode up to 35 numeric characters.
MSI		MSI Code uses only numerals (0 to 9).
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. This symbology can encode up to 35 alphanumeric characters or 2,710 numeric characters.
Pharmacode		Pharmacode represents only numeric data from 3 to 131070. It is a barcode standard used in the pharmaceutical industry for packaging. It is designed to be readable despite printing errors.
Plessey		Plessey uses hexadecimal digits to encode. It is a one-dimensional barcode used mainly in libraries.

PostNet		PostNet uses only numerals (0 to 9) with a check digit.
PZN		Pharmaceutical Central/General Number uses the same encoding algorithm as Code 39 but can carry only digits – 0123456789. The number of digits supported for encoding are 6 or 7. The letters 'PZN' and checksum digit are automatically added. It is mainly used to identify medicine and health-care products in Germany and other German-speaking countries.
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.
RM4SCC		Royal Mail (RM4SCC) uses only letters and numerals (with a check digit). This is the barcode used by the Royal Mail in the United Kingdom.
RSS14 (GS1 DataBar)		RSS14 is a 14-digit Reduced Space Symbology that uses EAN.UCC item identification for point-of-sale omnidirectional scanning. The RSS family of barcodes is also known as **GS1 DataBar**.
RSS14Stacked (GS1 DataBar Stacked)		RSS14Stacked uses the EAN.UCC information with Indicator digits as in the RSS14Truncated, but stacked in two rows for a smaller width. RSS14Stacked allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.

RSS14StackedOmnidirectional (GS1 DataBar Stacked Omnidirectional)		RSS14StackedOmnidirectional uses the EAN.UCC information with omnidirectional scanning as in the RSS14, but stacked in two rows for a smaller width.
RSS14Truncated (GS1 DataBar Truncated)		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.
RSSExpanded (GS1 DataBar Expanded)		RSSExpanded uses the EAN.UCC information as in the RSS14, but also adds AI elements such as weight and best-before dates. RSSExpanded allows you to insert an FNC1 character as a field separator for variable-length Application Identifiers (AIs). To insert FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at run time.
RSSExpandedStacked (GS1 DataBar Expanded Stacked)		RssExpandedStacked uses the EAN.UCC information with AI elements as in the RSSExpanded, but stacked in two rows for a smaller width. RSSExpandedStacked allows you to insert an FNC1 character as a field separator for variable-length Application Identifiers (AIs). To insert an FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at run time.
RSSLimited (GS1 DataBar Limited)		RSS Limited 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. RSSLimited allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.

SSCC_18		SSCC_18 is an 18-digit Serial Shipping Container Code. It is used to identify individual shipping containers for tracking purposes.
Telepen		Telepen has 2 different modes - alphanumeric-only and numeric-only. Both modes require a start character, a check digit, and a stop character. It is mainly used in manufacturing industries.
UCCEAN128		UCCEAN-128 complies to GS1-128 standards. GS1-128 uses a series of Application Identifiers to encode data. This barcode uses the complete ASCII character set. It also uses the FNC1 character as the first character position. Using AI's, it encodes best before dates, batch numbers, weights, and more such attributes. It is also used in HIBC applications.
UPC_A		UPC-A uses only numerals (11 numbers and a check digit).
UPC_E0		UPC-E0 uses only numerals. 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 numerals. Used typically for shelf labeling in the retail environment. The length of the input string for U.P.C. E1 is six numeric characters.

Note that the following barcodes support FNC1 characters:

- Aztec
- UCCEAN128
- MicroPDF417

- RSSExpanded
- RSSExpandedStacked

*The barcode EAN128FNC1 is now obsolete; you should use **UCCEAN128** instead. The UCCEAN128 provides similar functionality with better performance and some default properties. You should choose EAN128FNC1 only if you do not need the default properties.

Important Properties

Clicking the barcode control reveals its properties in the Properties window.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
BarHeight	Set the height, in inches, of the barcode's bars. If the bar height exceeds the height of the control, this property is ignored.
CaptionGrouping	Gets or sets a value indicating whether to add spaces between groups of characters in the caption to make long numbers easier to read. This property is only available with certain styles of barcode and is ignored with other styles.
CaptionLocation	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.
Font	Set the font for the caption. Only takes effect if you set the CaptionPosition property to a value other than None.
NarrowBarWidth	Also known as the X dimension, this is a value defining the width of the narrowest part of the barcode. Before using an extremely small value for this width, ensure that the scanner can read it. This value is specified in pixels (for example, 10 pixels).
NWRatio	Also known as the N dimension, this is a value defining the multiple of the ratio between the narrow and wide bars in symbologies that contain bars in only two widths. For example, if it is a 3 to 1 ratio, this value is 3.
Quiet Zone	Sets an area of blank space on each side of a barcode that tells the scanner where the symbology starts and stops. You can set separate values for the Left, Right, Top, and Bottom.
Rotation	Sets the amount of rotation to use for the barcode. You can select from None, Rotate90Degrees, Rotate180Degrees, or Rotate270Degrees.
SupplementOptions	Sets the 2/5-digit add-ons for EAN/UPC symbologies. You can specify Text, DataField, BarHeight, CaptionPosition, and Spacing for the supplement.
BackgroundColor	Select a background fill color for the barcode.
CheckSum	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.
Color	Select a color for the barcode and caption.
Symbology	Sets the symbology used to render the barcode. See the table below for details about each symbology.

Barcode Specific Properties

Aztec Options

Aztec options are available for the Aztec barcode style.

Error Correction: Indicates whether to allow code recovery if the barcode image is partly damaged, the value is ranging from 10% to 90%.

Layers: Indicates the number of the barcode layers.

Encoding: Select the barcode encoding from the drop-down list.

Code49 Options

Code49 options are available for the Code49 barcode style.

Use Grouping: Indicates whether to use grouping for the Code49 barcode. The possible values are **True** or **False** (default). If Grouping is set to True, any value not expressed by a single barcode is expressed by splitting it into several barcodes.

GroupNumber: Enter a number between 0 (default) and 8 for the barcode grouping. 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.

DataMatrix Options

DataMatrix options are available for the DataMatrix barcode style.

EccMode: Select the Ecc mode from the drop-down list. The possible values are **ECC000**, **ECC050**, **ECC080**, **ECC100**, **ECC140** or **ECC200**.

Ecc200 Symbol Size: Select the size of the ECC200 symbol from the drop-down list. The default value is **SquareAuto**.

Ecc200 Encoding Mode: Select the encoding mode for ECC200 from the drop-down list. The possible values are **Auto**, **ASCII**, **C40**, **Text**, **X12**, **EDIFACT**, or **Base256**.

Ecc000_140 Symbol Size: Select the size of the ECC000_140 barcode symbol from the drop-down list.

Structured Append: Select whether the barcode symbol is part of the structured append symbols. The possible values are **True** or **False**.

Structure Number: Enter the structure number of the barcode symbol within the structured append symbols.

File Identifier: Enter the file identifier of a related group of the structured append symbols. If you set the value to 0, the file identifier symbols are calculated automatically.

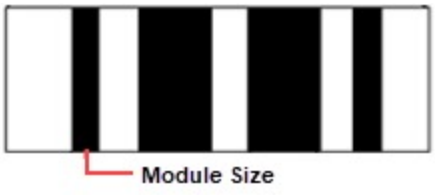
Encoding: Select the barcode encoding from the drop-down list.

EAN128FNC1 Options

EAN128FNC1 options are available for the EAN128FNC1 barcode style.

DPI: Specify the printer resolution.

Module Size: Enter the horizontal size of the barcode module.



Bar Adjust: Enter the adjustment size by dot units, which affects the size of the module and not the entire barcode.

GS1DataMatrix Options

GS1DataMatrix options are available for the GS1DataMatrix barcode style.

EccMode: Select the Ecc mode from the drop-down list. The possible values are **ECC000**, **ECC050**, **ECC080**, **ECC100**, **ECC140** or **ECC200**.

Ecc200 Symbol Size: Select the size of the ECC200 symbol from the drop-down list. The default value is **SquareAuto**.

Ecc200 Encoding Mode: Select the encoding mode for ECC200 from the drop-down list. The possible values are **Auto**, **ASCII**, **C40**, **Text**, **X12**, **EDIFACT**, or **Base256**.

Ecc000_140 Symbol Size: Select the size of the ECC000_140 barcode symbol from the drop-down list.

Structured Append: Select whether the barcode symbol is part of the structured append symbols. The possible values are **True** or **False**.

Structure Number: Enter the structure number of the barcode symbol within the structured append symbols.

File Identifier: Enter the file identifier of a related group of the structured append symbols. If you set the value to 0, the file identifier symbols are calculated automatically.

Encoding: Select the barcode encoding from the drop-down list.

GS1QRCode Options

ErrorLevel: Select the error correction level for the barcode from the drop-down list. Valid values are **M**, **L**, or **Q**. The available Error Level values change depending on the version you select.

Version: Enter the version of the MicroQRCode barcode style. Valid values are **M1**, **M2**, **M3**, or **M4.** Version M4 stores maximum amount of data.

Mask: Select the pattern for the barcode masking from the drop-down list. Valid values are **Mask00**, **Mask01**, **Mask10**, or **Mask11**.

Encoding: Select the barcode encoding from the drop-down list.

MaxiCode Options

MaxiCode option is available for the MaxiCode barcode.

Mode: Select the mode of the MaxiCode barcode. The available values are Mode2 to Mode6.

MicroPDF417 Options

MicroPDF417 options are available for the MicroPDF417 barcode style.

Compaction Mode: Select the type of the compaction mode from the drop-down list. The possible values are **Auto**, **TextCompactionMode**, **NumericCompactionMode,** or **ByteCompactionMode**.

Version: Select the version from the drop-down box to set the symbol size.

Segment Index: The segment index of the structured append symbol. The valid value is from 0 to 99998, and less than the value in Segment Count.

Segment Count: The segment count of the structured append symbol. The valid value is from 0 to 99999.

File ID: The file id of the structured append symbol. The valid value is from 0 to 899.

MicroQRCode Options

MicroQRCode options are available for the MicroQRCode barcode style.

ErrorLevel: Select the error correction level for the barcode from the drop-down list. Valid values are **M**, **L**, or **Q**. The available Error Level values change depending on the version you select.

Version: Enter the version of the MicroQRCode barcode style. Valid values are **M1**, **M2**, **M3**, or **M4.** Version M4 stores maximum amount of data.

Mask: Select the pattern for the barcode masking from the drop-down list. Valid values are **Mask00**, **Mask01**, **Mask10**, or **Mask11**.

Encoding: Select the barcode encoding from the drop-down list.

PDF417 Options

PDF417 options are available for the Pdf417 barcode style.

Columns: Enter column numbers for the barcode. Values for this property range from 1 to 30. The default value is -1 which automatically determines column numbers.

Rows: Enter row numbers for the barcode. Values range between 3 and 90. The default value is -1 which automatically determines row numbers.

ErrorLevel: Enter 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.

PDF 417 Barcode Type: Select the PDF417 barcode type from the drop-down list. The possible values are **Normal** or **Simple**. Simple is the compact type in which the right indicator is neither displayed nor printed.

QRCode Options

QRCode options are available for the QRCode barcode style.

Model: Select the model for the QRCode barcode style from the drop-down list. The possible values are **Model1**, the original model or **Model2**, the extended model. For **GS1QRCode**, Model1 is not supported.

ErrorLevel: Select the error correction level for the barcode from the drop-down list. The possible values are **L** (7% restorable), **M** (15% restorable), **Q** (25% restorable), and **H** (30% restorable). The higher the percentage, the larger the barcode becomes.

Version: Enter the version of the QRCode barcode style. 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.

Mask: Select the pattern for the barcode masking from the drop-down list. 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) mod 2 = 0
- Mask001 i mod 2 = 0
- Mask010 j mod 3 = 0
- Mask011 (i+j) mod 3 = 0
- Mask100 ((i div 2)+(j div 3)) mod 2 = 0
- Mask101 (ij) mod 2 + (ij) mod 3 = 0
- Mask110 ((ij) mod 2 +(ij) mod 3) mod 2 = 0
- Mask111 ((ij)mod 3 + (i+j) mod 2) mod 2 = 0

Use Connection: Select whether to use the connection for the barcode. The possible values are **True** or **False**. This property is used in conjunction with the ConnectionNumber property. This property does not apply to the GS1QRCode barcode.

ConnectionNumber: Enter the connection number for the barcode. 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. This property does not apply to the GS1QRCode barcode.

Encoding: Select the barcode encoding from the drop-down list.

RssExpandedStacked Options

RssExpandedStacked options are available for the RSSExpandedStacked barcode style.

Row Count: Enter the number of the barcode stacked rows.

UPC Supplementary Options

Supplementary options are available for UPC_A, UPC_E0, UPC_E1, EAN_13, and EAN_8 barcode styles.

Supplement Value: Enter the expression to set the value of the barcode supplement.

Caption Location: Select the location for the supplement caption from the drop-down list. The possible values are **None**, **Above**, or **Below**.

Supplement Bar Height: Enter the bar height for the barcode supplement.

Supplement Spacing: Enter the spacing between the main and supplement barcodes.

Barcode Dialog Properties

You can set the Barcode properties in the Barcode dialog. To open it, with the Barcode selected in the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the barcode that is unique within the report. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Value: Enter an expression or a static label, or choose a field expression from the drop-down list. You can access the expression editor by selecting **<Expression...>** in the list. The value of this expression or text is used to render the barcode in the report.

Invalid Barcode Text: Enter a message to display if the barcode contains invalid values (content, character, length).

Caption

Location: Select whether to display the caption above or below the barcode, or select **None** to display the barcode without a caption.

Text Alignment: Select the horizontal alignment of the caption. The default value of **General** centers the caption.

Barcode Settings

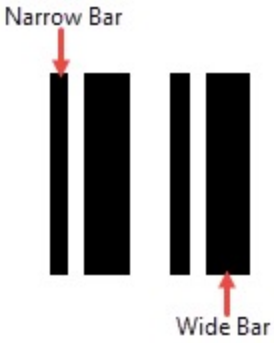
Symbology: Enter the type of barcode to use. ActiveReports supports all of the most popular symbologies.

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

When you choose a symbology which offers supplemental options, the additional options appear after the general collection of properties.

Bar Height: Enter a value in inches (for example, .25in) for the height of the barcode.

Narrow Bar Width (also known as X dimension): Enter a value in points (for example, 0.8pt) for the width of the narrowest part of the barcode. Before using an extremely small value for this width, ensure that the scanner can read it.



> **Tip**: For accurate scanning, the quiet zone should be ten times the Narrow Bar Width value.

Narrow Width Bar Ratio: Enter a value to define the multiple of the ratio between the narrow and wide bars in symbologies that contain bars in only two widths. For example, if it is a 3 to 1 ratio, this value is 3. Commonly used values are 2, 2.5, 2.75, and 3.

QuietZone

A quiet zone is an area of blank space on either side of a barcode that tells the scanner where the symbology starts and stops.

Left: Enter a size in inches of blank space to leave to the left of the barcode.

Right: Enter a size in inches of blank space to leave to the right of the barcode.

Top: Enter a size in inches of blank space to leave at the top of the barcode.

Bottom: Enter a size in inches of blank space to leave at the bottom of the barcode.

> **Note**: The units of measure listed for all of these properties are the default units of measure used if you do not specify. You may also specify **cm**, **mm**, **in**, **pt**, or **pc**.

Checksum

A checksum provides greater accuracy for many barcode symbologies.

Compute Checksum: Select whether to automatically calculate a checksum for the barcode.

> **Note**: If the symbology you choose requires a checksum, setting this value to **False** has no effect.

For more information, see the section on **Barcode Specific Properties**.

Appearance

Font

Family: Select a font family name or a theme font.

Size: Choose the size in points for the font or use a theme.

Style: Choose **Normal** or **Italic** or select a theme.

Weight: Choose from **Lighter**, **Thin**, **ExtraLight**, **Light**, **Normal**, **Medium**, **SemiBold**, **Bold**, **ExtraBold**, **Heavy**, or **Bolder**.

Color: Choose a color to use for the text.

Decoration: Choose from **None**, **Underline**, **Overline**, or **LineThrough**.

Border

Style: Select a style for the border.

Width: Enter a value in points to set the width of the border.

Color: Select a color to use for the border, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Background

Color: Select a color to use for the background, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Format

Format code: Select one of the common numeric formats provided or use a custom .NET formatting code to format dates or numbers. For more information, see MSDN's Formatting Types topic.

Amount of space to leave around report control

Top margin: Set the top padding in points.

Left margin: Set the left padding in points.

Right margin: Set the right padding in points.

Bottom margin: Set the bottom padding in points.

Rotation: Choose **None**, **Rotate90Degrees**, **Rotate180Degrees**, or **Rotate270Degrees**.

Visibility

Initial visibility

- **Visible:** The barcode is visible when the report runs.
- **Hidden:** The barcode is hidden when the report runs.
- **Expression:** Use an expression with a Boolean result to decide whether the barcode is visible.

Visibility can be toggled by another report item: Select this check box to display a toggle icon next to another report item. This enables the drop-down box where you can select the TextBox control that users can click to show or hide this barcode in the viewer.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this barcode. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Data Output

Element Name: Enter a name to be used in the XML output for this barcode.

Output: Choose **Auto**, **Yes**, or **No** to decide whether to include this barcode in the XML output. **Auto** exports the contents of the barcode only when the value is not a constant.

Render as: Choose **Auto**, **Element**, or **Attribute** to decide whether to render barcodes as Attributes or Elements in the exported XML file. **Auto** uses the report's setting for this property.

Bullet

You can use the Bullet control in your Page and RDLX reports. With this control, you can take a single value, the year-to-date revenue, for example, and compare it to a target value that you define in the control's properties. You can also define the beginning of the graph as the worst value and the end of the graph as the best value. To make the data visualization even more intuitive, you can also define a qualitative range (bad, satisfactory and good) for segments on the bullet graph and immediately see the position of the key measure within the bullet graph range.

Multiple bullet graphs can be combined into a data region, a table for example, to show single values side by side. You can orient bullets horizontally or vertically, and put them together as a stack to analyze several data dimensions at once. Bullet graphs are a great way to show progress towards a certain goal and serve as an alternative to dashboard gauges and meters.

Structure

The **Value** of the bullet defines the key measure displayed on the graph.

The **Target Value** defines a target for the Value to be compared to. You can select the target shape from a line, dot or square, and define its color and other properties.

The **Worst Value** and the **Best Value** define the value range on the graph.

In the **Range1Boundary** (bad/satisfactory boundary) and **Range2Boundary** (satisfactory/good boundary), you can optionally set the segments on the graph as qualitative ranges indicating bad, satisfactory and good sections.

Important Properties

By clicking on the Bullet control, you can set its properties in the Properties window.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
BestValue	Enter a value or expression to define the highest value on the bullet graph.
Interval	Set the interval at which you want to show tick marks.
Range1Boundary	Enter a value or expression to define the boundary between bad and satisfactory values.
Range2Boundary	Enter a value or expression to define the boundary between satisfactory and good values.
TargetValue	Define a target for the Value to be compared to.
TargetShape	Choose the shape of the target from Line, Dot, or Square. By default, the shape is set to Line.
TargetStyle	Choose the border color from Color Picker and set the width of the target value marker. This property applies only when the **Shape** is set to Line.
TickMark	Select the position of marks on the bullet control from None, Inside, or Outside.
Value	Enter a value or expression to use as the bullet value.
ValueColor	Choose a color for the defined value.
WorstValue	Enter a value or expression to define the lowest value on the bullet graph.

Bullet Dialog Properties

You can set the Bullet properties in the Bullet dialog. To open it, with the Bullet selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the Bullet that is unique within the report. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Data

Value: Enter an expression to use as the bullet value.

Target Value: Enter an expression to use as the target value of the bullet graph.

Appearance

Bullet Graph Orientation

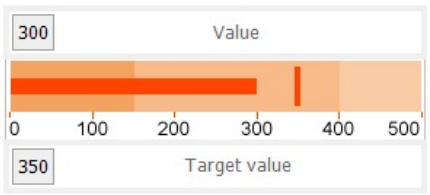
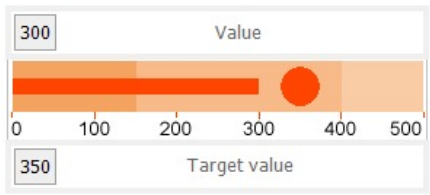
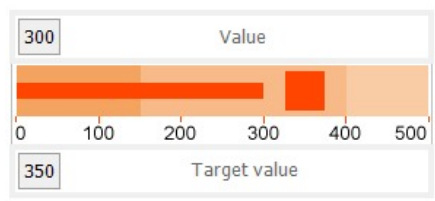
Horizontal: Select to display a horizontal bullet graph.

Vertical: Select to display a vertical bullet graph.

Value Style

Color: Select a color to use for the value marker, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color. The default value is **Black**.

Target Style

Line	Dot	Square
		

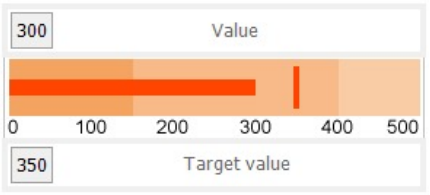
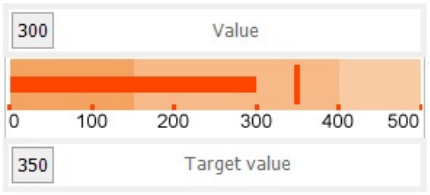
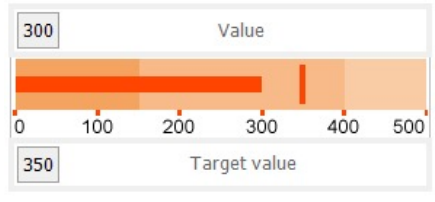
Target Type: Choose **Line**, **Dot** or **Square**. The default value is **Line**.

Color: Select a color to use for the target value marker, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color. The default value is **Black**.

Width: Enter a value in points to set the width of the target value marker. The default value is **3pt**.

> **Note:** The **Width** setting applies only when the **Target Type** is set to **Line**.

Tick Marks

None	Inside	Outside
		

Position: Choose **None**, **Inside** or **Outside**. The default value is **Outside**.

Color: Select a color to use for the tick marks, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color. The default value is **LightGray**.

Width: Enter a value in points to set the width of the tick marks. The default value is **1pt**.

Interval between tick marks: Set the interval at which you want to show tick marks.

Ranges

Worst Value: Enter a value or expression to define the lowest value on the graph.

Bad/Satisfactory Boundary: Enter a value or expression to define the boundary between bad and satisfactory values.

Display 3 Sections: Select this check box to show three separate value ranges (bad, satisfactory, and good) instead of two (bad and satisfactory). This enables the Satisfactory/Good Boundary.

Satisfactory/Good Boundary: Enter a value or expression to define the boundary between satisfactory and good values.

Best Value: Enter a value or expression to define the highest value on the graph.

Labels

Display Labels: Select this check box to display axis labels for the bullet graph. Selecting this box enables the rest of the properties on this page.

Format: Select one of the provided format codes or use a custom .NET formatting code to format dates or numbers. For more information, see MSDN's Formatting Types topic.

Font

Family: Choose the font family name. The default value is **Arial**.

Size: Choose the size in points for the font. The default value is **10pt**.

Style: Choose **Regular**, **Bold**, **Italic**, **Underline** or **Strikeout**. The default value is **Regular**.

Color: Select a Web or custom color for the font. The default value is **Black**.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this Bullet. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Visibility

Initial visibility

- **Visible:** The bullet graph is visible when the report runs.
- **Hidden:** The bullet graph is hidden when the report runs.
- **Expression:** Use an expression with a Boolean result to decide whether the bullet graph is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the bullet. The user can click the toggle item to show or hide this bullet.

Data Output

Element Name: Enter a name to be used in the XML output for this Bullet.

Output: Choose **Auto**, **Yes**, **No**, or **Content only** to decide whether to include this Bullet in the XML output. **Auto** exports the contents of the bullet graph only when the value is not a constant.

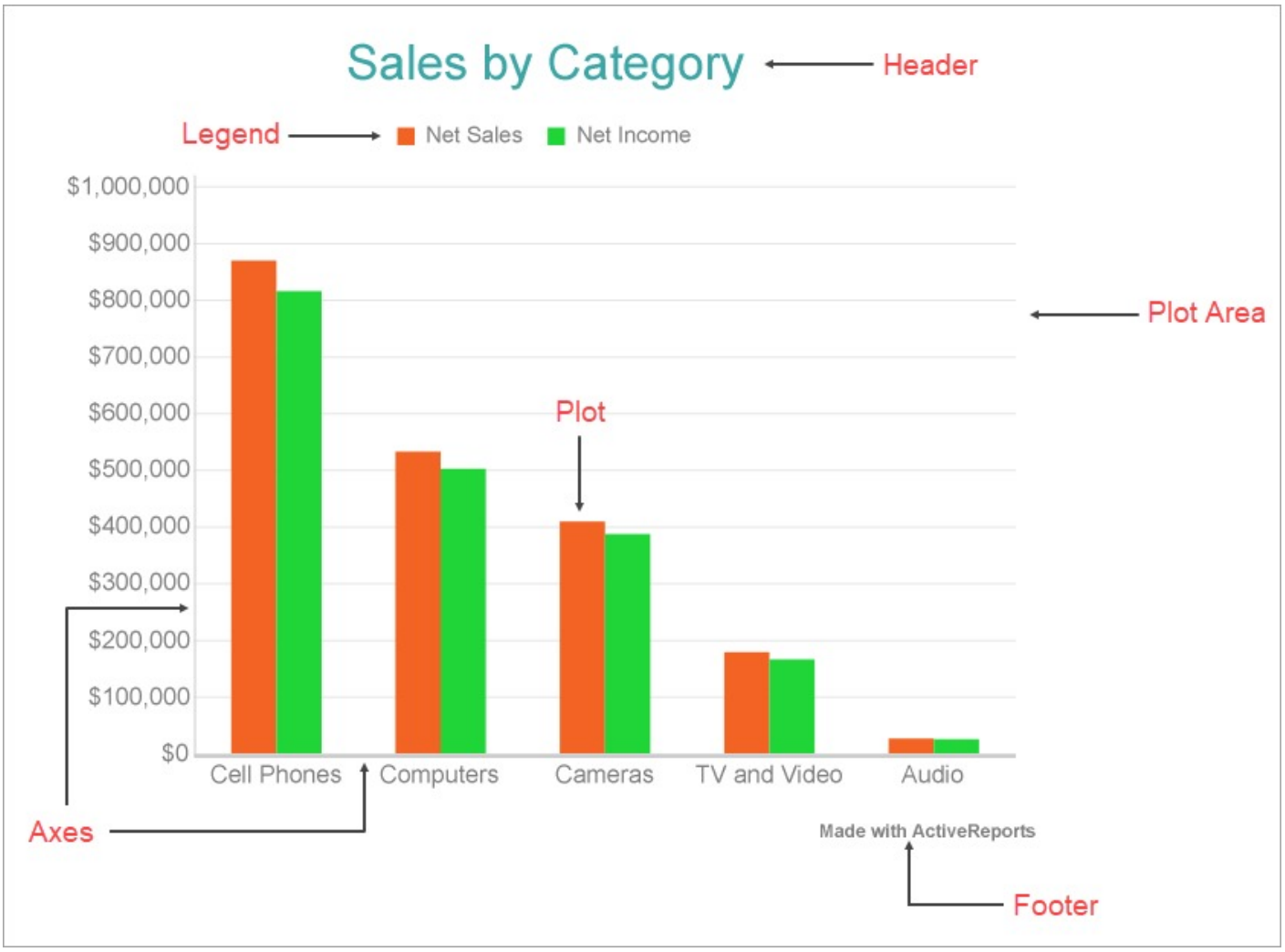
Chart

A **Chart** data region helps you to visualize your data graphically. It is a data visualization control primarily based on slicing data through encodings.

This article provides information about all the chart elements. Further subtopics provide details on every other aspect of charts that will help you design any chart supported by ActiveReports..

Chart Elements

The below image illustrates the elements that make up the **Chart** data region. These elements help you analyze the visual information and interpret numerical and relational data in a chart.



You can view all the chart elements in the **Report Explorer** and access properties through the Chart Smart Panels and Properties window. For more information on chart elements, see the below sections.

Plot Area

The plot area is the area of the chart where the data is plotted graphically. It also includes the chart axes and legend.

Plot

The plot is where the actual chart encodings are configured and converted to visual attributes, that is, graphs. It also includes other configuration settings such as applying Rules, adding Trendlines, setting clipping mode, offset, etc. For more information, see the Plots topic.

Axes

Typically, a chart consists of two axes that are used to plot the data - vertical axis (also known as Y-axis) and horizontal axis (also known as X-axis). You can change the axis type, hide the axis label or title, customize the line appearance, etc. For more information, see the Axes topic.

Legend

The legend serves as a key to the specific colors or patterns being used to show series values in the chart. To display legends for each plot on a chart, use VALUESNAME as the color field encoding. For more information, see the Legends topic.

Header

The chart header is used to display a title for the chart. The chart header appears above the plot area. Use the

header settings to control the text formatting, padding, background color, alignment, etc. for a chart header.

Footer

The chart footer is used to add a footer for the chart. The chart footer is displayed below the plot area. Use the footer settings to control the text formatting, padding, background color, alignment, etc. for a chart footer.

Supported Chart Types

Following is the list of supported chart types in Page/RDLX reports. Each of these are discussed under Plots topic.

- Column and Bar Charts
- Area Chart
- Line Chart
- Pie and Doughnut Charts
- Scatter and Bubble Charts
- Radar Scatter and Radial Bubble Charts
- Radar Line Chart
- Radar Area Chart
- Spiral Chart
- Polar Chart
- Gantt Chart
- Funnel and Pyramid Charts
- **Candlestick Chart (on-line documentation)**
- High Low Close Chart
- High Low Open Close Chart
- Range Charts
- Gauge Chart

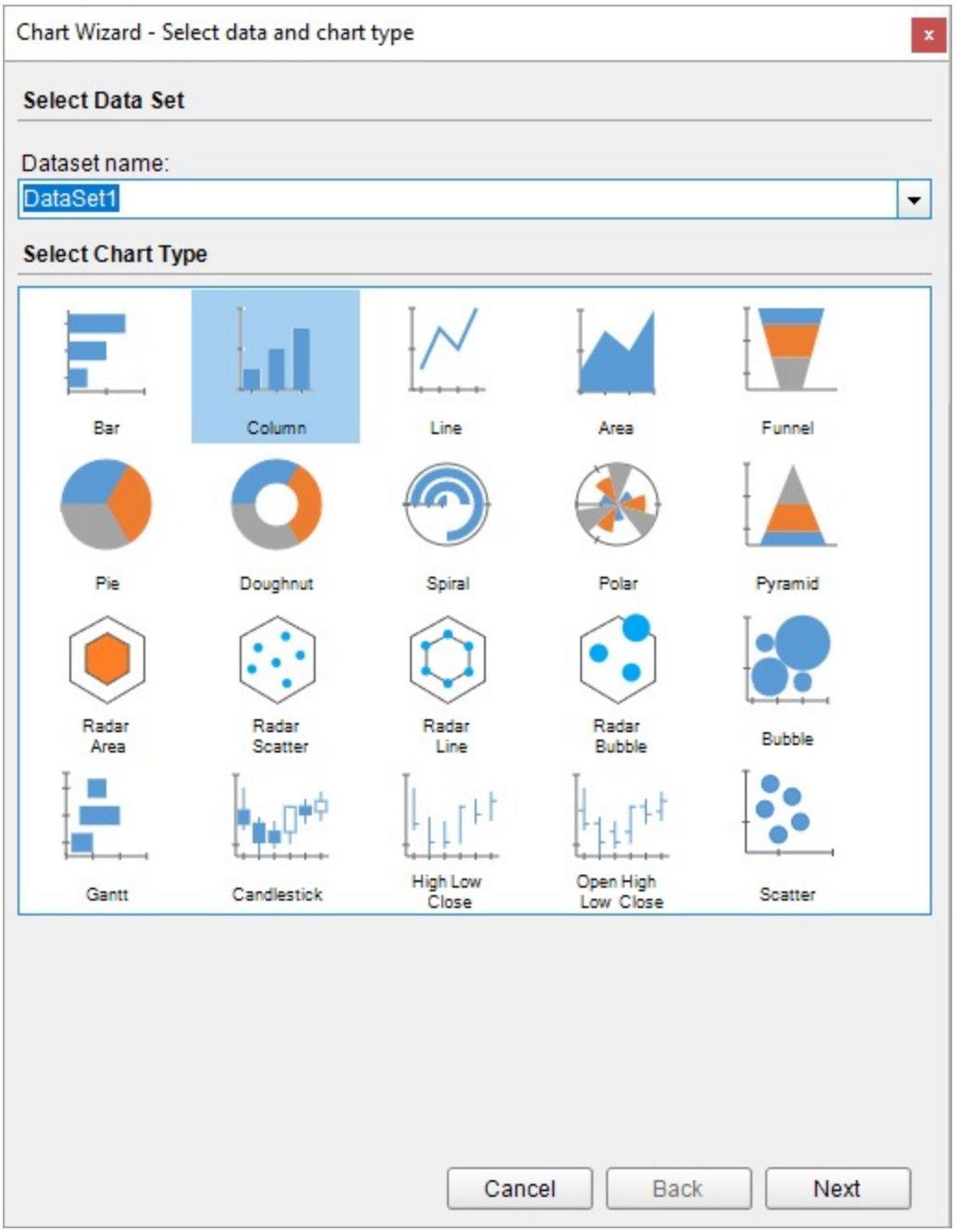
Chart Wizard

The Chart Wizard guides you through the process of creating a chart. When you drag and drop the Chart control onto the report design surface, the Chart Wizard is displayed.

> **Note**: The Chart Wizard is displayed only if a report is bound to a data set. For the information on how to bind a chart to data, see our walkthrough Create Clustered Column Chart.

Chart Wizard - Select data and chart type

The first screen of the Chart Wizard asks you to select a data set and a chart type.

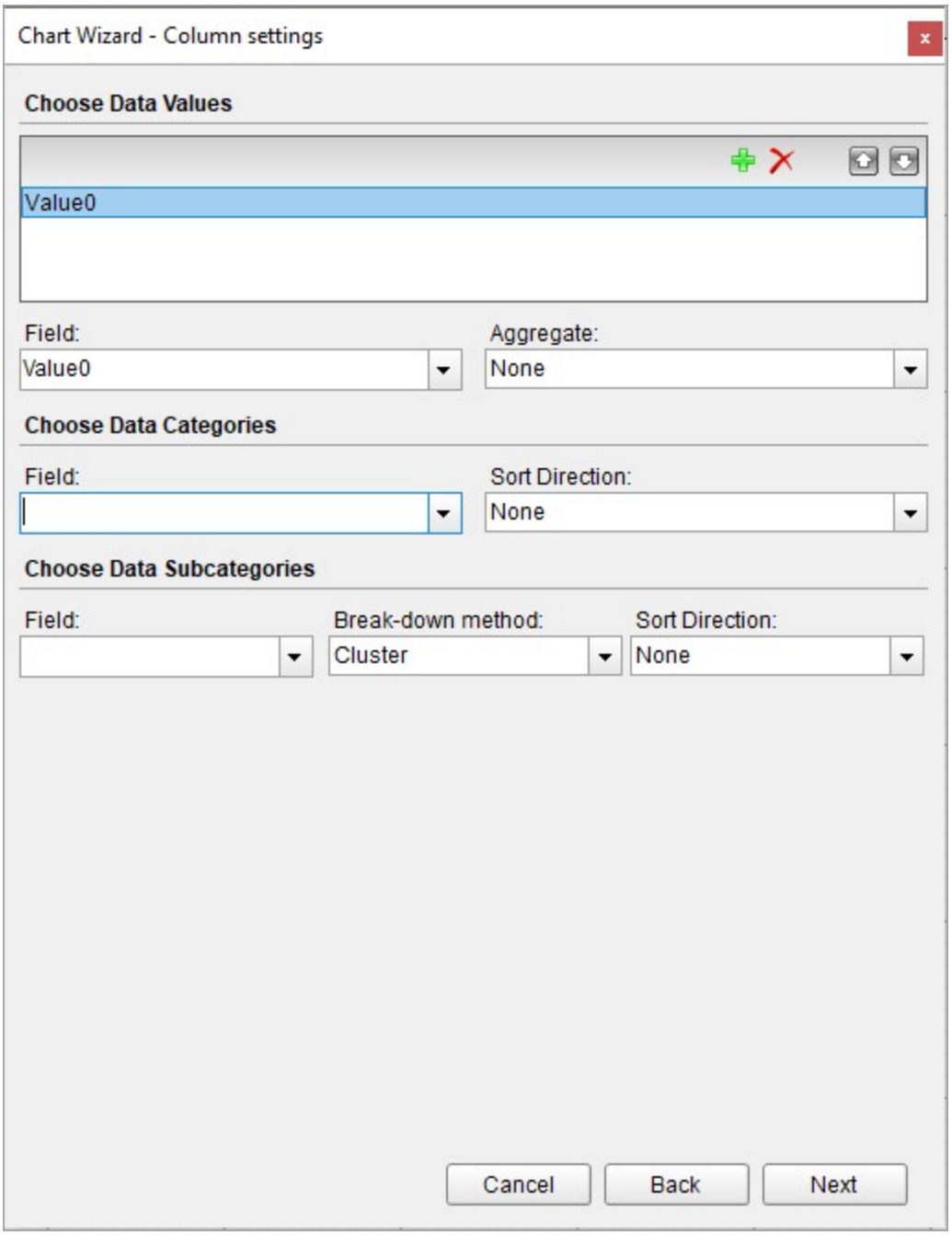


After a chart type is selected, specify the plot properties in the next screen of the wizard.

Chart Wizard - Settings

In the second screen of the Chart Wizard, specify the plot properties to configure the data values, category encodings, and detail encodings.

For Bar, Column, Area, Polar, Spiral



Choose Data Values

Add - adds a data field to the Y-axis.

Remove - removes a data field.

Up - moves a selected data field up.

Down - moves a selected data field down.

Field

The values in the Data Value Field specify the fields to be plotted on Y-Axis. If there is only one value, then the Y-Axis

Title is set to the corresponding field name. If there are multiple values, but no Details encoding is set, then the chart automatically displays the global legend filled with value field names.

Aggregate

Data field aggregate function. An Aggregate function evaluates all the data value fields into a single value. Choose the aggregate function from Average, Count, CountOfAll, List, Min, Max, PopulationStandardDeviation, PopulationVariance, Range, StandardDeviation, Sum, Variance.

Choose Data Categories

Field

The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field. The title of X-Axis is set to the corresponding category field name.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sort direction is automatically set to the Category Field.

Choose Data Subcategories

Field

Adds a subcategory group to the data.

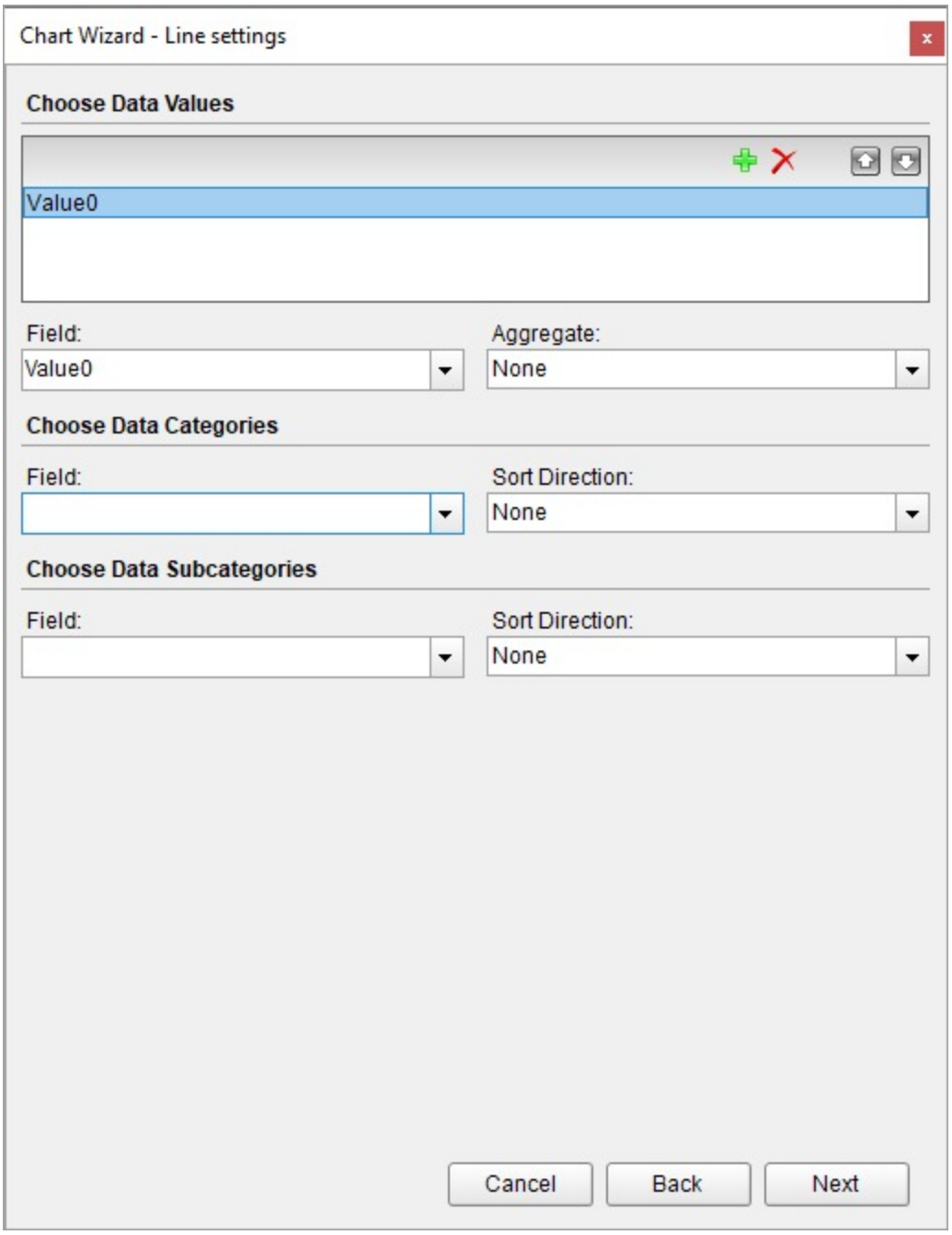
Break-down method

Select the way you want the additional groups to arrange from Cluster, Stacked, or Percentage.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sorting field is automatically set to the subcategory field.

For Line, Radar Line, Radar Area



Choose Data Values

Field

The values in the Data Value Field specify which fields should be plotted. If there is only one value, then the title of Y-Axis is set to the corresponding field name. If there are multiple values, but no Details encoding is set, then the chart automatically displays the global legend filled with value field names.

Aggregate

Data field aggregate function. An Aggregate function evaluates all the data value fields into a single value. Choose the aggregate function from Average, Count, CountOfAll, List, Min, Max, PopulationStandardDeviation, PopulationVariance,

Range, StandardDeviation, Sum, Variance.

Choose Data Categories

Field

The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field. The title of X-Axis is set to the corresponding category field name.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sort direction is automatically set to the category field.

Choose Data Subcategories

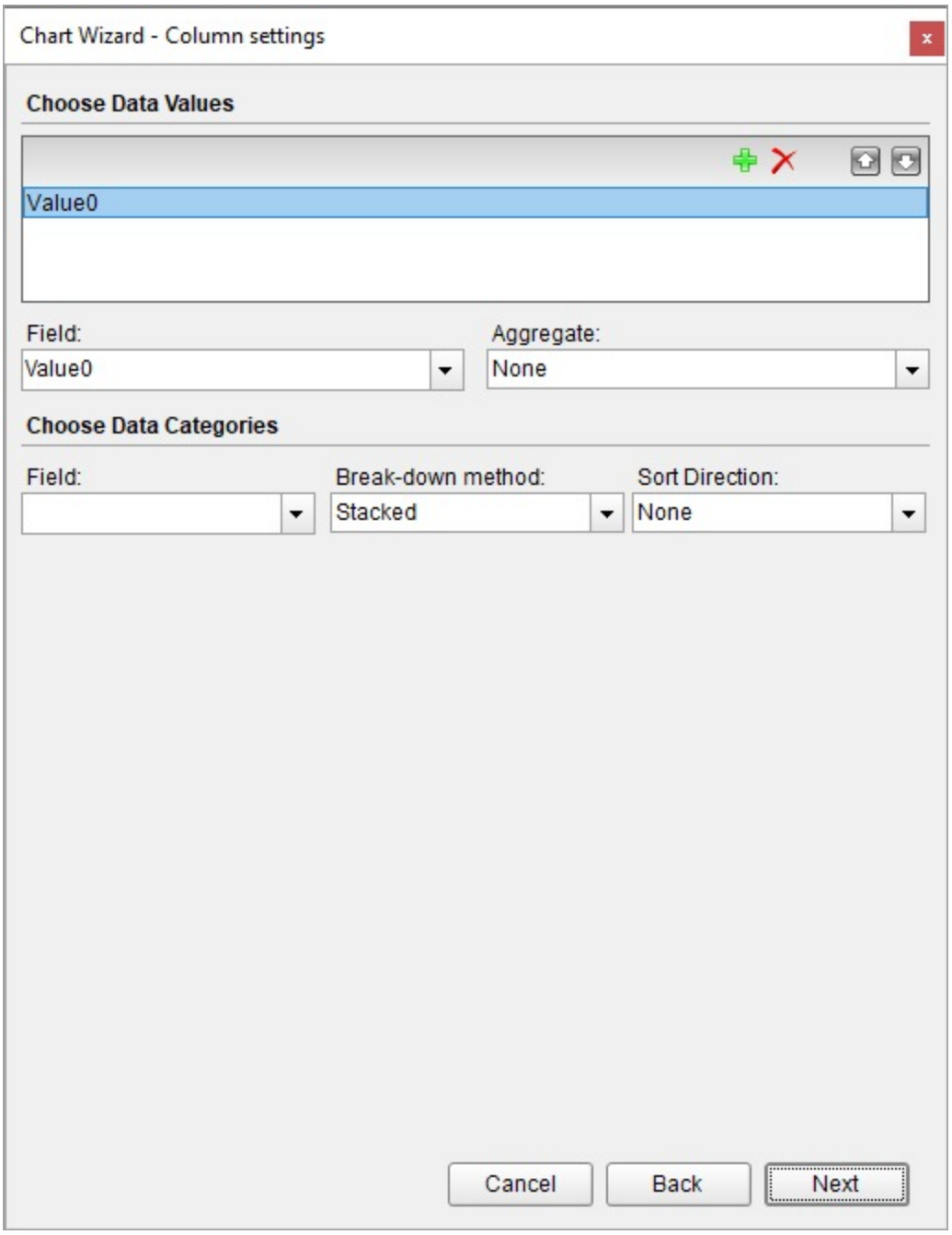
Field

Adds a subcategory group to the data.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sorting field is automatically set to the subcategory field.

For Pie, Donut, Funnel, Pyramid



Choose Data Categories

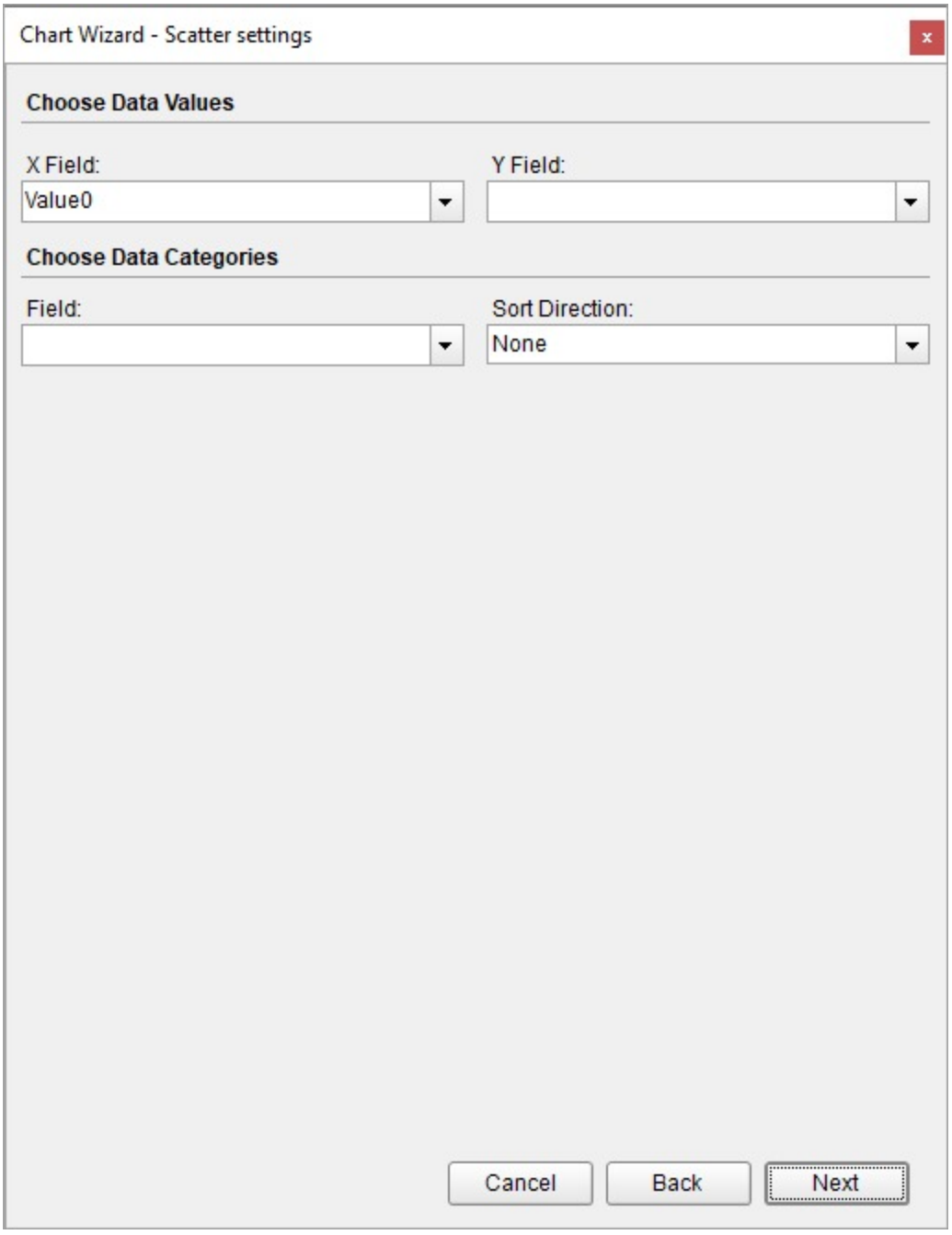
Field

The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field. The title of X-Axis is set to the corresponding category field name.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sort direction is automatically set to the Category Field.

For Scatter, Radar Scatter



Choose Data Values

For scatter plot charts, the values property includes mappings for both X and Y.

X Field

Enter an expression to use as an X value. The title of X-Axis is set to the X Field value.

Y Field

Enter an expression to use as a Y value. The title of Y-Axis is set to the Y Field value.

Choose Data Categories

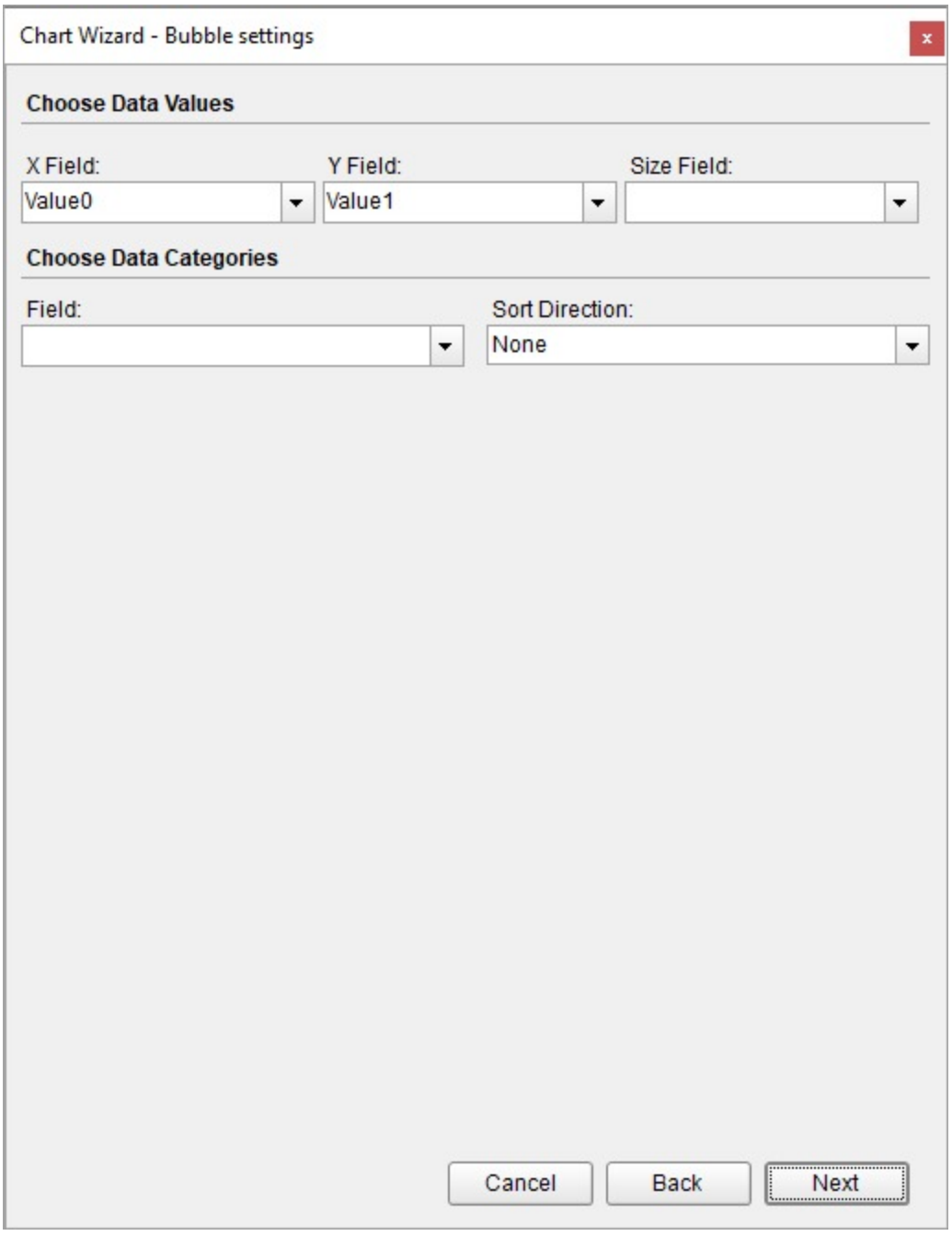
Field

The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sorting field is automatically set to the subcategory field.

For Bubble, Radar Bubble



For scatter plot charts, the values property includes mappings for both X and Y.

Choose Data Values

X Field

Enter an expression to use as an X value. The title of X-Axis is set to the X Field value.

Y Field

Enter an expression to use as a Y value. The title of Y-Axis is set to the Y Field value.

Size Field

Enter an expression to use as the bubble size value.

Choose Data Categories

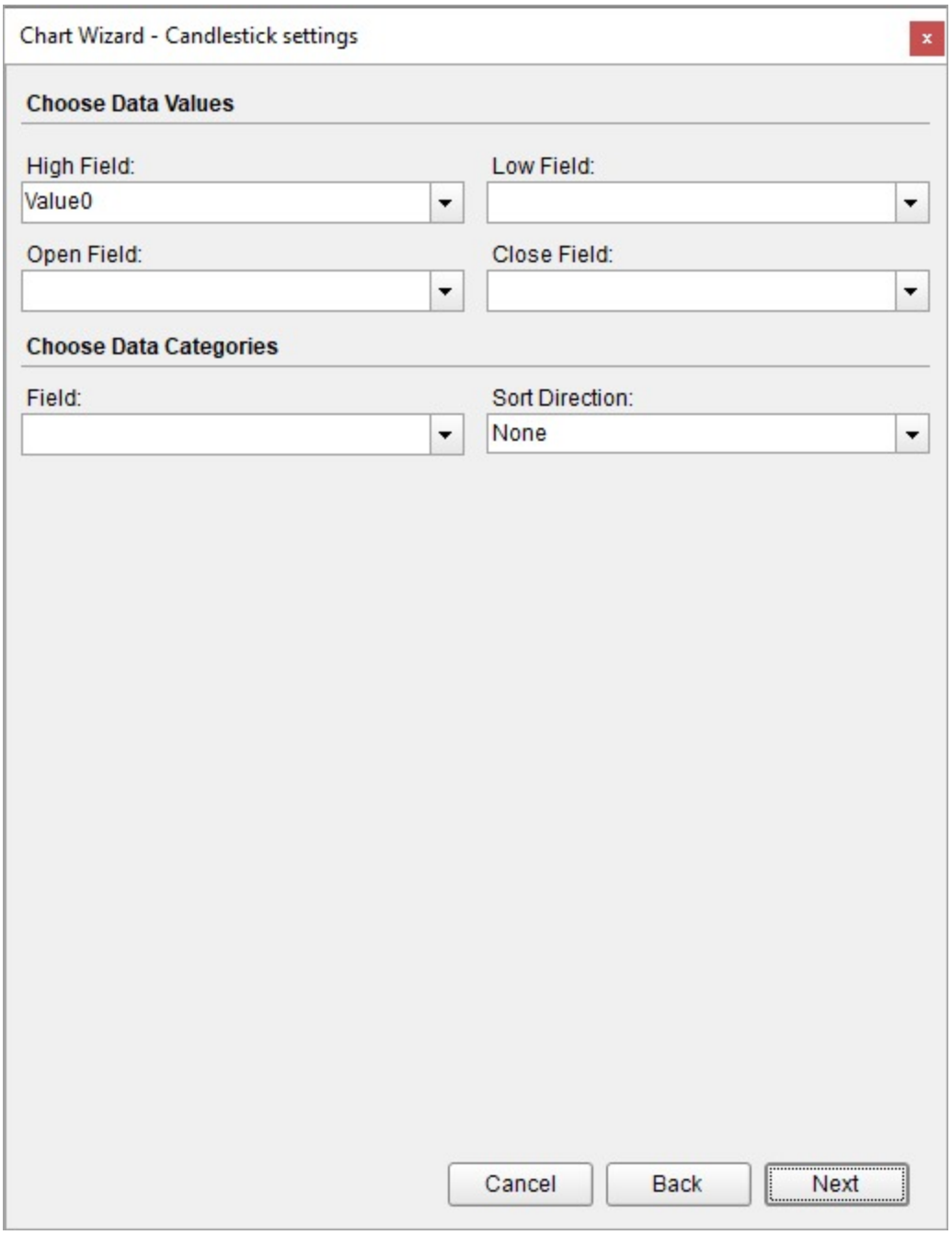
Field

The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sorting field is automatically set to the subcategory field.

For CandleStick, High-Low-Open-Close, High-Low-Close



Choose Data Values

High Field

Enter an expression to use as the high value.

Low Field

Enter an expression to use as the low value.

Open Field (for Open High Low Close and Candlestick chart type)

Enter an expression to use as the open value.

Close Field

Enter an expression to use as the close value.

Choose Data Categories

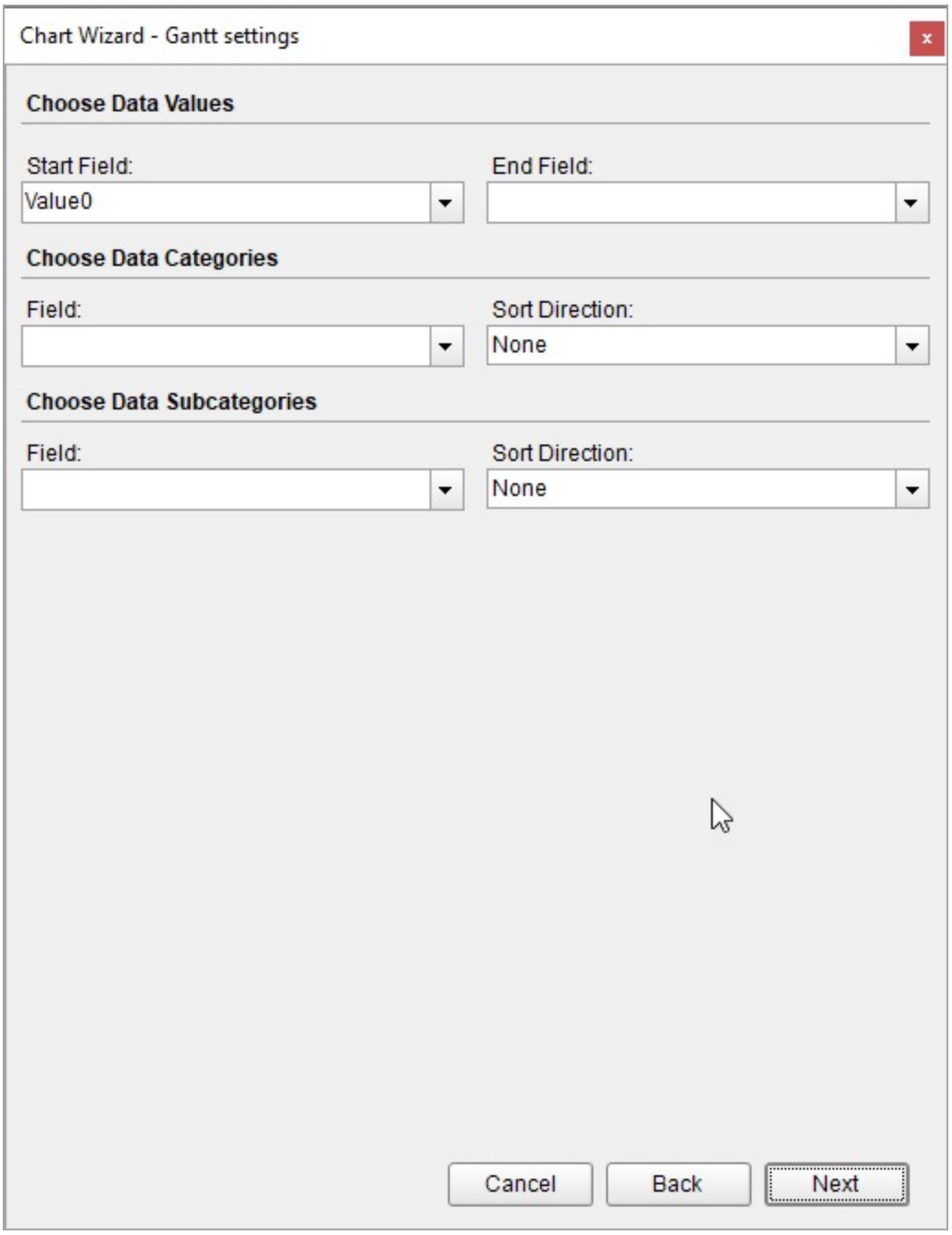
Field

The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field. The title of X-Axis is set to the corresponding category field name.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sort direction is automatically set to the category field.

For Gantt



Choose Data Values

Start Field

An expression to use as the start value or the lower value.

End Field

An expression to use as the end value or the upper value.

Choose Data Categories

Field

The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sort direction is automatically set to the category field.

Choose Data Subcategories

Field

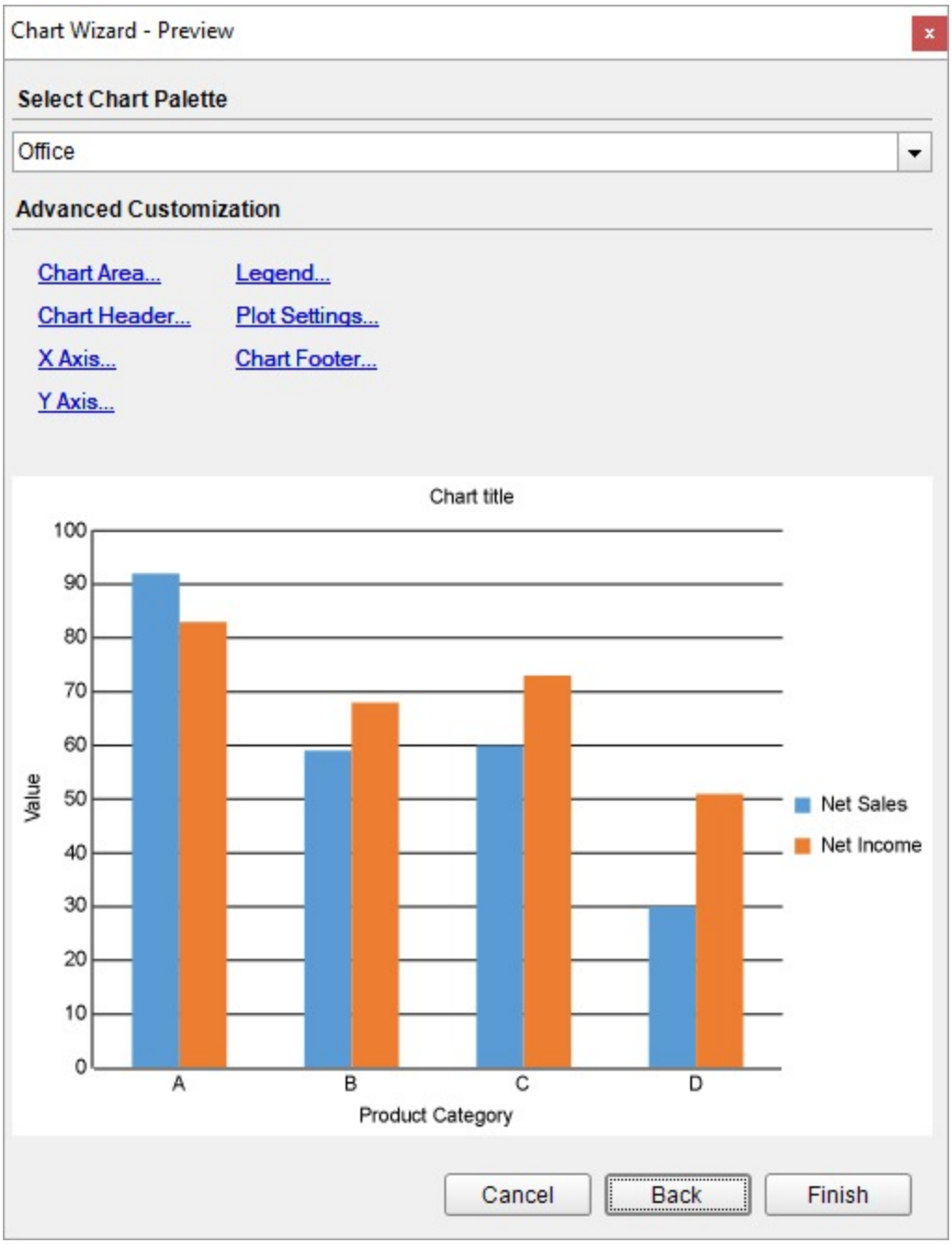
Adds a subcategory group to the data.

Sort Direction

Choose the data sorting direction from None, Ascending, or Descending. The sorting field is automatically set to the subcategory field.

Chart Wizard - Preview

Select a chart palette from the drop-down list. Click **Finish** to add the chart to the report or choose any of the advanced customization options by clicking it.



Select Chart Palette

Select a color palette for the chart from a list of predefined palettes.

Advanced Customization

Clicking any of the advanced customization options opens a corresponding chart smart panel.

- Chart Area..
- Chart Header...
- X Axis...

- Y Axis...
- Legend...
- Plot Settings...
- Chart Footer...

> **Note**: For the Pyramid, Doughnut, Pie, and Funnel chart types, the X Axis and Y Axis customization options are not available.

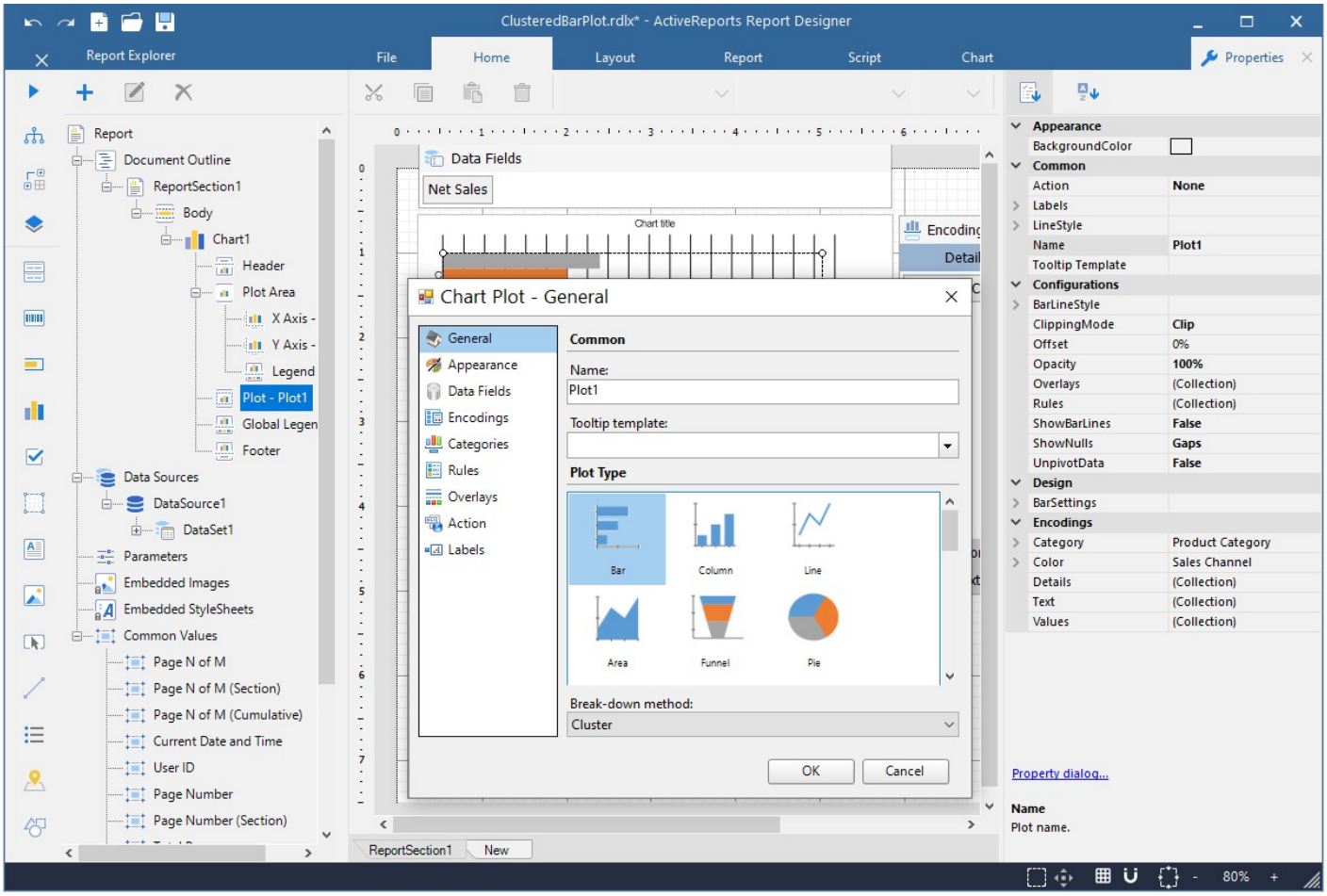
To learn about the chart smart panels, see Chart Smart Panels.

Chart Smart Panels

Smart panels are the dialogs that group commonly used properties of the chart elements using tabs and groups. ActiveReports provides you with the smart panels for these chart elements:

- Chart
- Chart Header and Chart Footer
- Plot
- X Axis and Y Axis
- Legend

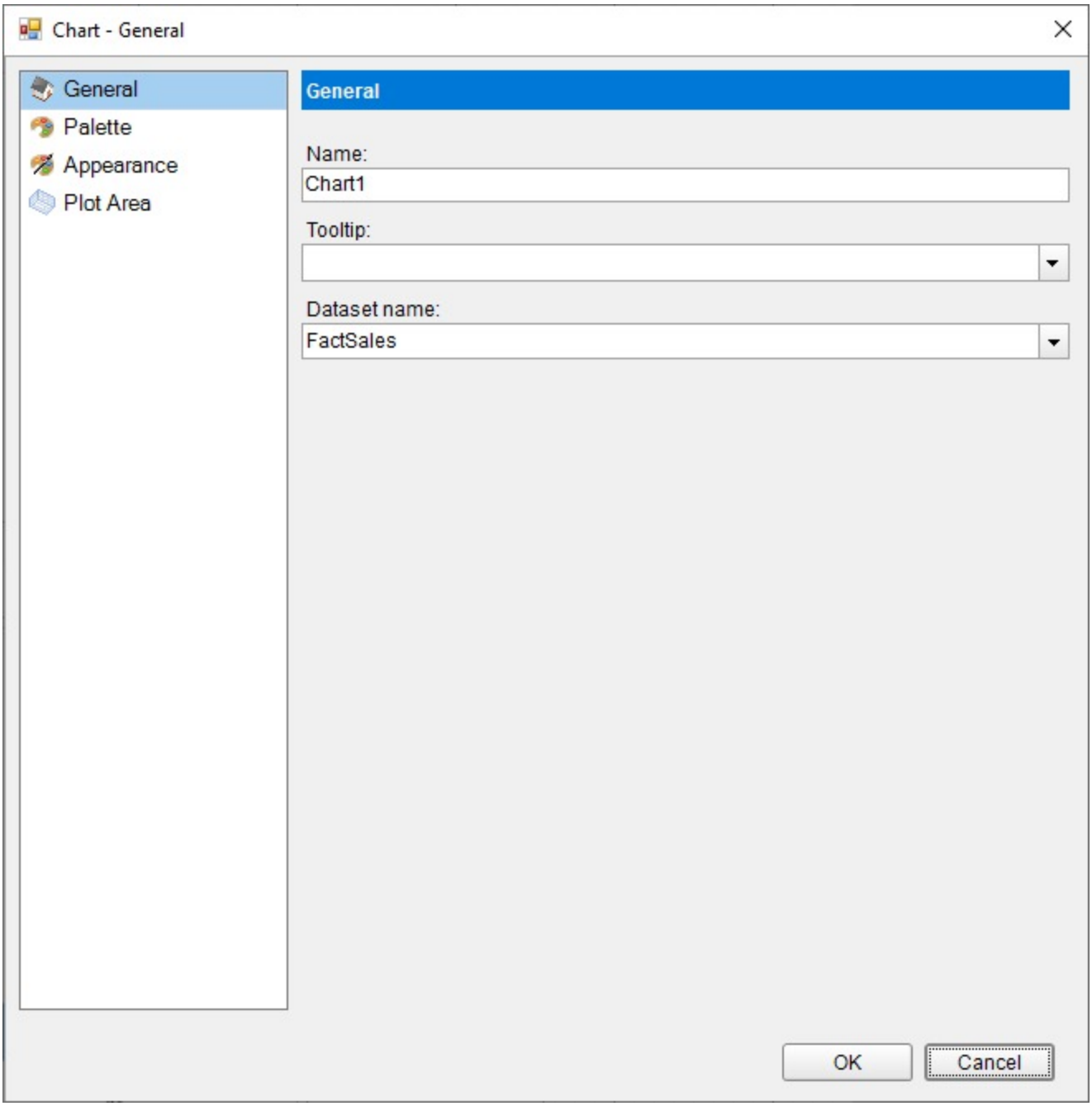
These smart panels can be accessed from the **Report Explorer**. You just need to select the chart element for which you want the smart panel to show and click the **Property dialog** link on the Properties pane.



The smart panels are also available from the last screen of the Chart Wizard for advanced chart customizations.

Chart Smart Panel

The Chart Smart Panel can also be accessed from the context menu on selecting the Chart on the design area.



General

Name: A name for the chart that is unique within the report. This name is displayed in the Document Outline and in XML exports.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Dataset Name: Select a dataset to bind to the chart. The combo box is populated with all of the datasets in the report's dataset collection.

Palette

Select a palette for the chart from the list of pre-defined palettes, including the Custom palette.

Appearance

Border

Style: Select a border style from the list of available options.

Width: Choose a width value between 0.25pt and 20pt for the thickness of the borderlines.

Color: Choose a Web or Custom color to use for the borderlines.

Background Fill Color: Choose a Web or Custom color to fill the background of the chart.

Plot Area

Border

Style: Choose an enumerated style for the **border**.

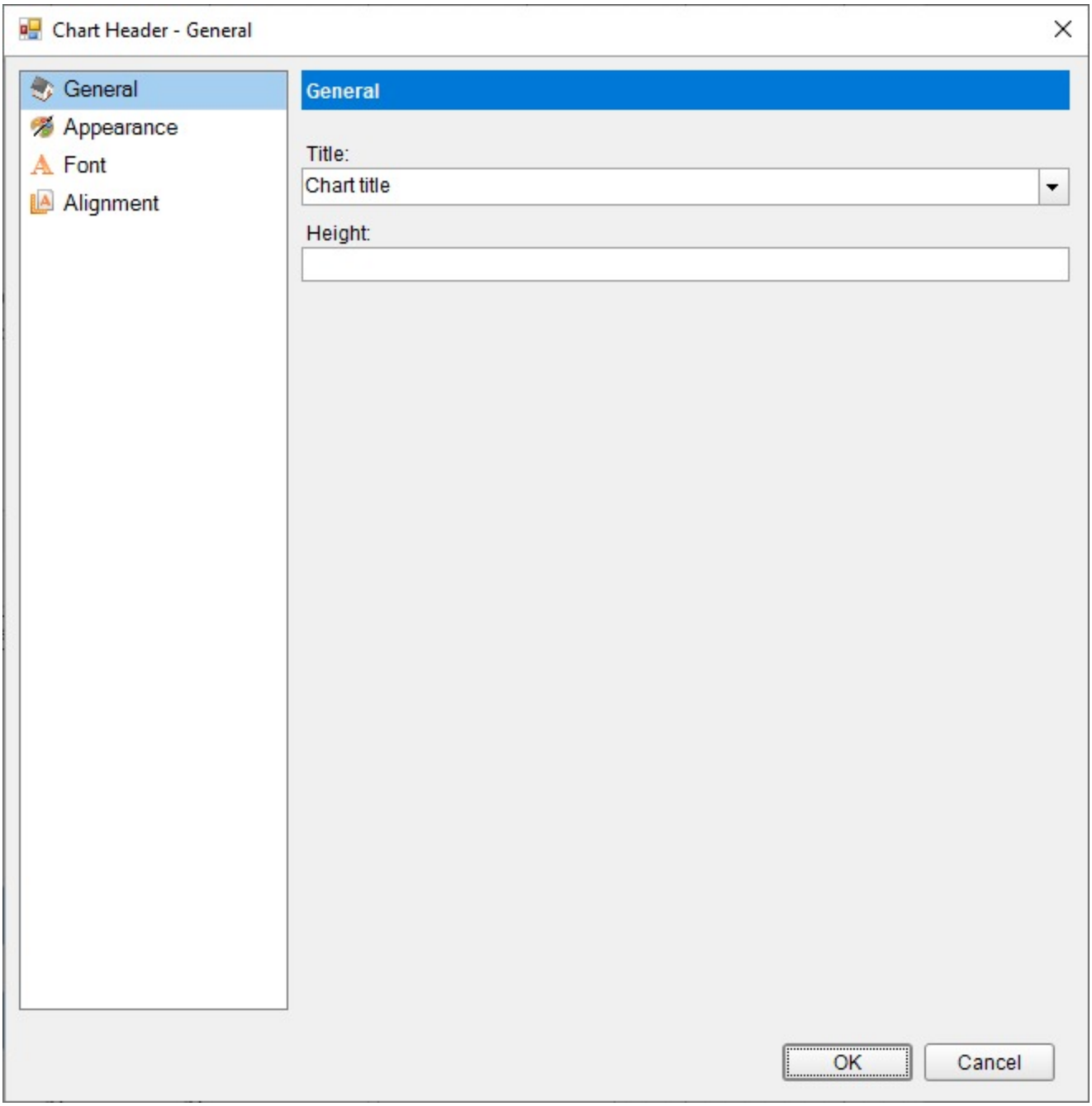
Width: Choose a width value between **0.25pt** and **20pt**.

Color: Select a Web or Custom color.

Background Fill Color: Select a Web or Custom color.

Padding: Set padding values - Top, Left, Right, and Bottom, for the chart.

Chart Header and Chart Footer Smart Panels



General

Text: Enter an expression or text to use for the chart header / footer.

Height: Set the height of the chart header/footer in percent.

Appearance

Border

Style: Select a border style from the list of available options.

Width: Choose a width value between 0.25pt and 20pt for the thickness of the borderlines.

Color: Choose a Web or Custom color to use for the borderlines.

Background Fill Color: Choose a Web or Custom color to fill the background of the chart header/footer.

Font

Family: Choose the font family name for the header/footer text.

Size: Choose the size in points for the header/footer text font.

Style: Choose one from the enumerated styles for the text font.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, LineThrough, and DoubleUnderline.

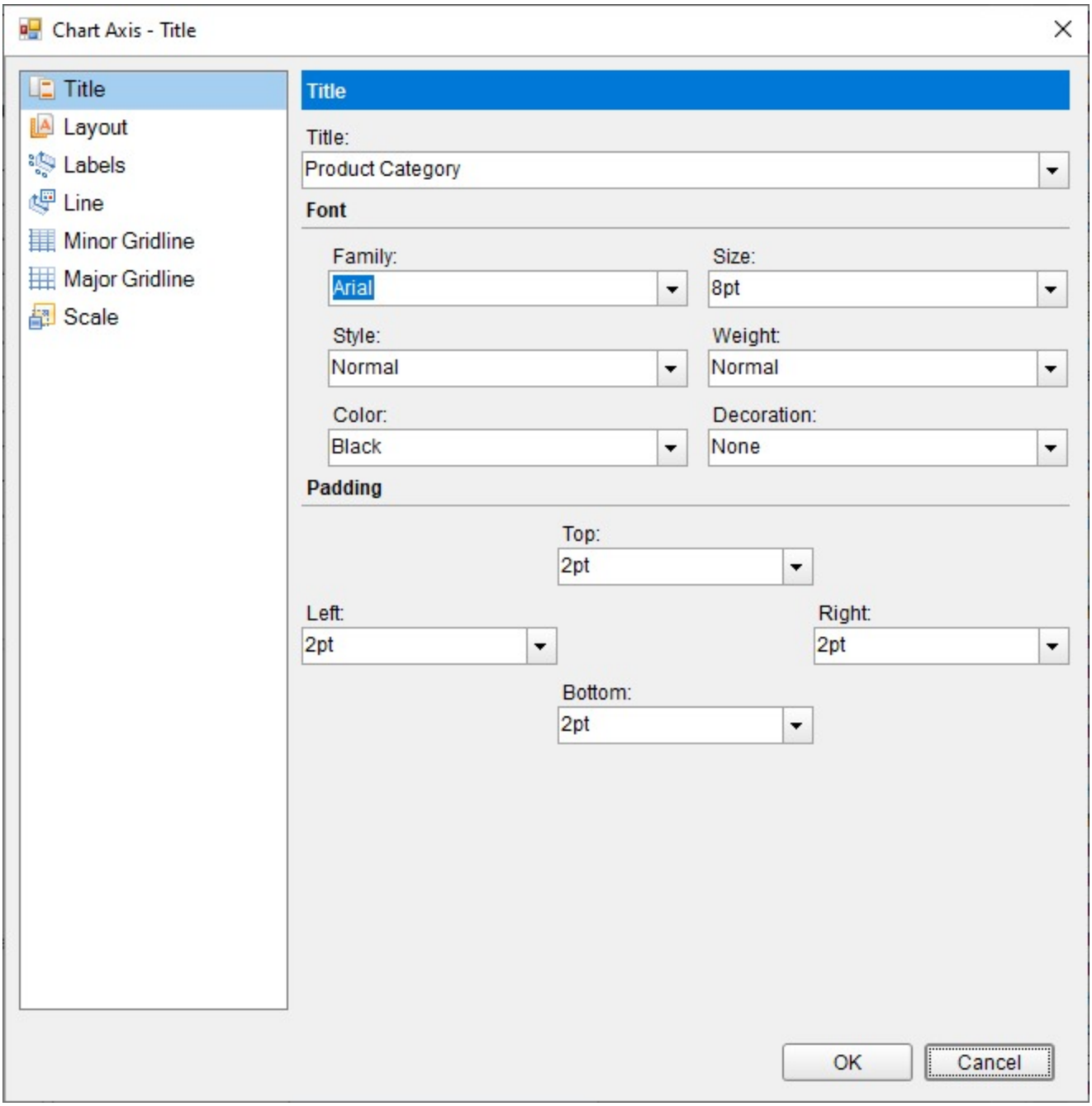
Alignment

Horizontal Alignment: Choose **Left**, **Center**, or **Right**.

Vertical Alignment: Choose **Top**, **Middle**, or **Bottom**.

Padding: Set the padding values - Top, Left, Right, and Bottom, for the chart header/footer.

Chart Axis Smart Panel



General

Axis Type: Select between X and Y axis types.

Plots: Select one or more chart axis plots from the list.

Title

Text: Enter an expression or text to use for the chart axis title.

Font

Family: Choose the font family name for the chart axis title.

Size: Choose the size in points for the chart axis title.

Style: Choose one from the enumerated styles for the text font.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, LineThrough, and DoubleUnderline.

Padding: Set the padding values - Top, Left, Right, and Bottom, for the chart axis title.

Layout

Overlapping Labels Mode: Select Auto or Show to specify whether the labels can be overlapped.

Size

Height: Set the height in percent relative to the overall chart height.

Width: Set the width in percent relative to the overall chart width.

MaxHeight: Set the maximum height in percent relative to the overall chart height.

MaxWidth: Set the width in percent relative to the overall chart width.

Position

Reversed: Select this check box to reverse the direction of the axis -top to bottom, or right to left.

Origin: Value or ordinal number, at which an axis crosses the perpendicular axis.

Position: Select from Far, Near, or None for the label position.

Labels

Show Labels: Select this checkbox to have the axis labels displayed.

Format: The formatting string used to render dates and numbers.

Label Field: Specify a custom value for the chart axis label.

Angle: Rotation angle for the axis label. The angle is measured in degrees with valid values from -90 to 90.

FontFamily: Choose the font family name for the chart axis label.

Size: Choose the size in points for the chart axis label.

Style: Choose one from the enumerated styles for the text font.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, LineThrough, and DoubleUnderline.

Padding: Set the padding values - Top, Left, Right, and Bottom, for the chart axis label.

Line

Show Line: Select this checkbox to have the axis line shown.

Appearance

Color: Select a Web or Custom color.

Width: Specify the width of the axis line in points.

Style: Choose from an enumerated style for the axis line.

Minor Gridline

Grid Interval: Set the interval at which you want to show minor grid lines or tick marks or both.

Grid Appearance

Show Grid: Select this check box to show minor grid lines for the axis.

Color: Select a color for the grid line.

Width: Enter a width value between 0.25pt and 20pt.

Style: Choose one from the enumerated styles for the gird line.

Tick mark appearance

Position: Select the tick mark position from None, Inside, Outside, or Cross.

- **None**: No tick mark is displayed.
- **Inside**: Tick marks are displayed inside the axis.
- **Outside**: Tick marks are displayed outside the axis.
- **Cross**: Tick marks are displayed crossing the axis.

Size: Specify the length of the minor grid tick marks.

Color: Select a Web or Custom color.

Width: Specify the width of the tick mark in points.

Style: Choose from an enumerated style for the minor grid tick mark.

Major Gridline

Grid Interval: Set the interval at which you want to show major grid lines or tick marks or both

Grid Appearance

Show Grid: Select this check box to show major grid lines for the axis.

Color: Select a color for the grid line.

Width: Enter a width value between 0.25pt and 20pt.

Style: Choose one from the enumerated styles for the gird line.

Tick mark appearance

Position: Select the tick mark position from None, Inside, Outside, or Cross.

- **None**: No tick mark is displayed.
- **Inside**: Tick marks are displayed inside the axis.
- **Outside**: Tick marks are displayed outside the axis.
- **Cross**: Tick marks are displayed crossing the axis.

Size: Specify the length of the major grid tick marks.

Color: Select a Web or Custom color.

Width: Specify the width of the tick mark in points.

Style: Choose from an enumerated style for the major grid tick mark.

Scale

Scale Type: Select the axis scale from Linear, Logarithmic, Ordinal, or Percentage.

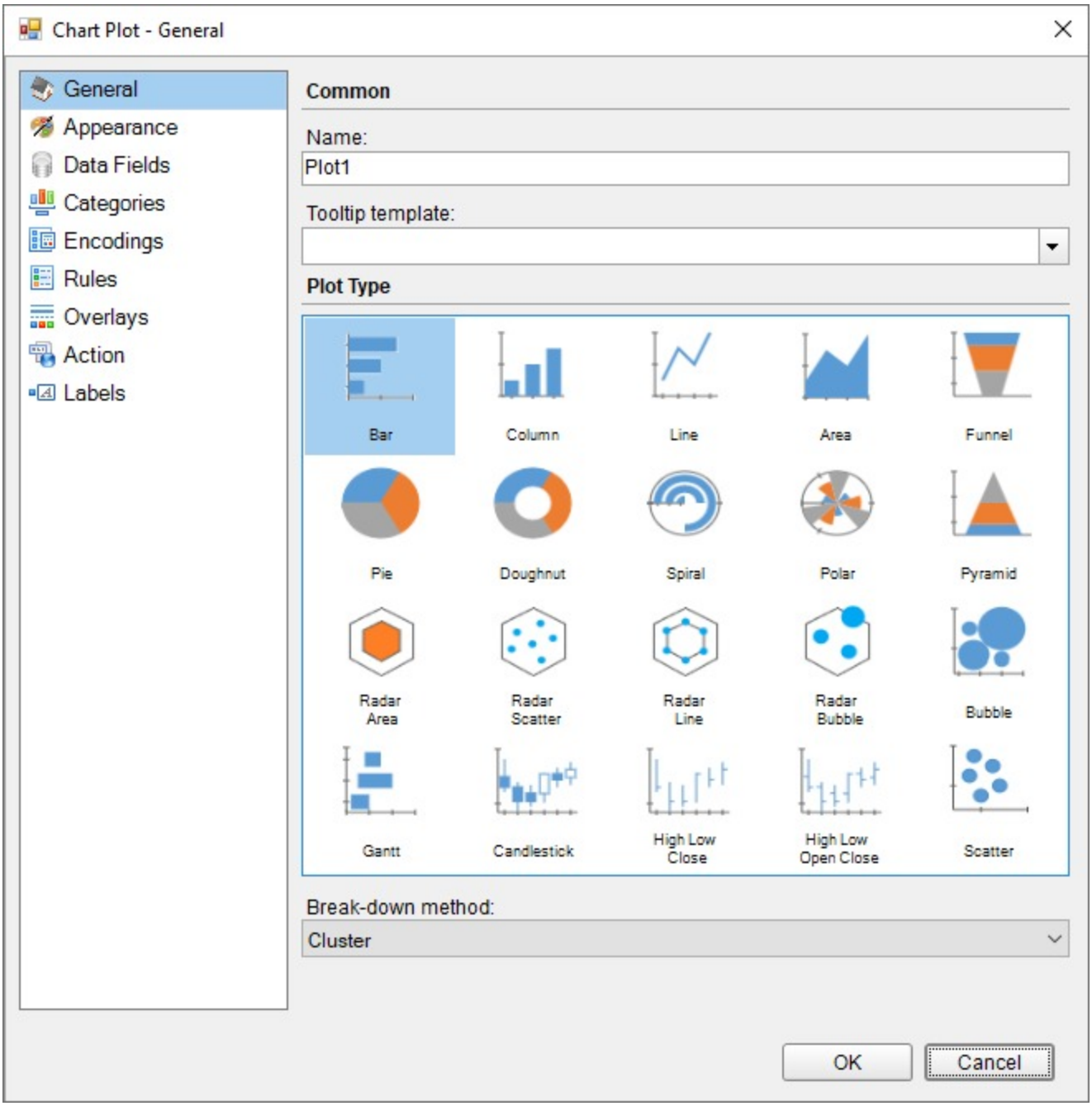
Minimum scale value: Leave this value blank to allow the data to determine the minimum value to use.

Maximum scale value: Leave this value blank to allow the data to determine the maximum value to use.

Logarithmic base: The axis logarithmic base for the logarithmic scale type.

See Axes for more information.

Chart Plot Smart Panel



General

Name: Enter the chart plot name.

Tooltip Template: Select the settings of the tooltip template from the list of predefined chart template tokens or enter your expression.

Plot Type: Select a chart plot type from the following available types:

- Bar
- Column
- Line

- Area
- Funnel
- Pie
- Doughnut
- Spiral
- Polar
- Pyramid
- Radar Area
- Radar Scatter
- Radar Line
- Radar Bubble
- Bubble
- Gantt
- Candlestick
- High Low Close
- Open High Low Close
- Scatter

See Plots topic for more information.

Break-down method: Select from Cluster, Stacked, or Percentage to break down the data values into subcategories and produce additional groups.

Data Fields

Add a data field value by clicking the **Add** icon.

General tab

Aggregate: Data field aggregate function.

Type: Select from Simple or Complex. Set to 'Complex' for Financial charts like CandleStick, High Low Close, Open High Low Close, and Gantt. For other charts, set to 'Simple'.

Values tab

Expression: Select from the list of dataset fields.

Name: Data field name.

Categories

Add a data category by clicking the **Add** icon.

Expression: The category mapping in the Data Categories Field groups the data into multiple categories for each unique value (or numerical range) that exists for the specified field.

Sorting

Sorting field: A field, used for sorting.

Sort direction: Choose the data sorting direction from None, Ascending, or Descending.

Sorting aggregate: Sorting aggregate function.

Encodings

Detail tab

Expression: Select a dataset value or enter an expression for a plot encoding.

Group: Select from None, Cluster, or Stack, the way additional groups should be arranged.

Exclude nulls: Select the checkbox to exclude the null details values in the dataset field.

Sorting field: A field, used for sorting.

Sort direction: Choose the data sorting direction from None, Ascending, or Descending.

Sorting aggregate: Sorting aggregate function.

Color tab

Expression: Select a dataset value or enter an expression for color encoding.

Aggregate: Color encoding aggregate function.

Show values name: Select the checkbox to show the values name.

Shape tab

Expression: Select a dataset value or enter an expression for a shape encoding.

Aggregate: Shape encoding aggregate function.

Size tab

Expression: Select a dataset value or enter an expression for a size encoding.

Aggregate: Size encoding aggregate function.

Labels

General tab

Template: Select a data label template for the chart plot.

Text position: Select the text position of the data label from Auto, Center, Inside, or Outside.

Overlapping Labels Mode: Specify whether the labels can overlap.

Offset: Gets or sets the text offset of the data label in pixels.

Appearance tab

Background color: Choose a Web or Custom color to fill the background of the chart plot label.

Font

Family: Choose the font family name for the chart plot label.

Size: Choose the size in points for the chart plot label.

Style: Choose one from the enumerated styles for the text font.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, LineThrough, and DoubleUnderline.

Border

Style: Select a border style from the list of available options.

Width: Choose a width value between 0.25pt and 20pt for the thickness of the borderlines.

Color: Choose a Web or Custom color to use for the borderlines.

Connecting Line

Style: Choose one from the enumerated styles for the labels line position.

Color: Choose a Web or Custom color.

Width: Set the line width in points.

Position: Select the position of the label line from Auto or Center.

Rules

General tab

Name: Enter the rule name.

Condition: Enter an expression for the rule to apply.

Properties tab

Target property: Select from a predefined list of properties, what property needs to be changed.

Target property value: Specify the target property value by entering an expression.

See Rules topic for more information.

Overlays

General tab

Name: Enter the chart overlay name.

Type: Set the type of the overlay by selecting it from the predefined list.

Display: Select the display position from Front or Back.

Appearance tab

BorderStyle: Select a border style from the list of available options.

Width: Choose a width value between 0.25pt and 20pt for the thickness of the borderlines.

Color: Choose a Web or Custom color to use for the borderlines.

Config tab

Detail Level: Specify if the overlay calculation should include the entire dataset or each detail group.

Field Name: Set the field name for the overlay to use.

Axis: Specify the axis, to which the overlay belongs.

Aggregate: Set the reference line aggregate function.

Legend Label: Enter the legend label text.

Value: Set the position of the specified axis.

See Trendlines topic for more information.

Action

Choose from the following actions to perform when the user clicks on the chart element.

None: The default behavior is to do nothing when a user clicks the chart element at run time.

Jump to report: For drill-through reporting, select this option and provide the name of a local report, the relative path of a report in another folder, or the full path of a report on another server.

Parameters

- **Name**: Supply the exact names of any parameters required for the targeted report. Note that the parameter names you supply in this must match parameters in the target report.

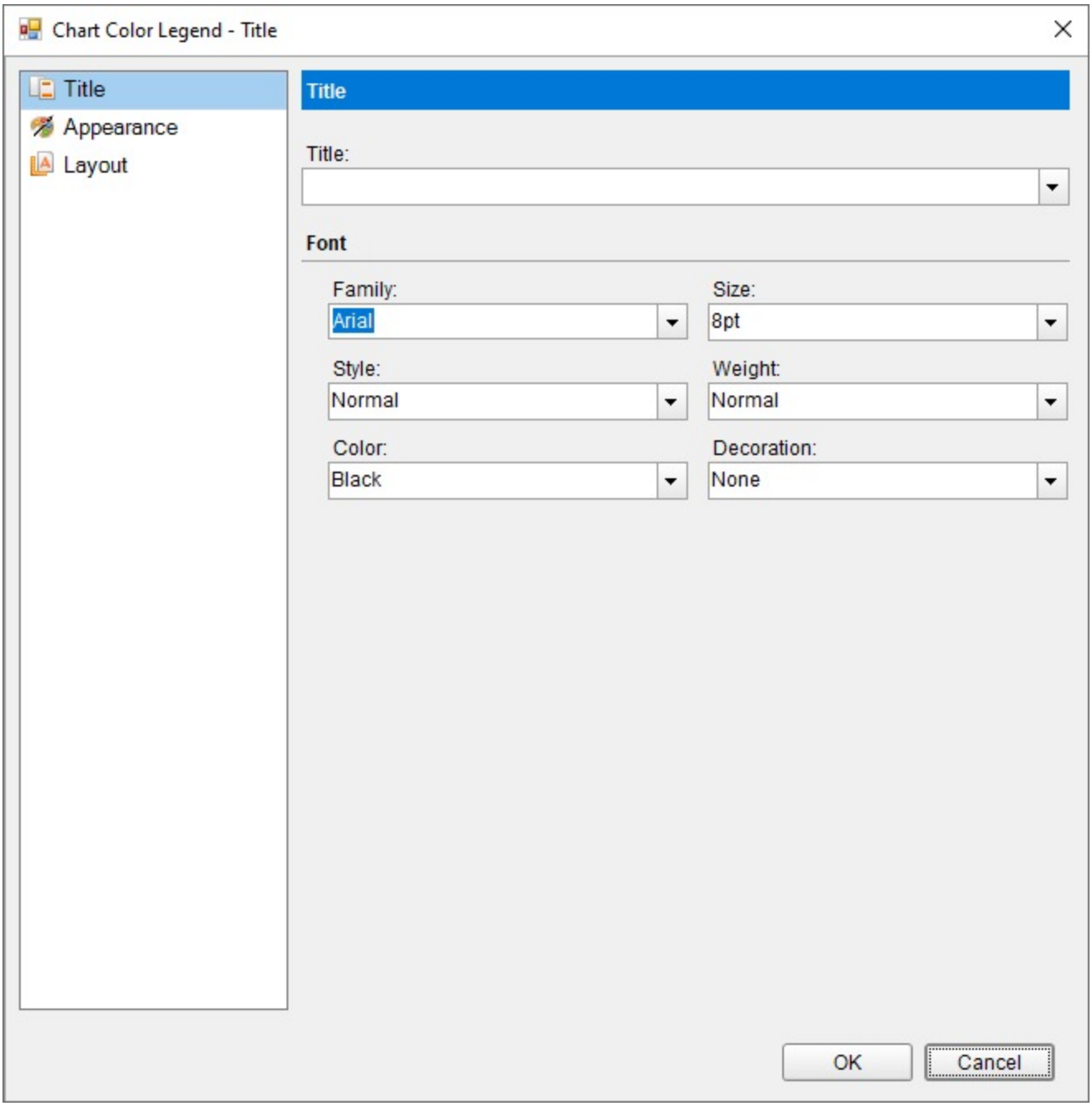
  > **Important:** The Parameter Name must exactly match the name of the parameter in the detail report. If any parameter is spelled differently, capitalized differently, or if an expected parameter is not supplied, the drill-through report will fail.

- **Value**: Enter a Parameter Value to pass to the detail report. This value must evaluate to a valid value for the parameter.
- **Omit**: Select this check box to omit this parameter from the report.

Jump to bookmark: Select this option and provide a valid Bookmark ID to allow the user to jump to the report control with that Bookmark ID.

Jump to URL: Select this option and provide a valid URL to create a hyperlink to a Web page.

Chart Color Legend Smart Panel



Title

Text: Enter an expression or text to use for the chart legend.

Font

Family: Choose the font family name for the chart legend.

Size: Choose the size in points for the chart legend.

Style: Choose one from the enumerated styles for the text font.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, LineThrough, and DoubleUnderline.

Appearance

Border

Style: Select a border style from the list of available options.

Width: Choose a width value between 0.25pt and 20pt for the thickness of the borderlines.

Color: Choose a Web or Custom color to use for the borderlines.

Font

Family: Choose the font family name for the chart legend.

Size: Choose the size in points for the header/footer text font.

Style: Choose one from the enumerated styles for the text font.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, LineThrough, and DoubleUnderline.

Background Fill Color: Choose a Web or Custom color to fill the background of the chart legend.

Layout

Hide Legend: Select this checkbox to hide the chart legend.

Position: Select an enumerated value from Bottom, Left, Right, or Top to determine the position of the legend relative to the chart area.

Orientation: Select the legend orientation from Horizontal or Vertical.

Size

MaxHeight: Sets the maximum height in percent relative to the overall chart height.

MaxWidth: Sets the maximum width in percent relative to the overall chart width.

Padding: Set the padding values - Top, Left, Right, and Bottom, for the chart legend.

See Legends topic for more information.

Plots

A plot in a Chart data region encodes Data values into geometrical shapes with the application of the encoding configuration. Let's say, for example, the plot of a bar chart can encode the list of countries using 'Category Encoding'.

Common Plot Properties

Name

A String value that represents the name of the plot which the user wants to keep for the chart.

Plot Template

Setting a Plot template (or the chart type) is the first step of designing a chart. The plot can be set from the Chart Wizard, from the toolbar menu available for the Chart data region, or from the **Plot Template** option in the context menu that appears on right-clicking the plot on the design area.

Refer to the following sub-topics on plots for detailed descriptions of various plot template configurations. Each of these plots is explained in the dedicated pages with end-to-end walkthroughs.

- Column and Bar Charts
- Area Chart
- Line Chart
- Pie and Doughnut Charts
- Scatter and Bubble Charts
- Radar Scatter and Radar Bubble Charts
- Radar Line Chart
- Radar Area Chart
- Spiral Chart
- Polar Chart
- Gantt Chart
- Funnel and Pyramid Charts
- **Candlestick Chart (on-line documentation)**
- High Low Close Chart
- High Low Open Close Chart
- Range Charts
- Gauge Chart

Multiplot Charts

Charts may be multiplot, when a chart has a common X axis with the Y axis displaying different values. The multiplot charts are used to compare data points between two plots. See **Multiplot Charts (on-line documentation)** for details.

Column and Bar Charts

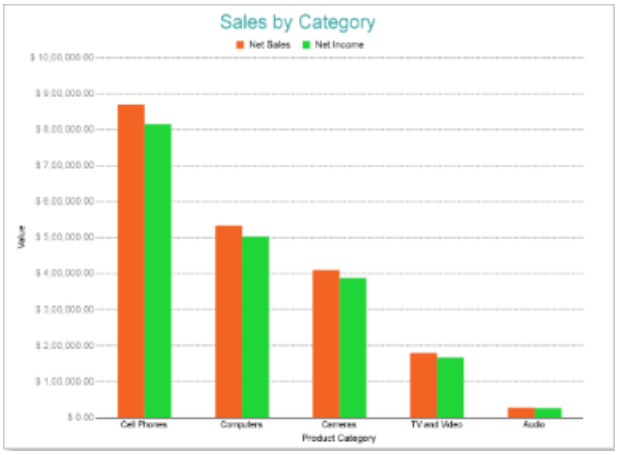
The Column and Bar charts present the comparison of data values across categories. The plot settings of these two charts are similar, however, they represent the data differently.

Column Charts

Column Charts plot the data values as vertical columns against the categories that are arranged horizontally. The y-axis values determine the heights of the columns, while the x-axis displays the category labels. The Column charts are vertical versions of bar charts and use the x-axis as a category axis.

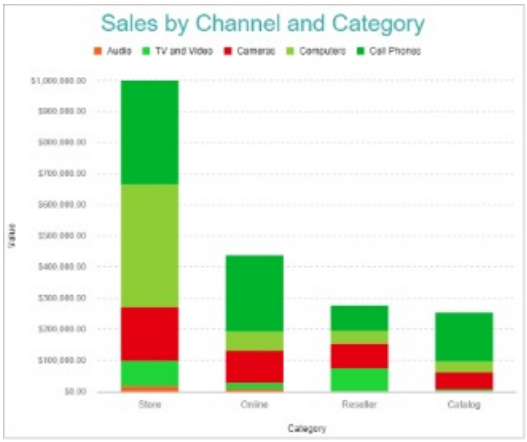
Clustered Column Chart

A Clustered Column chart is used to compare different values across different categories. See Create Clustered Column Chart walkthrough to learn how to create this chart.



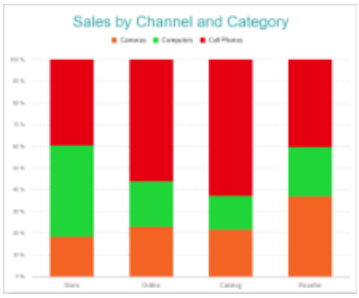
Stacked Column Chart

A Stacked Column chart is used to display the relationship of specific items to the whole across different categories and plot values. This chart stacks the data series vertically. The Create Stacked Column Chart walkthrough showcases plotting the Net Sales for each Product Category by Sales Channels.



Stacked Percentage Column Chart

A Percentage Stacked Column chart is used to perform comparisons of percentages that each of the values are contributing to the total, across all your categories. This chart stacks the data series vertically and also equalizes the plotted values to meet 100%.

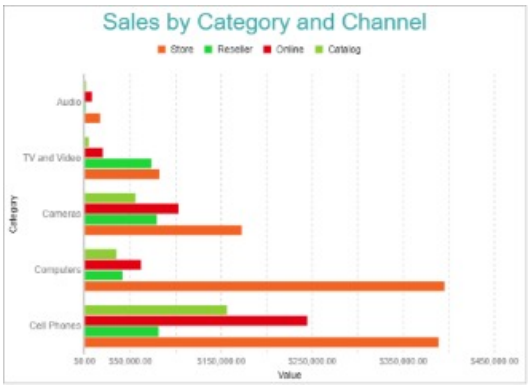


Bar Charts

Bar charts compare categorical data through horizontal bars, where the length of each bar represents the value of the corresponding category. In bar charts, categories are organized along the vertical axis and data values along the horizontal axis. For example, sales of various product categories can be presented through a bar chart. The Bar charts are preferred when the number of categories is too large to be used on an x-axis.

Clustered Bar Chart

A Clustered Bar chart can be used to display the comparisons of values across different categories. The Create Clustered Bar Chart walkthrough showcases plotting the chart comparing the Sales for different Product Categories and Sales Channel.

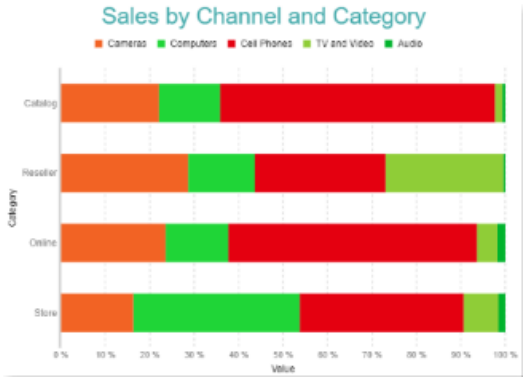


Stacked Bar Chart

A Stacked Bar chart is used to display the relationship of each category to the whole.

Stacked Percentage Bar Chart

A Percentage Stacked Bar chart is used to display the comparisons of the percentage that each value contributes to the total across different categories. The Create Stacked Percentage Bar Chart walkthrough showcases the Net Sales for each Product Category by Sales Channels with the percentage contribution of each subcategory within a category.



Column and Bar Plot Properties

The Column and Bar Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when a column or a bar is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each column or bar.

- **Template**: The template for the data label.
- **Offset**: The pixels by which the data label should move relative to the column or the bar edge.
- **OverlappingLabels:** Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **ConnectingLine**: The line that draws connecting the column or bar edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - Position: The position of the connecting line relative to the data label. 'Auto' (default) draws the line connecting the data label and the edge of the column or the bar, while 'Center' draws the line connecting the data label to the center of the column or bar.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the columns or the bars.
  - Center: Positions the data label text on the center of the column or the bar.
  - Inside: Positions the data label text inside the column or the bar.
  - Outside: Positions the data label text outside the column or the bar.
  - Auto: The default setting, same as Outside for column and bar.
- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.

LineStyle

The line style for the column and bar borders.

- **LineColor**: Specify the color of the border around columns or bars.
- **LineStyle:** Specify the line style of the border around columns or bars as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around columns or bars.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

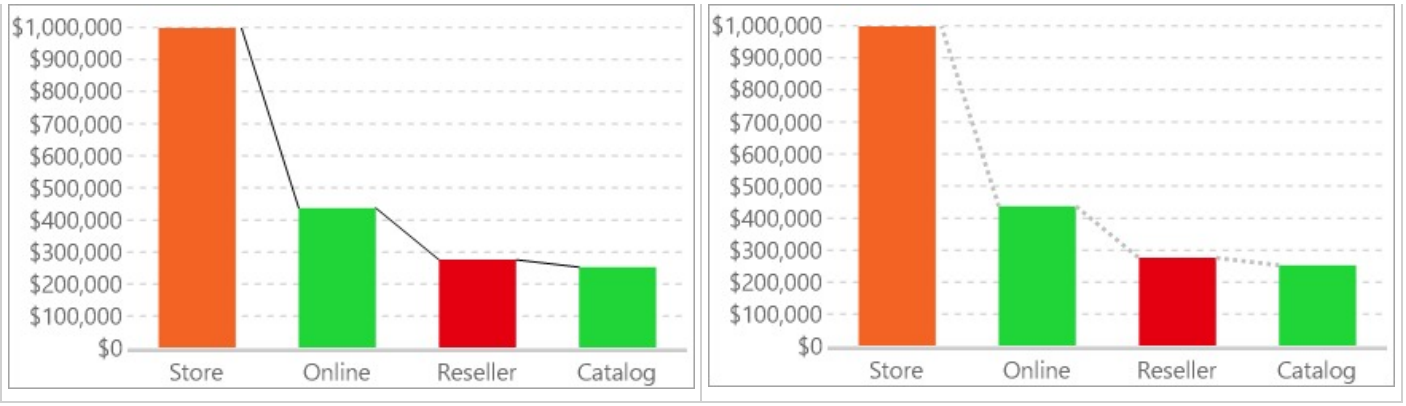
Configurations

BarLineStyle

The BarLineStyle settings are applied when the **ShowBarLines** property is set to True.

- **LineColor:** Specify the color of the bar line.
- **LineStyle:** Specify the bar line style as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the bar line width.

Bar line (default settings)	Bar line (custom settings)



ClippingMode

The Clipping Mode determines how a plot (columns or bars) extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area
- **Clip:** Clips off the excess bar or column lengths toward the right or the bottom
- **None:** Same as 'Fit' for bar and column plots.

Offset

The Offset is the percentage relative to the single-column width or a single bar height, by which the column and bar plots should move to the right or the bottom side, respectively.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means the columns or the bars are opaque while 0% opacity means that they are completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on the column or bar plot data points. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowBarLines

Determines whether to show the connecting lines between the columns or the bars of the same Details Encoding or the same Category Encoding (if the Details Encoding is empty) across different categories.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

UnpivotData

Determines whether to display multiple data fields as a single data field. In other words, if this property is set to 'True', the columns or the bars appear stacked in case they represent multiple data fields. By default, the property is set to 'False'.

Design

BarSettings

The BarSettings property specifies the percentage relative to the column width or the bar height that should be specified in order to change the shape of the columns or bars.

- **NeckHeight**: Creates a neck in the otherwise rectangular columns or bars. The resulting shape of the column or bar can have a different width or height as specified in the BottomWidth or the TopWidth properties.

- **BottomWidth**: Bottom width of the column or bar in percent.
- **Overlap**: Percentage overlap between the adjacent columns or bars representing different data values, relative to the containing category size.
- **TopWidth**: Top width of the column or bar in percent.
- **Width**: Width of the column or the bar.

Encodings

Values Encoding

The Values encoding specifies the data values. The Values property is the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Column and Bar plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Column and Bar plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

If multiple data values are added in the Values encoding, a clustered plot is created.

Category Encoding

The Category Encoding for Column and Bar plots forms the axis of the plot across which distinct categories are arranged. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

For example, see the Create Clustered Column Chart walkthrough where

- Values collection takes the `[Product Category]` field item
- Sorting field takes the `[Net Sales]`,
- Sort direction is Descending, and
- Sorting aggregate is Sum.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: ExcludeNulls property determines whether to exclude the null details values in the dataset field.
- **Group**: The Group property determines how the columns or bars are arranged when divided into subcategories.
  - Stack: Stacks the subcategories vertically in the case of column plots and side-by-side in the case of bar plots.
  - Cluster: Creates a cluster of groups for the subcategories. The Create Clustered Bar Chart walkthrough uses this value for grouping the details value. Note that a clustered plot can be created by providing two data field values, that is, two items in the Values encoding collection.
  - None: It is the default value.
- **SortDirection**: The SortDirection defines the ascending or descending order in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: The SortingField defines the order in which the subcategories are displayed.
- **Values**: The Values property is the collection and takes the field as a subcategory.

Color Encoding

The Color Encoding enables the color legend of the Category Encoding or Details Encoding to display a match between the column or bar colors and the data values. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values
The Values is the collection where the value of the Color expression is specified. However, the Column and Bar plots take the first item from the collection.

If the Details Encoding is empty, Column and Bar plots calculate distinct Color Encoding results for the categories produced by the Category Encoding, convert them to the background color of the corresponding columns or bars, and display the match in the legend.

Otherwise, Column and Bar plots calculate distinct Color Encoding results for the subcategories produced by the Details Encoding, convert them to the background color of the corresponding columns or bars subsections and display the match in the legend. In both cases, plots pick up colors from the Chart Palette.

For instance, in the Create Clustered Column Chart walkthrough, the Details encoding is empty but multiple data values are added. The **Show Values Name** property of the Color encoding is set to 'True', therefore, the colors are set to the `[Product Category]` expression showing the global legend with data value field names. Consequently, the report output shows the legend that matches Product Categories with corresponding column subsections background.

In most cases, you will use the same configuration for both Details Encoding and Color Encoding to enable a visual map of data values breakdown.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

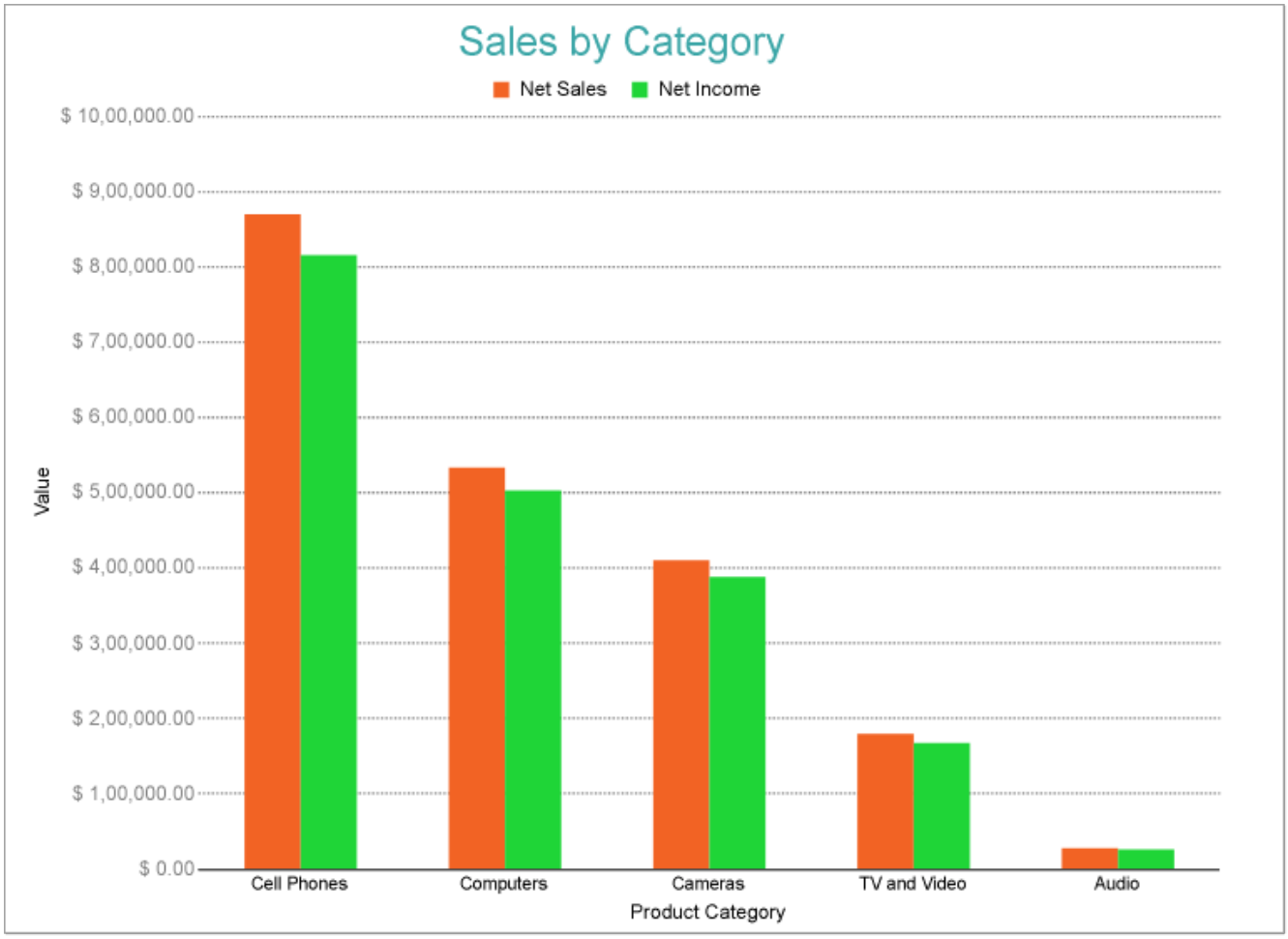
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Clustered Column Chart

This walkthrough creates a Clustered Column Chart. The chart consists of two columns comparing the 'Net Sales' and 'Net Income' per 'Product Category'. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields and modify the **Name** of the `[SalesAmount]` field to `[Net Sales]`.
4. On the same page, add two calculated fields:

Name	Value
Net Income	`=[Net Sales] - [UnitCost] - [DiscountAmount] - [ReturnAmount]`
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

5. Click **OK** to save the changes.

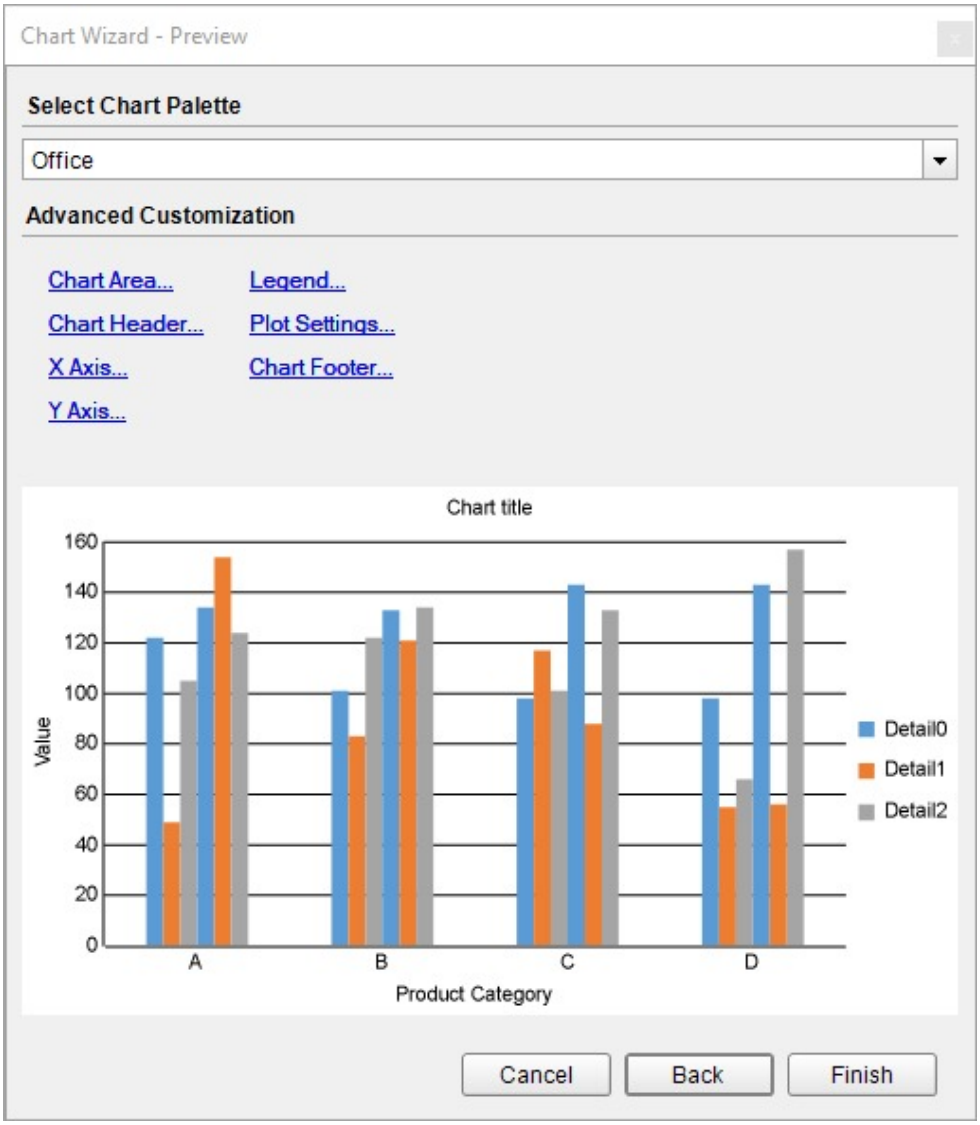
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Column'.
3. Click **Next** to proceed. Here, we will define two series values to form a cluster (or group), one column to show the 'Net Sales' and the other to show the 'Net Income'.
4. Under **Choose Data Values** select the following two values from the drop-down and set the corresponding aggregates:

Value	Aggregate
`[Net Sales]`	Sum
`[Net Income]`	Sum

5. In **Choose Data Categories**, select `[Product Category]` as the **Field**. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels accessed from the last screen of the wizard.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the product category to display in the order of decreasing total net sales. So fill in the following settings

Field	Settings
Sorting field	`=[Net Sales]`
Sorting direction	Descending

Sorting aggregate	Sum

3. Go to the **Encodings** page > **Color** tab, check on the **Show values name** option for legends to display.
4. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal points)'.
3. Now go to the **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 100000
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dotted
6. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Linear
    - **Minimum scale value**: 0
    - **Maximum scale value**: 1000000.
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
5. Click **OK** to complete setting up the X-axis.

Legend

1. To open the smart panel for advanced legend settings, right-click 'Global Legend' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 9pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the Legend.

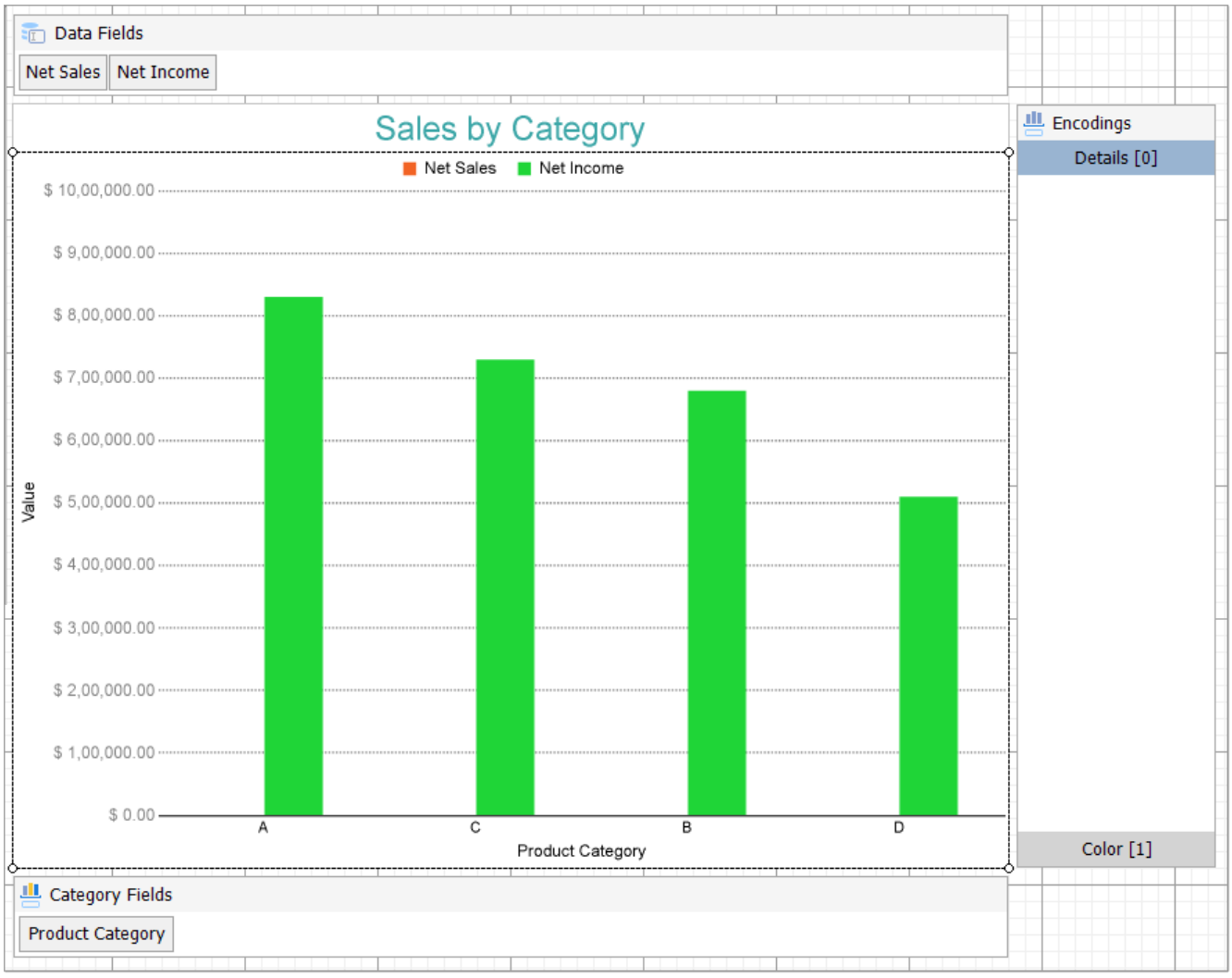
Chart Palette

To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.

1. Right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. In **Palette**, select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
3. Click **OK** to complete setting up the custom palette.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

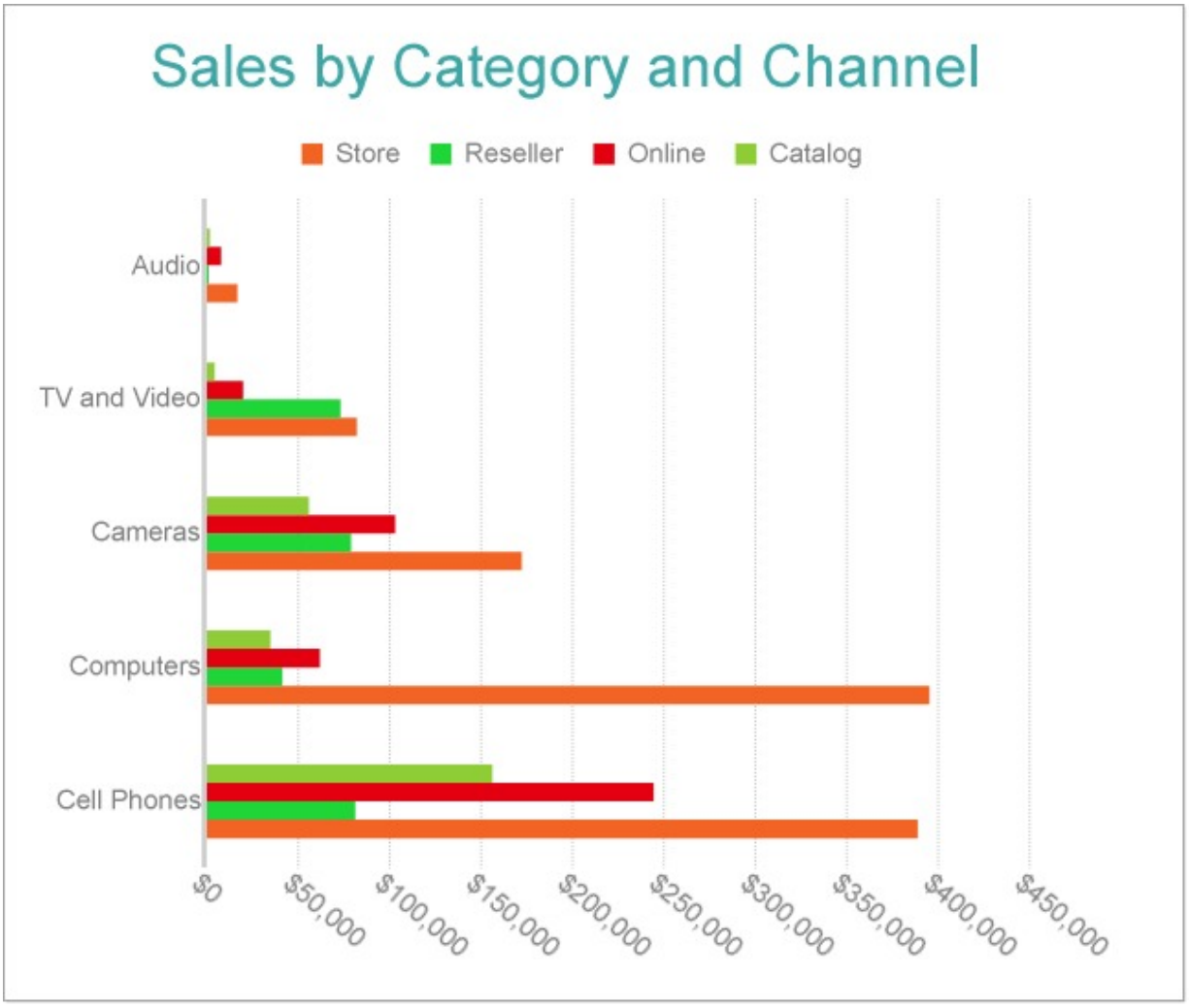


> **Note**: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Clustered Bar Chart

This walkthrough creates a Clustered Bar Chart. The chart compares the Store, Reseller, Online, and Catalog sales for the different sales categories as shown. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource**

icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields and modify the **Name** of the `[SalesAmount]` field to `[Net Sales]`.
4. On the same page, add three calculated fields:

Name	Value
Net Income	`=[Net Sales] - [UnitCost] - [DiscountAmount] - [ReturnAmount]`
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

5. Click **OK** to save the changes.

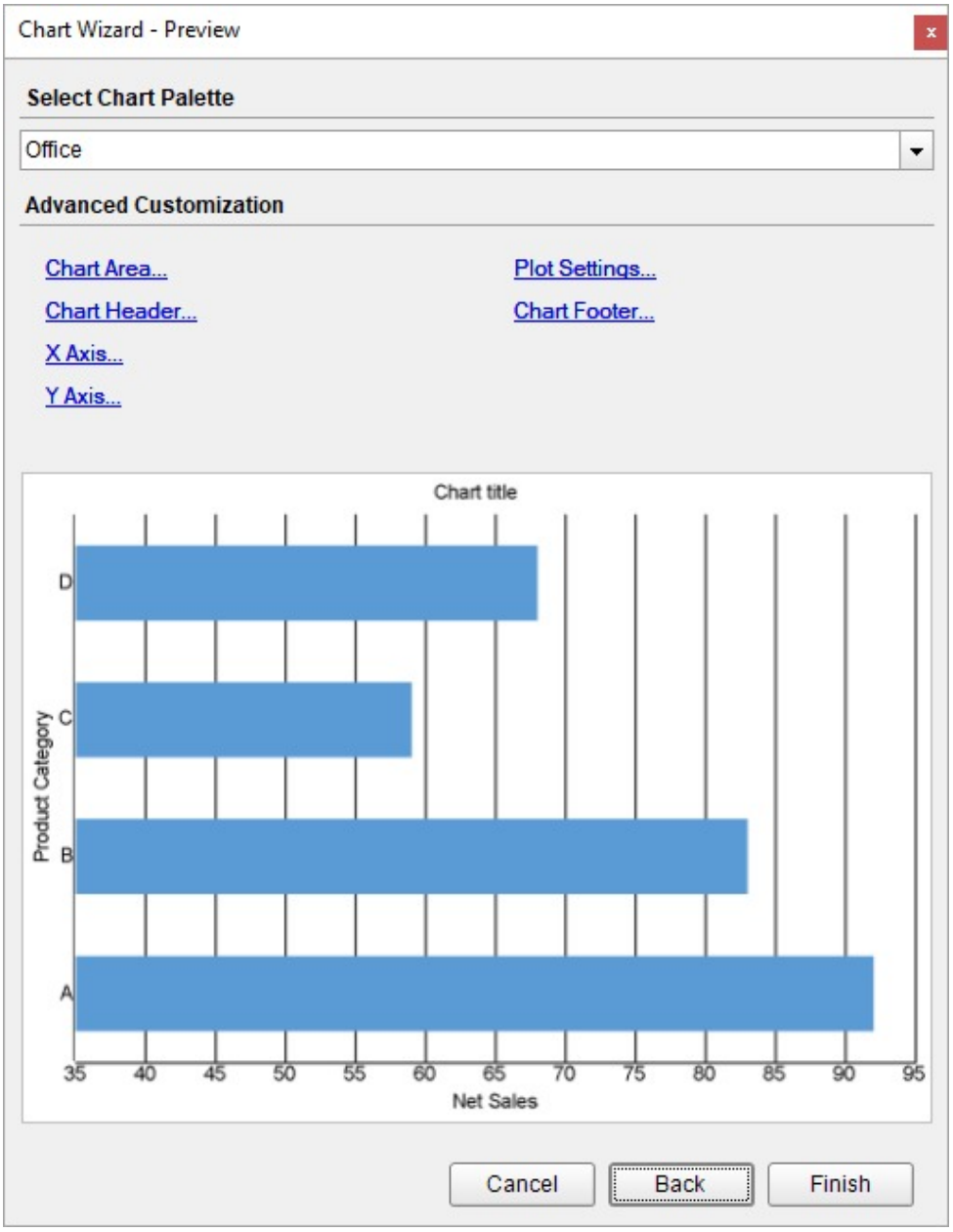
Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Bar'.
3. Click **Next** to proceed. Here, you need to specify the bar settings. We will define a data series value to display the net sales values across the horizontal axis. We will later sub-categorize the data series value to form a cluster.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[Net Sales]`	Sum

5. In **Choose Data Categories**, select `[Product Category]` as the **Field**. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the product category to display in the order of decreasing total net sales. So fill in the following settings:

Field	Settings
Sorting field	`=[Net Sales]`
Sorting direction	Descending
Sorting aggregate	Sum

3. Go to the **Encodings** page.
4. On the **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Sales Channel]` to display sales by store, reseller, online, and catalog.
    2. Under **Grouping**, set **Group** to 'Cluster' since we want to display the detail columns (i.e. sales by store, reseller, online, and catalog) in a cluster.
5. Navigate to the **Color** tab, add a new value and set **Expression** to `=[Sales Channel]` to display the legends based on the detail columns.
6. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the following properties.
    - **Format:** Currency (with 0 decimal points)
    - **Angle:** 45
4. Now go to the **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: Gray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page, check-on the **Show Grid** option and set the following properties.
    - **Grid Interval**: 50000
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dotted
7. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Linear
    - **Minimum scale value**: 0
    - **Maximum scale value**: 450000
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: Gray
4. Go to the **Line** page and set the following properties.
    - **Color:** Gray
    - **Width:** 2pt
5. Click **OK** to complete setting up the X-axis.

Chart Palette

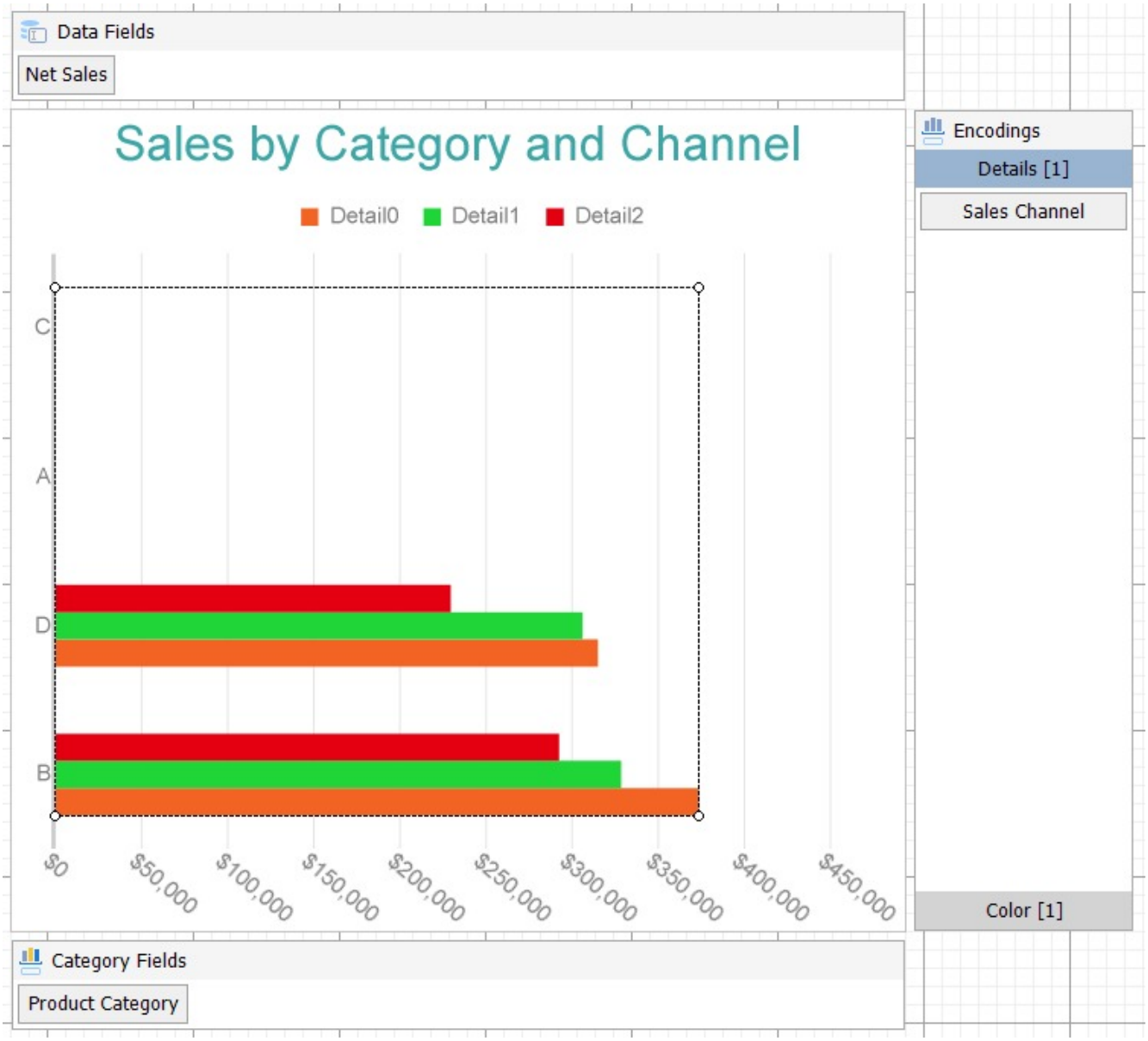
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 9pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category and Channel'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

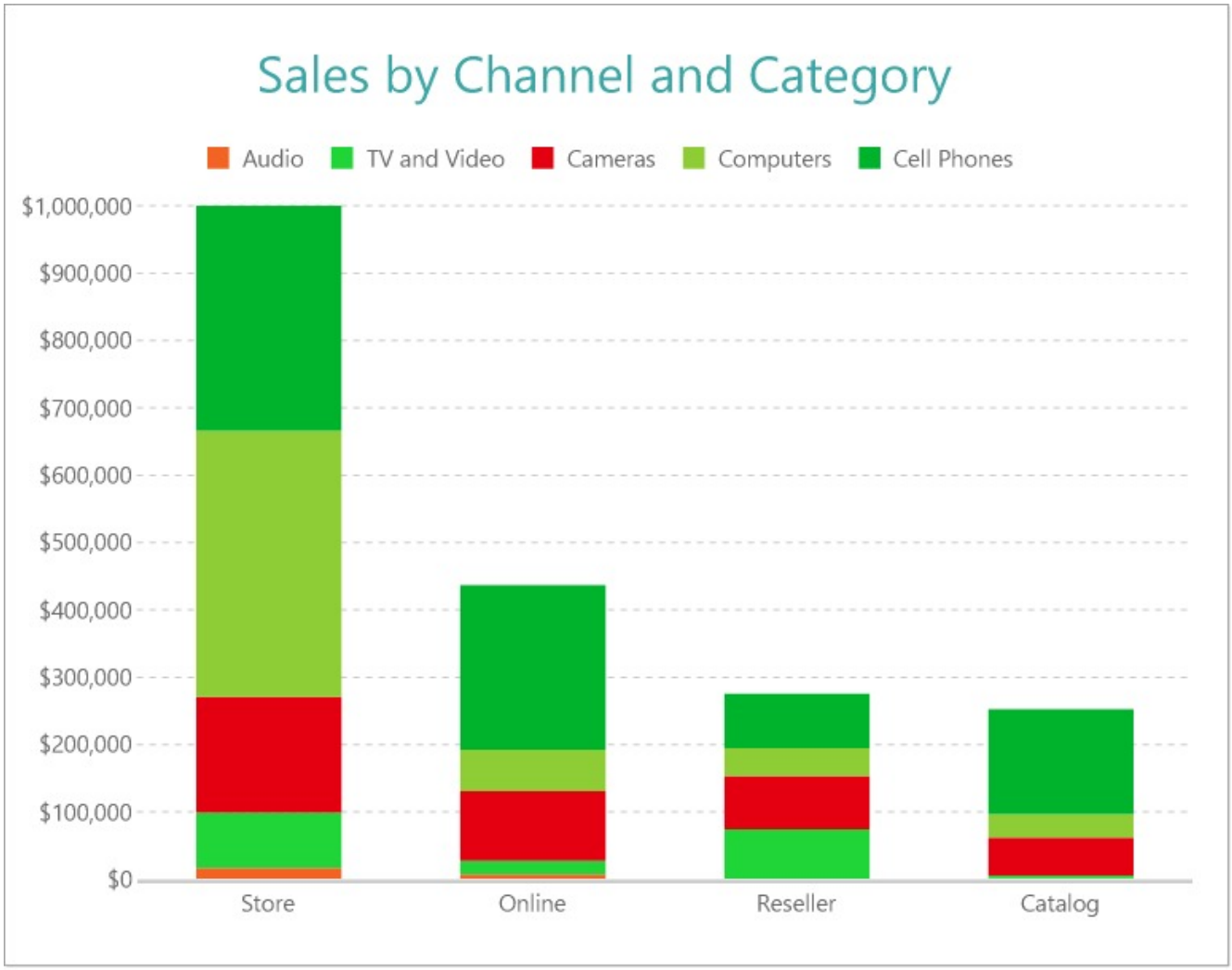


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Column Chart

This walkthrough creates a Stacked Column Chart. The chart shows the net sales for each product category by sales channels. In a stacked column chart, the data values are broken down into subcategories and are stacked on top of each other. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields and modify the **Name** of the `[SalesAmount]` field to `[Net Sales]`.
4. On the same page, add two calculated fields:

Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

5. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Column'.
3. Click **Next** to proceed. Here, you need to specify the column settings. We will define a data series value to display the net sales values across the horizontal axis. We will later sub-categorize the data series value to form a stack.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[Net Sales]`	Sum

5. In **Choose Data Categories**, select `[Sales Channel]` as the **Field**. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.

You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the sales channels to display in the order of decreasing total net sales. So fill in the following settings:

Field	Settings
Sorting field	`=[Net Sales]`
Sorting direction	Descending
Sorting aggregate	Sum

3. Go to the **Encodings** page.
4. On the **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Product Category]` to display net sales for cell phones, computers, cameras, tv and video, and audio.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to display the subcategories stacked on top of each other.
    3. Under **Sorting**, set **Sorting field** to `=[Net Sales]`, **Sort direction** to 'Ascending', and **Sorting aggregate** to 'Sum'. This will arrange the subcategories in the increasing order of their net sales in the stack.
5. Navigate to the **Color** tab, add a new value and set **Expression** to `=[Product Category]` to display the legends based on the subcategories.
6. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal points)'.
4. Now go to the **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 100000
    - **Show grid**: Check-on
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Style**: Dashed
7. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Linear
    - **Minimum scale value**: 0
    - **Maximum scale value**: 1000000
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
5. Click **OK** to complete setting up the X-axis.

Chart Palette

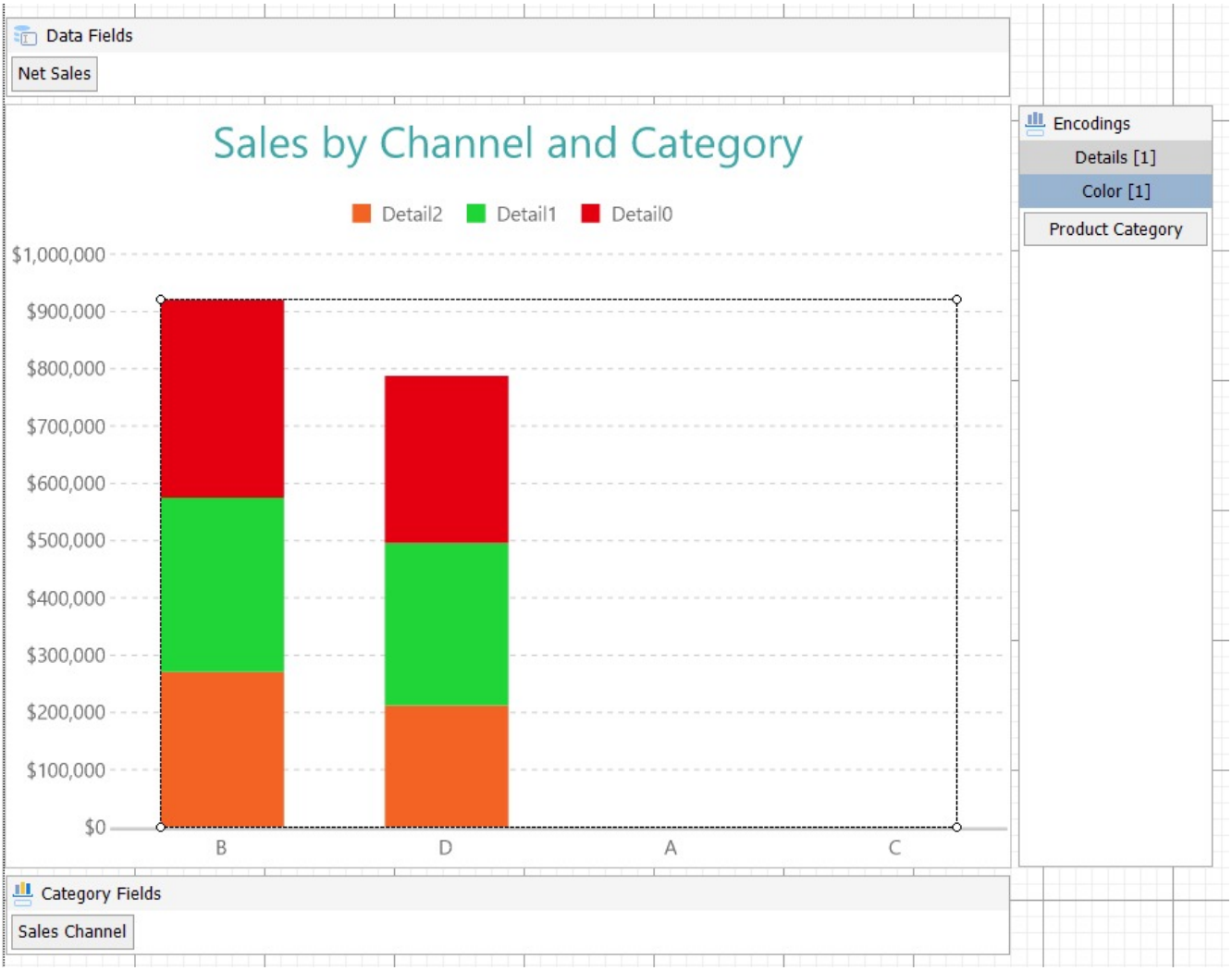
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 9pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Channel and Category'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

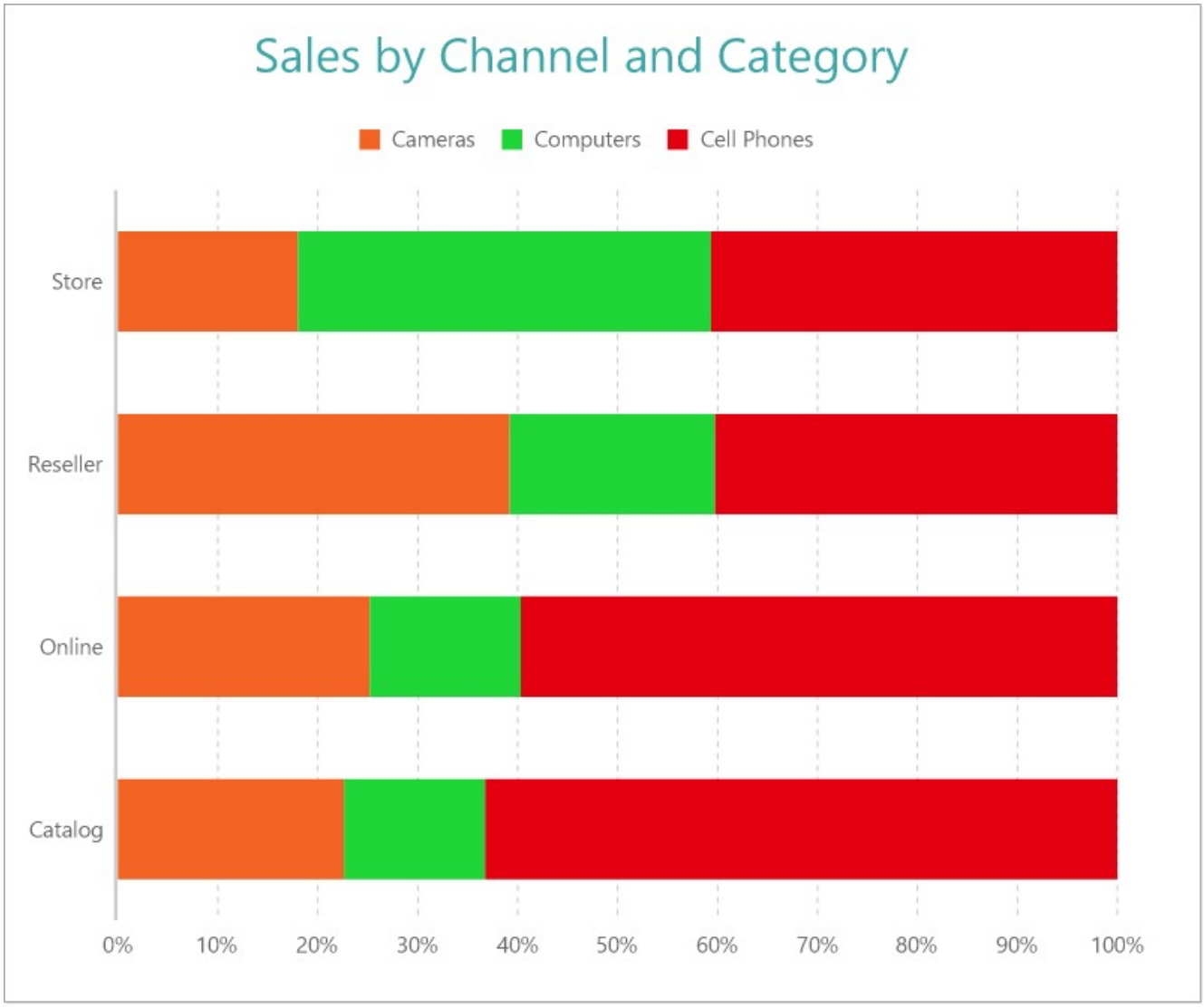


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Percentage Bar Chart

This walkthrough creates a Stacked Percentage Bar Chart. The chart shows the net sales for each product category by sales channels. The stacked percentage bar chart displays the percentage values that each subcategory contributes within a category. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields and modify the **Name** of the `[SalesAmount]` field to `[Net Sales]`.
4. On the same page, add two calculated fields:

Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

5. Then, navigate to the **Filters** page and add a new filter value, and set its properties as below.

Expression	Operator	Value
`=[ProductKey]`	GreaterThan	337

6. Click **OK** to save the changes.

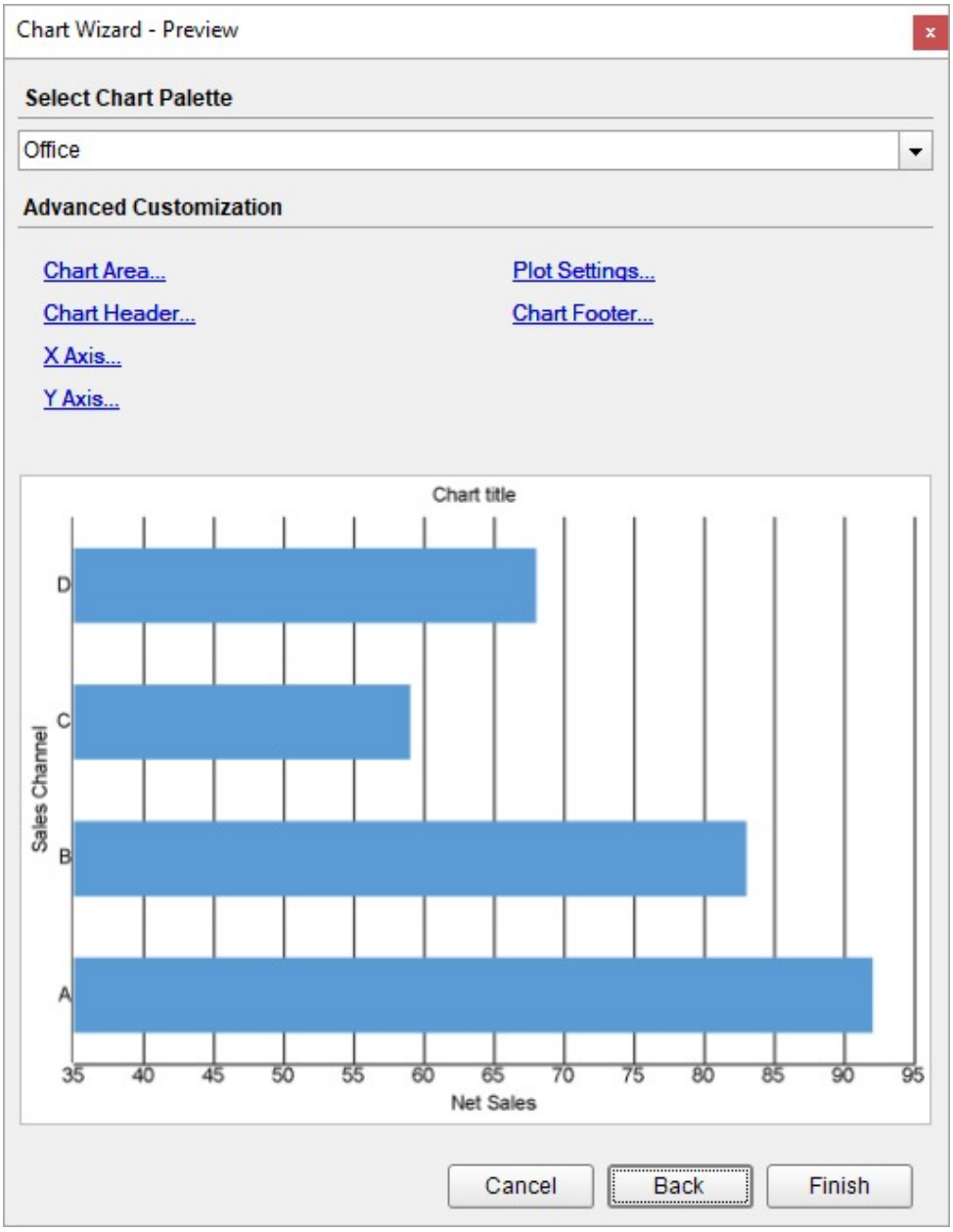
Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Bar'.
3. Click **Next** to proceed. Here, you need to specify the bar settings. We will define a data series value to display the net sales values across the horizontal axis. We will later sub-categorize the data series value to form a stack.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[Net Sales]`	Sum

5. In **Choose Data Categories**, select `=[Sales Channel]` as the **Field**. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the sales channels to display in the order of decreasing total net sales. So fill in the following settings:

Field	Settings
Sorting field	`=[Net Sales]`
Sorting direction	Descending
Sorting aggregate	Sum

3. Go to the **Encodings** page.
4. On the **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Product Category]` to subcategorize the net sales by product category for each sales channel.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to display the subcategories stacked on top of each other.
5. Navigate to the **Color** tab, add a new value and set **Expression** to `=[Product Category]` to display the legend based on the subcategories.
6. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties.
    - **Show grid**: Check-on
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Style**: Dashed
6. Go to the **Scale** page and set the **Scale Type** to 'Percentage'.
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
5. Click **OK** to complete setting up the X-axis.

Chart Palette

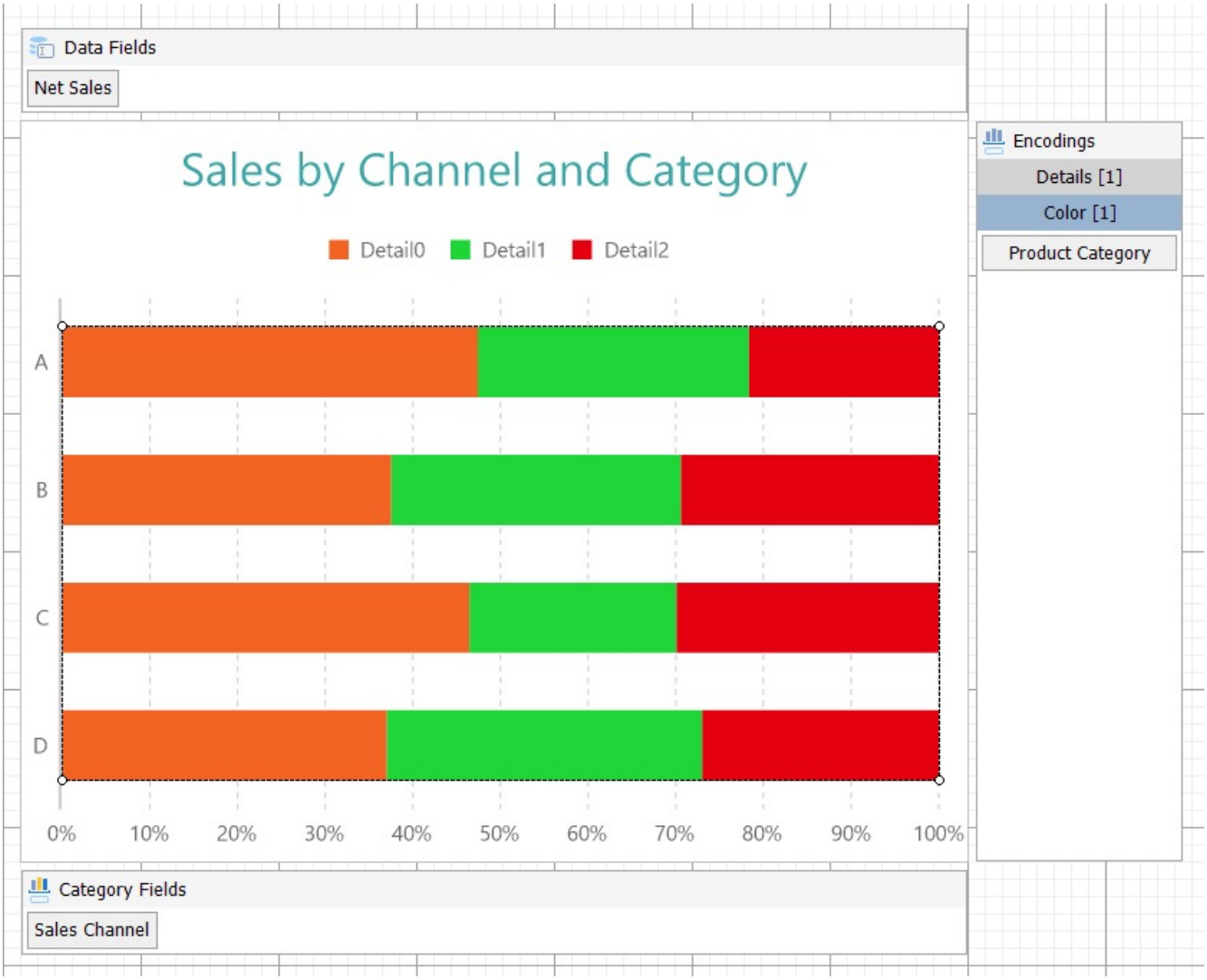
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 12pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Channel and Category'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

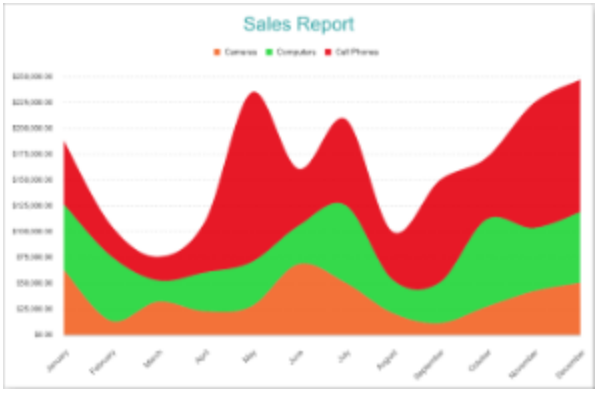
5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Area Chart

An **Area Chart** is similar to a line chart that is connected using line segments. The Area Chart arranges a period horizontally, connects them by line, encodes values into points, and fills the area between the line segments and x-axis with color. An area plot is a good chart option when you want to observe the volume change over a period. It gives the summation of the quantitative data.

Stacked Area Chart

A Stacked Area chart is a type of area chart that helps in breaking down data values into subcategories by placing corresponding area subsections on top of each other. The Create Stacked Area Chart walkthrough showcases plotting the Net Sales over a year with stacked areas for each Product Category.

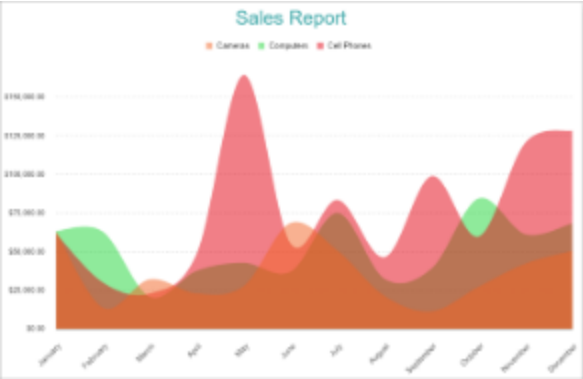


Stacked Percentage Area Chart

A Percentage Stacked Area chart combines the stacked area and the Percentage axis scale. Apart from showing changes in data values, the chart can also show each subcategory's contribution to a total. The Create Stacked Percent Area Chart walkthrough showcases plotting the percentage share of Product Categories in Net Sales.

Clustered Area Chart

A Clustered Area chart is used to break down data values into subcategories for granular analysis. For example, the Clustered Area Chart can be used to depict the total sales changes over a year segregated into clustered areas per product category.



Area Plot Properties

The Area Plot properties discussed below can be accessed from the Properties Panel by selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when the area plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each area chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the area plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.

- **TextPosition**: The position of the data label text relative to the area plot.
  - Center: Positions the data label text on the center of the area chart.
  - Inside: Positions the data label text inside the area chart.
  - Outside: Positions the data label text outside the area chart.
  - Auto: The default setting, same as Outside for area chart.

LineStyle

The line style for the area plot borders.

- **LineColor**: Specify the color of the border around the area plot.
- **LineStyle:** Specify the line style of the border around the area plot as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the area plot.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for area plots.

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default:** Indicates a straight line.
- **Spline:** Indicates a bezier curve.
- **StepCenter, StepLeft, and StepRight:** Indicates a stepped line with different step directions.

Opacity

The Opacity determines the opacity of areas filled with color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

UnpivotData

Determines whether to display multiple data fields as a single data field. By default, the property is set to 'False'.

Design

BarSettings

The BarSettings specifies the bar style settings.

- **NeckHeight**: Determines the bar neck height in percent.
- **BottomWidth**: Determines the bottom width in percent.
- **Overlap**: Determines the percentage bar overlap.
- **TopWidth**: Determines the top width in percent.
- **Width**: Determines the bar width in percent. The setting is valid only if **BottomWidth** and **TopWidth** are same.

Encodings

Category Encoding

The Category Encoding of an Area plot creates an area between the axis and the line generated from the connected data points representing the periodic change. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Colors Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Area plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This property determines whether to exclude the null details values in the dataset field.
- **Group**: This property determines the area subsection arrangement of the plot.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: This property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: This property defines the order in which the subcategories are displayed.
- **Values**: The Values property is the collection and takes the field as a subcategory.

Values Encoding

The Values encoding specifies the data values, and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Area plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Area plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

```
Count:{Text0}
```

```
Sum:{valueField.value}
```

Template Key

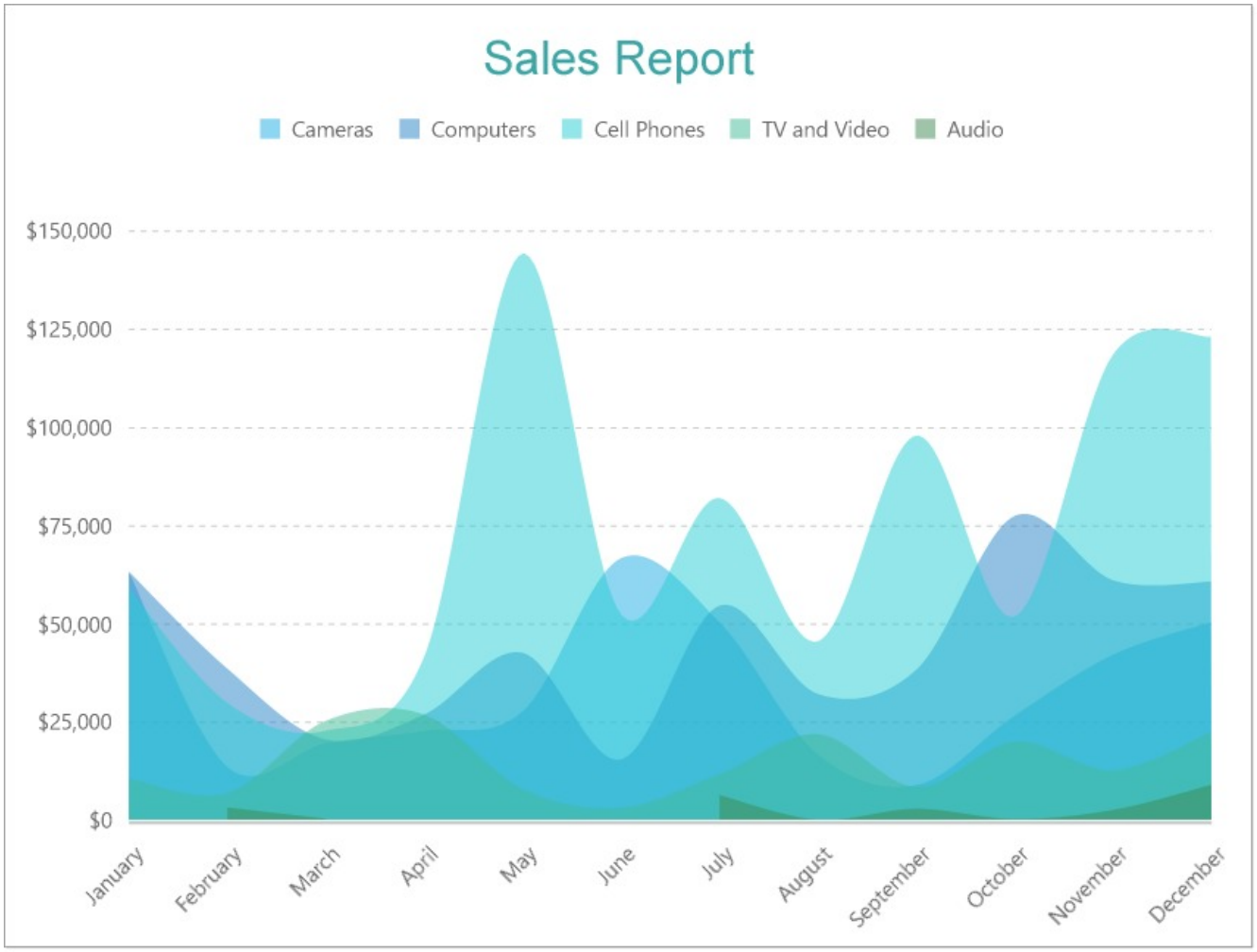
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Clustered Area Chart

This walkthrough creates a Clustered Area Chart. The chart displays the sales amount by product categories for each month. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.

2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Click **OK** to save the changes.
4. Go to the **Fields** page to view the available fields and modify the **Name** of the `[DateKey]` field to `[Sales Date]`.
5. On the same page, add a calculated field:

Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

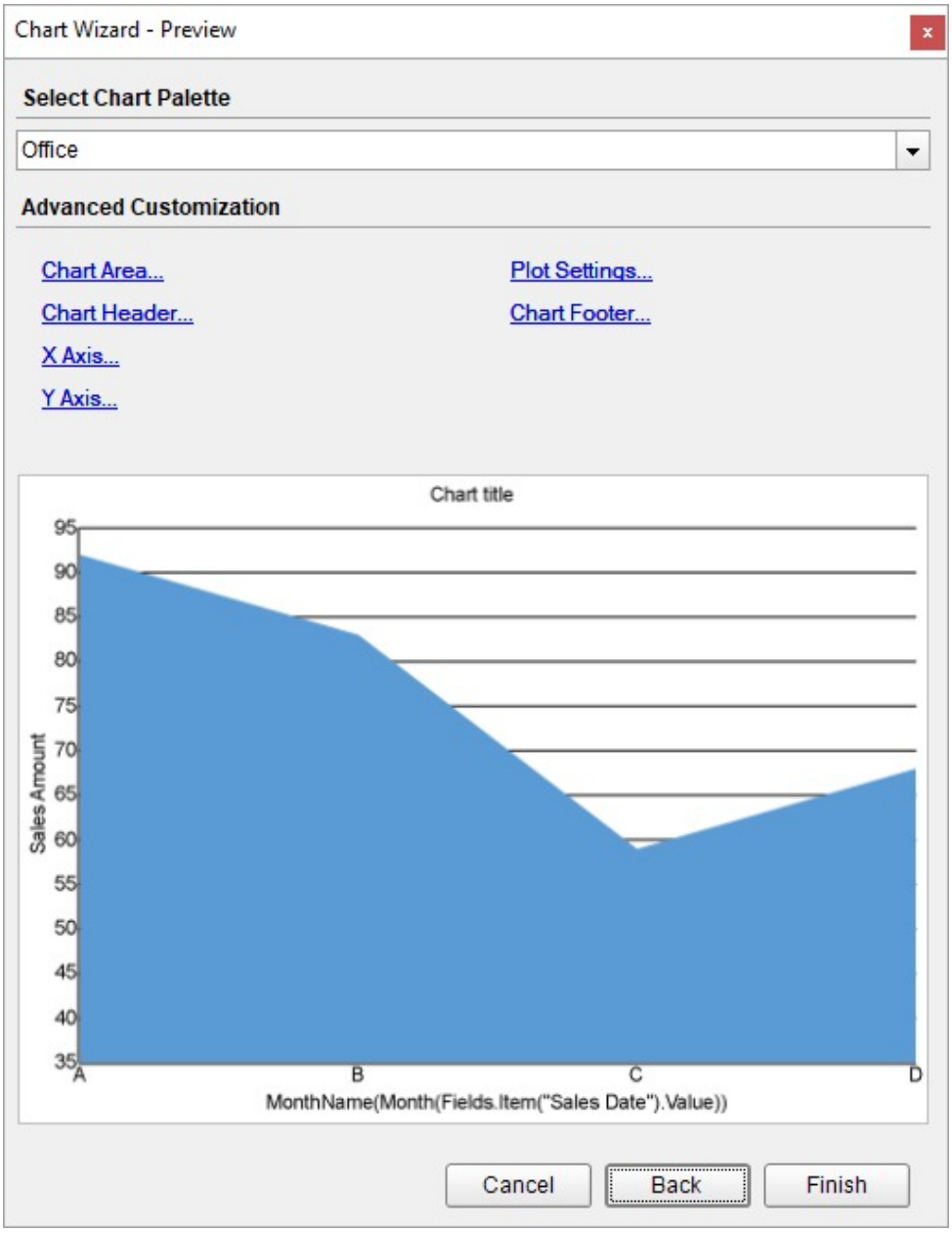
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Area'.
3. Click **Next** to proceed. Here, you need to specify the area settings.
4. In **Choose Data values** section, we will define a data series value to display the sales amount values along the vertical axis.
5. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

6. In **Choose Data Category**, set **Field** to `=MonthName(Month([Sales Date]))`. We will add more customizations to the category in later steps.
7. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the month names to display in chronological order. So fill in the following settings:

Field	Settings
Sorting field	`=Month([Sales Date])`
Sorting direction	Ascending

3. Go to the **Encodings** page.
4. On the **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Product Category]` to display the sales amount for cell phones, computers, cameras, tv and video, and audio.
    2. Under Grouping, set **Group** to 'Cluster'.
5. Then, navigate to the **Color** tab, add a new value, and set the **Expression** to `=[Product Category]`. This will display the legend based on the product categories in the chart.
6. Click **OK** to complete setting up the plot.
7. With 'Plot-Plot1' selected, go to the Properties pane and
    1. set the **Opacity** property to '50%' to change the opacity value for the plot fill color.
    2. set the **LineAspect** property to 'Spline' to display curved lines in the area chart.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal points)'.
4. Now, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 25000
    - **Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Go to the **Scale** page and set the following properties:
    - **Scale Type**: Linear
    - **Minimum scale value**: 0
    - **Maximum scale value**: 170000
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-45'.
4. Now, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
6. Click **OK** to complete setting up the X-axis.

Chart Palette

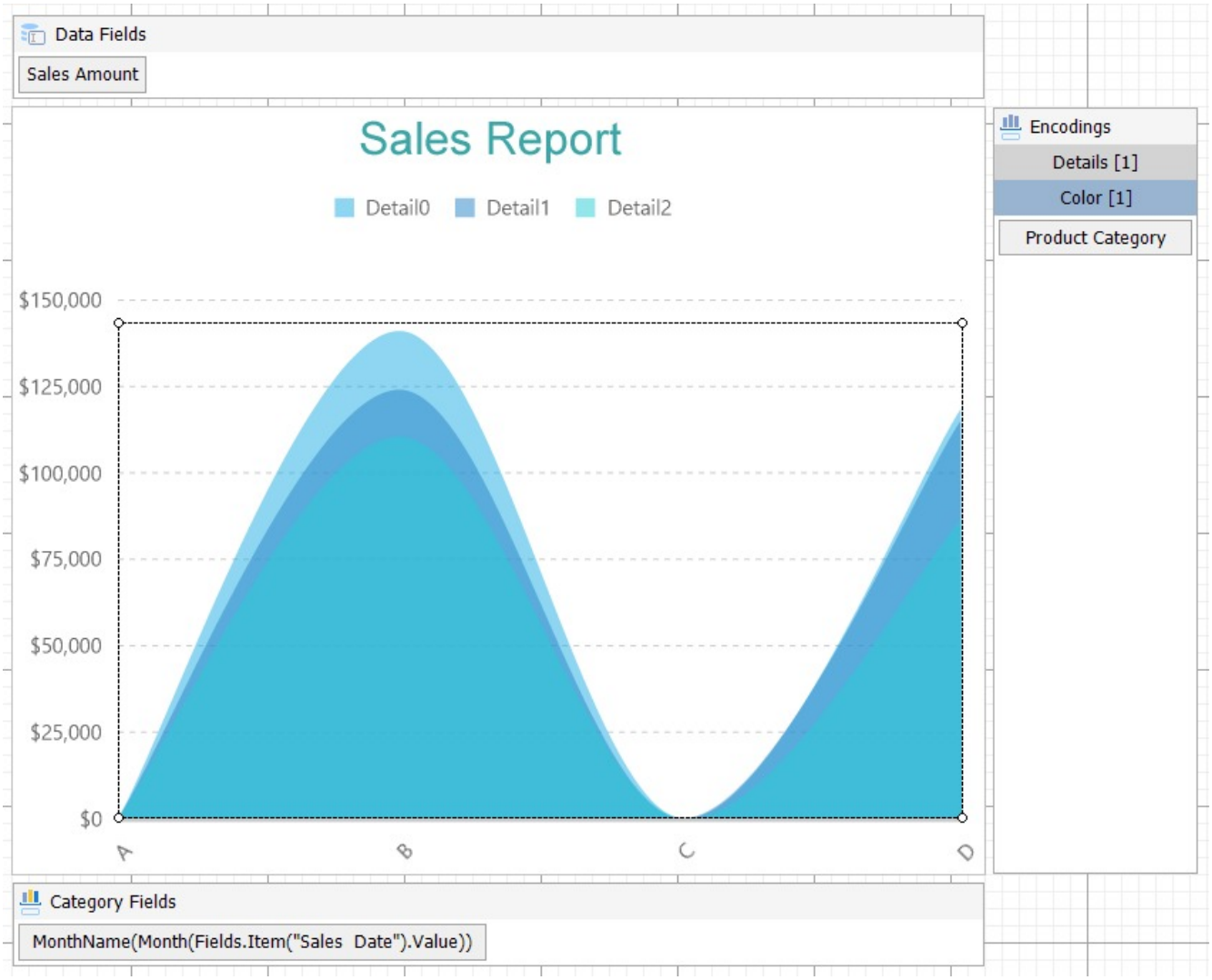
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. On the **Palette** page, choose the 'Blue2' palette from the drop-down.
3. Click **OK** to complete setting up the palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales Report'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

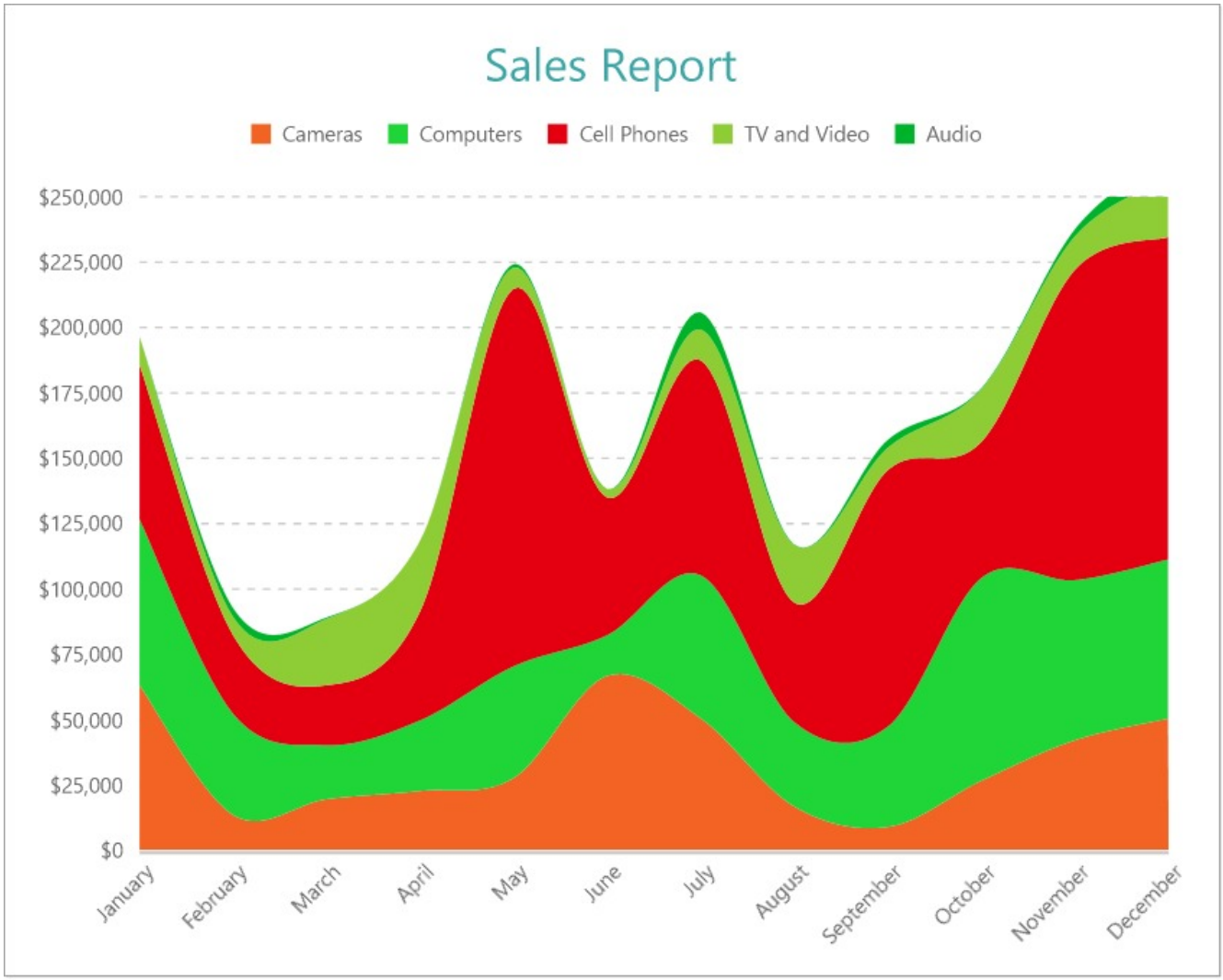


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Area Chart

This walkthrough creates a Stacked Area Chart. The chart displays the sales amount by product categories for each month. In a stacked area chart, the data values are broken down into subcategories and are stacked on top of each other. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields and modify the **Name** of the `[DateKey]` field to `[Sales Date]`.
4. On the same page, add a calculated field:

Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

5. Click **OK** to save the changes.

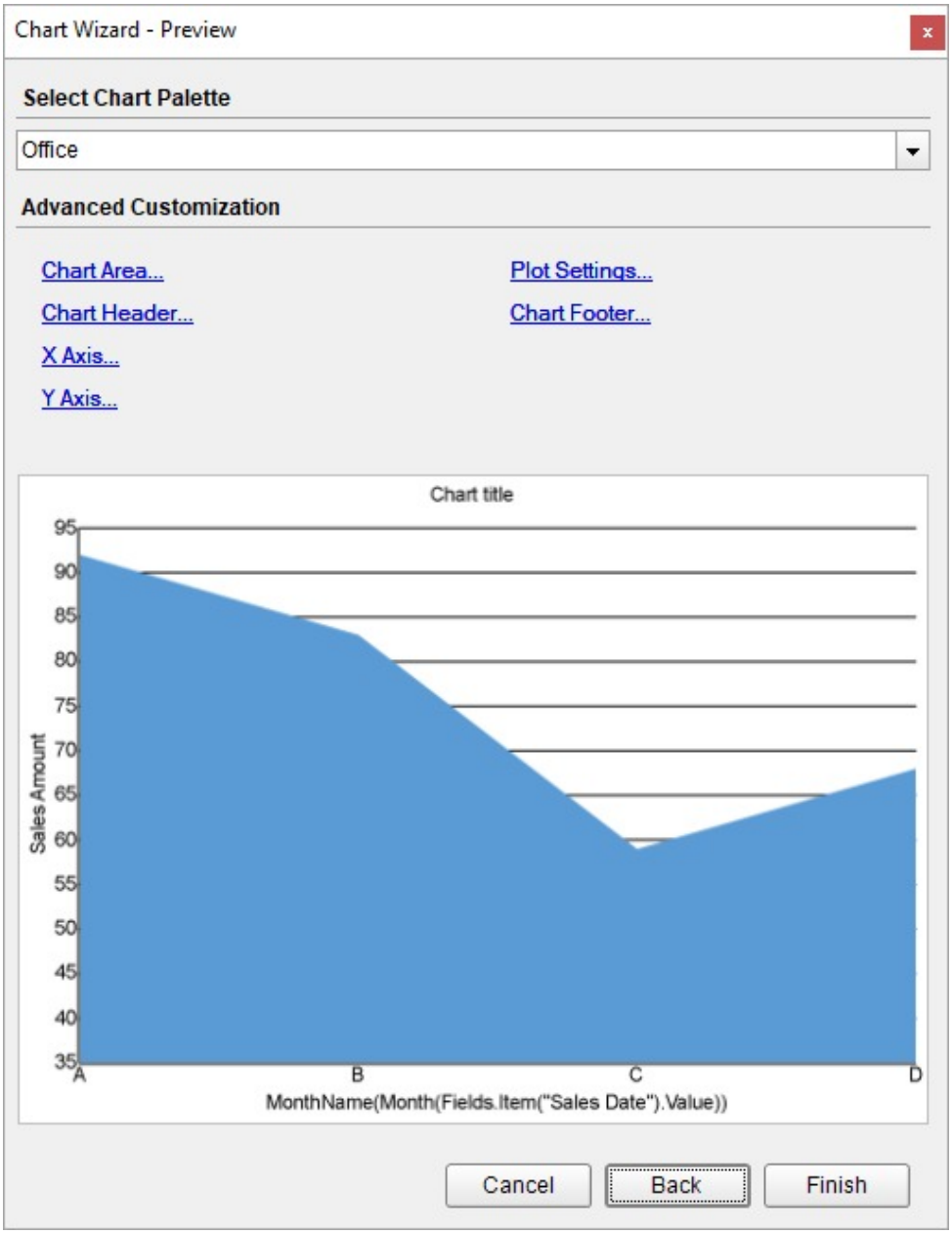
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Area'.
3. Click **Next** to proceed. Here, you need to specify the area settings.
4. In **Choose Data values** section, we will define a data series value to display the sales amount values along the vertical axis.
5. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

6. In **Choose Data Category**, set **Field** to `=MonthName(Month([Sales Date]))`.We will add more customizations to the category in later steps.
7. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the month names to display in the order of increasing month numbers. So fill in the following settings:

Field	Settings
Sorting field	`=Month([Sales Date])`
Sorting direction	Ascending

3. Go to the **Encodings** page.
4. On the **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Product Category]` to display the sales amount for cell phones, computers, cameras, tv and video, and audio.
    2. Under Grouping, set **Group** to 'Stack' since we want to subcategorize the sales amount by product categories (i.e. cell phones, computers, cameras, tv and video, and audio) in a stack.
5. Then, navigate to the **Color** tab, add a new value and set the **Expression** to `=[Product Category]`. This will display the legend based on the product categories in the chart.
6. Click **OK** to complete setting up the plot.
7. With 'Plot-Plot1' selected, go to the Properties pane and set the **Line Aspect** property for the plot to 'Spline' to display curved lines in the area chart.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal points)'.
4. Now, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 25000
    - **Show Grid**: Check-on
    - **Grid Appearance > Width**: 0.25pt
    - **Grid Appearance > Color**: #cccccc
    - **Grid Appearance > Style**: Dashed
7. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Linear
    - **Minimum scale value**: 0
    - **Maximum scale value**: 250000
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-45'.
4. Now, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
6. Click **OK** to complete setting up the X-axis.

Chart Palette

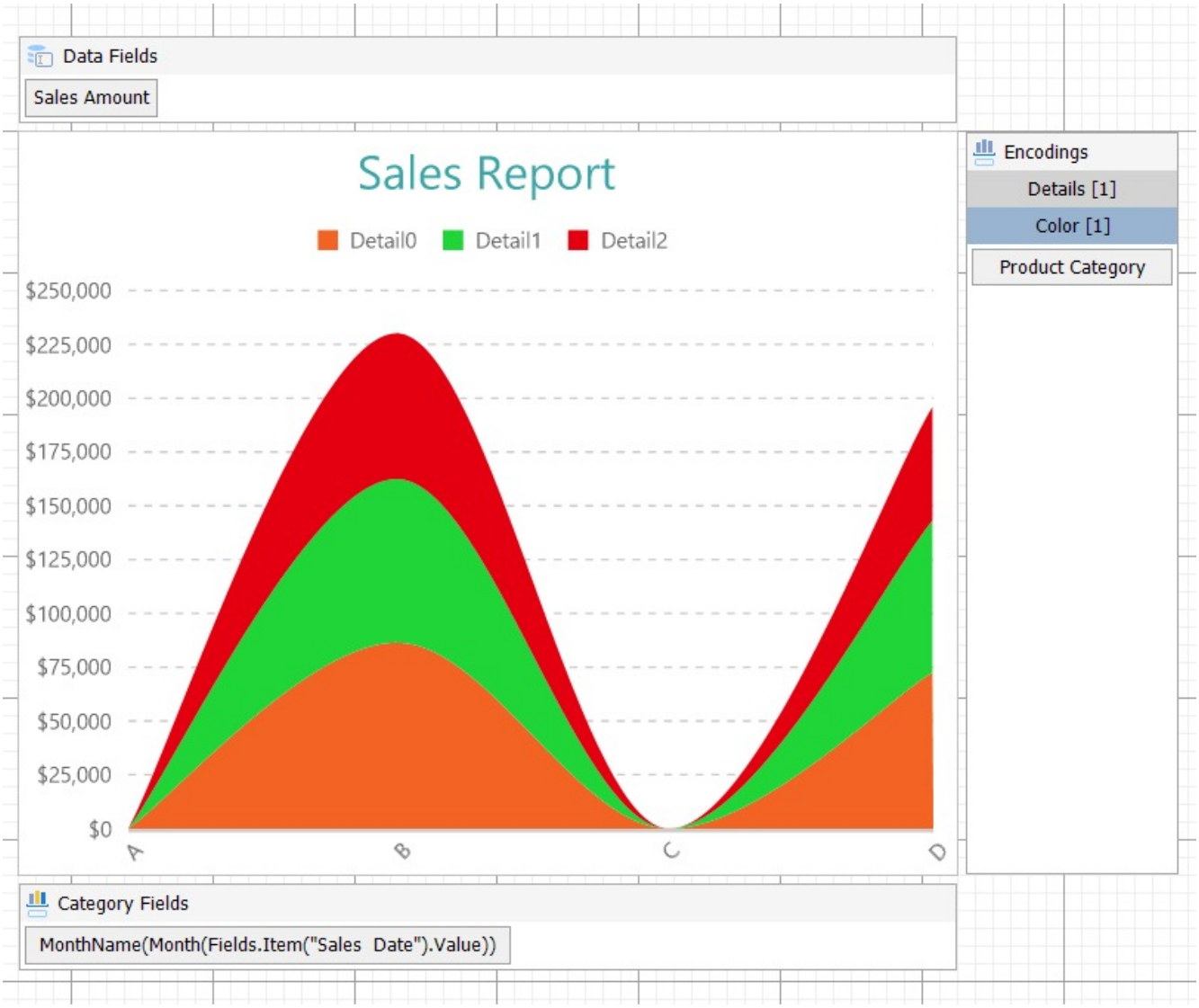
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. On the **Palette** page, select **Custom** from the drop-down and add the following colors:
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
3. Click **OK** to complete setting up the palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. On the **General** page, set **Title** to 'Sales Report'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

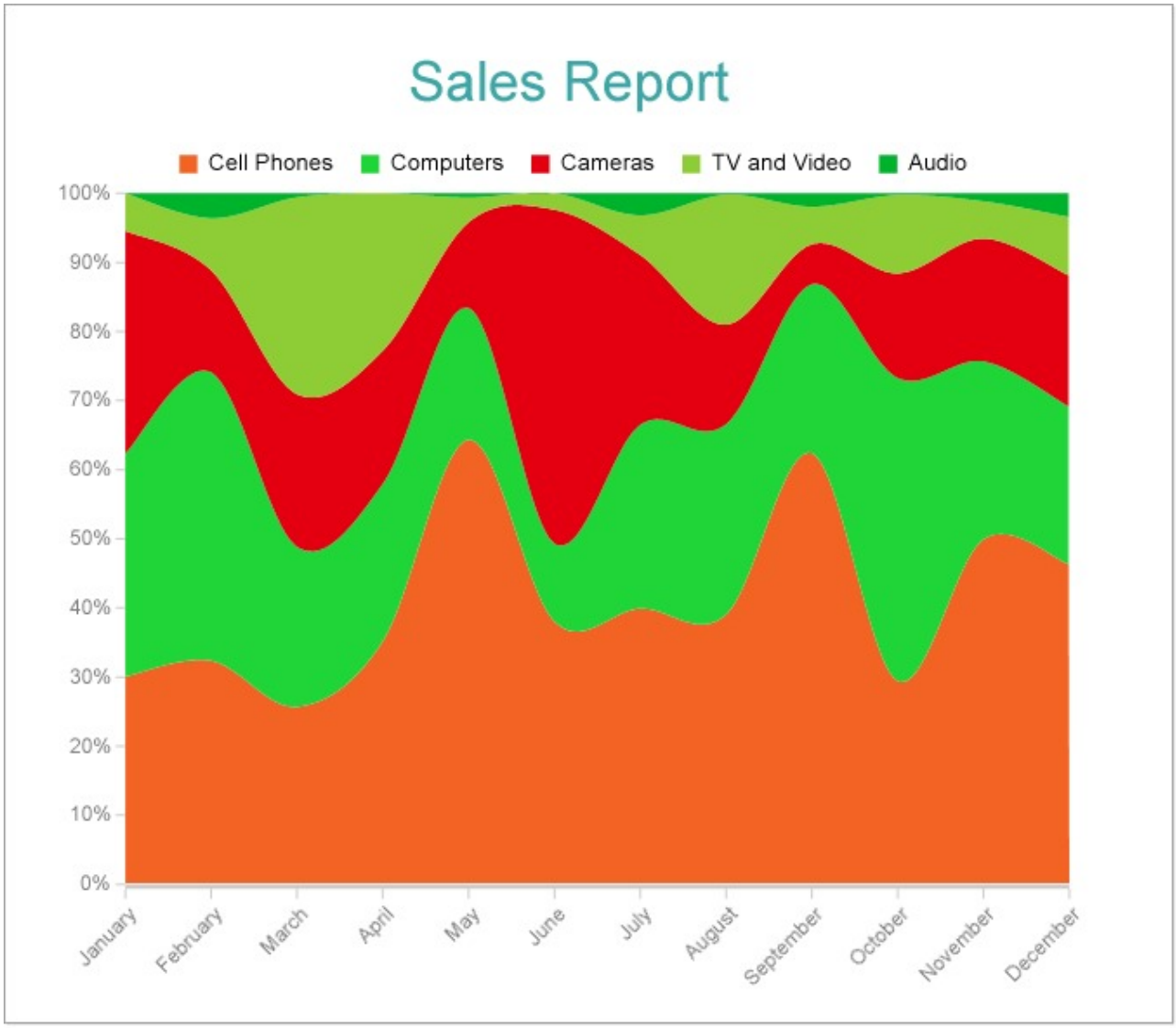


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Percentage Area Chart

This walkthrough creates a Stacked Percentage Area Chart. The stacked percentage area chart displays the percentage values that each Product Category contributes toward the Sales Amount in each month. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Click **OK** to save the changes.
4. Go to the **Fields** page to view the available fields and modify the **Name** of the `[DateKey]` field to `[Sales Date]`.
5. On the same page, add a calculated field:

Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

6. Click **OK** to save the changes.

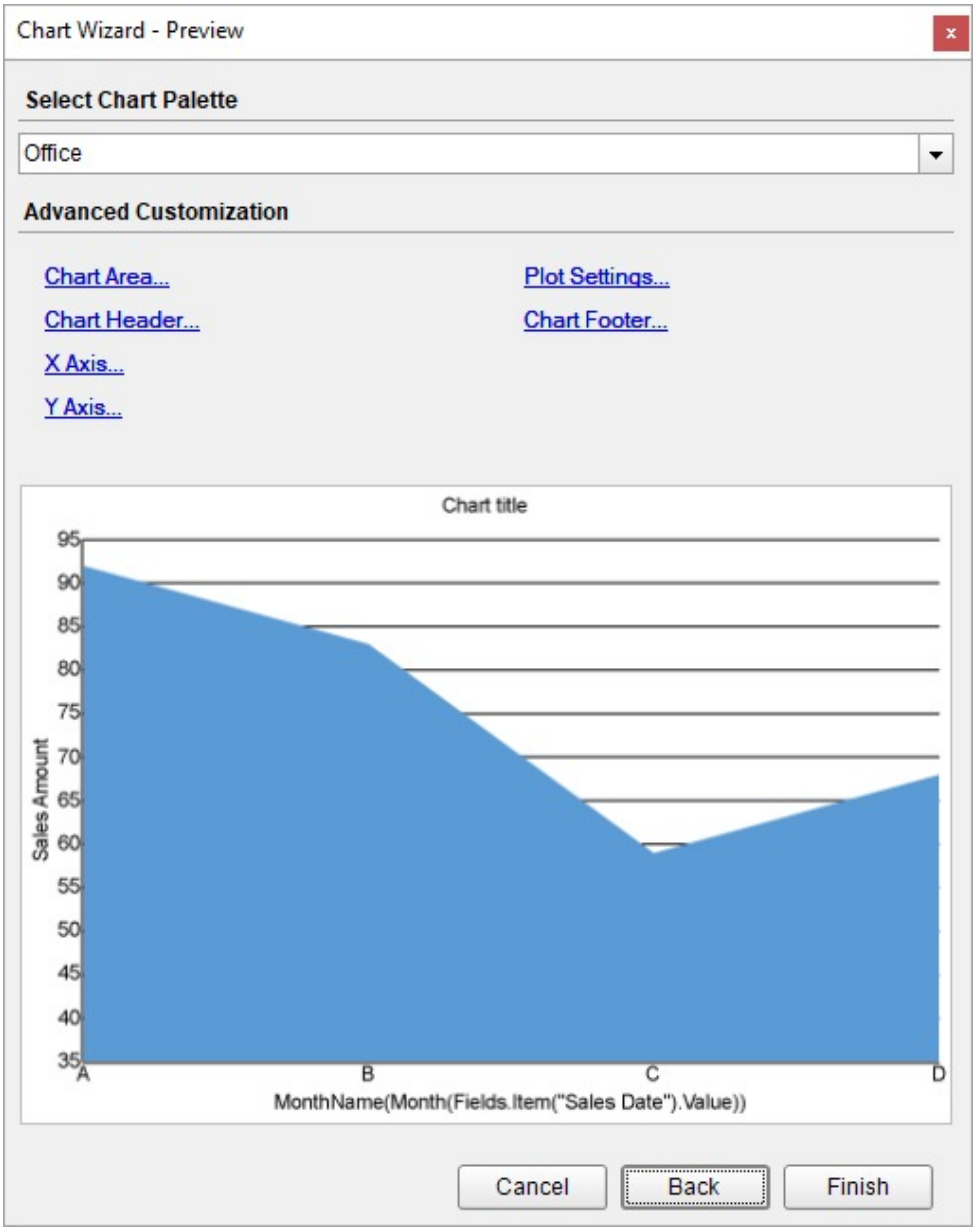
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Area'.
3. Click **Next** to proceed. Here, we will define a data series value to display the sale amount values along the vertical axis. We will later subcategorize the data series value to display as a stack.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Category**, set **Field** to `=MonthName(Month([Sales Date]))`.We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the month names to display in the order of increasing month numbers. So fill in the following settings:

Field	Settings
Sorting field	`=Month([Sales Date])`
Sorting direction	Ascending

3. Go to the **Encodings** page.
4. On the **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Product Category]` to display sales amount for different sales categories including cell phones, computers, cameras, tv and video, and audio.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to subcategorize the sales amount by product categories (i.e. cell phones, computers, cameras, tv and video, and audio) in a stack.
    3. Under **Sorting**, set **Sorting Field** to `=[SalesAmount]`, **Sorting Order** to 'Descending', and **Sorting Aggregate** to 'Sum'. This will arrange the subcategories in the decreasing order of sales amount.
5. Navigate to the **Color** tab, add a new value and set **Expression** to `=[Product Category]` to display legends based on the subcategories.
6. Click **OK** to complete setting up the plot.
7. With 'Plot-Plot1' selected, go to the Properties pane and set the **Line Aspect** property for the plot to 'Spline' to display curved lines in the area chart.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: Gray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties:
    - **Tick Mark > Position**: Outside
    - **Tick Mark > Color**: #cccccc
    - **Tick Mark > Style**: Solid
6. Go to the **Scale** page and set **Scale Type** to 'Percentage'.
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-45'.
4. Now go to the **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: Gray
5. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
6. Go to the **Major Gridline** page and set the following properties:
    - **Tick Mark > Position**: Outside
    - **Tick Mark > Color**: #cccccc
    - **Tick Mark > Style**: Solid
7. Click **OK** to complete setting up the X-axis.

Chart Palette

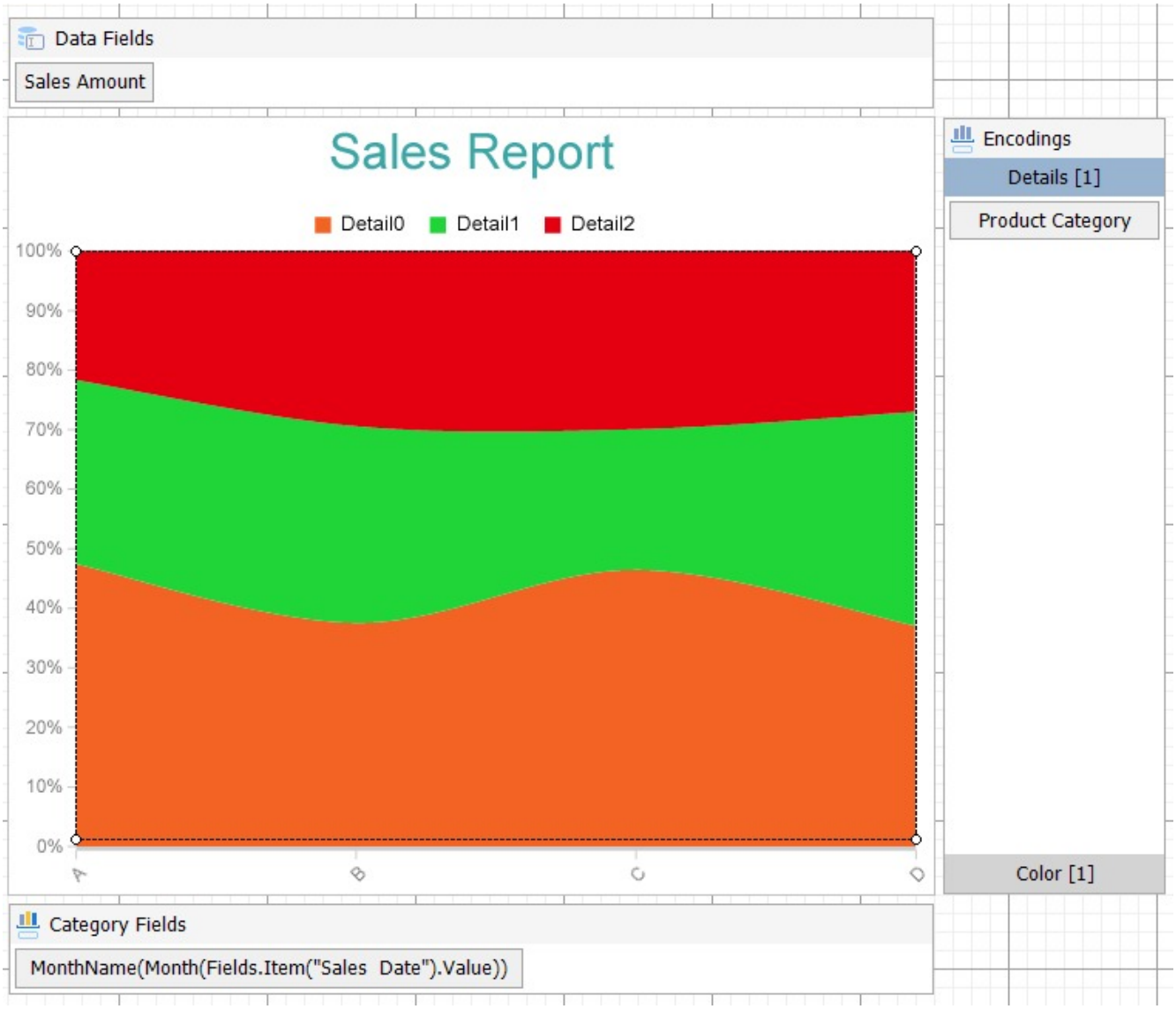
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. In **Palette**, select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 9pt
    - **Font > Color**: Gray
3. Go to the **Layout** page and set the following properties:
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog**.
2. Go to the **General** page and set **Title** to 'Sales Report'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Line Chart

The Line Charts represent the data points connected using straight lines. A line chart arranges a period horizontally, encodes data points into Symbols, and connects them by line segments. As the most basic chart type, Line Chart is highly beneficial in visualizing data trends as they compare values against periodic intervals such as temperature, time, etc. Some of the good examples that can be conveniently demonstrated through a line chart are the monthly average sale of a product and the closing prices of a stock in a given time frame.

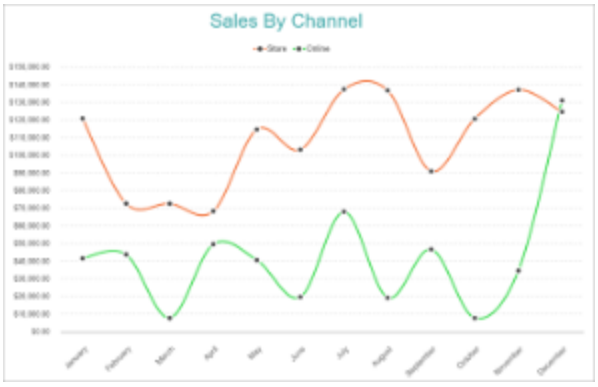
Single Line Chart

A Single Line chart helps you visualize the changes of a single data value. The Create Single Line Chart walkthrough

showcases plotting the changes in Product Returns over a year.

Multiple Line Chart

A Multiple Line chart lets you split the data values into subcategories for granular analysis of the total changes. The Create Multiple Line Chart walkthrough showcases plotting the change in the Net Sales of products over a year for two Sales Channels.



Multiple Values Line Chart

A Multiple Values Line chart lets you display changes of related or unrelated data values over the same period. For example, the Multiple Values Line Chart can be used to show the change of the Net Sales, the Cost of Goods Sold, and the Net Income for a product in a year as shown.

Line Plot Properties

The Line Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when the line plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each line chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the line plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the line plot.
  - Center: Positions the data label text on the center of the line chart.
  - Inside: Positions the data label text inside the line chart.
  - Outside: Positions the data label text outside the line chart.
  - Auto: The default setting, same as Outside for line chart.

LineStyle

The line style for the borders. Includes LineColor, LineStyle, and LineWidth properties for customization.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Symbols

Represents the properties that allow you to customize the look of symbols that form the Line plot.

- **BackgroundColor**: Change the color of the background.
- **LineColor**: Change the color of the line.
- **LineStyle**: Choose from different styles of line, such as Dotted, Dashed, Solid, Double, Groove, etc.
- **LineWidth**: Select the width of the lines in points.
- **Shape**: Choose from different shapes of symbols such as Dot, Box, Diamond, Triangle, X, Dash, Plus, etc.
- **Visible**: Set to true to display data point symbols.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Line plots.

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default**: Indicates a straight line.
- **Spline**: Indicates a bezier curve.
- **Step Center, Step Left, and Step Right**: Indicates a stepped line with different step directions.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

SwapAxes

Indicates whether the axes are swapped.

SymbolOpacity

Represents the opacity value of symbol fill color.

Encodings

Category Encoding

The Category Encoding of a line plot is a set of properties that determine the period over which the plot generates connected data points representing the Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Colors Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Line plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the subsection arrangement of the plot.
    - Stack: You can use this value to configure a Stacked line plot.

    - Cluster: You can use this value to configure subsections that overlap each other.
    - None: Equals to Cluster.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Shape Encoding

The Shape Encoding enables the shape legend of the Details or Category Encoding and includes the Aggregate function and shape expression, which is elaborated below:

Action

The action to take when the shape legend is clicked.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Value

The Value property is the collection and usually takes a bound field. However, the Line plots take the first item from the collection.

Size Encoding

The Size Encoding enables the Aggregate function and Size expression. The Size Encoding works solely with numeric values and breaks down data values into ranges that determine the symbol size.

Action

The action to take when the size legend is clicked.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Value

The Value property is the collection and usually takes a bound field. However, the Line plots take the first item from the collection.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Line plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Line plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

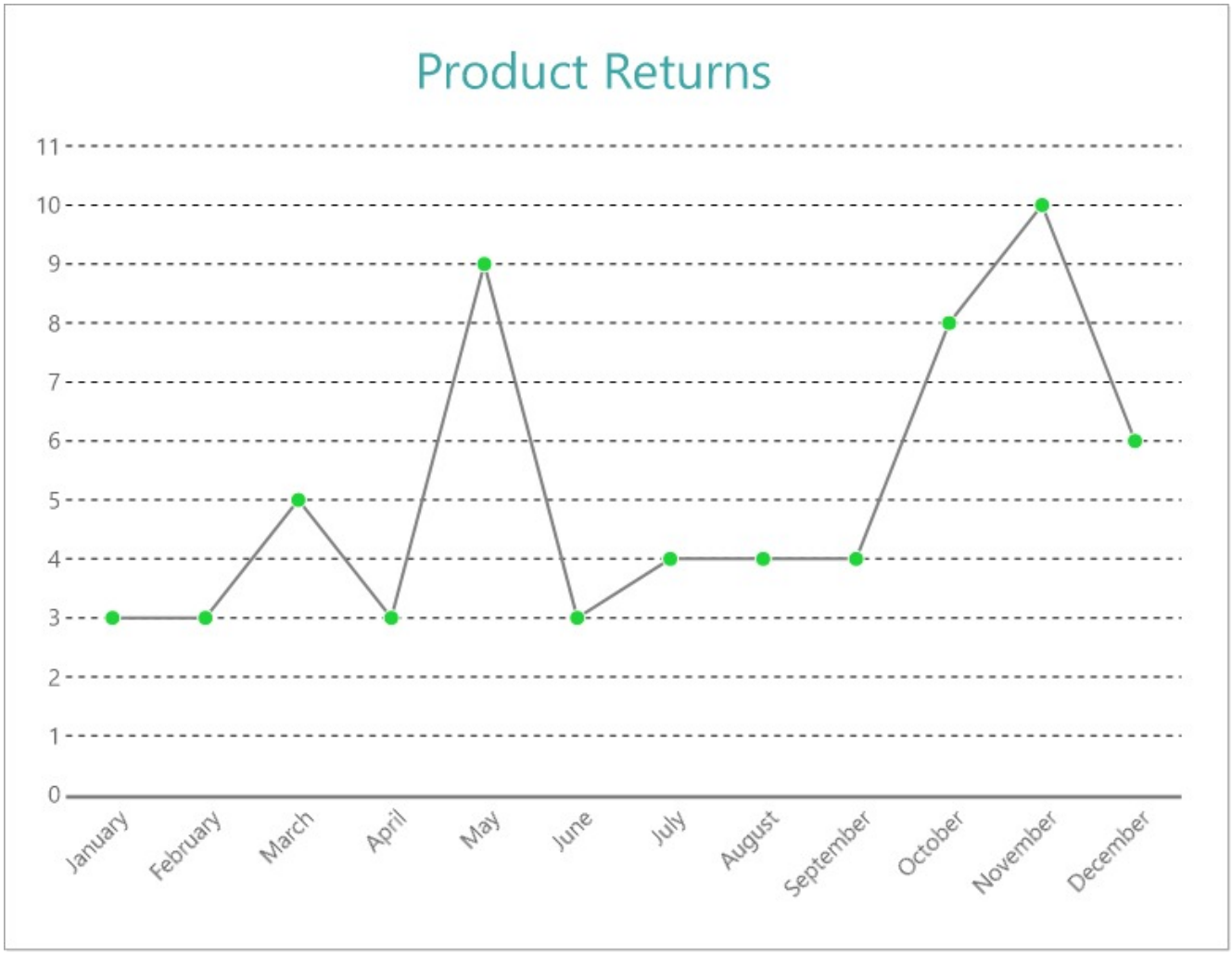
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Single Line Chart

This walkthrough creates a Single Line Chart. The chart shows the trend in the return quantity over the year. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL: https://demodata.mescius.io/contoso/odata/v1/FactSales For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Click **OK** to save the changes.
4. Go to the **Fields** page to view the available fields and modify the **Name** of the `[DateKey]` and `[ReturnQuantity]` fields to `[Sales Date]` and `[Return Quantity]`, respectively.

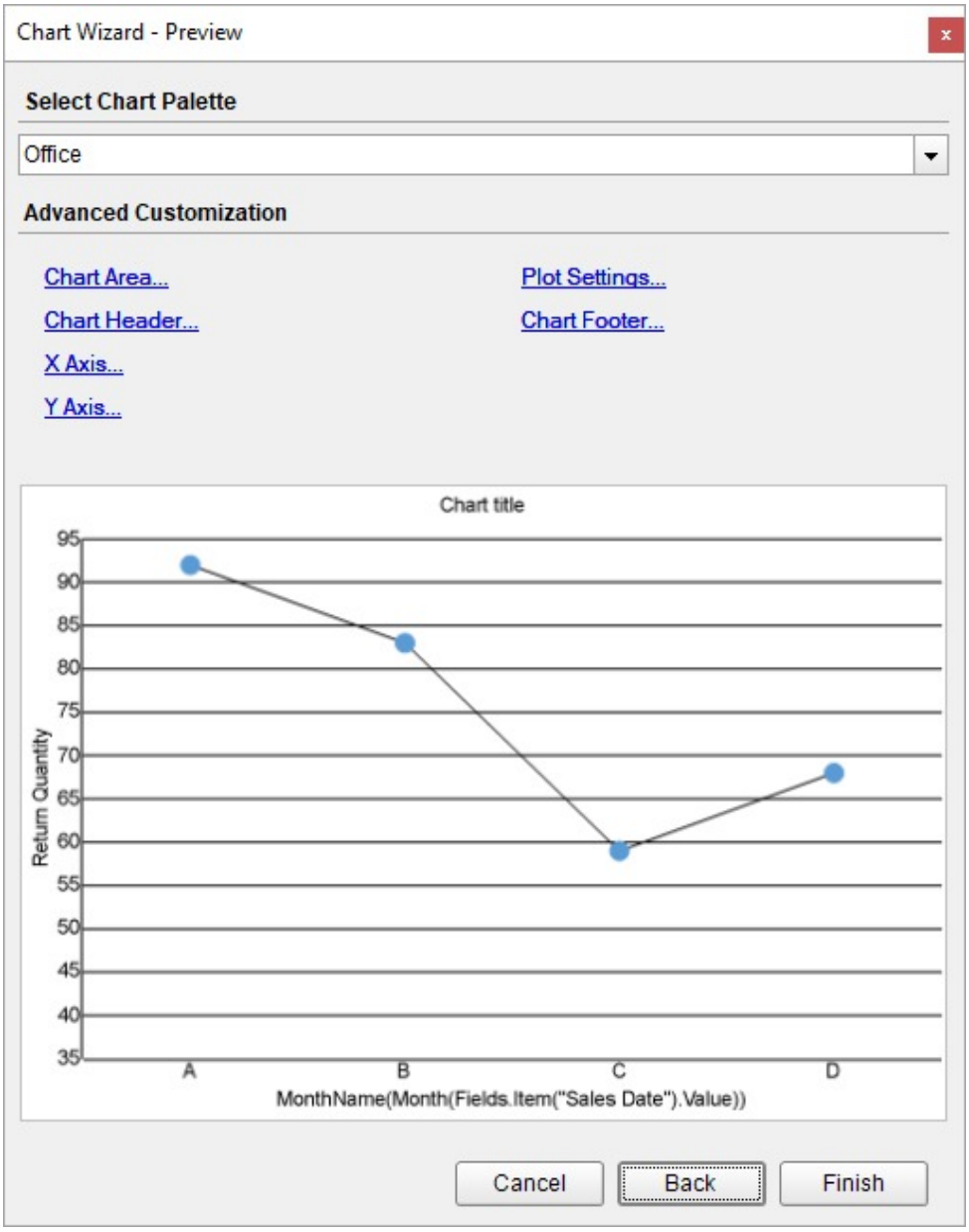
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Line'.
3. Click **Next** to proceed. Here, you need to specify the line settings. We will define a data series value to display the return quantity values along the horizontal axis.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[Return Quantity]`	Sum

5. In **Choose Data Categories**, set **Field** to `=MonthName(Month([Sales Date]))`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the month names to display in the order of increasing month numbers. So fill in the following settings:

Field	Settings
Sorting field	`=Month([Sales Date])`
Sorting direction	Ascending

3. Go to the **Appearance** page and set the following properties.
    - **Line Style > Width**: 1.5pt
    - **Line Style > Color**: #808080
    - **Symbol Settings > Shape**: Dot
    - **Symbol Settings > Background Color**: #1fd537
    - **Symbol Border Settings > Style**: Solid
    - **Symbol Border Settings > Color**: White
4. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties.
    - **Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Linear
    - **Minimum scale value**: 0
    - **Maximum scale value**: 11
7. Click **OK** to complete setting up the Y-axis.

X-Axis

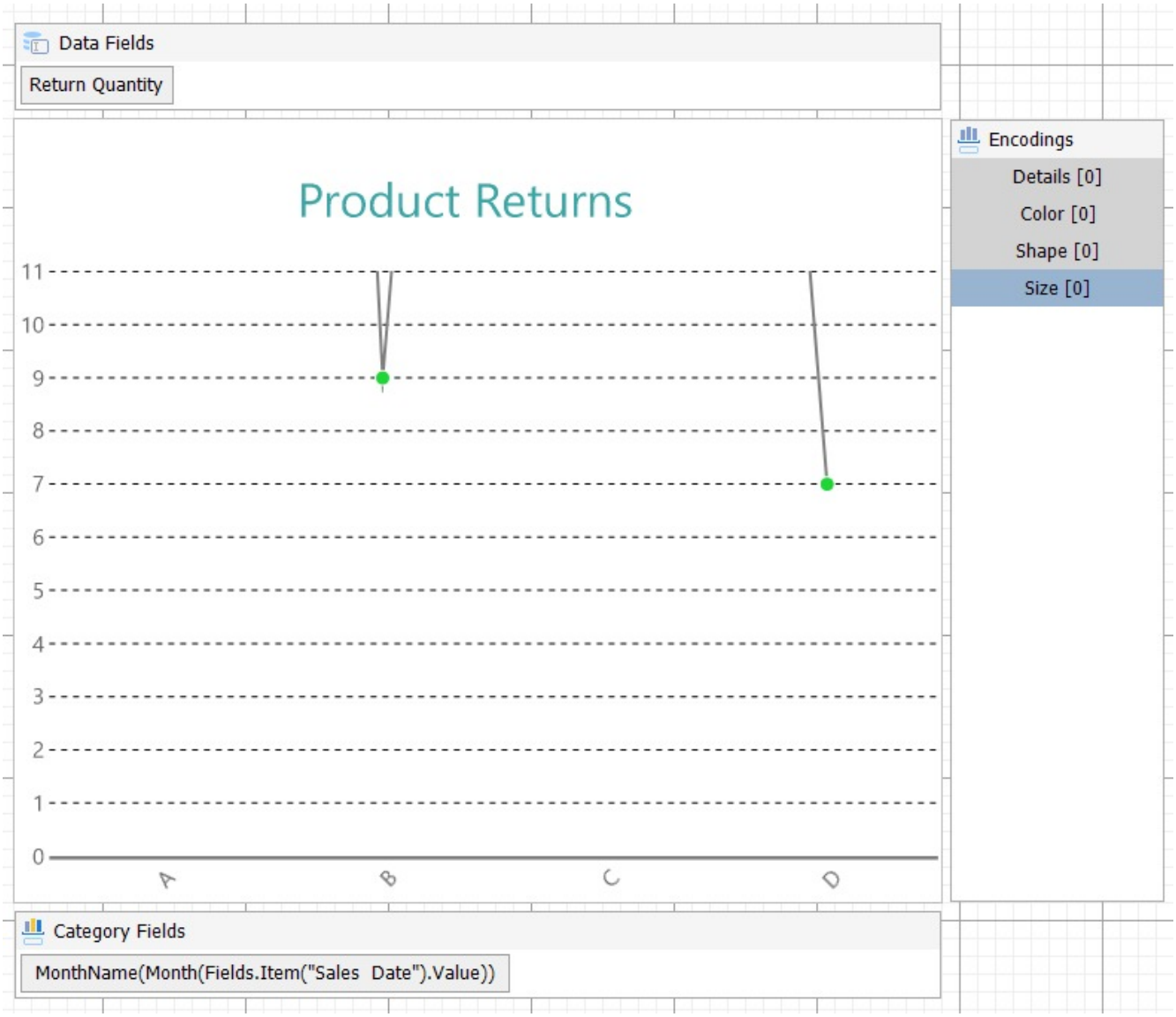
1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-45'.
4. Now go to the **Appearance** tab and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
6. Click **OK** to complete setting up the X-axis.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Product Returns'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt

- **Color**: #3da7a8

4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

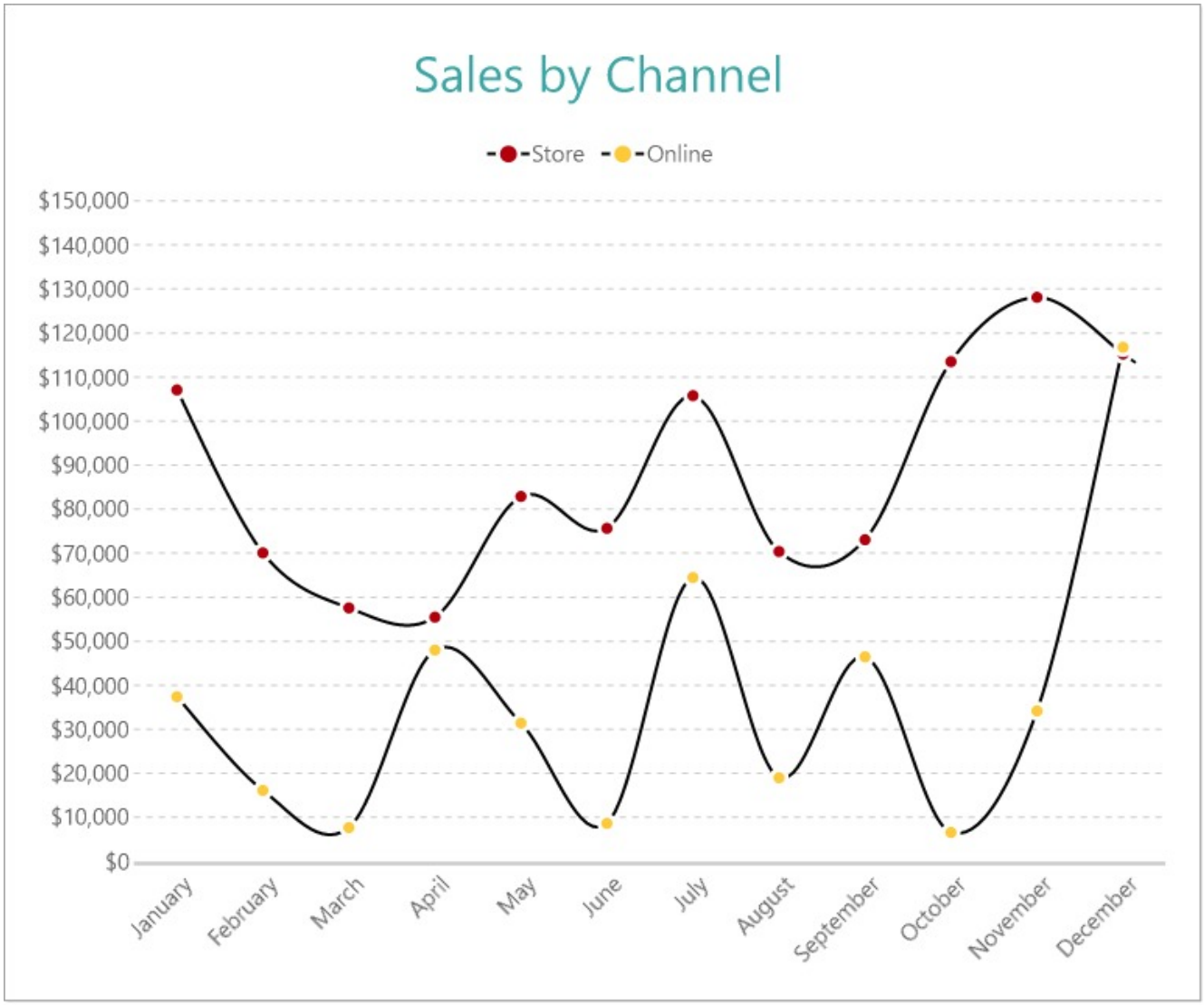


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Multiple Line Chart

This walkthrough creates a Multiple Line Chart. The chart shows the sales trend for 'Store' and 'Online' over a year. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields and modify the **Name** of the `[DateKey]` field to `[Sales Date]`.
4. On the same page, add one calculated field:

Name	Value
Channel Name	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online")`

5. Go to the **Filters** page and add a new filter value, and set its properties as below.

Expression	Operator	Values
`=[ChannelKey]`	In	=1 =2

6. Click **OK** to save the changes.

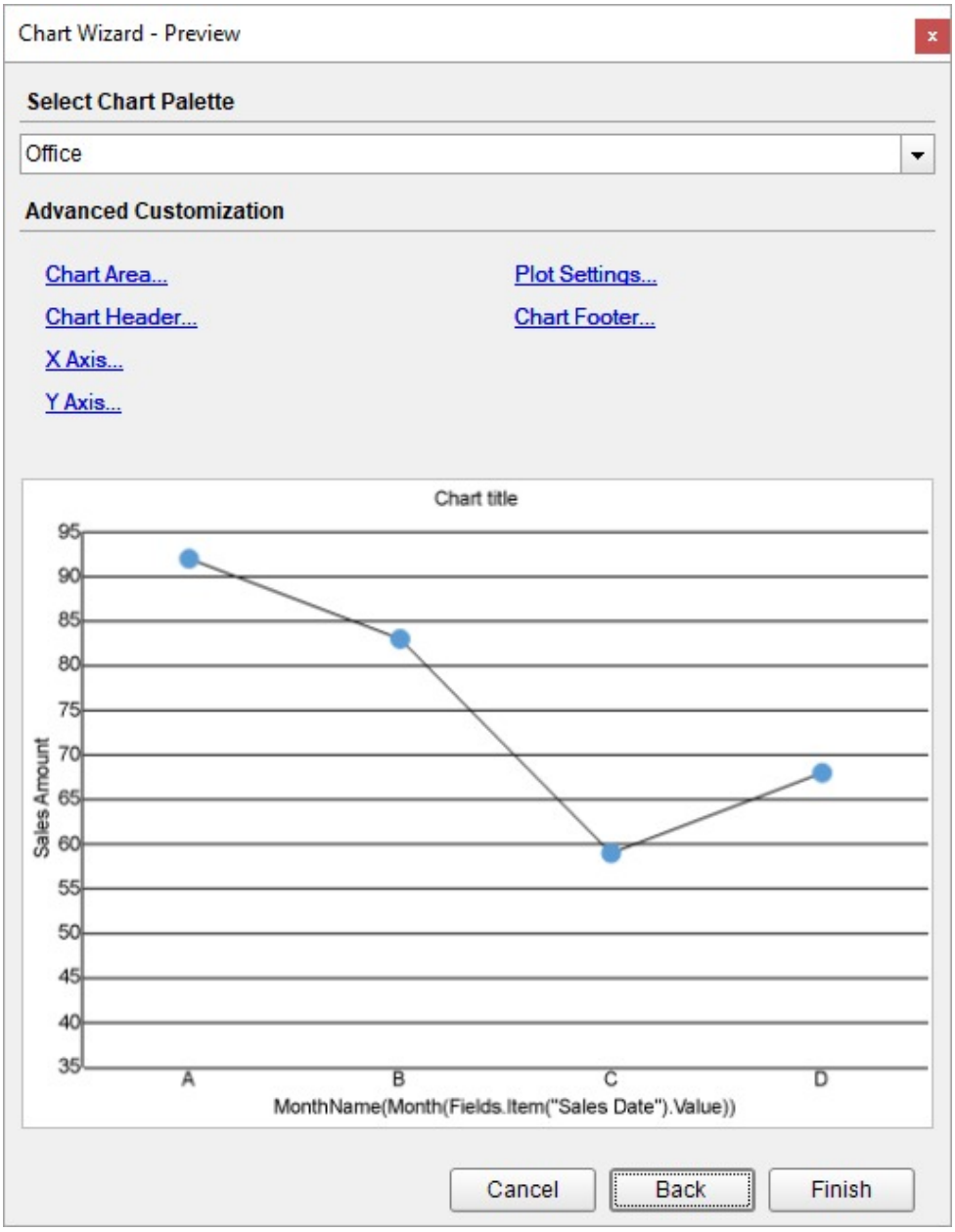
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Line'.
3. Click **Next** to proceed. Here, you need to specify the line settings. We will define a data series value to display the sales amount values across the vertical axis.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Categories**, select set **Field** to `=MonthName(Month([Sales Date]))`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the month names to display in chronological order. So fill in the following settings:

Field	Settings
Sorting field	`=Month([Sales Date])`
Sorting direction	Ascending

3. Go to the **Encodings** page.
4. On the **Details** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Channel Name]` to display the sales amount for 'Online' and 'Store' channels.
    2. Under Grouping, set **Group** to 'Cluster'.
5. Then, navigate to the **Color** tab, add a new value and set the **Expression** to `=[Channel Name]`. This will display the legend based on the channel names in the chart.
6. Go to the **Appearance** page and set the following properties.
    - **Line Style > Width**: 1.5pt
    - **Line Style > Color**: Black
    - **Symbol Settings > Shape**: Dot
    - **Symbol Border Settings > Style**: Solid
    - **Symbol Border Settings > Color**: White
    - **Symbol Border Settings > Width**: 2pt
7. Click **OK** to complete setting up the plot.
8. With 'Plot-Plot1' selected, go to the Properties window and set the **Line Aspect** property for the plot to 'Spline' to display curved lines in the chart.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal points)'.
4. Now, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 10000
    - **Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Linear
    - **Minimum scale value**: 0
    - **Maximum scale value**: 150000
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-45'.
4. Now, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.

    - **Color:** #cccccc
    - **Width:** 2pt
6. Click **OK** to complete setting up the X-axis.

Legend - Color

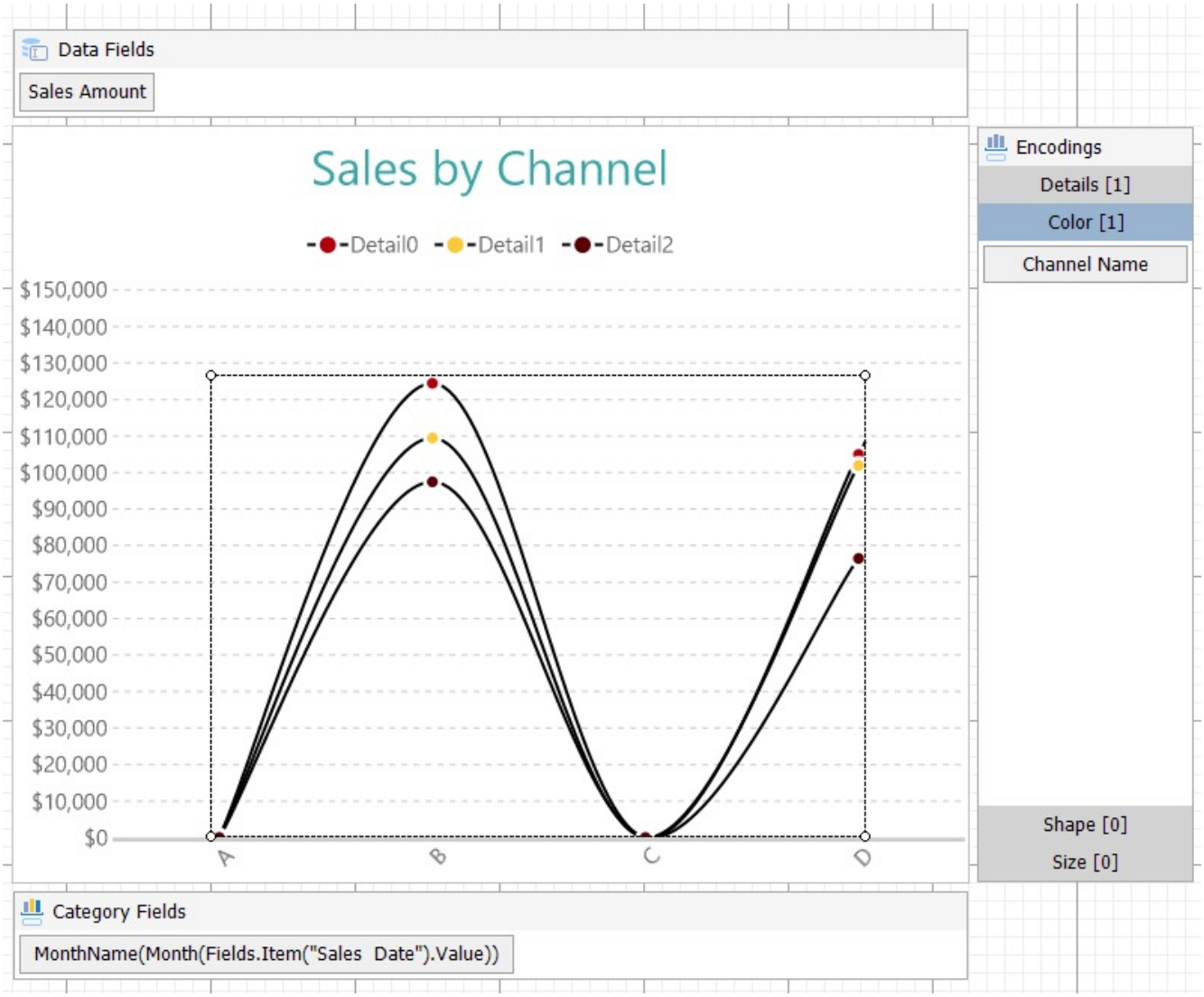
1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: DimGray
4. Click **OK** to complete setting up the chart legend.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. On the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #b3000c
    - #ffca3a
3. Click **OK** to complete setting up the custom palette.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales By Channel'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Pie and Doughnut Charts

Pie charts are circular charts that display the contribution of each category which is represented by a slice. Each and every slice represents a category of data that makes up the whole pie. A pie chart is able to provide information at a glance itself. For example, you can use a pie chart to compare growth areas within a business such as profit and turnover.

A doughnut chart is like a pie chart with a center hole. It encodes the data values into rings divided into segments. For instance, a donut chart can be used to display a survey of book sales in various genres in a bookstore.

Single Value Pie Chart

A Single Value Pie chart visualizes parts of a single data value. For instance, the Single Value Plot can be used to show

the contribution of product categories in Net Sales. See Create Pie Chart walkthrough to learn how to create this chart.

Multiple Value Pie or Doughnut Chart

A Multiple Value Pie or Doughnut chart can be used to display various values divided into common parts using a Doughnut or Pie plot. For example, the Doughnut Plot can be used to show the contribution of product categories into Discounts, Returns, and Unit Cost. See Create Doughnut Chart walkthrough to learn how to create a Multiple Value Doughnut Chart.

Circular Bar Chart

A Circular Bar chart places doughnut segments adjacent to each other. For instance, the Circular Bar Chart can be used to depict the contribution of sales channels into Net Income. See Create Circular Bar Chart walkthrough to learn how to create a this chart.

Pie and Doughnut Plot Properties

The Pie and Doughnut Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the Pie Plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each Pie Chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the Pie Plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the Pie Plot.
  - Center: Positions the data label text on the center of the Pie Chart.
  - Inside: Positions the data label text inside the Pie Chart.
  - Outside: Positions the data label text outside the Pie Chart.
  - Auto: The default setting, same as Outside for Pie Chart.

LineStyle

The line style for the borders. Includes LineColor, LineStyle, and LineWidth properties for customization.

- **LineColor**: Specify the color of the border around the segments or slices.
- **LineStyle:** Specify the line style of the border around the segments or slices as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the segments or slices.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Pie Plots.

InnerRadius

Inner radius of the plot is the percentage relative to the chart's outer radius. It defines the hole size in the center of the disk

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default**: Indicates a straight line.
- **Spline**: Indicates a bezier curve.
- **Step Center, Step Left, and Step Right**: Indicates a stepped line with different step directions.

Opacity

The Opacity determines the opacity of areas filled with color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules

topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

StartAngle

Indicates the arc angle that defines the clockwise rotation of the plot. A full rotation makes 360 degrees.

Sweep

Indicates the arc degrees from 0 to 360 which decides the length of the arc occupied by the plot.

UnpivotData

Determines whether to display multiple data fields as a single data field. By default, the property is set to 'False'.

Design

BarSettings

The BarSettings specifies the bar-style settings.

- **NeckHeight**: Determines the bar neck height in percent.
- **BottomWidth**: Determines the Bottom width in percent.
- **Overlap**: Determines the Percentage bar overlap.
- **TopWidth**: Determines the Top width in percent.
- **Width**: Determines the bar width in percent.

Encodings

Color Encoding

The Color Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Pie Plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked Pie Plot.
  - Cluster: You can use this value to configure subsections that overlap each other.
  - None: Equals to Cluster.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Pie Plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Pie Plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

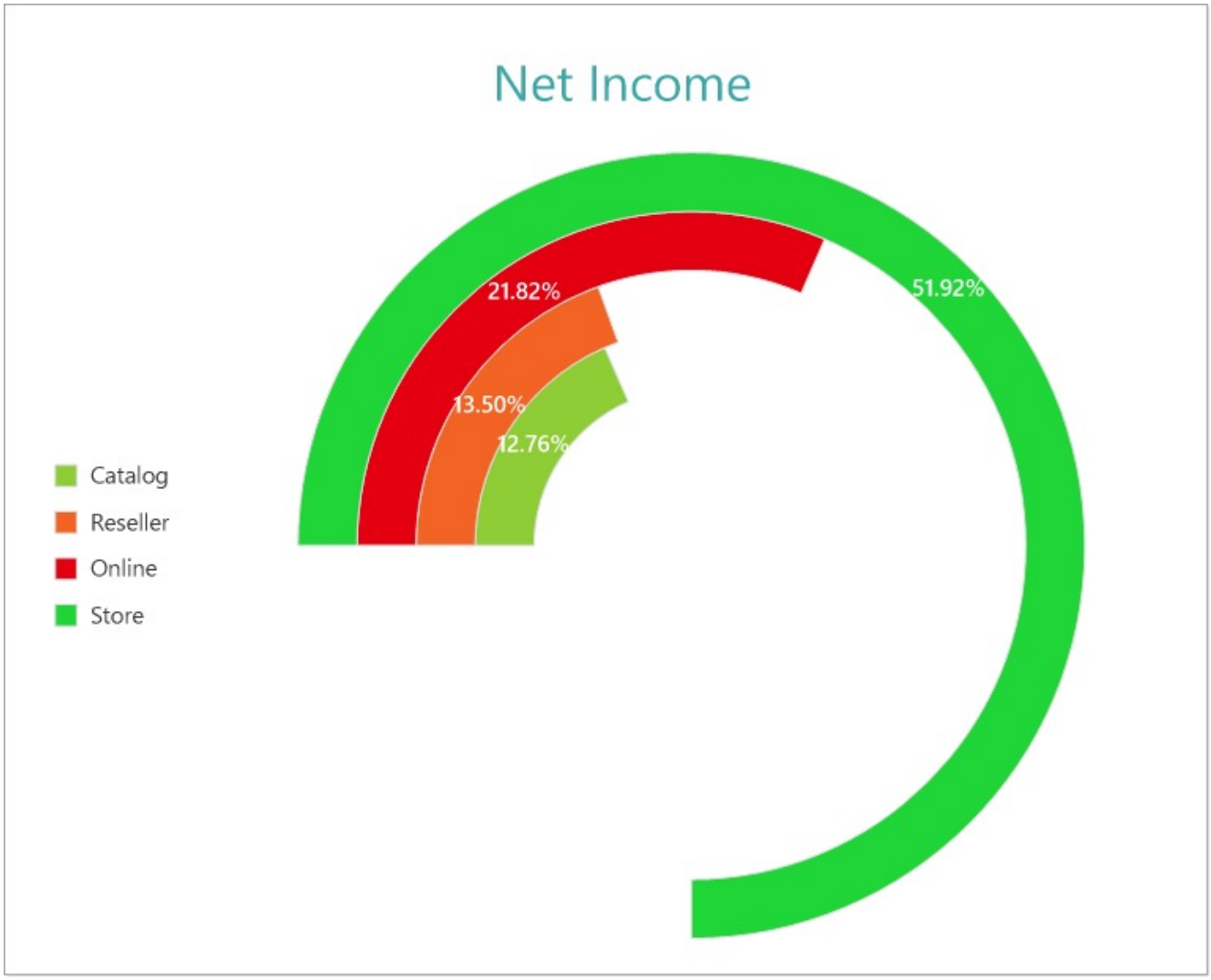
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Circular Bar Chart

This walkthrough creates a Circular Bar Chart. The chart shows the percentage values that each sales channel contributes to the net income. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.

2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource**  icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields. On the same page, add two calculated fields:

Name	Value
Net Income	`=[SalesAmount] - [UnitCost] - [DiscountAmount] - [ReturnAmount]`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

4. Click **OK** to save the changes.

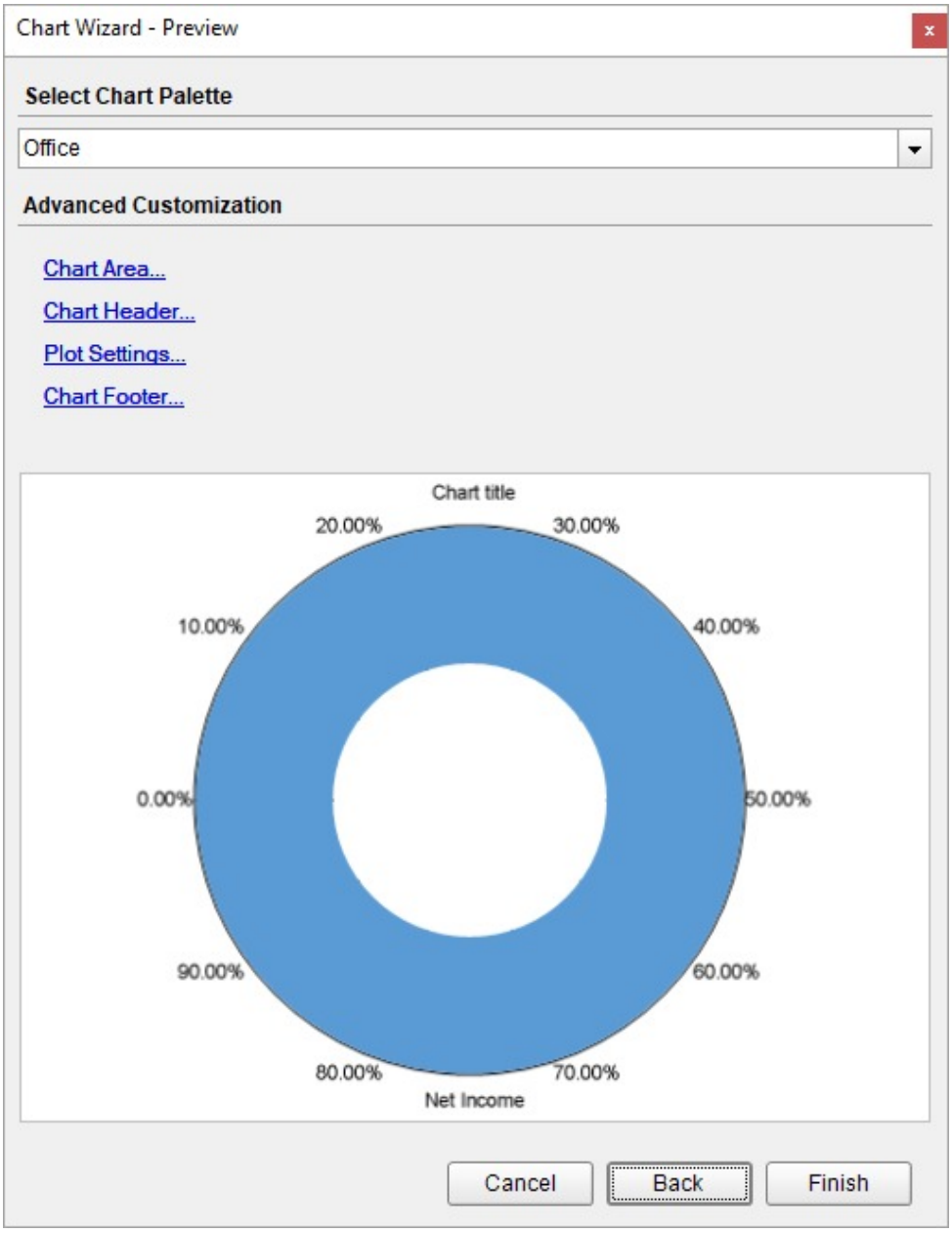
Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Doughnut'.
3. Click **Next** to proceed. Here, you need to specify the doughnut settings. We will define a data series value to display the net income for each sales channel.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[Net Income]`	Sum

5. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Encodings** page.
3. On the **Detail** tab, add a new value, and set its properties as below.

    1. Set **Expression** to `=[Sales Channel]` to display the net income for the store, reseller, online, and catalog.
    2. Under **Grouping**, set **Group** to 'Cluster'.
    3. Under **Sorting**, set **Sorting** field to `=[Net Sales]`, **Sort** direction to 'Ascending', and **Sorting aggregate** to 'Sum'. This will arrange the subcategories in the increasing order of their net sales in the cluster.
4. Navigate to the **Color** page, add a new value, and set the **Expression** to `=[Sales Channel]` to display legend based on the sales channels in the chart.
5. Go to the **Labels** page > **General** tab and set the following properties.
    - **Template**: `{PercentageCategory:p2}`
    - **Text Position**: Center
6. Go to the **Appearance** tab and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Weight**: SemiBold
    - **Font > Color**: #ffffff
    - **Connecting Line > Style**: Solid
    - **Connecting Line > Color**: Gainsboro
7. Click **OK** to complete setting up the plot.
8. With 'Plot-Plot1' selected, go to the Properties pane and
    1. set the **Inner Radius** property for the doughnut to '40%' to reduce the inner size.
    2. set the **Sweep** property to '270' to change the arc length for the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option as we are already displaying the percentage values of each sales channel in the chart.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the Y-axis.

Chart Palette

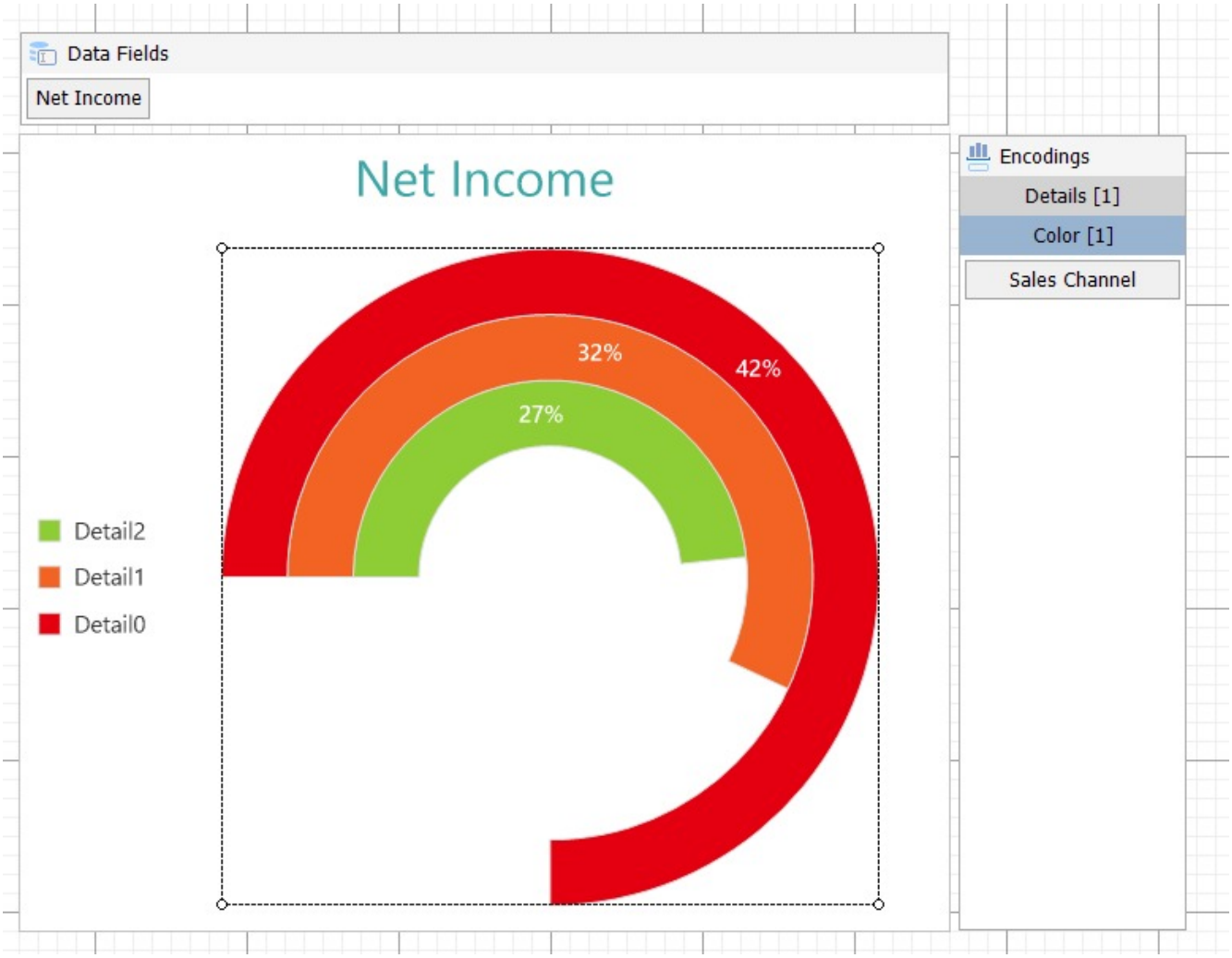
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #8fcd37
    - #f26324
    - #e40010
    - #1fd537
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: #3c3c3c
3. Go to the **Layout** page and set the **Position** to 'Left'.
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog**.
2. Go to the **General** page and set **Title** to 'Net Income'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



> **Note**: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Doughnut Chart

This walkthrough creates a Doughnut Chart. The chart shows the contribution of sales channels to the total unit cost, return, and discount. The final chart appears like this:

Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:

https://demodata.mescius.io/contoso/odata/v1/FactSales
For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Click **OK** to save the changes.
4. Go to the **Fields** page to view the available fields and modify the **Name** of the `[ReturnAmount]` and `[DiscountAmount]` fields to `[Returns]` and `[Discount]`, respectively.
5. On the same page, add two calculated fields:

Name	Value
Net Income	`=[SalesAmount] - [UnitCost] - [Discount] - [Returns]`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

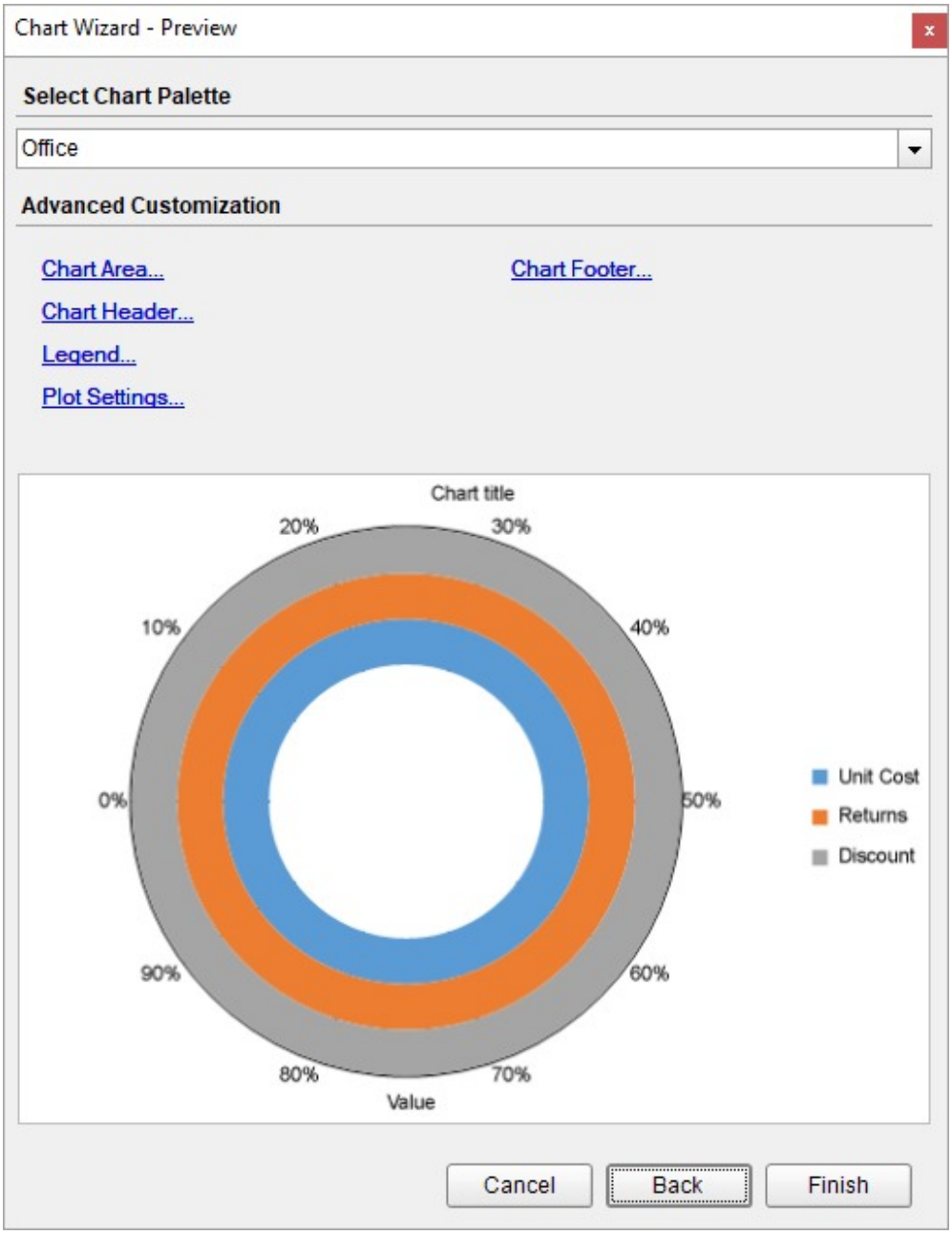
Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Doughnut'.
3. Click **Next** to proceed. Here, you need to specify the doughnut settings.
4. In **Choose Data values** section, we will define three data series values to display the sales amount, unit cost, and discount for the sales channels.
   Select the following three fields from the drop-down and set the corresponding aggregates:

Field	Aggregate
`=[UnitCost]`	Sum
`=[Returns]`	Sum
`=[Discount]`	Sum

5. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Encodings** page.

3. On the **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Sales Channel]` to display sales data for the subcategories i.e. store, reseller, online, and catalog.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to display the subcategories (i.e. sales data by store, reseller, online, and catalog) in a stack.
4. Navigate to the **Color** page, add a new value, and set the **Expression** to `=[Sales Channel]` to display legends based on the subcategories.
5. Go to the **Labels** page > **General** tab and set the following properties:
    - **Template**: `{valueField.name}`
    - **Text Position**: Center
6. Go to the **Appearance** tab and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Weight**: SemiBold
    - **Font > Color**: #ffffff
    - **Connecting Line > Style**: Solid
    - **Connecting Line > Color**: Gainsboro
7. Click **OK** to complete setting up the plot.
8. With 'Plot-Plot1' selected, go to the Properties pane and set the **Inner Radius** property for the doughnut to '20%' to reduce the inner size.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option as we are already displaying the sales channel labels in the chart.
4. Go to the **Line** page and set the following properties:
    - **Color**: Gainsboro
    - **Width**: 0.25pt
5. Click **OK** to complete setting up the Y-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #8fcd37
    - #f26324
    - #e40010
    - #1fd537
3. Click **OK** to complete setting up the custom palette.

Legend - Color

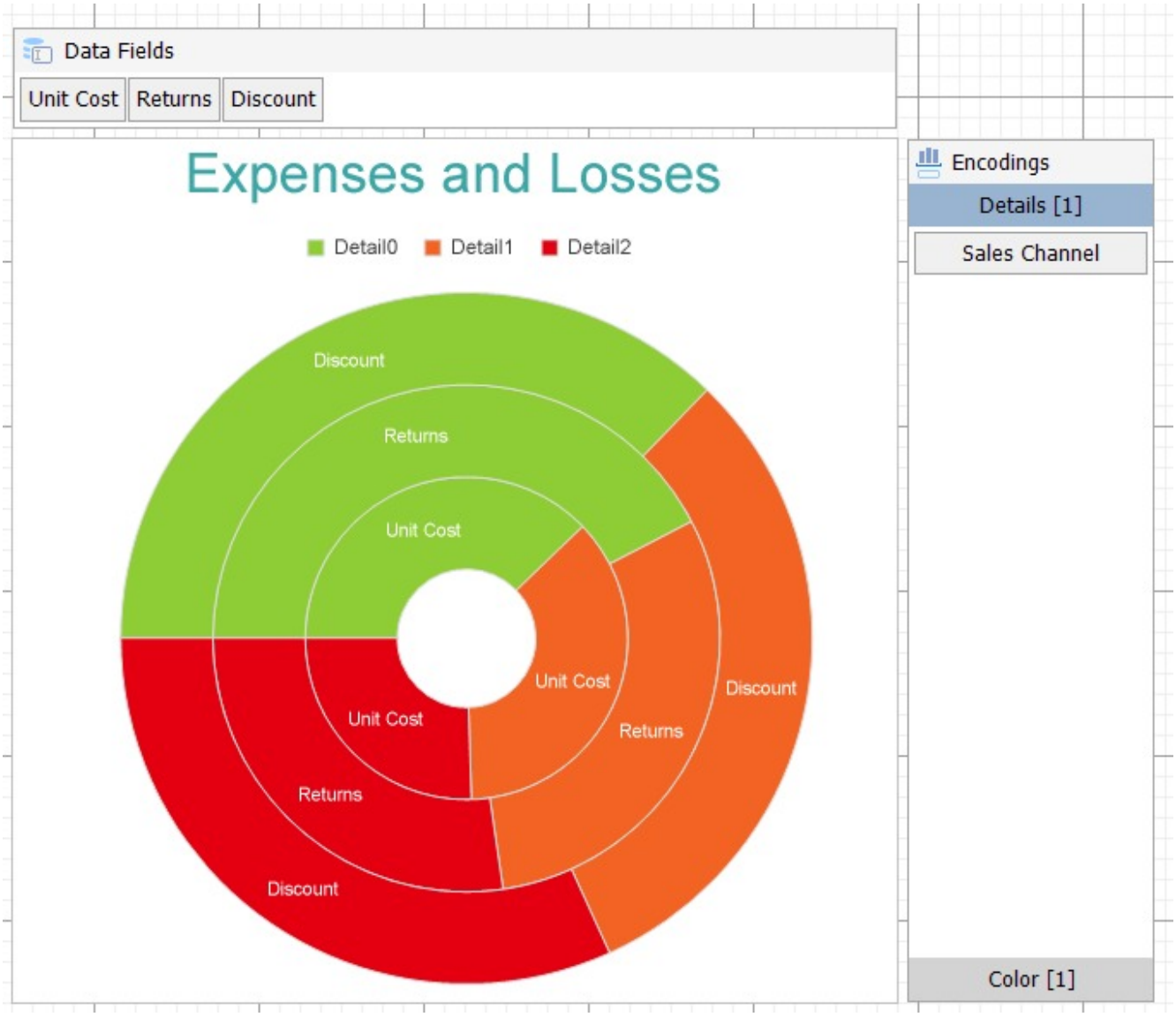
1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose Property Dialog**.**
2. Go to the **Appearance** page and set the following properties:
    - **Font > Size**: 10pt
    - **Font > Color**: #3c3c3c
3. Go to the **Layout** page and set the following properties:
    - **Position**: Top

- **Orientation**: Horizontal

4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Expenses and Losses'.
3. Go to the **Font** page and set the properties as below.
   - **Size**: 24pt
   - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

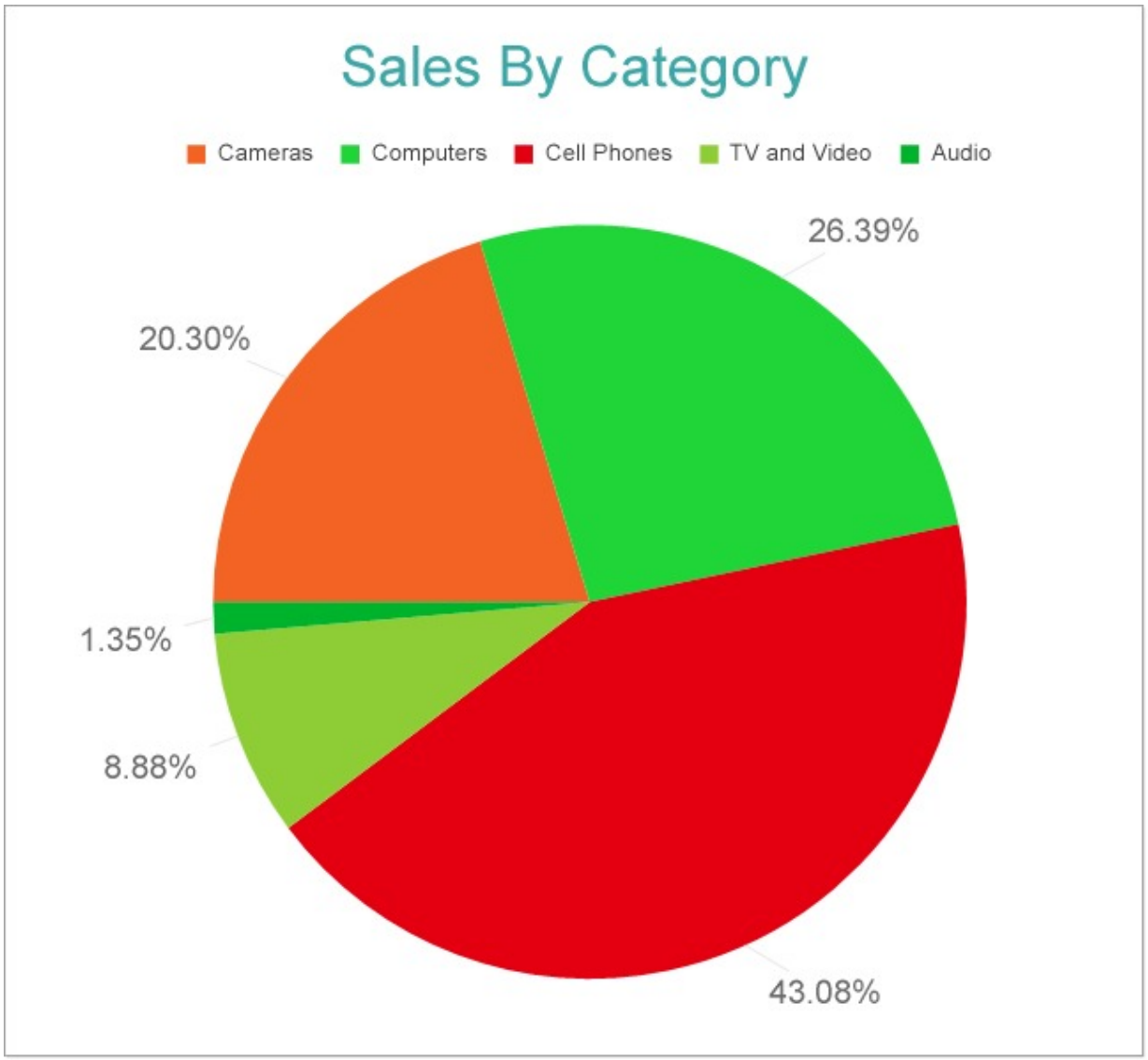


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Pie Chart

This walkthrough creates a Pie Chart. The chart shows how the total sales amount is divided between different product categories such as cameras, computers, cell phones, TV and video, and audio. Each slice in the chart represents the percentage values that each part of the category contributes. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.

3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

```
$.value[*]
```

3. Go to the **Fields** page to view the available fields. On the same page, add following calculated field:

Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

4. Click **OK** to save the changes.

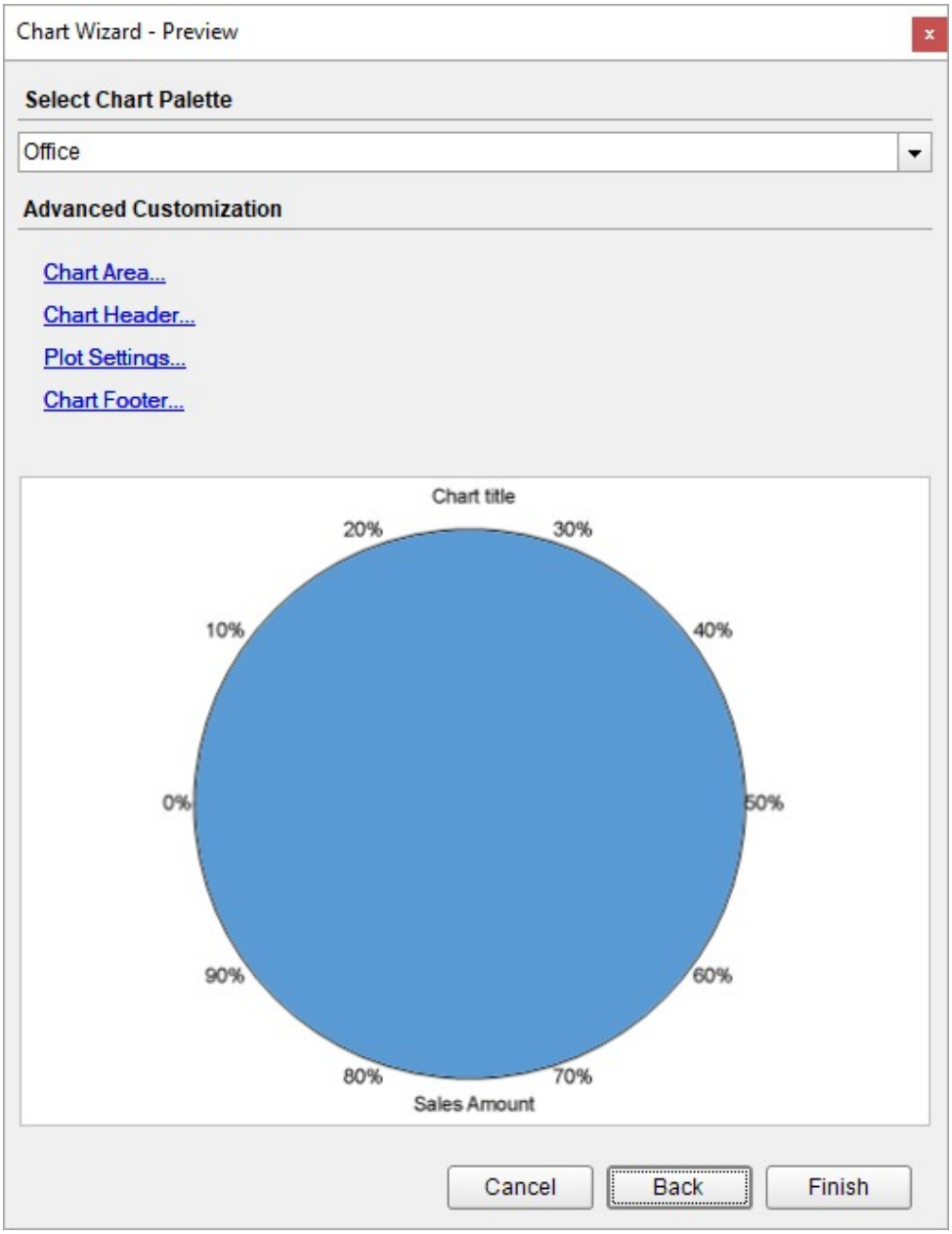
Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Pie'.
3. Click **Next** to proceed. Here, you need to specify the pie settings. We will define a data series value to show the sales amount values in a pie.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Encodings** page >**Detail** tab, add a new value, and set the **Expression** to `=[Product Category]` to display sales amount for different sales categories like cell phones, computers, cameras, TV, video, and audio.

3. In the same tab under Grouping, set **Group** to 'Stack'.
4. Navigate to the **Color** page, add a new value, and set the **Expression** to `=[Product Category]` to display legends based on the subcategories.
5. Go to the **Labels** page > **General** tab and set the following properties.
    - **Template**: `{PercentageCategory:p2}`
    - **Text Position:** Outside
    - **Offset:** 16
6. Now go **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Color:** DimGray
    - **Font > Size:** 14pt
    - **Connecting Line > Style**: Solid
    - **Connecting Line > Color**: #e6e6e6
    - **Connecting Line > Width**: 0.25pt
7. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**
2. Go to the **Title** page and remove the text from the **Title** field to hide the axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option as we are already displaying the product category labels in the chart.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the Y-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
3. Click **OK** to complete setting up the custom palette.

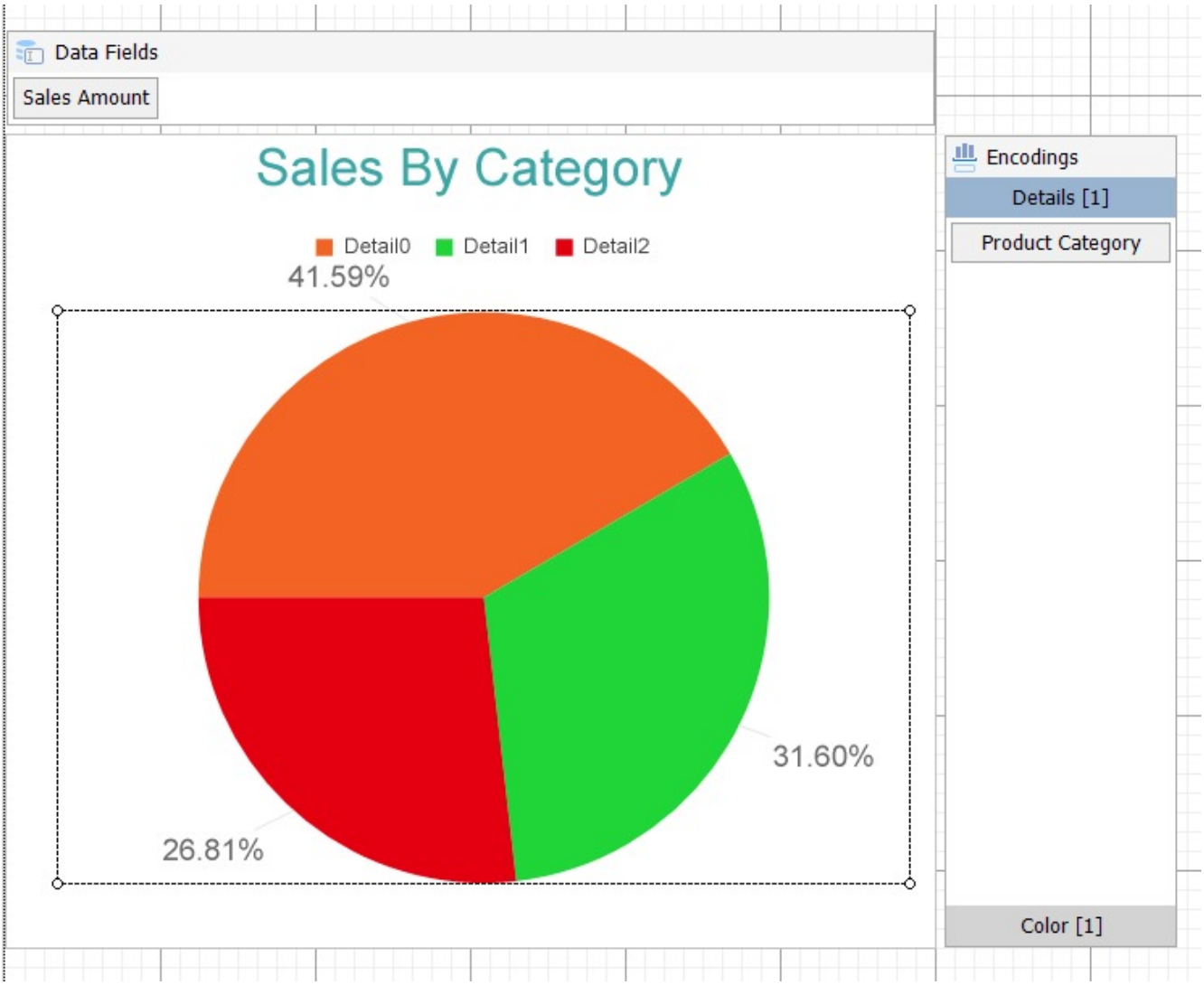
Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose Property Dialog**.**
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: #3c3c3c
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales By Category'.

3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

5. Once you are done with configuring and customizing the chart, preview the report.

Scatter and Bubble Charts

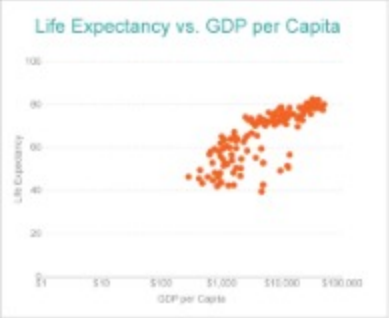
Scatter and Bubble plots are chart types used to visualize relationships between value points. Both plots arrange data values along the vertical and horizontal axes and display points called Symbols on the intersections.

Scatter Chart

A scatter chart visualizes relationships between two variables. The Scatter graph uses dots to represent two different numeric variables. For example, the scatter chart can be used to depict the correlation between variables like GDP per capita and life expectancy.

Simple Scatter Chart

A Simple Scatter Plot visualizes relationships between two variables. For instance, the Simple Scatter Charts can be used to depict the correlation between life expectancy and GDP per Capita. See Create Scatter Chart walkthrough to learn how to create this chart.

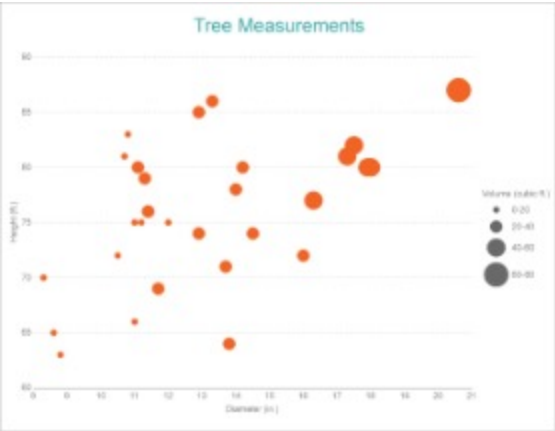


Bubble Chart

Bubble charts are used while plotting data points in terms of three numeric parameters. Note that the third numeric parameter is represented by the bubble's diameter. A bubble plot simply adds the third variable by encoding it into the size of the displayed symbols. For instance, the bubble chart can be used to depict the correlation between the diameter of a tree, its height, and volume.

Simple Bubble Chart

A Simple Bubble chart adds a third variable into visualization by encoding it into the size of the showcased symbols. For instance, the simple bubble chart can be used to display the correlation between a tree's diameter, its height and volume. See Create Bubble Chart walkthrough to learn how to create this chart.



Multi-Category Scatter and Bubble Charts

The Multi-Category Scatter and Bubble chart splits the data values into categories and visualizes them using the shape

and color of the symbol. For example, the multi-category scatter chart can be used to display the correlation between the life expectancy and GDP per capita categorized by the continent.

Scatter and Bubble Plot Properties

The Scatter and Bubble Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when the Scatter Plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each area chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the Scatter Plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or

a LineThrough.
- **TextPosition**: The position of the data label text relative to the plot.
  - Center: Positions the data label text on the center of the chart.
  - Inside: Positions the data label text inside the chart.
  - Outside: Positions the data label text outside the chart.
  - Auto: The default setting, same as Outside.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Symbols

Represents the properties that allow you to customize the look of symbols that form the Scatter plot.

- **BackgroundColor**: Change the color of the background.
- **LineColor**: Change the color of the line.
- **LineStyle**: Choose from different styles of line, such as Dotted, Dashed, Solid, Double, Groove, etc.
- **LineWidth**: Select the width of the lines in points.
- **Shape**: Choose from different shapes of symbols such as Dot, Box, Diamond, Triangle, X, Dash, Plus, etc.
- **Visible**: Set to true to display data point symbols.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Scatter Plots.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

SymbolOpacity

Represents the opacity value of symbol fill color.

Encodings

Category Encoding

The Category Encoding of a Scatter Plot is a set of properties that determine the period over which the plot generates connected data points representing those above Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Color Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Scatter Plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the area subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked Scatter Plot.
  - Cluster: You can use this value to configure area subsections that overlap each other.
  - None: Equals to Cluster.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the

subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Shape Encoding

The Shape Encoding enables the shape legend of the Details or Category Encoding in Scatter charts. It includes the Aggregate function and the shape expression, which is elaborated below:

Action

The action to take when the shape legend is clicked.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Value

The Value property is the collection and usually takes a bound field. However, the Scatter plots take the first item from the collection.

Size Encoding

The Size Encoding enables the Aggregate function and Size expression in Bubble charts to enable the size legend. The Size Encoding works solely with numeric values and breaks down data values into ranges that determine the symbol size. You can also set the **Action** to take when the size legend is clicked.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Scatter Plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Scatter Plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

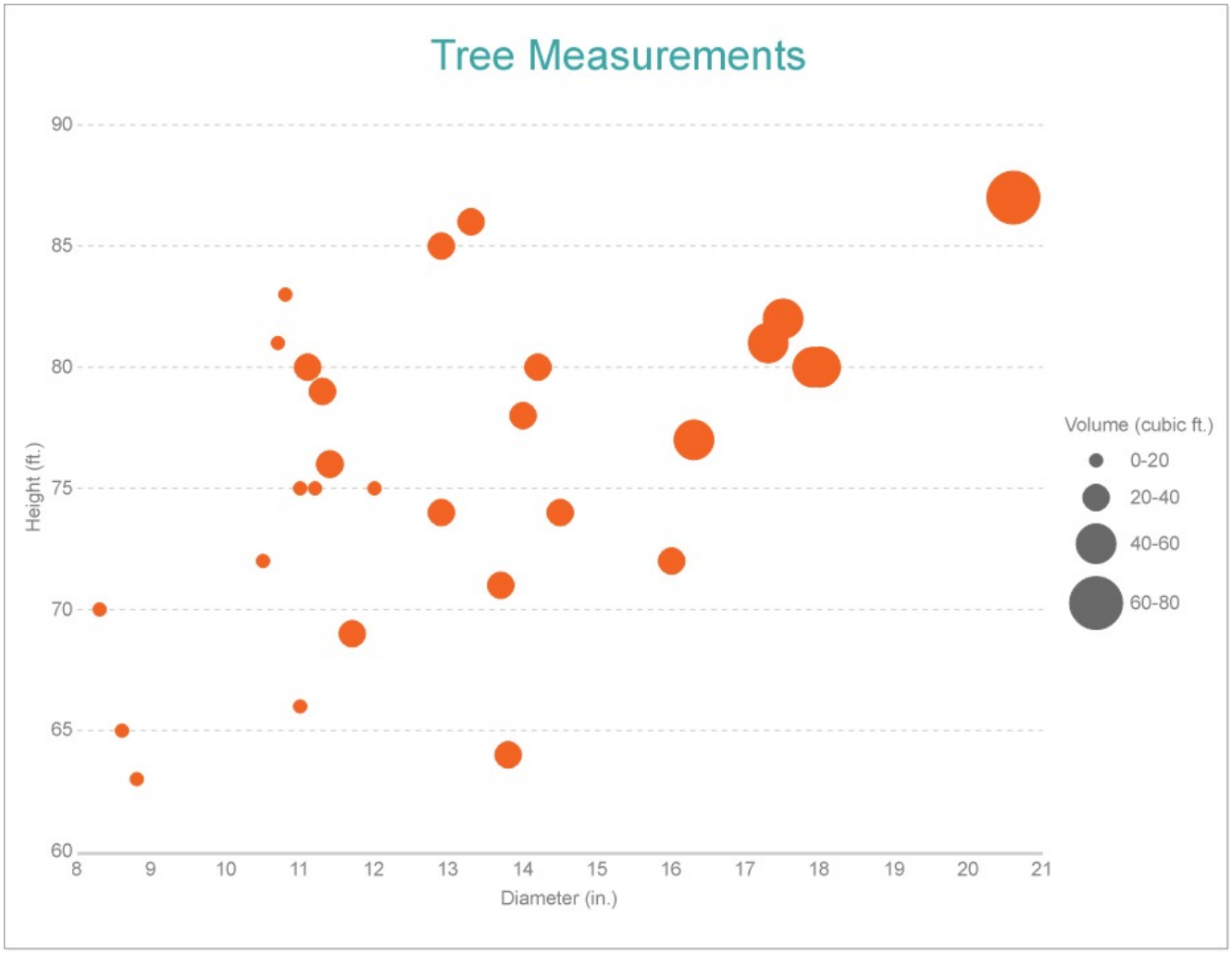
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Bubble Chart

This walkthrough creates a Bubble Chart. The chart shows the correlation between the height, diameter, and volume of the trees. The final chart appears like this:



Create a Report

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
  {
    "Girth": 8.3,
    "Height": 70,
    "Volume": 10.3
  },
  {
    "Girth": 8.6,
    "Height": 65,
    "Volume": 10.3
  },
  {
    "Girth": 8.8,
    "Height": 63,
    "Volume": 10.2
  },
  {
    "Girth": 10.5,
    "Height": 72,
    "Volume": 16.4
  },
  {
    "Girth": 10.7,
    "Height": 81,
    "Volume": 18.8
  },
  {
    "Girth": 10.8,
    "Height": 83,
    "Volume": 19.7
  },
  {
    "Girth": 11,
    "Height": 66,
    "Volume": 15.6
  },
```

```
{
  "Girth": 11,
  "Height": 75,
  "Volume": 18.2
},
{
  "Girth": 11.1,
  "Height": 80,
  "Volume": 22.6
},
{
  "Girth": 11.2,
  "Height": 75,
  "Volume": 19.9
},
{
  "Girth": 11.3,
  "Height": 79,
  "Volume": 24.2
},
{
  "Girth": 11.4,
  "Height": 76,
  "Volume": 21
},
{
  "Girth": 11.4,
  "Height": 76,
  "Volume": 21.4
},
{
  "Girth": 11.7,
  "Height": 69,
  "Volume": 21.3
},
{
  "Girth": 12,
  "Height": 75,
  "Volume": 19.1
},
{
  "Girth": 12.9,
  "Height": 74,
  "Volume": 22.2
},
{
  "Girth": 12.9,
  "Height": 85,
  "Volume": 33.8
},
{
```

```
  "Girth": 13.3,
  "Height": 86,
  "Volume": 27.4
},
{
  "Girth": 13.7,
  "Height": 71,
  "Volume": 25.7
},
{
  "Girth": 13.8,
  "Height": 64,
  "Volume": 24.9
},
{
  "Girth": 14,
  "Height": 78,
  "Volume": 34.5
},
{
  "Girth": 14.2,
  "Height": 80,
  "Volume": 31.7
},
{
  "Girth": 14.5,
  "Height": 74,
  "Volume": 36.3
},
{
  "Girth": 16,
  "Height": 72,
  "Volume": 38.3
},
{
  "Girth": 16.3,
  "Height": 77,
  "Volume": 42.6
},
{
  "Girth": 17.3,
  "Height": 81,
  "Volume": 55.4
},
{
  "Girth": 17.5,
  "Height": 82,
  "Volume": 55.7
},
{
```

```
    "Girth": 17.9,
    "Height": 80,
    "Volume": 58.3
  },
  {
    "Girth": 18,
    "Height": 80,
    "Volume": 51.5
  },
  {
    "Girth": 18,
    "Height": 80,
    "Volume": 51
  },
  {
    "Girth": 20.6,
    "Height": 87,
    "Volume": 77
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'Measurements'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

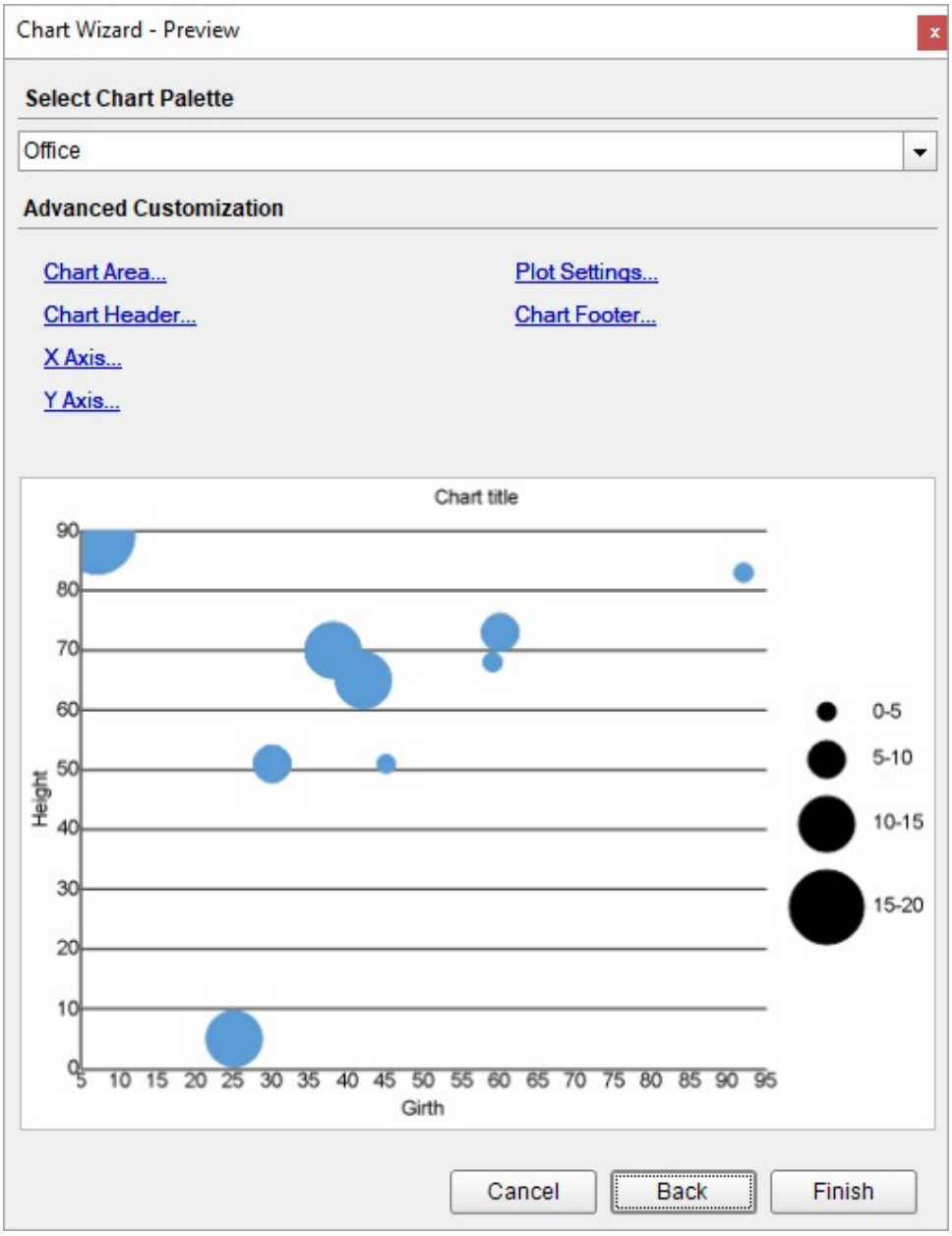
Query
`$.[*]`

3. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Measurements' and the **Chart Type** as 'Bubble'.
3. Click **Next** to proceed. Here, you need to specify the bubble settings.
4. In **Choose Data values** section, we will define three data values to show the 'Girth', 'Height', and 'Volume' of the trees.
    1. From the drop-down, set the **X Field** to `=[Girth]`.
    2. From the drop-down, set the **Y Field** to `=[Height]`.
    3. From the drop-down, set the **Size Field** to `=[Volume]`.
5. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and set the following properties:
    - **Title**: Height (ft.)

    - **Font > Size**: 12pt
    - **Font > Color**: Gray
3. Go to the **Labels** page and set the following properties:
    - **Font > Size:** 12pt
    - **Font > Color**: Gray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties:
    - **Grid Interval**: 5
    - **Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Go to the **Scale** page and set the following properties:
    - **Scale Type**: Linear
    - **Minimum scale value**: 60
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and set the following properties.
    - **Title**: Diameter (in.)
    - **Font > Size**: 12pt
    - **Font > Color**: Gray
3. Go to the **Labels** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Gray
4. Go to the **Line** page and set the following properties.
    - **Color:** #cccccc
    - **Width:** 2pt
5. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
    - #b3000c
    - #ffca3a
3. Click **OK** to complete setting up the custom palette.

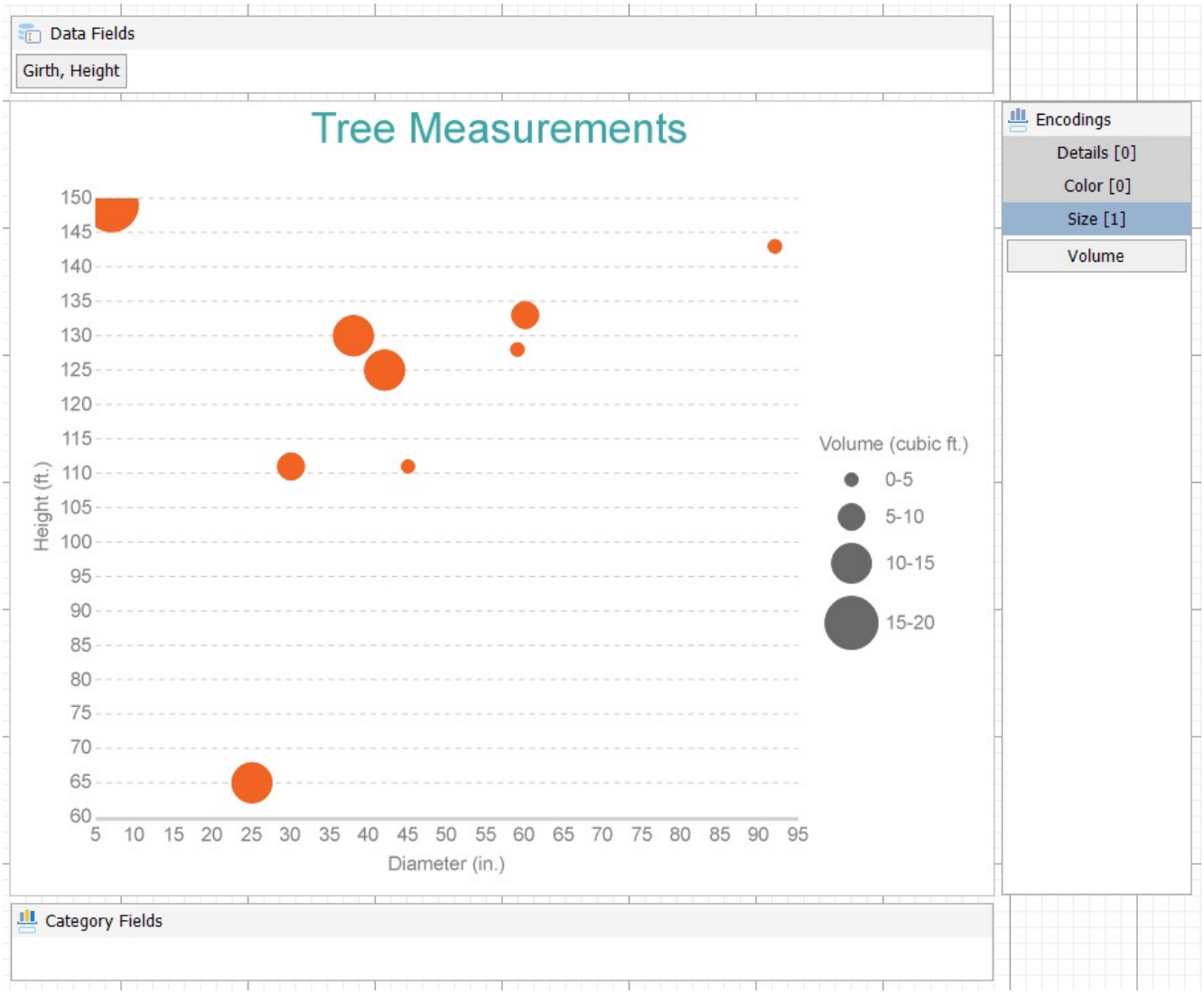
Legend- Size

1. To open the smart panel for the legend, right-click 'Legend - Size' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and set the following properties.
    - **Title**: Volume (cubic ft.)
    - **Font > Size**: 12pt
    - **Font >Color**: Gray

3. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 12pt
    - **Background Fill Color > Icon Color**: DimGray
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Tree Measurements'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

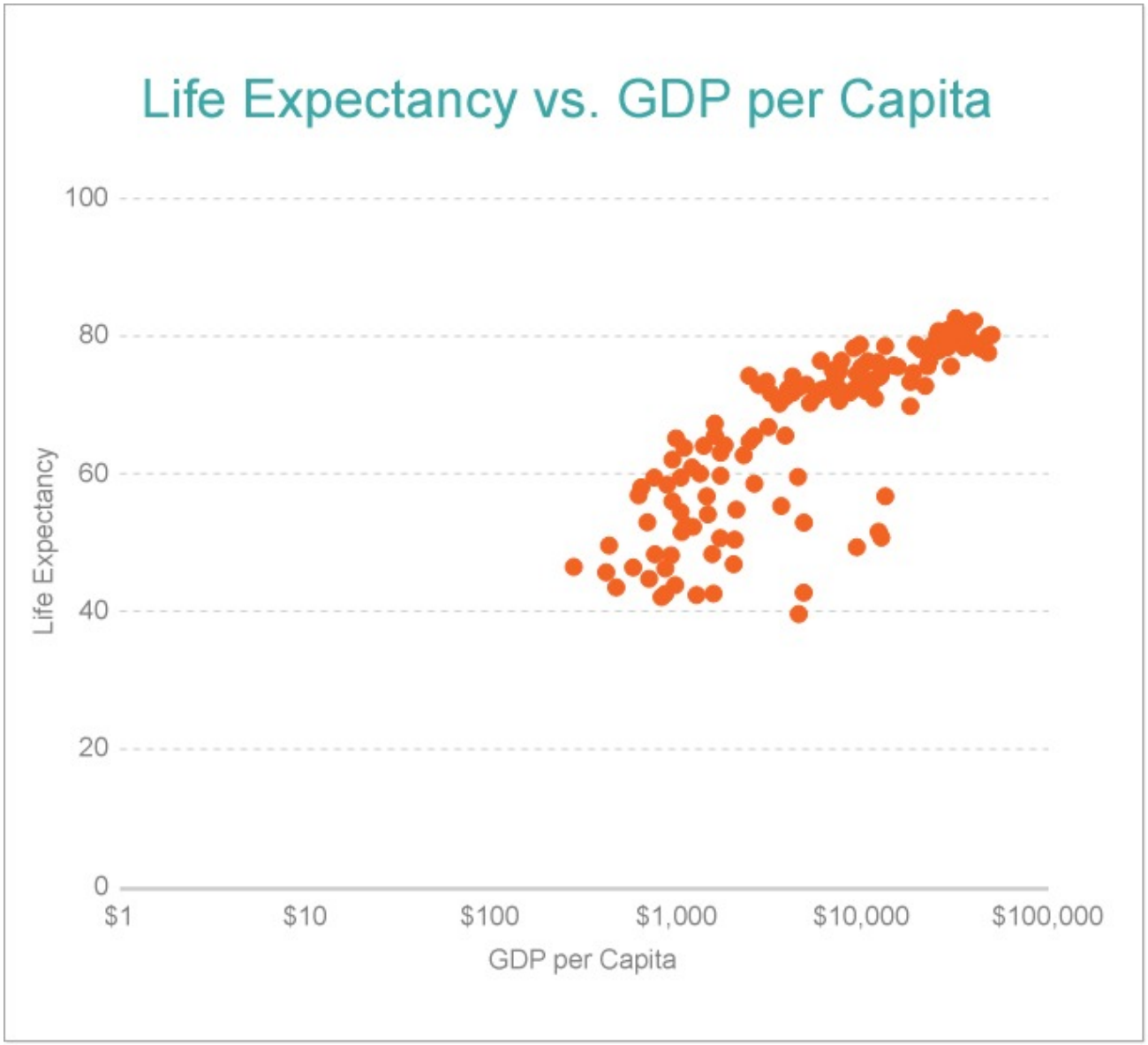


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Scatter Chart

This walkthrough creates a Scatter Chart. The chart shows the relationship between the 'GDP per Capita' and 'Life Expectancy'. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.

4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
 {
   "country": "Afghanistan",
   "continent": "Asia",
   "lifeExp": 43.828,
   "gdpPercap": 974.5803384,
   "pop": 31889923
 },
 {
   "country": "Albania",
   "continent": "Europe",
   "lifeExp": 76.423,
   "gdpPercap": 5937.029526,
   "pop": 3600523
 },
 {
   "country": "Algeria",
   "continent": "Africa",
   "lifeExp": 72.301,
   "gdpPercap": 6223.367465,
   "pop": 33333216
 },
 {
   "country": "Angola",
   "continent": "Africa",
   "lifeExp": 42.731,
   "gdpPercap": 4797.231267,
   "pop": 12420476
 },
 {
   "country": "Argentina",
   "continent": "Americas",
   "lifeExp": 75.32,
   "gdpPercap": 12779.37964,
   "pop": 40301927
 },
 {
   "country": "Australia",
   "continent": "Oceania",
   "lifeExp": 81.235,
   "gdpPercap": 34435.36744,
   "pop": 20434176
 },
 {
   "country": "Austria",
   "continent": "Europe",
```

```
  "lifeExp": 79.829,
  "gdpPercap": 36126.4927,
  "pop": 8199783
},
{
  "country": "Bahrain",
  "continent": "Asia",
  "lifeExp": 75.635,
  "gdpPercap": 29796.04834,
  "pop": 708573
},
{
  "country": "Bangladesh",
  "continent": "Asia",
  "lifeExp": 64.062,
  "gdpPercap": 1391.253792,
  "pop": 150448339
},
{
  "country": "Belgium",
  "continent": "Europe",
  "lifeExp": 79.441,
  "gdpPercap": 33692.60508,
  "pop": 10392226
},
{
  "country": "Benin",
  "continent": "Africa",
  "lifeExp": 56.728,
  "gdpPercap": 1441.284873,
  "pop": 8078314
},
{
  "country": "Bolivia",
  "continent": "Americas",
  "lifeExp": 65.554,
  "gdpPercap": 3822.137084,
  "pop": 9119152
},
{
  "country": "Bosnia and Herzegovina",
  "continent": "Europe",
  "lifeExp": 74.852,
  "gdpPercap": 7446.298803,
  "pop": 4552198
},
{
  "country": "Botswana",
  "continent": "Africa",
  "lifeExp": 50.728,
  "gdpPercap": 12569.85177,
```

```
  "pop": 1639131
},
{
  "country": "Brazil",
  "continent": "Americas",
  "lifeExp": 72.39,
  "gdpPercap": 9065.800825,
  "pop": 190010647
},
{
  "country": "Bulgaria",
  "continent": "Europe",
  "lifeExp": 73.005,
  "gdpPercap": 10680.79282,
  "pop": 7322858
},
{
  "country": "Burkina Faso",
  "continent": "Africa",
  "lifeExp": 52.295,
  "gdpPercap": 1217.032994,
  "pop": 14326203
},
{
  "country": "Burundi",
  "continent": "Africa",
  "lifeExp": 49.58,
  "gdpPercap": 430.0706916,
  "pop": 8390505
},
{
  "country": "Cambodia",
  "continent": "Asia",
  "lifeExp": 59.723,
  "gdpPercap": 1713.778686,
  "pop": 14131858
},
{
  "country": "Cameroon",
  "continent": "Africa",
  "lifeExp": 50.43,
  "gdpPercap": 2042.09524,
  "pop": 17696293
},
{
  "country": "Canada",
  "continent": "Americas",
  "lifeExp": 80.653,
  "gdpPercap": 36319.23501,
  "pop": 33390141
```

```
},
{
  "country": "Central African Republic",
  "continent": "Africa",
  "lifeExp": 44.741,
  "gdpPercap": 706.016537,
  "pop": 4369038
},
{
  "country": "Chad",
  "continent": "Africa",
  "lifeExp": 50.651,
  "gdpPercap": 1704.063724,
  "pop": 10238807
},
{
  "country": "Chile",
  "continent": "Americas",
  "lifeExp": 78.553,
  "gdpPercap": 13171.63885,
  "pop": 16284741
},
{
  "country": "China",
  "continent": "Asia",
  "lifeExp": 72.961,
  "gdpPercap": 4959.114854,
  "pop": 1318683096
},
{
  "country": "Colombia",
  "continent": "Americas",
  "lifeExp": 72.889,
  "gdpPercap": 7006.580419,
  "pop": 44227550
},
{
  "country": "Comoros",
  "continent": "Africa",
  "lifeExp": 65.152,
  "gdpPercap": 986.1478792,
  "pop": 710960
},
{
  "country": "Congo, Dem. Rep.",
  "continent": "Africa",
  "lifeExp": 46.462,
  "gdpPercap": 277.5518587,
  "pop": 64606759
},
{
```

```
  "country": "Congo, Rep.",
  "continent": "Africa",
  "lifeExp": 55.322,
  "gdpPercap": 3632.557798,
  "pop": 3800610
},
{
  "country": "Costa Rica",
  "continent": "Americas",
  "lifeExp": 78.782,
  "gdpPercap": 9645.06142,
  "pop": 4133884
},
{
  "country": "Cote d'Ivoire",
  "continent": "Africa",
  "lifeExp": 48.328,
  "gdpPercap": 1544.750112,
  "pop": 18013409
},
{
  "country": "Croatia",
  "continent": "Europe",
  "lifeExp": 75.748,
  "gdpPercap": 14619.22272,
  "pop": 4493312
},
{
  "country": "Cuba",
  "continent": "Americas",
  "lifeExp": 78.273,
  "gdpPercap": 8948.102923,
  "pop": 11416987
},
{
  "country": "Czech Republic",
  "continent": "Europe",
  "lifeExp": 76.486,
  "gdpPercap": 22833.30851,
  "pop": 10228744
},
{
  "country": "Denmark",
  "continent": "Europe",
  "lifeExp": 78.332,
  "gdpPercap": 35278.41874,
  "pop": 5468120
},
{
  "country": "Djibouti",
```

```
  "continent": "Africa",
  "lifeExp": 54.791,
  "gdpPercap": 2082.481567,
  "pop": 496374
},
{
  "country": "Dominican Republic",
  "continent": "Americas",
  "lifeExp": 72.235,
  "gdpPercap": 6025.374752,
  "pop": 9319622
},
{
  "country": "Ecuador",
  "continent": "Americas",
  "lifeExp": 74.994,
  "gdpPercap": 6873.262326,
  "pop": 13755680
},
{
  "country": "Egypt",
  "continent": "Africa",
  "lifeExp": 71.338,
  "gdpPercap": 5581.180998,
  "pop": 80264543
},
{
  "country": "El Salvador",
  "continent": "Americas",
  "lifeExp": 71.878,
  "gdpPercap": 5728.353514,
  "pop": 6939688
},
{
  "country": "Equatorial Guinea",
  "continent": "Africa",
  "lifeExp": 51.579,
  "gdpPercap": 12154.08975,
  "pop": 551201
},
{
  "country": "Eritrea",
  "continent": "Africa",
  "lifeExp": 58.04,
  "gdpPercap": 641.3695236,
  "pop": 4906585
},
{
  "country": "Ethiopia",
  "continent": "Africa",
  "lifeExp": 52.947,
```

```
  "gdpPercap": 690.8055759,
  "pop": 76511887
},
{
  "country": "Finland",
  "continent": "Europe",
  "lifeExp": 79.313,
  "gdpPercap": 33207.0844,
  "pop": 5238460
},
{
  "country": "France",
  "continent": "Europe",
  "lifeExp": 80.657,
  "gdpPercap": 30470.0167,
  "pop": 61083916
},
{
  "country": "Gabon",
  "continent": "Africa",
  "lifeExp": 56.735,
  "gdpPercap": 13206.48452,
  "pop": 1454867
},
{
  "country": "Gambia",
  "continent": "Africa",
  "lifeExp": 59.448,
  "gdpPercap": 752.7497265,
  "pop": 1688359
},
{
  "country": "Germany",
  "continent": "Europe",
  "lifeExp": 79.406,
  "gdpPercap": 32170.37442,
  "pop": 82400996
},
{
  "country": "Ghana",
  "continent": "Africa",
  "lifeExp": 60.022,
  "gdpPercap": 1327.60891,
  "pop": 22873338
},
{
  "country": "Greece",
  "continent": "Europe",
  "lifeExp": 79.483,
  "gdpPercap": 27538.41188,
```

```
  "pop": 10706290
},
{
  "country": "Guatemala",
  "continent": "Americas",
  "lifeExp": 70.259,
  "gdpPercap": 5186.050003,
  "pop": 12572928
},
{
  "country": "Guinea",
  "continent": "Africa",
  "lifeExp": 56.007,
  "gdpPercap": 942.6542111,
  "pop": 9947814
},
{
  "country": "Guinea-Bissau",
  "continent": "Africa",
  "lifeExp": 46.388,
  "gdpPercap": 579.231743,
  "pop": 1472041
},
{
  "country": "Haiti",
  "continent": "Americas",
  "lifeExp": 60.916,
  "gdpPercap": 1201.637154,
  "pop": 8502814
},
{
  "country": "Honduras",
  "continent": "Americas",
  "lifeExp": 70.198,
  "gdpPercap": 3548.330846,
  "pop": 7483763
},
{
  "country": "Hong Kong, China",
  "continent": "Asia",
  "lifeExp": 82.208,
  "gdpPercap": 39724.97867,
  "pop": 6980412
},
{
  "country": "Hungary",
  "continent": "Europe",
  "lifeExp": 73.338,
  "gdpPercap": 18008.94444,
  "pop": 9956108
},
```

```
{
  "country": "Iceland",
  "continent": "Europe",
  "lifeExp": 81.757,
  "gdpPercap": 36180.78919,
  "pop": 301931
},
{
  "country": "India",
  "continent": "Asia",
  "lifeExp": 64.698,
  "gdpPercap": 2452.210407,
  "pop": 1110396331
},
{
  "country": "Indonesia",
  "continent": "Asia",
  "lifeExp": 70.65,
  "gdpPercap": 3540.651564,
  "pop": 223547000
},
{
  "country": "Iran",
  "continent": "Asia",
  "lifeExp": 70.964,
  "gdpPercap": 11605.71449,
  "pop": 69453570
},
{
  "country": "Iraq",
  "continent": "Asia",
  "lifeExp": 59.545,
  "gdpPercap": 4471.061906,
  "pop": 27499638
},
{
  "country": "Ireland",
  "continent": "Europe",
  "lifeExp": 78.885,
  "gdpPercap": 40675.99635,
  "pop": 4109086
},
{
  "country": "Israel",
  "continent": "Asia",
  "lifeExp": 80.745,
  "gdpPercap": 25523.2771,
  "pop": 6426679
},
{
```

```
  "country": "Italy",
  "continent": "Europe",
  "lifeExp": 80.546,
  "gdpPercap": 28569.7197,
  "pop": 58147733
},
{
  "country": "Jamaica",
  "continent": "Americas",
  "lifeExp": 72.567,
  "gdpPercap": 7320.880262,
  "pop": 2780132
},
{
  "country": "Japan",
  "continent": "Asia",
  "lifeExp": 82.603,
  "gdpPercap": 31656.06806,
  "pop": 127467972
},
{
  "country": "Jordan",
  "continent": "Asia",
  "lifeExp": 72.535,
  "gdpPercap": 4519.461171,
  "pop": 6053193
},
{
  "country": "Kenya",
  "continent": "Africa",
  "lifeExp": 54.11,
  "gdpPercap": 1463.249282,
  "pop": 35610177
},
{
  "country": "Korea, Dem. Rep.",
  "continent": "Asia",
  "lifeExp": 67.297,
  "gdpPercap": 1593.06548,
  "pop": 23301725
},
{
  "country": "Korea, Rep.",
  "continent": "Asia",
  "lifeExp": 78.623,
  "gdpPercap": 23348.13973,
  "pop": 49044790
},
{
  "country": "Kuwait",
  "continent": "Asia",
```

```
  "lifeExp": 77.588,
  "gdpPercap": 47306.98978,
  "pop": 2505559
},
{
  "country": "Lebanon",
  "continent": "Asia",
  "lifeExp": 71.993,
  "gdpPercap": 10461.05868,
  "pop": 3921278
},
{
  "country": "Lesotho",
  "continent": "Africa",
  "lifeExp": 42.592,
  "gdpPercap": 1569.331442,
  "pop": 2012649
},
{
  "country": "Liberia",
  "continent": "Africa",
  "lifeExp": 45.678,
  "gdpPercap": 414.5073415,
  "pop": 3193942
},
{
  "country": "Libya",
  "continent": "Africa",
  "lifeExp": 73.952,
  "gdpPercap": 12057.49928,
  "pop": 6036914
},
{
  "country": "Madagascar",
  "continent": "Africa",
  "lifeExp": 59.443,
  "gdpPercap": 1044.770126,
  "pop": 19167654
},
{
  "country": "Malawi",
  "continent": "Africa",
  "lifeExp": 48.303,
  "gdpPercap": 759.3499101,
  "pop": 13327079
},
{
  "country": "Malaysia",
  "continent": "Asia",
  "lifeExp": 74.241,
```

```
  "gdpPercap": 12451.6558,
  "pop": 24821286
},
{
  "country": "Mali",
  "continent": "Africa",
  "lifeExp": 54.467,
  "gdpPercap": 1042.581557,
  "pop": 12031795
},
{
  "country": "Mauritania",
  "continent": "Africa",
  "lifeExp": 64.164,
  "gdpPercap": 1803.151496,
  "pop": 3270065
},
{
  "country": "Mauritius",
  "continent": "Africa",
  "lifeExp": 72.801,
  "gdpPercap": 10956.99112,
  "pop": 1250882
},
{
  "country": "Mexico",
  "continent": "Americas",
  "lifeExp": 76.195,
  "gdpPercap": 11977.57496,
  "pop": 108700891
},
{
  "country": "Mongolia",
  "continent": "Asia",
  "lifeExp": 66.803,
  "gdpPercap": 3095.772271,
  "pop": 2874127
},
{
  "country": "Montenegro",
  "continent": "Europe",
  "lifeExp": 74.543,
  "gdpPercap": 9253.896111,
  "pop": 684736
},
{
  "country": "Morocco",
  "continent": "Africa",
  "lifeExp": 71.164,
  "gdpPercap": 3820.17523,
  "pop": 33757175
```

```
},
{
  "country": "Mozambique",
  "continent": "Africa",
  "lifeExp": 42.082,
  "gdpPercap": 823.6856205,
  "pop": 19951656
},
{
  "country": "Myanmar",
  "continent": "Asia",
  "lifeExp": 62.069,
  "gdpPercap": 944,
  "pop": 47761980
},
{
  "country": "Namibia",
  "continent": "Africa",
  "lifeExp": 52.906,
  "gdpPercap": 4811.060429,
  "pop": 2055080
},
{
  "country": "Nepal",
  "continent": "Asia",
  "lifeExp": 63.785,
  "gdpPercap": 1091.359778,
  "pop": 28901790
},
{
  "country": "Netherlands",
  "continent": "Europe",
  "lifeExp": 79.762,
  "gdpPercap": 36797.93332,
  "pop": 16570613
},
{
  "country": "New Zealand",
  "continent": "Oceania",
  "lifeExp": 80.204,
  "gdpPercap": 25185.00911,
  "pop": 4115771
},
{
  "country": "Nicaragua",
  "continent": "Americas",
  "lifeExp": 72.899,
  "gdpPercap": 2749.320965,
  "pop": 5675356
},
```

```
{
  "country": "Niger",
  "continent": "Africa",
  "lifeExp": 56.867,
  "gdpPercap": 619.6768924,
  "pop": 12894865
},
{
  "country": "Nigeria",
  "continent": "Africa",
  "lifeExp": 46.859,
  "gdpPercap": 2013.977305,
  "pop": 135031164
},
{
  "country": "Norway",
  "continent": "Europe",
  "lifeExp": 80.196,
  "gdpPercap": 49357.19017,
  "pop": 4627926
},
{
  "country": "Oman",
  "continent": "Asia",
  "lifeExp": 75.64,
  "gdpPercap": 22316.19287,
  "pop": 3204897
},
{
  "country": "Pakistan",
  "continent": "Asia",
  "lifeExp": 65.483,
  "gdpPercap": 2605.94758,
  "pop": 169270617
},
{
  "country": "Panama",
  "continent": "Americas",
  "lifeExp": 75.537,
  "gdpPercap": 9809.185636,
  "pop": 3242173
},
{
  "country": "Paraguay",
  "continent": "Americas",
  "lifeExp": 71.752,
  "gdpPercap": 4172.838464,
  "pop": 6667147
},
{
  "country": "Peru",
```

```
  "continent": "Americas",
  "lifeExp": 71.421,
  "gdpPercap": 7408.905561,
  "pop": 28674757
},
{
  "country": "Philippines",
  "continent": "Asia",
  "lifeExp": 71.688,
  "gdpPercap": 3190.481016,
  "pop": 91077287
},
{
  "country": "Poland",
  "continent": "Europe",
  "lifeExp": 75.563,
  "gdpPercap": 15389.92468,
  "pop": 38518241
},
{
  "country": "Portugal",
  "continent": "Europe",
  "lifeExp": 78.098,
  "gdpPercap": 20509.64777,
  "pop": 10642836
},
{
  "country": "Puerto Rico",
  "continent": "Americas",
  "lifeExp": 78.746,
  "gdpPercap": 19328.70901,
  "pop": 3942491
},
{
  "country": "Reunion",
  "continent": "Africa",
  "lifeExp": 76.442,
  "gdpPercap": 7670.122558,
  "pop": 798094
},
{
  "country": "Romania",
  "continent": "Europe",
  "lifeExp": 72.476,
  "gdpPercap": 10808.47561,
  "pop": 22276056
},
{
  "country": "Rwanda",
  "continent": "Africa",
```

```
  "lifeExp": 46.242,
  "gdpPercap": 863.0884639,
  "pop": 8860588
},
{
  "country": "Sao Tome and Principe",
  "continent": "Africa",
  "lifeExp": 65.528,
  "gdpPercap": 1598.435089,
  "pop": 199579
},
{
  "country": "Saudi Arabia",
  "continent": "Asia",
  "lifeExp": 72.777,
  "gdpPercap": 21654.83194,
  "pop": 27601038
},
{
  "country": "Senegal",
  "continent": "Africa",
  "lifeExp": 63.062,
  "gdpPercap": 1712.472136,
  "pop": 12267493
},
{
  "country": "Serbia",
  "continent": "Europe",
  "lifeExp": 74.002,
  "gdpPercap": 9786.534714,
  "pop": 10150265
},
{
  "country": "Sierra Leone",
  "continent": "Africa",
  "lifeExp": 42.568,
  "gdpPercap": 862.5407561,
  "pop": 6144562
},
{
  "country": "Singapore",
  "continent": "Asia",
  "lifeExp": 79.972,
  "gdpPercap": 47143.17964,
  "pop": 4553009
},
{
  "country": "Slovak Republic",
  "continent": "Europe",
  "lifeExp": 74.663,
  "gdpPercap": 18678.31435,
```

```
  "pop": 5447502
},
{
  "country": "Slovenia",
  "continent": "Europe",
  "lifeExp": 77.926,
  "gdpPercap": 25768.25759,
  "pop": 2009245
},
{
  "country": "Somalia",
  "continent": "Africa",
  "lifeExp": 48.159,
  "gdpPercap": 926.1410683,
  "pop": 9118773
},
{
  "country": "South Africa",
  "continent": "Africa",
  "lifeExp": 49.339,
  "gdpPercap": 9269.657808,
  "pop": 43997828
},
{
  "country": "Spain",
  "continent": "Europe",
  "lifeExp": 80.941,
  "gdpPercap": 28821.0637,
  "pop": 40448191
},
{
  "country": "Sri Lanka",
  "continent": "Asia",
  "lifeExp": 72.396,
  "gdpPercap": 3970.095407,
  "pop": 20378239
},
{
  "country": "Sudan",
  "continent": "Africa",
  "lifeExp": 58.556,
  "gdpPercap": 2602.394995,
  "pop": 42292929
},
{
  "country": "Swaziland",
  "continent": "Africa",
  "lifeExp": 39.613,
  "gdpPercap": 4513.480643,
  "pop": 1133066
```

```
},
{
  "country": "Sweden",
  "continent": "Europe",
  "lifeExp": 80.884,
  "gdpPercap": 33859.74835,
  "pop": 9031088
},
{
  "country": "Switzerland",
  "continent": "Europe",
  "lifeExp": 81.701,
  "gdpPercap": 37506.41907,
  "pop": 7554661
},
{
  "country": "Syria",
  "continent": "Asia",
  "lifeExp": 74.143,
  "gdpPercap": 4184.548089,
  "pop": 19314747
},
{
  "country": "Taiwan",
  "continent": "Asia",
  "lifeExp": 78.4,
  "gdpPercap": 28718.27684,
  "pop": 23174294
},
{
  "country": "Tanzania",
  "continent": "Africa",
  "lifeExp": 52.517,
  "gdpPercap": 1107.482182,
  "pop": 38139640
},
{
  "country": "Thailand",
  "continent": "Asia",
  "lifeExp": 70.616,
  "gdpPercap": 7458.396327,
  "pop": 65068149
},
{
  "country": "Togo",
  "continent": "Africa",
  "lifeExp": 58.42,
  "gdpPercap": 882.9699438,
  "pop": 5701579
},
{
```

```
  "country": "Trinidad and Tobago",
  "continent": "Americas",
  "lifeExp": 69.819,
  "gdpPercap": 18008.50924,
  "pop": 1056608
},
{
  "country": "Tunisia",
  "continent": "Africa",
  "lifeExp": 73.923,
  "gdpPercap": 7092.923025,
  "pop": 10276158
},
{
  "country": "Turkey",
  "continent": "Europe",
  "lifeExp": 71.777,
  "gdpPercap": 8458.276384,
  "pop": 71158647
},
{
  "country": "Uganda",
  "continent": "Africa",
  "lifeExp": 51.542,
  "gdpPercap": 1056.380121,
  "pop": 29170398
},
{
  "country": "United Kingdom",
  "continent": "Europe",
  "lifeExp": 79.425,
  "gdpPercap": 33203.26128,
  "pop": 60776238
},
{
  "country": "United States",
  "continent": "Americas",
  "lifeExp": 78.242,
  "gdpPercap": 42951.65309,
  "pop": 301139947
},
{
  "country": "Uruguay",
  "continent": "Americas",
  "lifeExp": 76.384,
  "gdpPercap": 10611.46299,
  "pop": 3447496
},
{
  "country": "Venezuela",
```

```
  "continent": "Americas",
  "lifeExp": 73.747,
  "gdpPercap": 11415.80569,
  "pop": 26084662
},
{
  "country": "Vietnam",
  "continent": "Asia",
  "lifeExp": 74.249,
  "gdpPercap": 2441.576404,
  "pop": 85262356
},
{
  "country": "West Bank and Gaza",
  "continent": "Asia",
  "lifeExp": 73.422,
  "gdpPercap": 3025.349798,
  "pop": 4018332
},
{
  "country": "Yemen, Rep.",
  "continent": "Asia",
  "lifeExp": 62.698,
  "gdpPercap": 2280.769906,
  "pop": 22211743
},
{
  "country": "Zambia",
  "continent": "Africa",
  "lifeExp": 42.384,
  "gdpPercap": 1271.211593,
  "pop": 11746035
},
{
  "country": "Zimbabwe",
  "continent": "Africa",
  "lifeExp": 43.487,
  "gdpPercap": 469.7092981,
  "pop": 12311143
}
]
```


For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'Stats'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

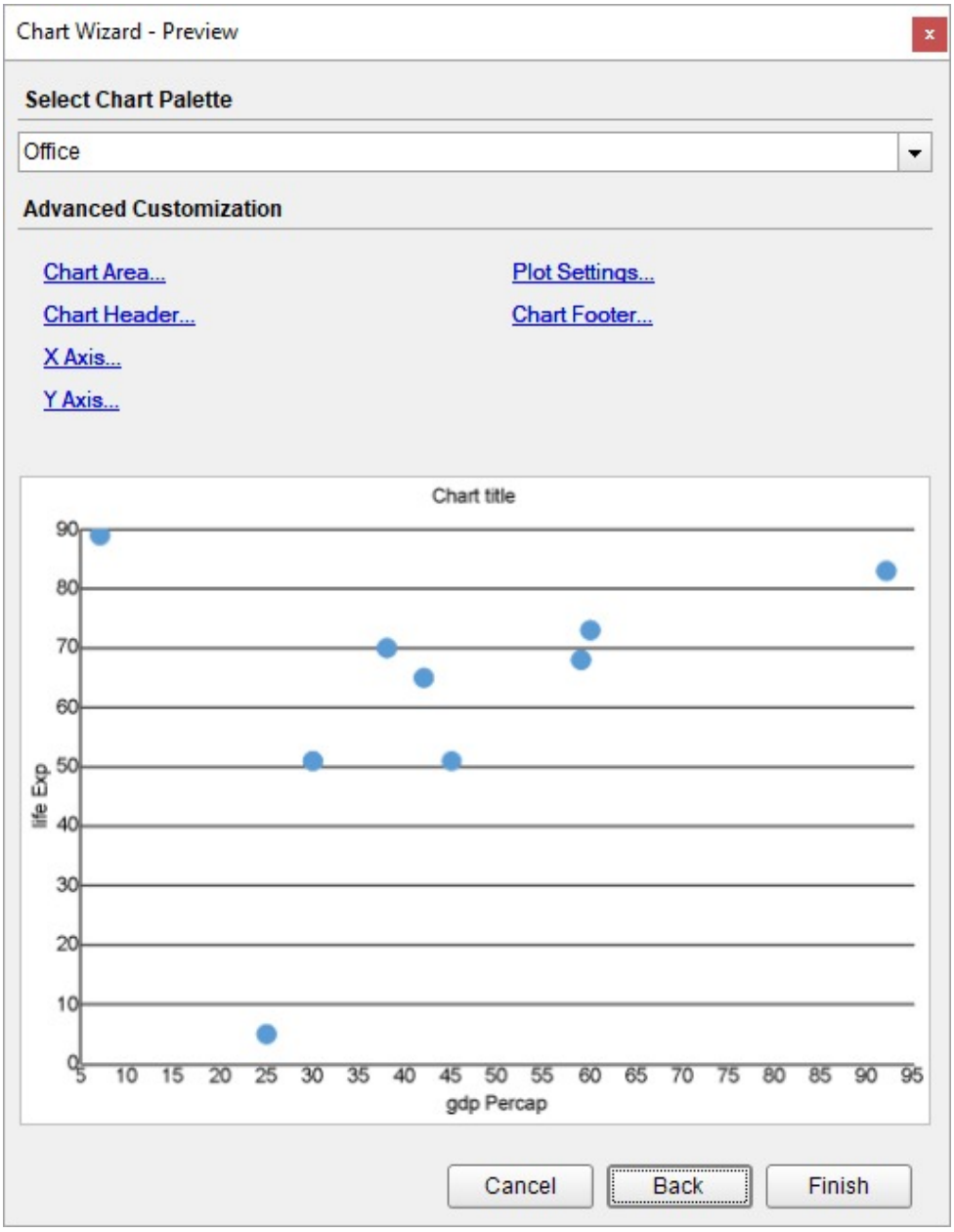
Query
`$.[*]`

3. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Stats' and the **Chart Type** as 'Scatter'.
3. Click **Next** to proceed. Here, you need to specify the scatter settings.
4. In **Choose Data values** section, we will define two data values to display 'GDP Per Capita' and 'Life Expectancy' along the horizontal and vertical axes of the chart.
    1. From the drop-down, set the **X Field** to `=[gdpPercap]`.
    2. From the drop-down, set the **Y Field** to `=[lifeExp]`.
5. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and set the following properties.
    - **Title**: Life Expectancy

    - **Font > Size**: 12pt
    - **Font > Color**: Gray
3. Go to the **Labels** page and set the following properties:
    - **Font > Size:** 12pt
    - **Font > Color**: Gray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties:
    - **Grid Interval**: 20
    - **Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog.**
2. Go to the **Title** page and set the following properties:
    - **Title**: GDP per Capita
    - **Font > Size**: 12pt
    - **Font > Color**: Gray
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal points)'.
4. Then, navigate to the Appearance tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Gray
5. Go to the **Line** page and set the following properties:
    - **Color:** #cccccc
    - **Width:** 2pt
6. Go to the **Scale** page and set the **Scale Type** to 'Logarithmic'.
7. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page, select **Custom** from the drop-down, and add the '#f26324' color.
3. Click **OK** to complete setting up the custom palette.

Chart Header

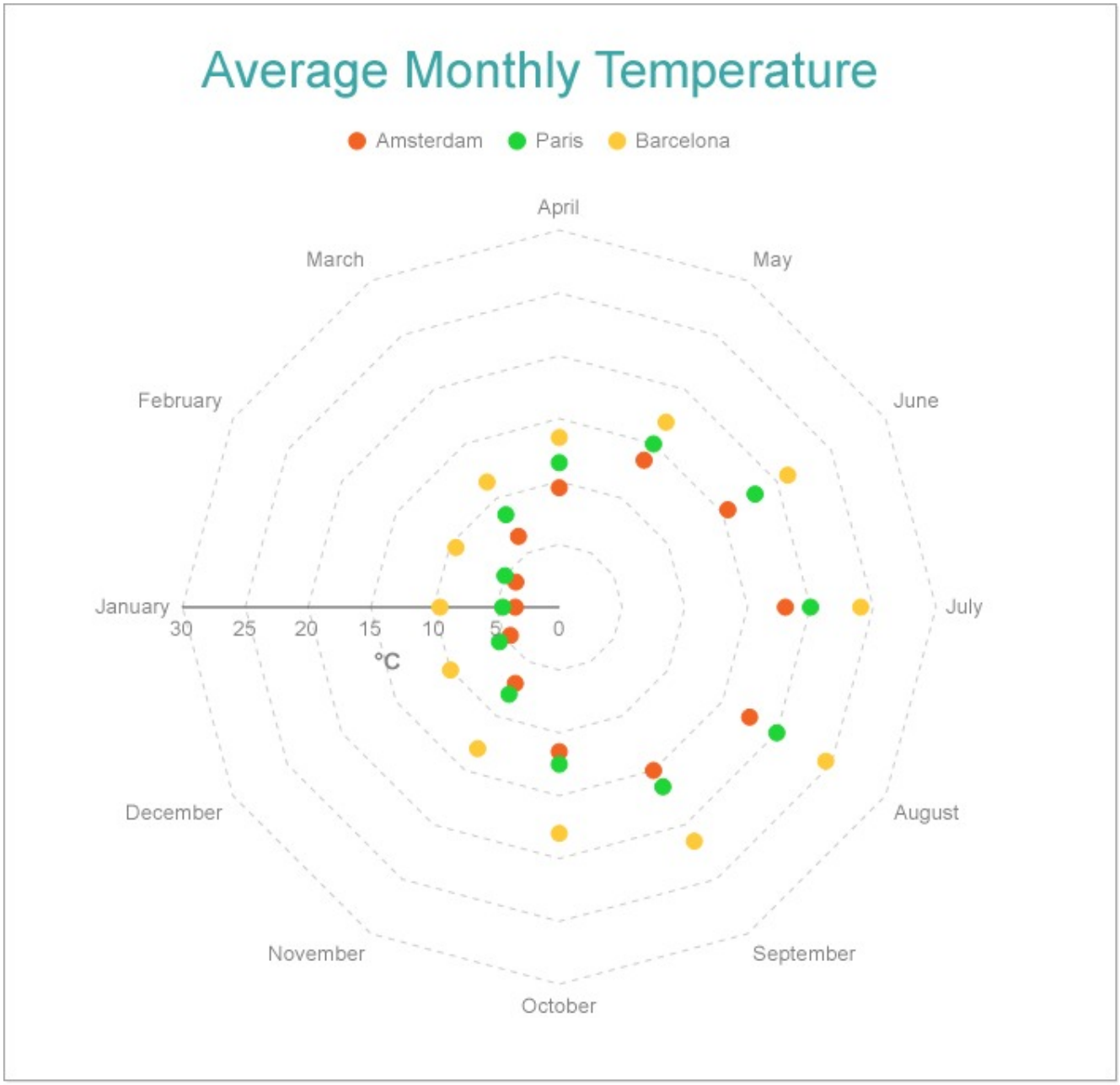
1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Life Expectancy vs. GDP per Capita'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Multi-Category Radar Scatter Chart

This walkthrough creates a Multi-Category Radar Scatter Chart. The chart shows the average monthly temperature for three cities - Amsterdam, Paris, and Barcelona. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
  {
    "City": "Amsterdam",
    "Month": "January",
    "MinT": 1,
    "MaxT": 6,
    "Precipitation": 65
  },
  {
    "City": "Amsterdam",
    "Month": "February",
    "MinT": 1,
    "MaxT": 7,
    "Precipitation": 50
  },
  {
    "City": "Amsterdam",
    "Month": "March",
    "MinT": 3,
    "MaxT": 10,
    "Precipitation": 50
  },
  {
    "City": "Amsterdam",
    "Month": "April",
    "MinT": 5,
    "MaxT": 14,
    "Precipitation": 40
  },
  {
    "City": "Amsterdam",
    "Month": "May",
    "MinT": 9,
    "MaxT": 18,
    "Precipitation": 55
  },
  {
    "City": "Amsterdam",
    "Month": "June",
    "MinT": 11,
    "MaxT": 20,
    "Precipitation": 65
  },
  {
    "City": "Amsterdam",
    "Month": "July",
    "MinT": 13,
    "MaxT": 23,
```

```
    "Precipitation": 80
  },
  {
    "City": "Amsterdam",
    "Month": "August",
    "MinT": 13,
    "MaxT": 22,
    "Precipitation": 100
  },
  {
    "City": "Amsterdam",
    "Month": "September",
    "MinT": 11,
    "MaxT": 19,
    "Precipitation": 85
  },
  {
    "City": "Amsterdam",
    "Month": "October",
    "MinT": 8,
    "MaxT": 15,
    "Precipitation": 85
  },
  {
    "City": "Amsterdam",
    "Month": "November",
    "MinT": 4,
    "MaxT": 10,
    "Precipitation": 85
  },
  {
    "City": "Amsterdam",
    "Month": "December",
    "MinT": 2,
    "MaxT": 7,
    "Precipitation": 80
  },
  {
    "City": "Paris",
    "Month": "January",
    "MinT": 2,
    "MaxT": 7,
    "Precipitation": 50
  },
  {
    "City": "Paris",
    "Month": "February",
    "MinT": 2,
    "MaxT": 8,
    "Precipitation": 40
  },
```

```
{
  "City": "Paris",
  "Month": "March",
  "MinT": 5,
  "MaxT": 12,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "April",
  "MinT": 7,
  "MaxT": 16,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "May",
  "MinT": 10,
  "MaxT": 20,
  "Precipitation": 65
},
{
  "City": "Paris",
  "Month": "June",
  "MinT": 13,
  "MaxT": 23,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "July",
  "MinT": 15,
  "MaxT": 25,
  "Precipitation": 60
},
{
  "City": "Paris",
  "Month": "August",
  "MinT": 15,
  "MaxT": 25,
  "Precipitation": 55
},
{
  "City": "Paris",
  "Month": "September",
  "MinT": 12,
  "MaxT": 21,
  "Precipitation": 50
},
{
```

```
  "City": "Paris",
  "Month": "October",
  "MinT": 9,
  "MaxT": 16,
  "Precipitation": 60
},
{
  "City": "Paris",
  "Month": "November",
  "MinT": 5,
  "MaxT": 11,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "December",
  "MinT": 3,
  "MaxT": 8,
  "Precipitation": 60
},
{
  "City": "Barcelona",
  "Month": "January",
  "MinT": 5,
  "MaxT": 14,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "February",
  "MinT": 5,
  "MaxT": 14,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "March",
  "MinT": 7,
  "MaxT": 16,
  "Precipitation": 35
},
{
  "City": "Barcelona",
  "Month": "April",
  "MinT": 9,
  "MaxT": 18,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "May",
```

```
  "MinT": 13,
  "MaxT": 21,
  "Precipitation": 55
},
{
  "City": "Barcelona",
  "Month": "June",
  "MinT": 17,
  "MaxT": 25,
  "Precipitation": 30
},
{
  "City": "Barcelona",
  "Month": "July",
  "MinT": 20,
  "MaxT": 28,
  "Precipitation": 20
},
{
  "City": "Barcelona",
  "Month": "August",
  "MinT": 20,
  "MaxT": 29,
  "Precipitation": 65
},
{
  "City": "Barcelona",
  "Month": "September",
  "MinT": 17,
  "MaxT": 26,
  "Precipitation": 85
},
{
  "City": "Barcelona",
  "Month": "October",
  "MinT": 14,
  "MaxT": 22,
  "Precipitation": 100
},
{
  "City": "Barcelona",
  "Month": "November",
  "MinT": 9,
  "MaxT": 17,
  "Precipitation": 65
},
{
  "City": "Barcelona",
  "Month": "December",
  "MinT": 6,
```

```
    "MaxT": 14,
    "Precipitation": 40
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Climate'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Go to the **Fields** page to view the available fields. On the same page, add one calculated field:

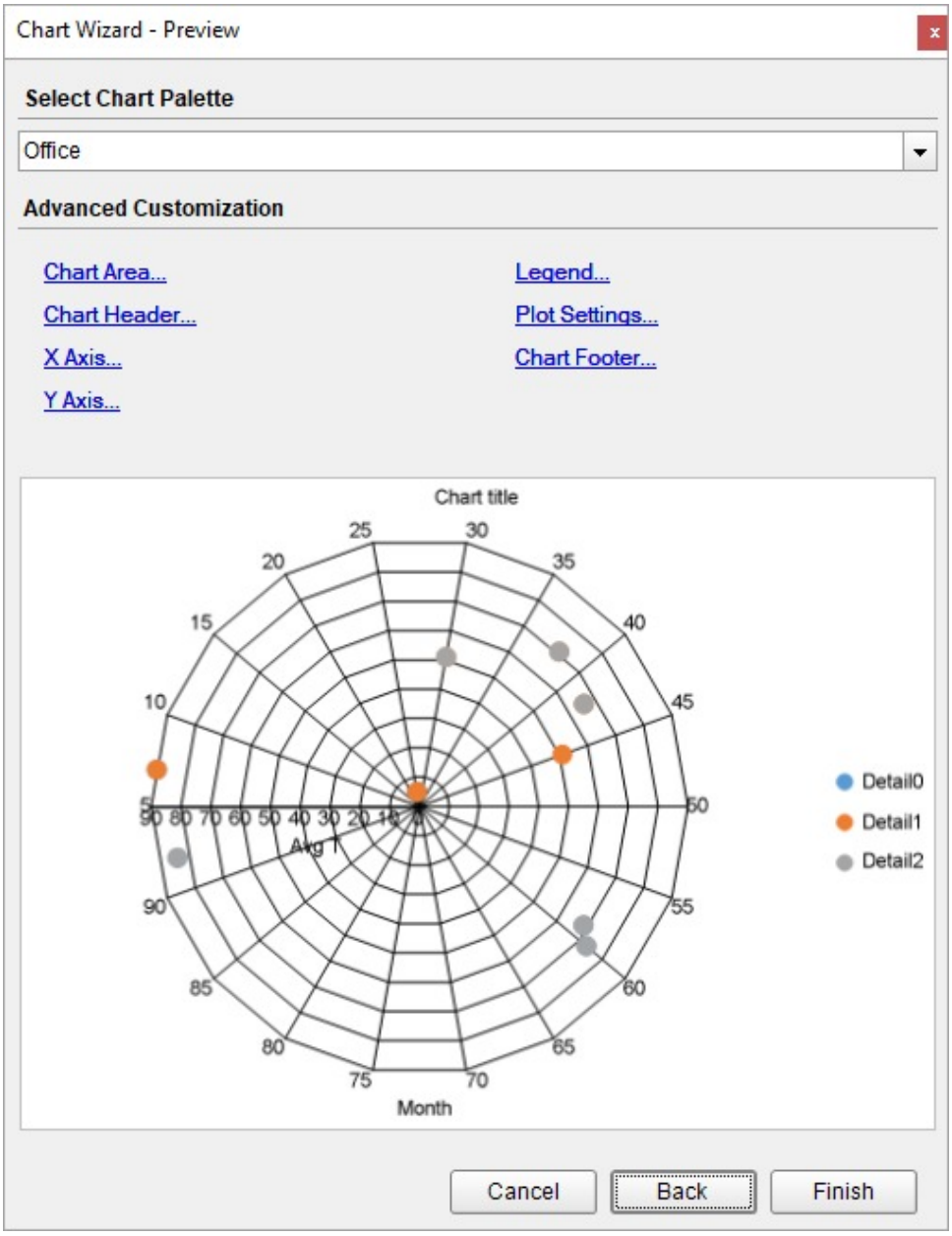
Name	Value
AvgT	`=([MinT] + [MaxT]) / 2`

4. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Climate' and the **Chart Type** as 'Radar Scatter'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Radar Scatter chart.
4. In **Choose Data values** section, we will define two data values to display.
    - From the drop-down, set the **X Field** to `=[Month]`.
    - From the drop-down, set the **Y Field** to `=[AvgT]`.
5. Under the **Data Categories** section, set the Field to `=[City]` create a multi-category chart.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Data Fields** page and remove the 'Month' value under the **Values** tab.
3. On the **Categories** page, add a new value, and set the Expression to `=[Month]` to display the average temperature for

every month.
4. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and set the following properties:
    - **Title**: °C
    - **Font > Size**: 12pt
    - **Font > Color**: Gray
    - **Font > Weight**: SemiBold
3. Go to the **Labels** page > **Appearance** tab and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Line** page and set the following properties:
    - **Show Line**: Check-on
    - **Color**: #3c3c3c
    - **Width**: 0.5pt
    - **Style**: Solid
5. Go to the **Major Gridline** page and set the following properties:
    - **Grid Interval**: 5
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Go to the **Scale** page and set the following properties:
    - **Scale Type**: Linear
    - **Maximum scale value**: 30
    - **Minimum scale value**: 0
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
5. Click **OK** to complete setting up the X-axis.

Chart Palette

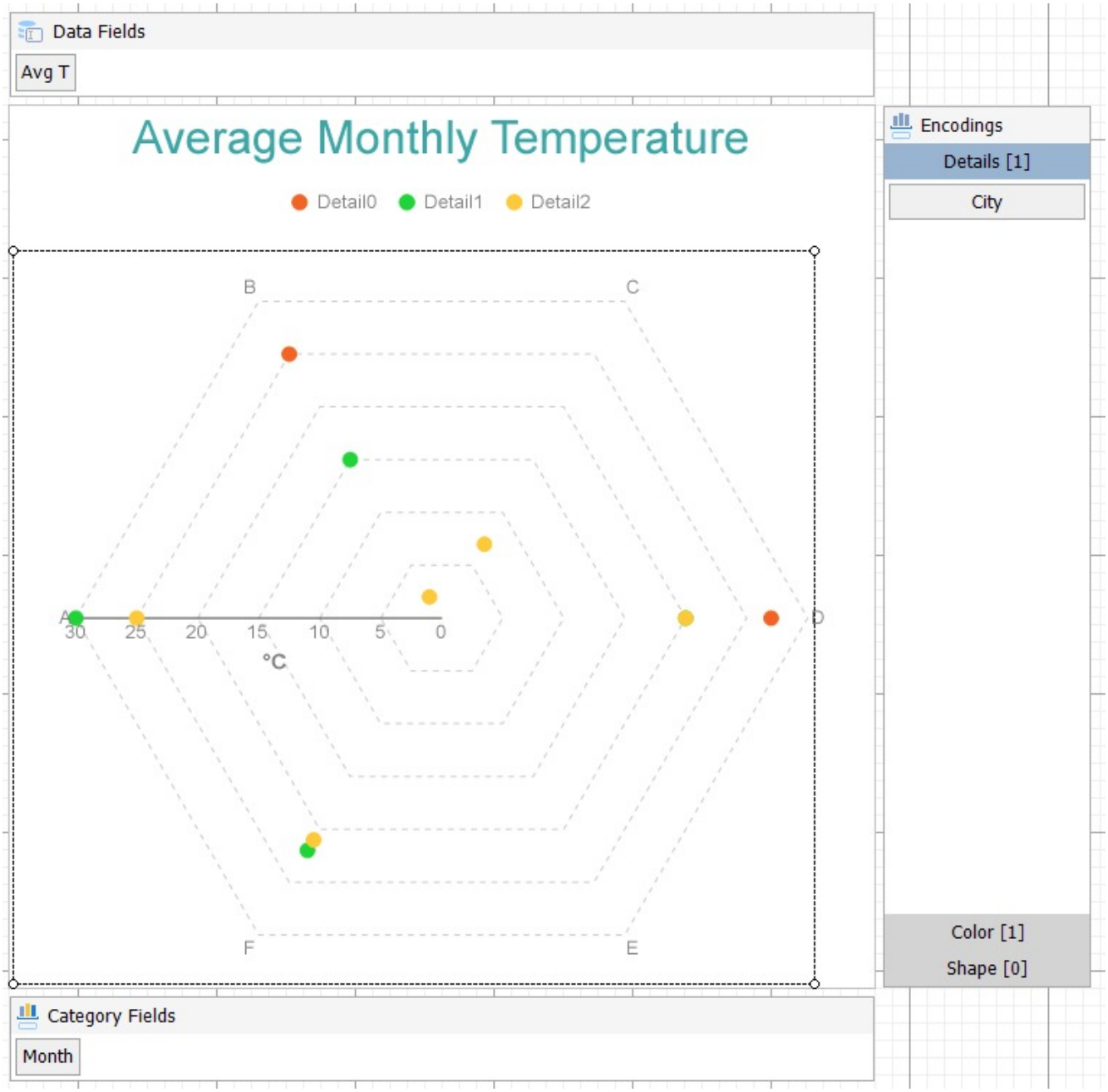
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, click the 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: Gray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Average Monthly Temperature'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Radar Scatter and Radar Bubble Charts

Radar Scatter and Radar Bubble charts are useful in showing ordered measurements of one or two variables across a category range. Radar Scatter and Radar Bubble plots arrange categories on a circle and link the corresponding points with straight lines. Data values are represented by value points called Symbols. The symbol size in a Radar Bubble plot can represent extra information about the data point. In Radar and Radar Bubble charts, the symbols are laid out along the radial lines. The major difference between the Radar Scatter plot and Radar Bubble plot is that the former visualizes

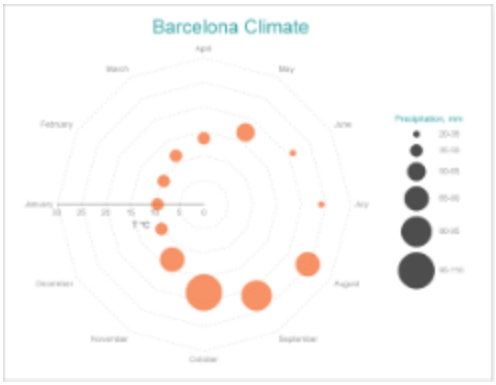
a single variable measurement, while the latter encodes the size of data points.

Simple Radar Scatter Chart

A Simple Radar Scatter chart visualizes measurements of a single variable. For instance, the simple radar bubble plot can be used to display the average monthly temperature and precipitation in Paris, France.

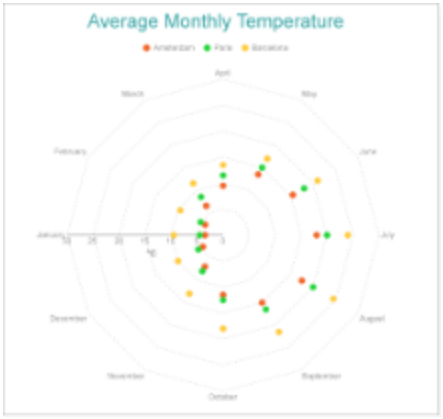
Simple Radar Bubble Chart

A radar bubble chart encodes surplus data using data point sizes. For instance, the simple radar bubble plot can be used to display the average monthly temperature and precipitation in Paris, France. See Create Simple Radar Bubble Chart walkthrough to learn how to create this chart.



Multi-Category Radar Scatter and Bubble Charts

A Multi-Category Radar Scatter and Bubble charts can be used to configure Radar Scatter and Bubble charts to split data values into subcategories and depict them with symbols' colors, shapes, or both. For instance, the Multi-Category Radar Scatter and Bubble charts can be used to depict the average monthly temperature in three European cities that are distinguished by data points' colors. See Create Multi-Category Radar Scatter Chart walkthrough to learn how to create this chart.



Radar Scatter and Bubble Plot Properties

The Radar Scatter Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when the Radar Scatter Plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each area chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the Radar Scatter Plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.

- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the Radar Scatter Plot.
  - Center: Positions the data label text on the center of the area chart.
  - Inside: Positions the data label text inside the area chart.
  - Outside: Positions the data label text outside the area chart.
  - Auto: The default setting, same as Outside for area chart.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Symbols

Represents the properties that allow you to customize the look of symbols that form the Radar Scatter plot.

- **BackgroundColor**: Change the color of the background.
- **LineColor**: Change the color of the line.
- **LineStyle**: Choose from different styles of lines such as Dotted, Dashed, Solid, Double, Groove, etc.
- **LineWidth**: Select the width of the lines in points.
- **Shape**: Choose from different shapes of symbols such as Dot, Box, Diamond, Triangle, X, Dash, Plus, etc.
- **Visible**: Set to true to display data point symbols.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Radar Scatter Plots.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

StartAngle

Indicates the arc angle that defines the clockwise rotation of the chart. A full rotation makes 360 degrees.

SymbolOpacity

Represents the opacity value of symbol fill color.

Encodings

Category Encoding

The Category Encoding of a Radar Scatter Plot is a set of properties that determine the period over which the plot generates connected data points representing the Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Color Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Radar Scatter Plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the area subsection arrangement of the plot.
    - Stack: You can use this value to configure a Stacked Radar Scatter Plot.
    - Cluster: You can use this value to configure area subsections that overlap each other.

    - None: Equals to Cluster.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Shape Encoding

The Shape Encoding enables the shape legend of the Details or Category Encoding in Radar Scatter charts. It includes the Aggregate function and shape expression, which is elaborated below:

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Value

The Value property is the collection and usually takes a bound field. However, the Radar Scatter plots take the first item from the collection.

Size Encoding

The Size Encoding enables the Aggregate function and Size expression in Radar Bubble charts. The Size Encoding works solely with numeric values and breaks down data values into ranges that determine the symbol size.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Radar Scatter Plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Radar Scatter Plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

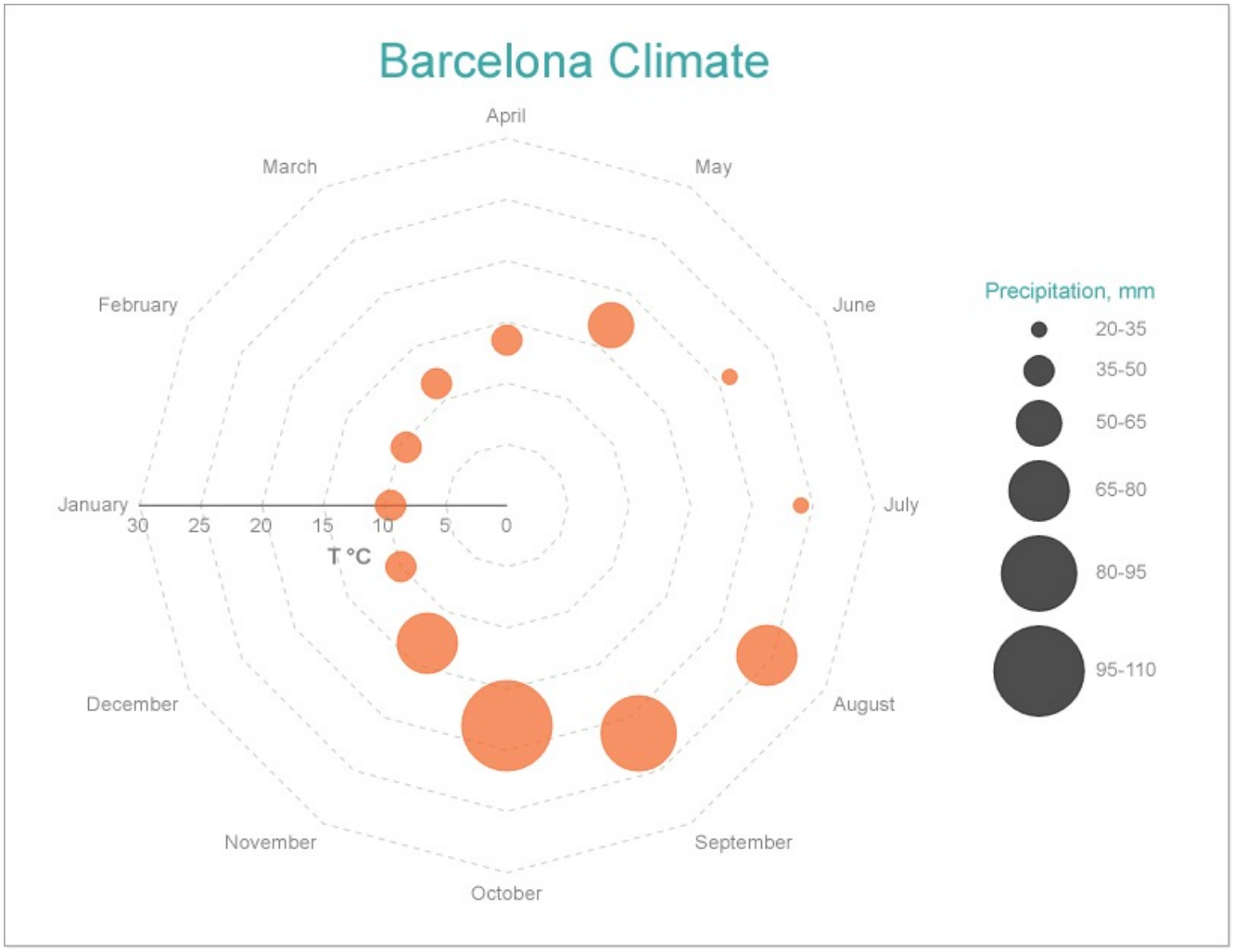
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Radar Bubble Chart

This walkthrough creates a Radar Bubble Chart. The chart shows the average monthly temperature and precipitation for Barcelona city. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
  {
    "City": "Amsterdam",
    "Month": "January",
    "MinT": 1,
    "MaxT": 6,
    "Precipitation": 65
  },
  {
    "City": "Amsterdam",
    "Month": "February",
    "MinT": 1,
    "MaxT": 7,
    "Precipitation": 50
  },
  {
    "City": "Amsterdam",
    "Month": "March",
    "MinT": 3,
    "MaxT": 10,
    "Precipitation": 50
  },
  {
    "City": "Amsterdam",
    "Month": "April",
    "MinT": 5,
    "MaxT": 14,
    "Precipitation": 40
  },
  {
    "City": "Amsterdam",
    "Month": "May",
    "MinT": 9,
    "MaxT": 18,
    "Precipitation": 55
  },
  {
```

```
  "City": "Amsterdam",
  "Month": "June",
  "MinT": 11,
  "MaxT": 20,
  "Precipitation": 65
},
{
  "City": "Amsterdam",
  "Month": "July",
  "MinT": 13,
  "MaxT": 23,
  "Precipitation": 80
},
{
  "City": "Amsterdam",
  "Month": "August",
  "MinT": 13,
  "MaxT": 22,
  "Precipitation": 100
},
{
  "City": "Amsterdam",
  "Month": "September",
  "MinT": 11,
  "MaxT": 19,
  "Precipitation": 85
},
{
  "City": "Amsterdam",
  "Month": "October",
  "MinT": 8,
  "MaxT": 15,
  "Precipitation": 85
},
{
  "City": "Amsterdam",
  "Month": "November",
  "MinT": 4,
  "MaxT": 10,
  "Precipitation": 85
},
{
  "City": "Amsterdam",
  "Month": "December",
  "MinT": 2,
  "MaxT": 7,
  "Precipitation": 80
},
{
  "City": "Paris",
```

```
  "Month": "January",
  "MinT": 2,
  "MaxT": 7,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "February",
  "MinT": 2,
  "MaxT": 8,
  "Precipitation": 40
},
{
  "City": "Paris",
  "Month": "March",
  "MinT": 5,
  "MaxT": 12,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "April",
  "MinT": 7,
  "MaxT": 16,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "May",
  "MinT": 10,
  "MaxT": 20,
  "Precipitation": 65
},
{
  "City": "Paris",
  "Month": "June",
  "MinT": 13,
  "MaxT": 23,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "July",
  "MinT": 15,
  "MaxT": 25,
  "Precipitation": 60
},
{
  "City": "Paris",
  "Month": "August",
  "MinT": 15,
```

```
  "MaxT": 25,
  "Precipitation": 55
},
{
  "City": "Paris",
  "Month": "September",
  "MinT": 12,
  "MaxT": 21,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "October",
  "MinT": 9,
  "MaxT": 16,
  "Precipitation": 60
},
{
  "City": "Paris",
  "Month": "November",
  "MinT": 5,
  "MaxT": 11,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "December",
  "MinT": 3,
  "MaxT": 8,
  "Precipitation": 60
},
{
  "City": "Barcelona",
  "Month": "January",
  "MinT": 5,
  "MaxT": 14,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "February",
  "MinT": 5,
  "MaxT": 14,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "March",
  "MinT": 7,
  "MaxT": 16,
```

```
    "Precipitation": 35
  },
  {
    "City": "Barcelona",
    "Month": "April",
    "MinT": 9,
    "MaxT": 18,
    "Precipitation": 40
  },
  {
    "City": "Barcelona",
    "Month": "May",
    "MinT": 13,
    "MaxT": 21,
    "Precipitation": 55
  },
  {
    "City": "Barcelona",
    "Month": "June",
    "MinT": 17,
    "MaxT": 25,
    "Precipitation": 30
  },
  {
    "City": "Barcelona",
    "Month": "July",
    "MinT": 20,
    "MaxT": 28,
    "Precipitation": 20
  },
  {
    "City": "Barcelona",
    "Month": "August",
    "MinT": 20,
    "MaxT": 29,
    "Precipitation": 65
  },
  {
    "City": "Barcelona",
    "Month": "September",
    "MinT": 17,
    "MaxT": 26,
    "Precipitation": 85
  },
  {
    "City": "Barcelona",
    "Month": "October",
    "MinT": 14,
    "MaxT": 22,
    "Precipitation": 100
  },
```

```
  {
    "City": "Barcelona",
    "Month": "November",
    "MinT": 9,
    "MaxT": 17,
    "Precipitation": 65
  },
  {
    "City": "Barcelona",
    "Month": "December",
    "MinT": 6,
    "MaxT": 14,
    "Precipitation": 40
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Climate'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Go to the **Fields** page to view the available fields. On the same page, add one calculated field:

Name	Value
AvgT	`=([MinT] + [MaxT]) / 2`

4. Go to the **Filters** page, add a new filter value, and set its properties as below.

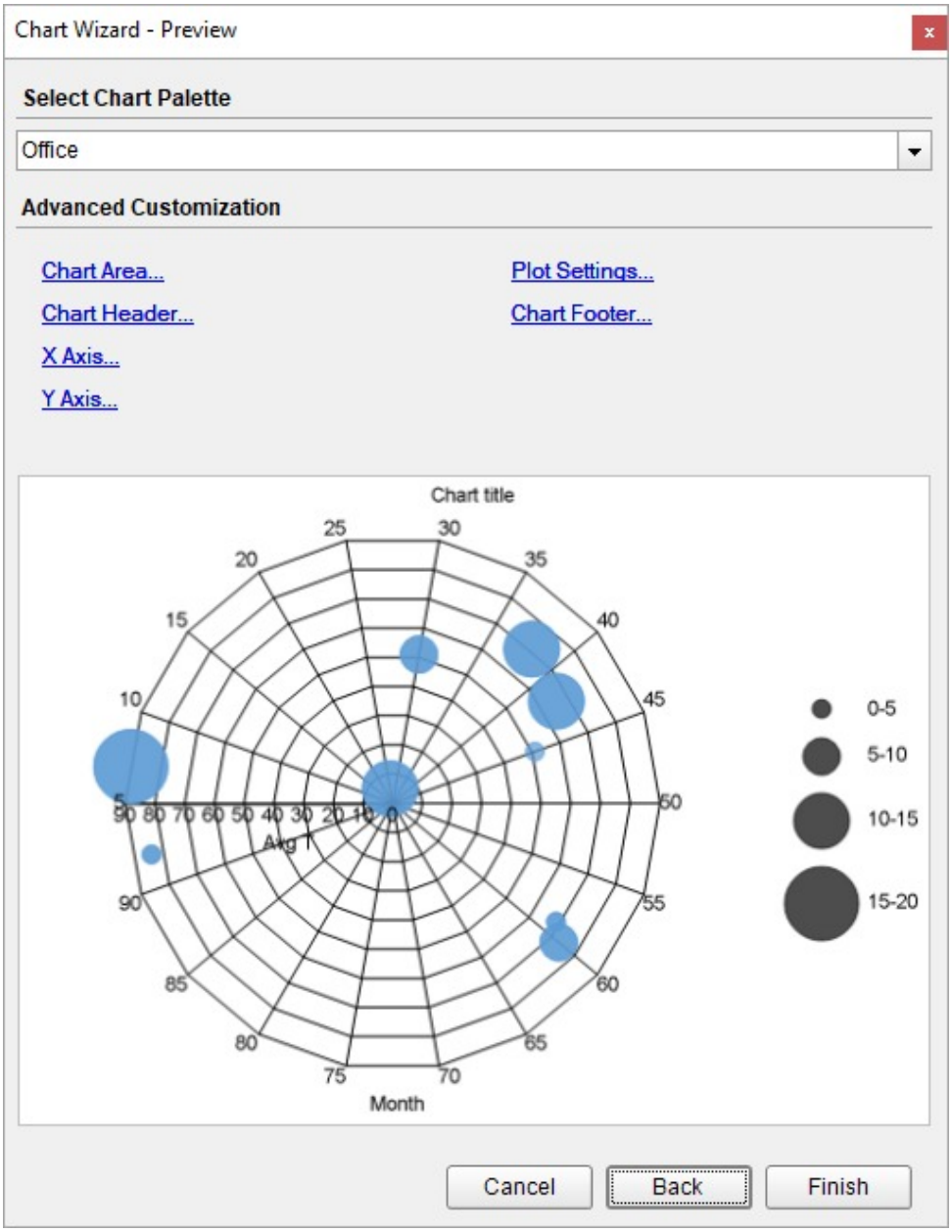
Expression	Operator	Value
`=[City]`	Equal	Barcelona

5. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Climate' and the **Chart Type** as 'Radar Bubble'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Radar Bubble chart.
4. In **Choose Data values** section, we will define three data values to display.
    - From the drop-down, set the **X Field** to `=[Month]`.
    - From the drop-down, set the **Y Field** to `=[AvgT]`.
    - From the drop-down, set the **Size Field** to `=[Precipitation]`.
5. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Data Fields** page and remove the 'Month' value under the **Values** tab.
3. In the **Categories** page, add a new value, and set the Expression to `=[Month]` to display the average temperature for

every month.
4. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and set the following properties:
    - **Title**: T °C
    - **Font > Size**: 12pt
    - **Font > Color**: Gray
    - **Font > Weight**: SemiBold
3. Go to the **Labels** page > **Appearance** tab and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Line** page and set the following properties:
    - **Show Line**: Check-on
    - **Color**: #3c3c3c
    - **Width**: 0.5pt
    - **Style**: Solid
5. Go to the **Major Gridline** page and set the following properties:
    - **Grid Interval**: 5
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Go to the **Scale** page and set the following properties:
    - **Scale Type**: Linear
    - **Maximum scale value**: 30
    - **Minimum scale value**: 0
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and set the following properties.
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
5. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select **Custom** from the drop-down and add the following colors.
    - #f26324
3. Click **OK** to complete setting up the custom palette.

Legend - Size

1. To open the smart panel for the legend, click the 'Legend - Size' on the Report Explorer and choose **Property Dialog**.

2. Go to the **Title** page and set the following properties.
    - **Title**: Precipitation, mm
    - **Size**: 12pt
    - **Color**: #3da7a8
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: Gray
    - **Background Fill Color > Icon Color**: DimGray

    By default, charts like line (line and smooth), bubble, radar bubble, and radar line automatically calculate the size of the markers.
4. To specify custom marker size in the chart, follow the below steps:
    1. Navigate to the **Ranges** page and add four ranges using the Add button.
    2. Define the upper bound of the ranges in the **To** field, as follows:

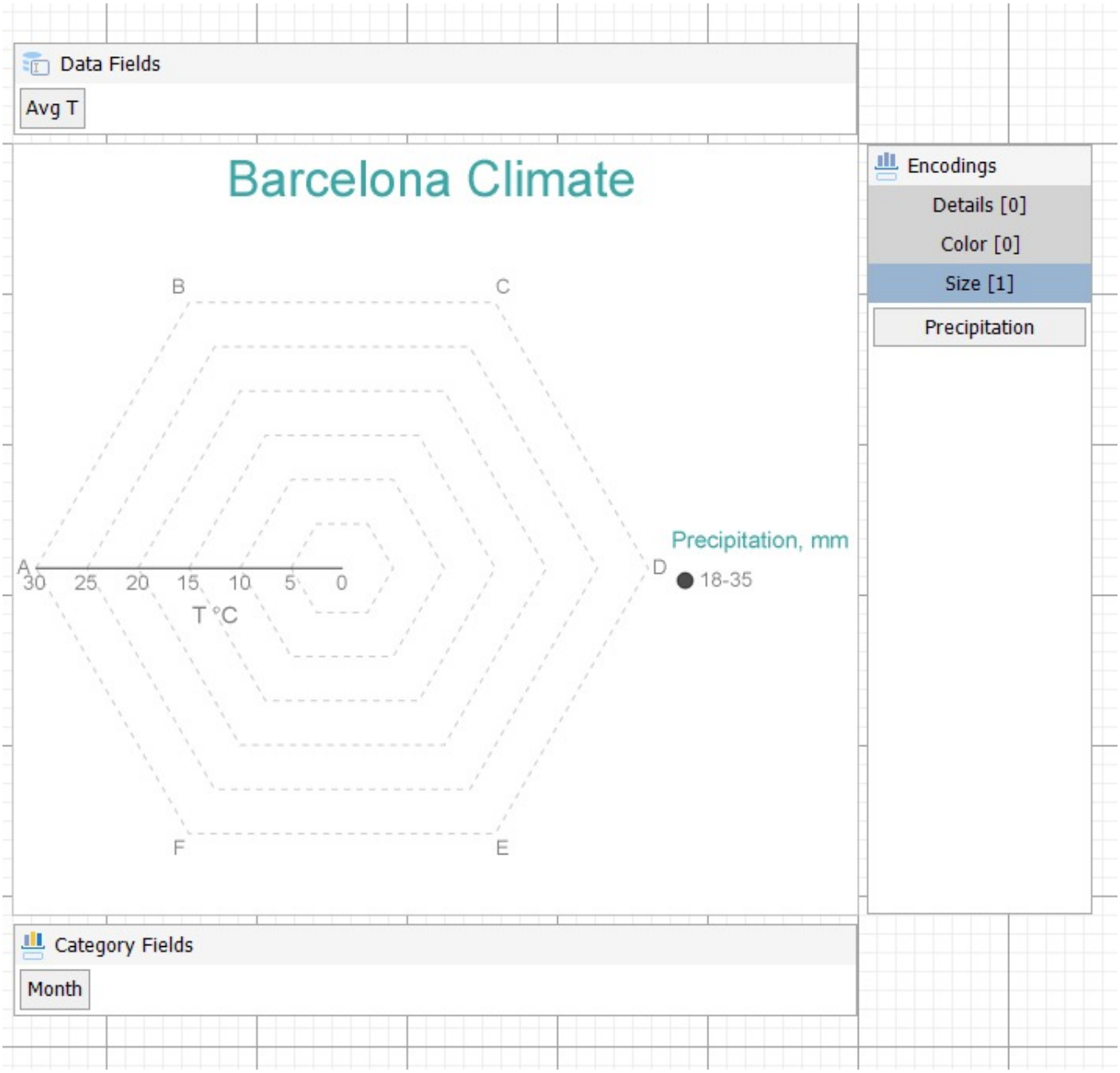
Range	To
Range1	35
Range2	50
Range3	65
Range4	80
Range5	95
Range6	110

    You can also choose to display custom titles instead of the range values using the **Title** field in the legend.

5. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to **'**Barcelona Climate'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

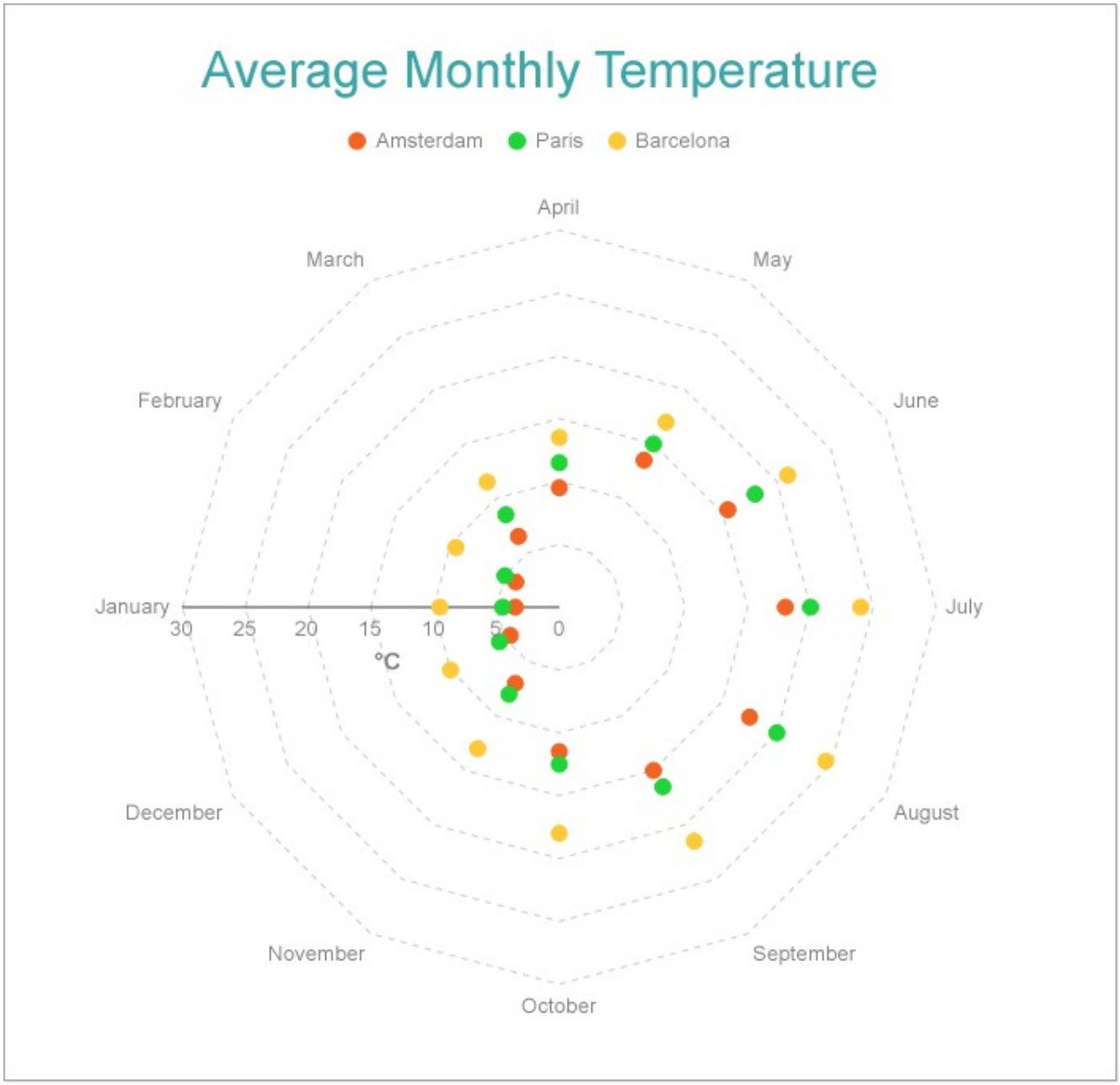


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Multi-Category Radar Scatter Chart

This walkthrough creates a Multi-Category Radar Scatter Chart. The chart shows the average monthly temperature for three cities - Amsterdam, Paris, and Barcelona. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
  {
    "City": "Amsterdam",
    "Month": "January",
    "MinT": 1,
    "MaxT": 6,
    "Precipitation": 65
  },
  {
    "City": "Amsterdam",
    "Month": "February",
    "MinT": 1,
    "MaxT": 7,
    "Precipitation": 50
  },
  {
    "City": "Amsterdam",
    "Month": "March",
    "MinT": 3,
    "MaxT": 10,
    "Precipitation": 50
  },
  {
    "City": "Amsterdam",
    "Month": "April",
    "MinT": 5,
    "MaxT": 14,
    "Precipitation": 40
  },
  {
    "City": "Amsterdam",
    "Month": "May",
    "MinT": 9,
    "MaxT": 18,
    "Precipitation": 55
  },
  {
    "City": "Amsterdam",
    "Month": "June",
    "MinT": 11,
    "MaxT": 20,
    "Precipitation": 65
  },
  {
    "City": "Amsterdam",
    "Month": "July",
    "MinT": 13,
    "MaxT": 23,
```

```
  "Precipitation": 80
},
{
  "City": "Amsterdam",
  "Month": "August",
  "MinT": 13,
  "MaxT": 22,
  "Precipitation": 100
},
{
  "City": "Amsterdam",
  "Month": "September",
  "MinT": 11,
  "MaxT": 19,
  "Precipitation": 85
},
{
  "City": "Amsterdam",
  "Month": "October",
  "MinT": 8,
  "MaxT": 15,
  "Precipitation": 85
},
{
  "City": "Amsterdam",
  "Month": "November",
  "MinT": 4,
  "MaxT": 10,
  "Precipitation": 85
},
{
  "City": "Amsterdam",
  "Month": "December",
  "MinT": 2,
  "MaxT": 7,
  "Precipitation": 80
},
{
  "City": "Paris",
  "Month": "January",
  "MinT": 2,
  "MaxT": 7,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "February",
  "MinT": 2,
  "MaxT": 8,
  "Precipitation": 40
},
```

```
{
  "City": "Paris",
  "Month": "March",
  "MinT": 5,
  "MaxT": 12,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "April",
  "MinT": 7,
  "MaxT": 16,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "May",
  "MinT": 10,
  "MaxT": 20,
  "Precipitation": 65
},
{
  "City": "Paris",
  "Month": "June",
  "MinT": 13,
  "MaxT": 23,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "July",
  "MinT": 15,
  "MaxT": 25,
  "Precipitation": 60
},
{
  "City": "Paris",
  "Month": "August",
  "MinT": 15,
  "MaxT": 25,
  "Precipitation": 55
},
{
  "City": "Paris",
  "Month": "September",
  "MinT": 12,
  "MaxT": 21,
  "Precipitation": 50
},
{
```

```
  "City": "Paris",
  "Month": "October",
  "MinT": 9,
  "MaxT": 16,
  "Precipitation": 60
},
{
  "City": "Paris",
  "Month": "November",
  "MinT": 5,
  "MaxT": 11,
  "Precipitation": 50
},
{
  "City": "Paris",
  "Month": "December",
  "MinT": 3,
  "MaxT": 8,
  "Precipitation": 60
},
{
  "City": "Barcelona",
  "Month": "January",
  "MinT": 5,
  "MaxT": 14,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "February",
  "MinT": 5,
  "MaxT": 14,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "March",
  "MinT": 7,
  "MaxT": 16,
  "Precipitation": 35
},
{
  "City": "Barcelona",
  "Month": "April",
  "MinT": 9,
  "MaxT": 18,
  "Precipitation": 40
},
{
  "City": "Barcelona",
  "Month": "May",
```

```
  "MinT": 13,
  "MaxT": 21,
  "Precipitation": 55
},
{
  "City": "Barcelona",
  "Month": "June",
  "MinT": 17,
  "MaxT": 25,
  "Precipitation": 30
},
{
  "City": "Barcelona",
  "Month": "July",
  "MinT": 20,
  "MaxT": 28,
  "Precipitation": 20
},
{
  "City": "Barcelona",
  "Month": "August",
  "MinT": 20,
  "MaxT": 29,
  "Precipitation": 65
},
{
  "City": "Barcelona",
  "Month": "September",
  "MinT": 17,
  "MaxT": 26,
  "Precipitation": 85
},
{
  "City": "Barcelona",
  "Month": "October",
  "MinT": 14,
  "MaxT": 22,
  "Precipitation": 100
},
{
  "City": "Barcelona",
  "Month": "November",
  "MinT": 9,
  "MaxT": 17,
  "Precipitation": 65
},
{
  "City": "Barcelona",
  "Month": "December",
  "MinT": 6,
```

```
    "MaxT": 14,
    "Precipitation": 40
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Climate'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Go to the **Fields** page to view the available fields. On the same page, add one calculated field:

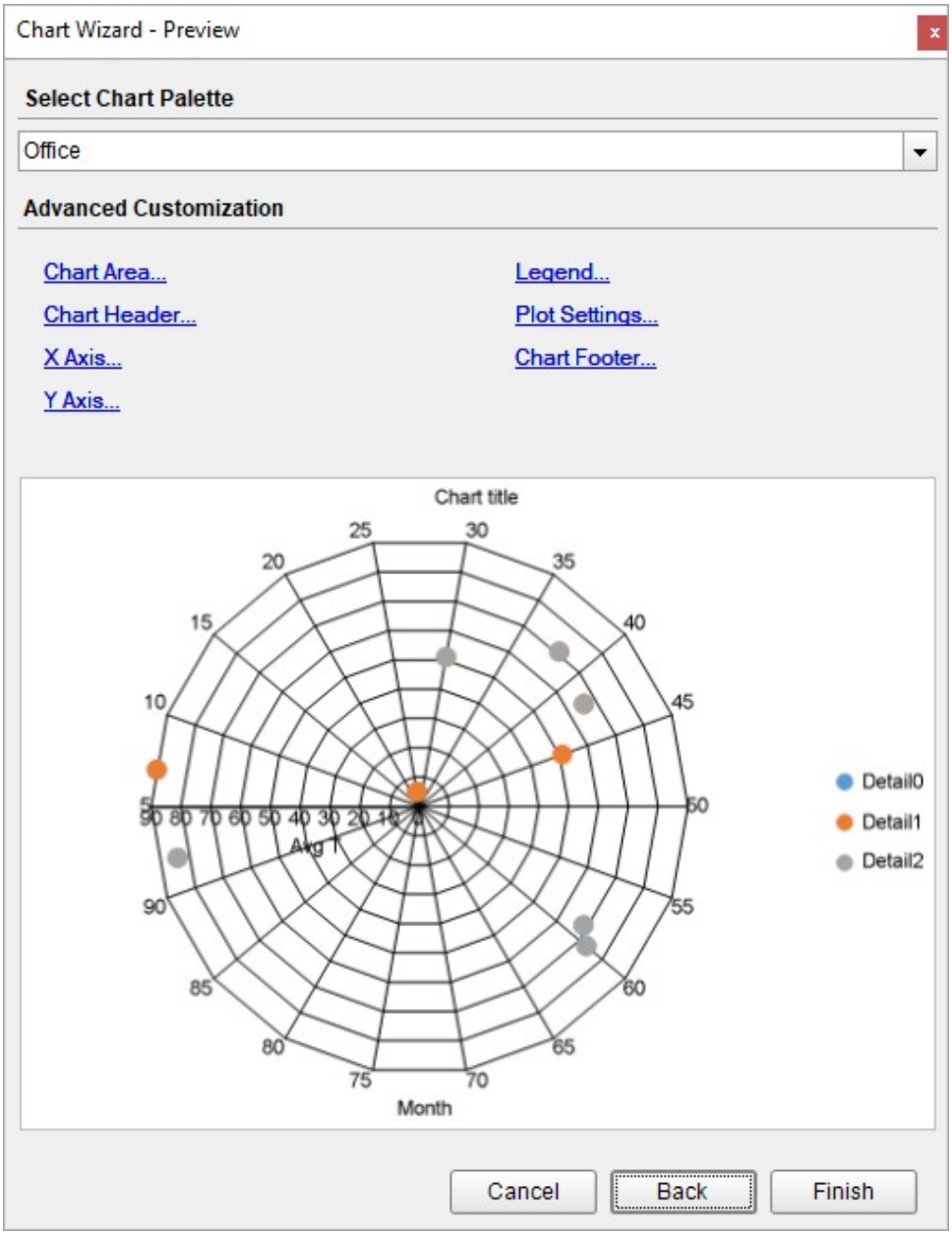
Name	Value
AvgT	`=([MinT] + [MaxT]) / 2`

4. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Climate' and the **Chart Type** as 'Radar Scatter'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Radar Scatter chart.
4. In **Choose Data values** section, we will define two data values to display.
    - From the drop-down, set the **X Field** to `=[Month]`.
    - From the drop-down, set the **Y Field** to `=[AvgT]`.
5. Under the **Data Categories** section, set the Field to `=[City]` create a multi-category chart.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Data Fields** page and remove the 'Month' value under the **Values** tab.
3. On the **Categories** page, add a new value, and set the Expression to `=[Month]` to display the average temperature for

every month.
4. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and set the following properties:
    - **Title**: °C
    - **Font > Size**: 12pt
    - **Font > Color**: Gray
    - **Font > Weight**: SemiBold
3. Go to the **Labels** page > **Appearance** tab and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Line** page and set the following properties:
    - **Show Line**: Check-on
    - **Color**: #3c3c3c
    - **Width**: 0.5pt
    - **Style**: Solid
5. Go to the **Major Gridline** page and set the following properties:
    - **Grid Interval**: 5
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Go to the **Scale** page and set the following properties:
    - **Scale Type**: Linear
    - **Maximum scale value**: 30
    - **Minimum scale value**: 0
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
5. Click **OK** to complete setting up the X-axis.

Chart Palette

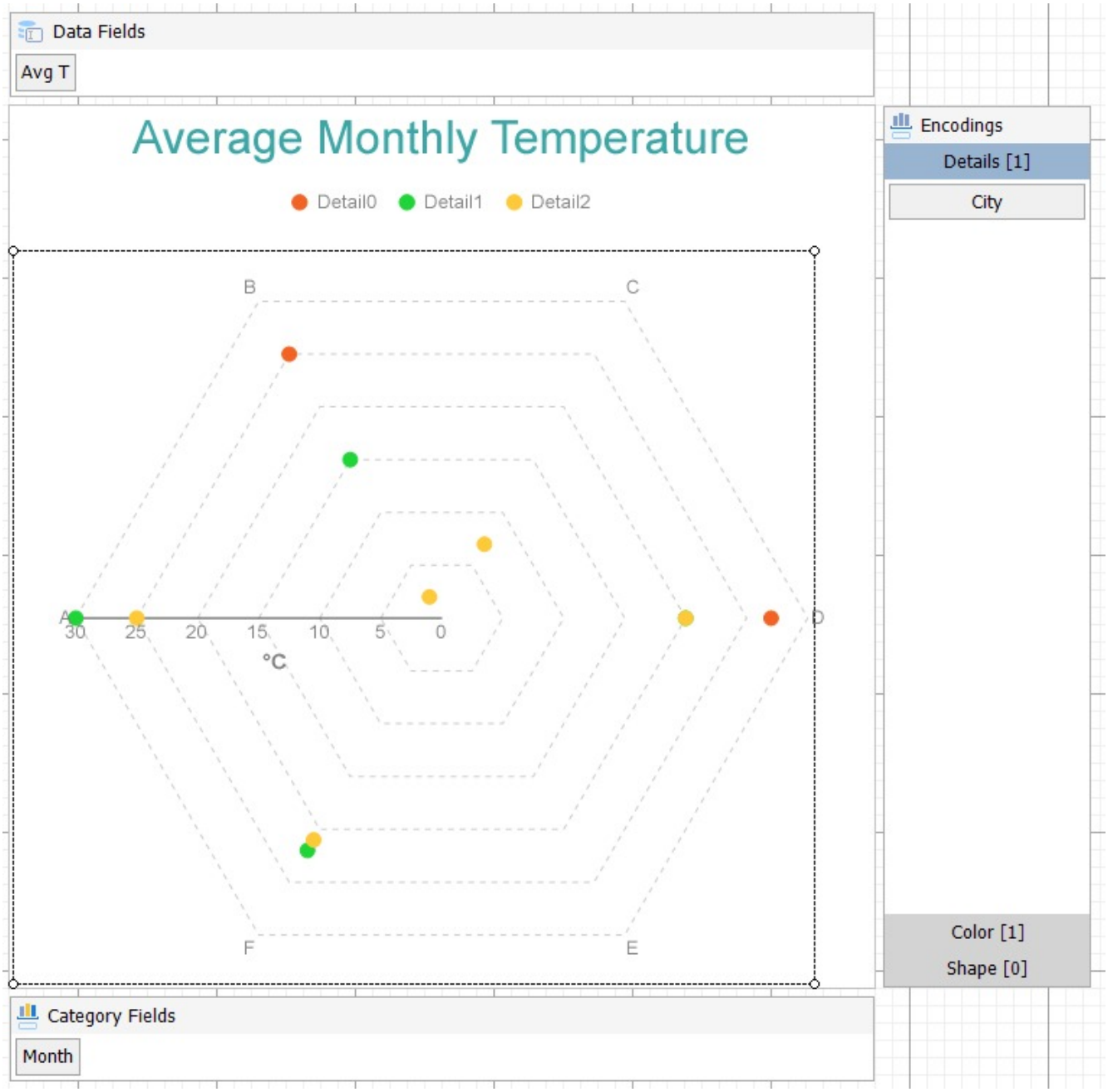
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select **Custom** from the drop-down and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, click the 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: Gray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Average Monthly Temperature'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Radar Line Chart

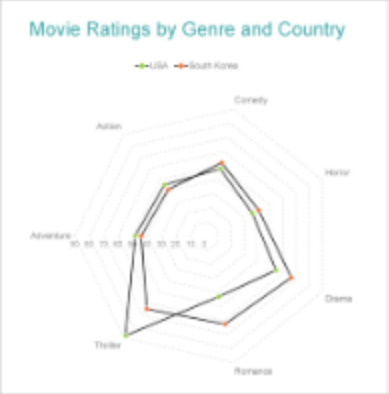
Radar Line Charts are plot types useful to depict ordered measurements of single or multiple variables across different category ranges. A radar line chart arranges categories radially and connects corresponding points with straight lines. The data values are represented by Symbols or shapes laid out across radial lines and connected via straight or curved lines.

Radar Line Chart with Multiple Values

A radar line chart can be used to depict measurements of one or multiple variables across various category ranges. For example, the Radar Line Plot with Multiple Values can be used to display two types of average ratings for many movie genres.

Radar Line Chart with Multiple Lines

A multiple-line radar chart lets you split data values into subcategories for finer analysis. For instance, the Radar Line with Multiple Lines chart can be used to depict average movie ratings in different countries for several movie genres. See Radar Line Chart with Multiple Lines walkthrough to learn how to create this chart.



Radar Line Plot Properties

The Radar Line Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when the Radar Line Plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The properties related to label settings in a radar line chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the Radar Line Plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the Radar Line Plot.
  - Center: Positions the data label text on the center of the chart.
  - Inside: Positions the data label text inside the chart.
  - Outside: Positions the data label text outside the chart.
  - Auto: The default setting, same as Outside for chart.

LineStyle

The line style for borders. Includes LineColor, LineStyle, and LineWidth properties for customization.

- **LineColor**: Specify the color of the line connecting the data points.
- **LineStyle:** Specify the line style of the line connecting the data points as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the line connecting the data points.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Symbols

Represents the properties that allow you to customize the look of symbols that form the Radar Line plot.

- **BackgroundColor**: Change the color of the background.
- **LineColor**: Change the color of the line.
- **LineStyle**: Choose from different styles of lines such as Dotted, Dashed, Solid, Double, Groove, etc.

- **LineWidth**: Select the width of the lines in points.
- **Shape**: Choose from different shapes of symbols such as Dot, Box, Diamond, Triangle, X, Dash, Plus, etc.
- **Visible**: Set to true to display data point symbols.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Radar Line Plots.

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default**: Indicates a straight line.
- **Spline**: Indicates a bezier curve.
- **Step Center, Step Left, and Step Right**: Indicates a stepped line with different step directions.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

StartAngle

Indicates the arc angle that defines the clockwise rotation of the chart. A full rotation makes 360 degrees.

SymbolOpacity

Represents the opacity value of symbol fill color.

Encodings

Category Encoding

The Category Encoding of a Radar Line Plot is a set of properties that determine the period over which the plot generates connected data points representing the Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Color Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Radar Line Plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the area subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked Radar Line Plot.
  - Cluster: You can use this value to configure area subsections that overlap each other.
  - None: Equals to Cluster.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.

- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Shape Encoding

The Shape Encoding enables the shape legend of the Details or Category Encoding in Radar Line charts. It includes the Aggregate function and the shape expression, which is elaborated below:

Action

The action to take when the shape legend is clicked.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Value

The Value property is the collection and usually takes a bound field. However, the Radar Line plots take the first item from the collection.

Size Encoding

The Size Encoding enables the Aggregate function and Size expression in Radar Line charts. The Size Encoding works solely with numeric values and breaks down data values into ranges that determine the symbol size.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Radar Line Plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Radar Line Plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

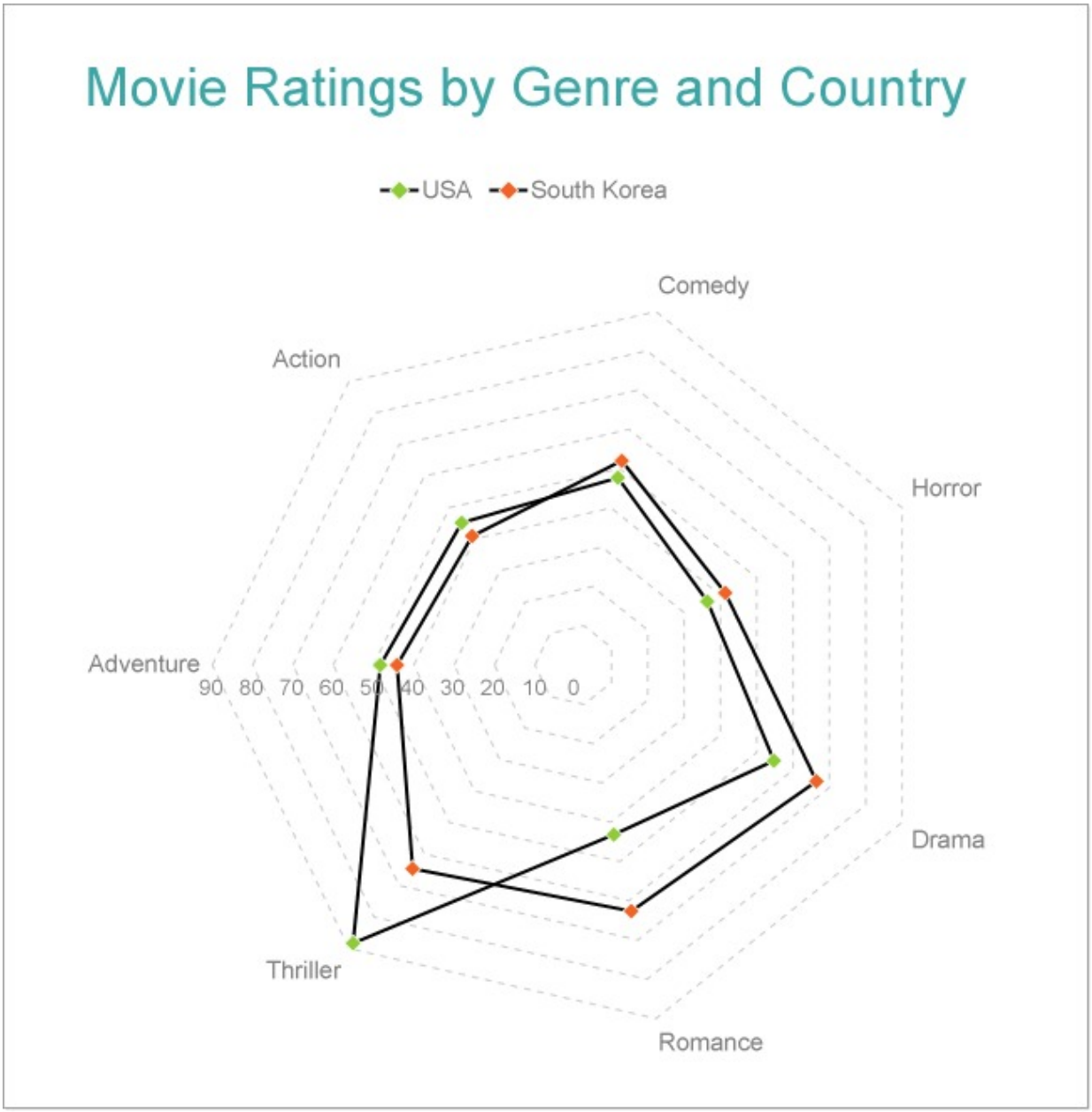
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Radar Line Chart with Multiple Lines

This walkthrough creates a Radar Line Chart with Multiple Lines. The chart shows the average movie ratings by certain genres in different countries. Such a chart helps you to analyze the multivariate data for deeper insights. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data

```
[
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 87,
    "AudienceRating": 81,
    "Budget": 8,
    "Country": "UK"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 9,
    "AudienceRating": 44,
    "Budget": 105,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 30,
    "AudienceRating": 52,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 93,
    "AudienceRating": 84,
    "Budget": 18,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 55,
    "AudienceRating": 70,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 39,
    "AudienceRating": 63,
    "Budget": 200,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 40,
    "AudienceRating": 71,
    "Budget": 30,
    "Country": "USA"
```

```
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 48,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 93,
  "AudienceRating": 93,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 5,
  "AudienceRating": 51,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 89,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 13,
  "AudienceRating": 40,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 89,
  "AudienceRating": 64,
  "Budget": 7,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 71,
  "Budget": 19,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 4,
  "AudienceRating": 46,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 54,
  "AudienceRating": 84,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 89,
  "AudienceRating": 56,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 43,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 72,
  "Budget": 200,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 37,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Adventure",
```

```
  "RottenTomatoesRating": 30,
  "AudienceRating": 46,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 6,
  "AudienceRating": 35,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 64,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 21,
  "AudienceRating": 35,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 94,
  "AudienceRating": 78,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 66,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 23,
  "AudienceRating": 31,
```

```
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 55,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 34,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 55,
  "AudienceRating": 69,
  "Budget": 78,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 83,
  "AudienceRating": 92,
  "Budget": 237,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 67,
  "AudienceRating": 74,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 63,
  "AudienceRating": 59,
  "Budget": 70,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 7,
    "AudienceRating": 32,
    "Budget": 45,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 44,
    "AudienceRating": 38,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 79,
    "AudienceRating": 60,
    "Budget": 20,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 9,
    "AudienceRating": 33,
    "Budget": 45,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 35,
    "AudienceRating": 50,
    "Budget": 70,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 65,
    "AudienceRating": 57,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 19,
    "AudienceRating": 50,
    "Budget": 17,
    "Country": "USA"
  },
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 25,
  "AudienceRating": 63,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 84,
  "AudienceRating": 80,
  "Budget": 4,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 52,
  "Budget": 150,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 41,
  "AudienceRating": 56,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 88,
  "AudienceRating": 86,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 69,
  "AudienceRating": 66,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 65,
  "Budget": 68,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 11,
  "AudienceRating": 56,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 90,
  "AudienceRating": 77,
  "Budget": 33,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 47,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 61,
  "AudienceRating": 62,
  "Budget": 26,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 48,
  "Budget": 42,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 86,
  "AudienceRating": 63,
  "Budget": 2,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 36,
  "AudienceRating": 66,
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
```

```
  "AudienceRating": 64,
  "Budget": 37,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 75,
  "Budget": 140,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 23,
  "AudienceRating": 42,
  "Budget": 26,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 13,
  "AudienceRating": 43,
  "Budget": 85,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 86,
  "AudienceRating": 61,
  "Budget": 10,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 62,
  "AudienceRating": 84,
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 50,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 81,
  "AudienceRating": 77,
```

```
  "Budget": 6,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 48,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 48,
  "Budget": 125,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 67,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 13,
  "AudienceRating": 56,
  "Budget": 140,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 55,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 34,
  "Budget": 90,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 70,
  "Country": "South Korea"
```

```
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 84,
  "AudienceRating": 63,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 19,
  "AudienceRating": 45,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 20,
  "AudienceRating": 56,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 12,
  "AudienceRating": 47,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 44,
  "AudienceRating": 50,
  "Budget": 163,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 62,
  "AudienceRating": 54,
  "Budget": 13,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 78,
  "AudienceRating": 81,
  "Budget": 50,
  "Country": "USA"
},
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 80,
  "AudienceRating": 51,
  "Budget": 7,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 1,
  "AudienceRating": 63,
  "Budget": 6,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 17,
  "AudienceRating": 35,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 66,
  "AudienceRating": 58,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 29,
  "AudienceRating": 66,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 48,
  "Budget": 21,
  "Country": "Germany"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 42,
  "AudienceRating": 69,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 71,
  "AudienceRating": 72,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 52,
  "AudienceRating": 45,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 49,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 47,
  "AudienceRating": 63,
  "Budget": 21,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 12,
  "AudienceRating": 31,
  "Budget": 58,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 47,
  "Budget": 69,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
```

```
  "AudienceRating": 28,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 91,
  "AudienceRating": 81,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 84,
  "AudienceRating": 81,
  "Budget": 37,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 59,
  "AudienceRating": 37,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 48,
  "AudienceRating": 47,
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 78,
  "AudienceRating": 75,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 92,
  "AudienceRating": 61,
  "Budget": 30,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 27,
    "AudienceRating": 44,
    "Budget": 75,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 15,
    "AudienceRating": 28,
    "Budget": 30,
    "Country": "South Korea"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 7,
    "AudienceRating": 38,
    "Budget": 50,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 26,
    "AudienceRating": 49,
    "Budget": 40,
    "Country": "South Korea"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 93,
    "AudienceRating": 79,
    "Budget": 15,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 45,
    "AudienceRating": 38,
    "Budget": 45,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 40,
    "AudienceRating": 56,
    "Budget": 50,
    "Country": "South Korea"
```

```
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 64,
  "AudienceRating": 38,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 6,
  "AudienceRating": 28,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 72,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 75,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 36,
  "AudienceRating": 46,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 93,
  "AudienceRating": 80,
  "Budget": 85,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
  "AudienceRating": 38,
  "Budget": 20,
  "Country": "UK"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 23,
  "AudienceRating": 56,
  "Budget": 175,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 55,
  "Budget": 21,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 74,
  "AudienceRating": 53,
  "Budget": 5,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 62,
  "AudienceRating": 37,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 55,
  "Budget": 31,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 62,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 25,
  "AudienceRating": 43,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Action",
```

```
    "RottenTomatoesRating": 35,
    "AudienceRating": 55,
    "Budget": 130,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 28,
    "AudienceRating": 76,
    "Budget": 85,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 78,
    "AudienceRating": 83,
    "Budget": 125,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 41,
    "AudienceRating": 50,
    "Budget": 24,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 61,
    "AudienceRating": 56,
    "Budget": 40,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 22,
    "AudienceRating": 52,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 40,
    "AudienceRating": 51,
    "Budget": 1,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 13,
    "AudienceRating": 59,
```

```
  "Budget": 13,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 11,
  "AudienceRating": 53,
  "Budget": 73,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 71,
  "AudienceRating": 71,
  "Budget": 24,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 73,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 75,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 52,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 25,
  "AudienceRating": 52,
  "Budget": 19,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 71,
  "AudienceRating": 68,
  "Budget": 35,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 75,
    "AudienceRating": 68,
    "Budget": 30,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 38,
    "AudienceRating": 57,
    "Budget": 52,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 68,
    "AudienceRating": 58,
    "Budget": 75,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 8,
    "AudienceRating": 36,
    "Budget": 35,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 22,
    "AudienceRating": 50,
    "Budget": 150,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 33,
    "AudienceRating": 62,
    "Budget": 175,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 29,
    "AudienceRating": 43,
    "Budget": 13,
    "Country": "South Korea"
  },
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 73,
  "AudienceRating": 63,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 51,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 56,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
  "AudienceRating": 47,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 56,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 3,
  "AudienceRating": 61,
  "Budget": 25,
  "Country": "Germany"
},
{
```

```
  "Genre": "Drama",
  "RottenTomatoesRating": 80,
  "AudienceRating": 90,
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 48,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 60,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 75,
  "AudienceRating": 40,
  "Budget": 0,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 82,
  "AudienceRating": 86,
  "Budget": 53,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 10,
  "AudienceRating": 59,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 31,
  "Budget": 112,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 58,
```

```
  "AudienceRating": 85,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 35,
  "AudienceRating": 44,
  "Budget": 36,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 26,
  "AudienceRating": 64,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 20,
  "AudienceRating": 45,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 67,
  "Budget": 150,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 67,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 44,
  "AudienceRating": 66,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 54,
  "AudienceRating": 65,
```

```
  "Budget": 12,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 83,
  "AudienceRating": 75,
  "Budget": 250,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 78,
  "AudienceRating": 82,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 42,
  "AudienceRating": 60,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 87,
  "AudienceRating": 70,
  "Budget": 83,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 54,
  "AudienceRating": 68,
  "Budget": 7,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 65,
  "AudienceRating": 76,
  "Budget": 11,
  "Country": "UK"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 15,
  "AudienceRating": 61,
  "Budget": 18,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 69,
  "AudienceRating": 72,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 44,
  "AudienceRating": 45,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 64,
  "AudienceRating": 57,
  "Budget": 36,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 46,
  "AudienceRating": 57,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 93,
  "AudienceRating": 84,
  "Budget": 150,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 69,
  "AudienceRating": 69,
  "Budget": 150,
  "Country": "Germany"
},
```

```
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 32,
  "AudienceRating": 57,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 38,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 71,
  "AudienceRating": 57,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 83,
  "AudienceRating": 72,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 73,
  "Budget": 85,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 38,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 44,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 36,
  "AudienceRating": 59,
  "Budget": 75,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 72,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 38,
  "AudienceRating": 55,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 86,
  "AudienceRating": 93,
  "Budget": 160,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 59,
  "Budget": 185,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 88,
  "AudienceRating": 87,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 39,
  "AudienceRating": 51,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 67,
```

```
  "AudienceRating": 65,
  "Budget": 2,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 82,
  "AudienceRating": 90,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 94,
  "AudienceRating": 91,
  "Budget": 186,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 74,
  "AudienceRating": 80,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 63,
  "Budget": 85,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 42,
  "AudienceRating": 84,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 4,
  "AudienceRating": 59,
  "Budget": 79,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 85,
  "AudienceRating": 77,
  "Budget": 0,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 42,
    "AudienceRating": 37,
    "Budget": 16,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 38,
    "AudienceRating": 55,
    "Budget": 45,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 13,
    "AudienceRating": 24,
    "Budget": 47,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 61,
    "AudienceRating": 55,
    "Budget": 45,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 16,
    "AudienceRating": 50,
    "Budget": 83,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 94,
    "AudienceRating": 89,
    "Budget": 8,
    "Country": "South Korea"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 19,
    "AudienceRating": 63,
    "Budget": 80,
    "Country": "USA"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 45,
  "AudienceRating": 58,
  "Budget": 12,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 76,
  "AudienceRating": 83,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 25,
  "AudienceRating": 48,
  "Budget": 70,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 11,
  "AudienceRating": 45,
  "Budget": 75,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 78,
  "AudienceRating": 26,
  "Budget": 10,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 52,
  "Budget": 117,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 91,
  "AudienceRating": 83,
  "Budget": 33,
  "Country": "Germany"
},
{
```

```
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 50,
  "Budget": 50,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 26,
  "AudienceRating": 38,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 34,
  "AudienceRating": 46,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 49,
  "Budget": 19,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 41,
  "Budget": 58,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 19,
  "AudienceRating": 36,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 62,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 8,
  "AudienceRating": 55,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 28,
  "AudienceRating": 62,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 69,
  "AudienceRating": 73,
  "Budget": 27,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 48,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 40,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 79,
  "AudienceRating": 86,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 48,
  "AudienceRating": 55,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 18,
  "AudienceRating": 40,
```

```
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 33,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 73,
  "AudienceRating": 66,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 29,
  "AudienceRating": 71,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 22,
  "AudienceRating": 51,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 61,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 76,
  "Budget": 66,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 86,
  "AudienceRating": 73,
  "Budget": 4,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 63,
    "AudienceRating": 77,
    "Budget": 55,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 9,
    "AudienceRating": 46,
    "Budget": 50,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 16,
    "AudienceRating": 36,
    "Budget": 35,
    "Country": "South Korea"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 19,
    "AudienceRating": 42,
    "Budget": 60,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 2,
    "AudienceRating": 31,
    "Budget": 30,
    "Country": "UK"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 90,
    "AudienceRating": 72,
    "Budget": 25,
    "Country": "UK"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 93,
    "AudienceRating": 84,
    "Budget": 17,
    "Country": "Germany"
  },
```

```
{
  "Genre": "Action",
  "RottenTomatoesRating": 34,
  "AudienceRating": 68,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 70,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 93,
  "AudienceRating": 86,
  "Budget": 145,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 95,
  "AudienceRating": 89,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 38,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 54,
  "AudienceRating": 54,
  "Budget": 40,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 61,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 56,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 47,
  "AudienceRating": 54,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 63,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 51,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 43,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 73,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 9,
```

```
  "AudienceRating": 29,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 83,
  "AudienceRating": 84,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 49,
  "AudienceRating": 61,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 77,
  "AudienceRating": 54,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 31,
  "AudienceRating": 72,
  "Budget": 130,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 23,
  "AudienceRating": 37,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 70,
  "AudienceRating": 70,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 8,
  "AudienceRating": 48,
```

```
  "Budget": 56,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 29,
  "AudienceRating": 59,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 21,
  "AudienceRating": 27,
  "Budget": 3,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 73,
  "AudienceRating": 67,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 60,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 31,
  "AudienceRating": 56,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 51,
  "AudienceRating": 50,
  "Budget": 37,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 37,
  "AudienceRating": 40,
  "Budget": 80,
  "Country": "Germany"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 25,
  "AudienceRating": 57,
  "Budget": 50,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 95,
  "AudienceRating": 84,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 64,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 49,
  "AudienceRating": 57,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 56,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 34,
  "AudienceRating": 66,
  "Budget": 5,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 51,
  "AudienceRating": 40,
  "Budget": 18,
  "Country": "USA"
},
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 20,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 70,
  "AudienceRating": 74,
  "Budget": 85,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 5,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 37,
  "AudienceRating": 54,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 0,
  "AudienceRating": 36,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 55,
  "AudienceRating": 65,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 0,
  "Country": "USA"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 68,
  "AudienceRating": 79,
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 47,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 21,
  "AudienceRating": 82,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 82,
  "AudienceRating": 56,
  "Budget": 1,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 0,
  "AudienceRating": 0,
  "Budget": 3,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 68,
  "AudienceRating": 58,
  "Budget": 5,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 11,
```

```
  "AudienceRating": 41,
  "Budget": 26,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 50,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 74,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 95,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 68,
  "AudienceRating": 74,
  "Budget": 26,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 74,
  "AudienceRating": 45,
  "Budget": 24,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 60,
  "AudienceRating": 72,
  "Budget": 50,
```

```
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 34,
    "AudienceRating": 61,
    "Budget": 250,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 45,
    "AudienceRating": 74,
    "Budget": 300,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 88,
    "AudienceRating": 62,
    "Budget": 3,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 7,
    "AudienceRating": 31,
    "Budget": 15,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 64,
    "AudienceRating": 52,
    "Budget": 40,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 8,
    "AudienceRating": 55,
    "Budget": 20,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 17,
    "AudienceRating": 37,
    "Budget": 60,
    "Country": "UK"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 71,
  "Budget": 200,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 8,
  "AudienceRating": 46,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 47,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 47,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 62,
  "Budget": 230,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 59,
  "AudienceRating": 46,
  "Budget": 12,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 50,
  "Budget": 50,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Drama",
  "RottenTomatoesRating": 85,
  "AudienceRating": 61,
  "Budget": 12,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 70,
  "Budget": 48,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 81,
  "Budget": 110,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 72,
  "Budget": 58,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 11,
  "AudienceRating": 41,
  "Budget": 42,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 57,
  "AudienceRating": 59,
  "Budget": 4,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 28,
  "AudienceRating": 70,
  "Budget": 16,
  "Country": "South Korea"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 22,
  "AudienceRating": 43,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 83,
  "AudienceRating": 87,
  "Budget": 93,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 59,
  "Budget": 200,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 77,
  "AudienceRating": 75,
  "Budget": 28,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 20,
  "AudienceRating": 68,
  "Budget": 140,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 62,
  "AudienceRating": 62,
  "Budget": 110,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 30,
  "AudienceRating": 39,
```

```
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 18,
  "AudienceRating": 70,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 40,
  "AudienceRating": 67,
  "Budget": 11,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 81,
  "AudienceRating": 83,
  "Budget": 60,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 58,
  "AudienceRating": 57,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 10,
  "AudienceRating": 32,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 41,
  "Budget": 3,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 77,
  "Budget": 55,
```

```
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 49,
    "AudienceRating": 81,
    "Budget": 58,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 15,
    "AudienceRating": 49,
    "Budget": 100,
    "Country": "UK"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 46,
    "AudienceRating": 61,
    "Budget": 19,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 80,
    "AudienceRating": 80,
    "Budget": 7,
    "Country": "South Korea"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 16,
    "AudienceRating": 25,
    "Budget": 25,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 57,
    "AudienceRating": 60,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 70,
    "AudienceRating": 81,
    "Budget": 90,
    "Country": "UK"
  },
```

```
{
  "Genre": "Action",
  "RottenTomatoesRating": 60,
  "AudienceRating": 79,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 66,
  "Budget": 39,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 82,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 7,
  "AudienceRating": 40,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 68,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 19,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 50,
  "AudienceRating": 41,
  "Budget": 7,
  "Country": "UK"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 81,
  "AudienceRating": 47,
  "Budget": 15,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 0,
  "AudienceRating": 0,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 22,
  "AudienceRating": 38,
  "Budget": 13,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 45,
  "AudienceRating": 42,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 79,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 92,
  "AudienceRating": 81,
  "Budget": 32,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 61,
  "Budget": 120,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 61,
```

```
  "AudienceRating": 54,
  "Budget": 258,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 40,
  "Budget": 27,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 94,
  "AudienceRating": 91,
  "Budget": 140,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 76,
  "AudienceRating": 86,
  "Budget": 70,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 54,
  "AudienceRating": 68,
  "Budget": 65,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 26,
  "AudienceRating": 81,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 67,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 49,
  "AudienceRating": 20,
```

```
  "Budget": 22,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 65,
  "AudienceRating": 27,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 28,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 37,
  "AudienceRating": 62,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 23,
  "AudienceRating": 48,
  "Budget": 82,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 59,
  "Budget": 5,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 82,
  "AudienceRating": 78,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 88,
  "AudienceRating": 87,
  "Budget": 20,
  "Country": "South Korea"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 36,
  "Budget": 65,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 39,
  "AudienceRating": 42,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 36,
  "AudienceRating": 44,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 28,
  "AudienceRating": 46,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 92,
  "AudienceRating": 85,
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 83,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 57,
  "Budget": 32,
  "Country": "USA"
},
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 48,
  "AudienceRating": 48,
  "Budget": 29,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 64,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 71,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 72,
  "AudienceRating": 67,
  "Budget": 51,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 74,
  "AudienceRating": 78,
  "Budget": 130,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 97,
  "AudienceRating": 91,
  "Budget": 16,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 20,
  "AudienceRating": 47,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Thriller",
```

```
  "RottenTomatoesRating": 78,
  "AudienceRating": 74,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 62,
  "AudienceRating": 57,
  "Budget": 21,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 62,
  "Budget": 41,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 48,
  "AudienceRating": 68,
  "Budget": 80,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 7,
  "AudienceRating": 41,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 93,
  "AudienceRating": 91,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 27,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 63,
```

```
  "AudienceRating": 84,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 65,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 81,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 52,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 78,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 49,
  "AudienceRating": 63,
  "Budget": 155,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 66,
  "AudienceRating": 55,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 56,
  "AudienceRating": 65,
  "Budget": 25,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 72,
    "AudienceRating": 54,
    "Budget": 20,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 67,
    "AudienceRating": 79,
    "Budget": 18,
    "Country": "USA"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 94,
    "AudienceRating": 96,
    "Budget": 185,
    "Country": "USA"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 19,
    "AudienceRating": 34,
    "Budget": 100,
    "Country": "South Korea"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 76,
    "AudienceRating": 70,
    "Budget": 20,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 89,
    "AudienceRating": 88,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 23,
    "AudienceRating": 31,
    "Budget": 70,
    "Country": "UK"
```

```
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 60,
  "AudienceRating": 68,
  "Budget": 14,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 39,
  "AudienceRating": 43,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 58,
  "Budget": 82,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 22,
  "AudienceRating": 51,
  "Budget": 12,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 91,
  "AudienceRating": 88,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 29,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 65,
  "Budget": 55,
  "Country": "UK"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
  "AudienceRating": 75,
  "Budget": 22,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 87,
  "AudienceRating": 89,
  "Budget": 90,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 55,
  "Budget": 180,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 44,
  "AudienceRating": 47,
  "Budget": 120,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 35,
  "AudienceRating": 58,
  "Budget": 80,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 19,
  "AudienceRating": 29,
  "Budget": 48,
  "Country": "Germany"
},
{
  "Genre": "Drama",
```

```
  "RottenTomatoesRating": 17,
  "AudienceRating": 51,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 3,
  "AudienceRating": 22,
  "Budget": 5,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 30,
  "AudienceRating": 41,
  "Budget": 60,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 75,
  "AudienceRating": 91,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 11,
  "AudienceRating": 50,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 54,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 97,
  "AudienceRating": 83,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 85,
  "AudienceRating": 76,
```

```
  "Budget": 13,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 138,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 42,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 39,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 47,
  "Budget": 19,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 54,
  "Budget": 13,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 61,
  "AudienceRating": 47,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 75,
  "Budget": 40,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 94,
    "AudienceRating": 72,
    "Budget": 4,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 52,
    "AudienceRating": 78,
    "Budget": 80,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 66,
    "AudienceRating": 85,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 6,
    "AudienceRating": 42,
    "Budget": 150,
    "Country": "South Korea"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 73,
    "AudienceRating": 32,
    "Budget": 2,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 19,
    "AudienceRating": 66,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 84,
    "AudienceRating": 82,
    "Budget": 40,
    "Country": "South Korea"
  },
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 39,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 55,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 38,
  "Budget": 62,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 32,
  "AudienceRating": 57,
  "Budget": 65,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 43,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 72,
  "AudienceRating": 64,
  "Budget": 18,
  "Country": "USA"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 14,
  "AudienceRating": 40,
  "Budget": 145,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 97,
  "AudienceRating": 87,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 33,
  "AudienceRating": 52,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 65,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 57,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 12,
  "AudienceRating": 47,
  "Budget": 59,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 74,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 9,
```

```
  "AudienceRating": 53,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 20,
  "AudienceRating": 43,
  "Budget": 37,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 75,
  "AudienceRating": 65,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 4,
  "AudienceRating": 29,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 46,
  "AudienceRating": 32,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 50,
  "AudienceRating": 48,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 56,
  "AudienceRating": 78,
  "Budget": 11,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 90,
  "AudienceRating": 78,
```

```
    "Budget": 75,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 22,
    "AudienceRating": 43,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 56,
    "AudienceRating": 59,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 18,
    "AudienceRating": 61,
    "Budget": 15,
    "Country": "UK"
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Ratings'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Go to the **Filters** page, add a new filter value, and set its properties as below.

Expression	Operator	Values
`=[Country]`	In	`USA` `South Korea`

4. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

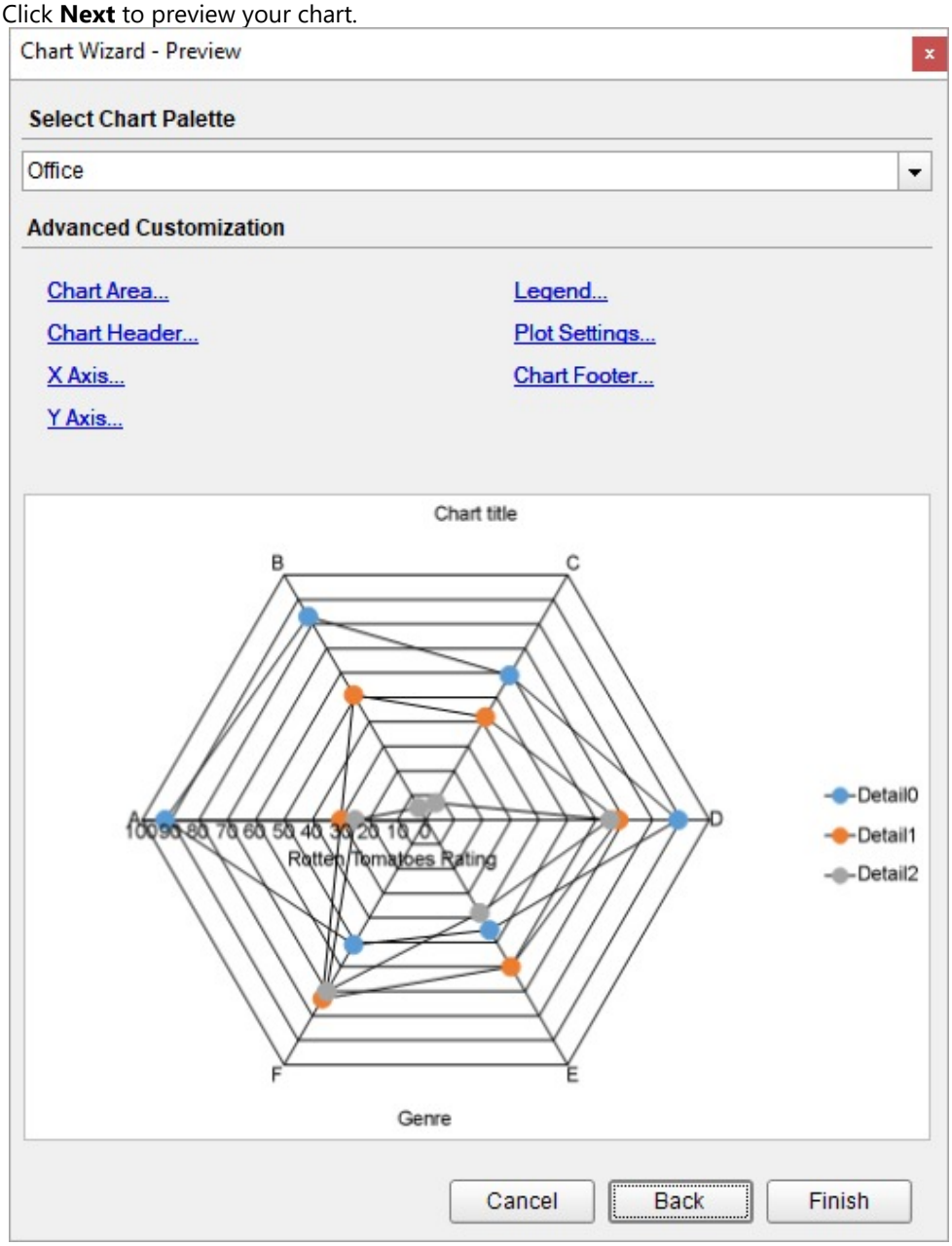
1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data

and the chart type.

2. Select the **Dataset Name** as 'Ratings' and the **Chart Type** as 'Radar Line'.
3. Click **Next** to proceed. Here, we will define a data value to display the rotten tomatoes ratings in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[RottenTomatoesRating]`	Average

5. In **Choose Data Category**, set **Field** to `=[Genre]`. We will add more customizations to the category in later steps.
6. In **Choose Data Subcategories**, set **Field** to `=[Country]` to further categorize the movie ratings based on different countries (like the USA and South Korea).
7. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Line Style > Width**: 1.5pt
    - **Symbol Settings > Shape**: Diamond
    - **Symbol Border Settings > Style**: Solid
    - **Symbol Border Settings > Color**: White
    - **Symbol Border Settings > Width**: 0.25pt
3. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 10
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and set the following properties:
    - **Font > Size:** 10pt
    - **Font > Color**: Gray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the X-axis.

Chart Palette

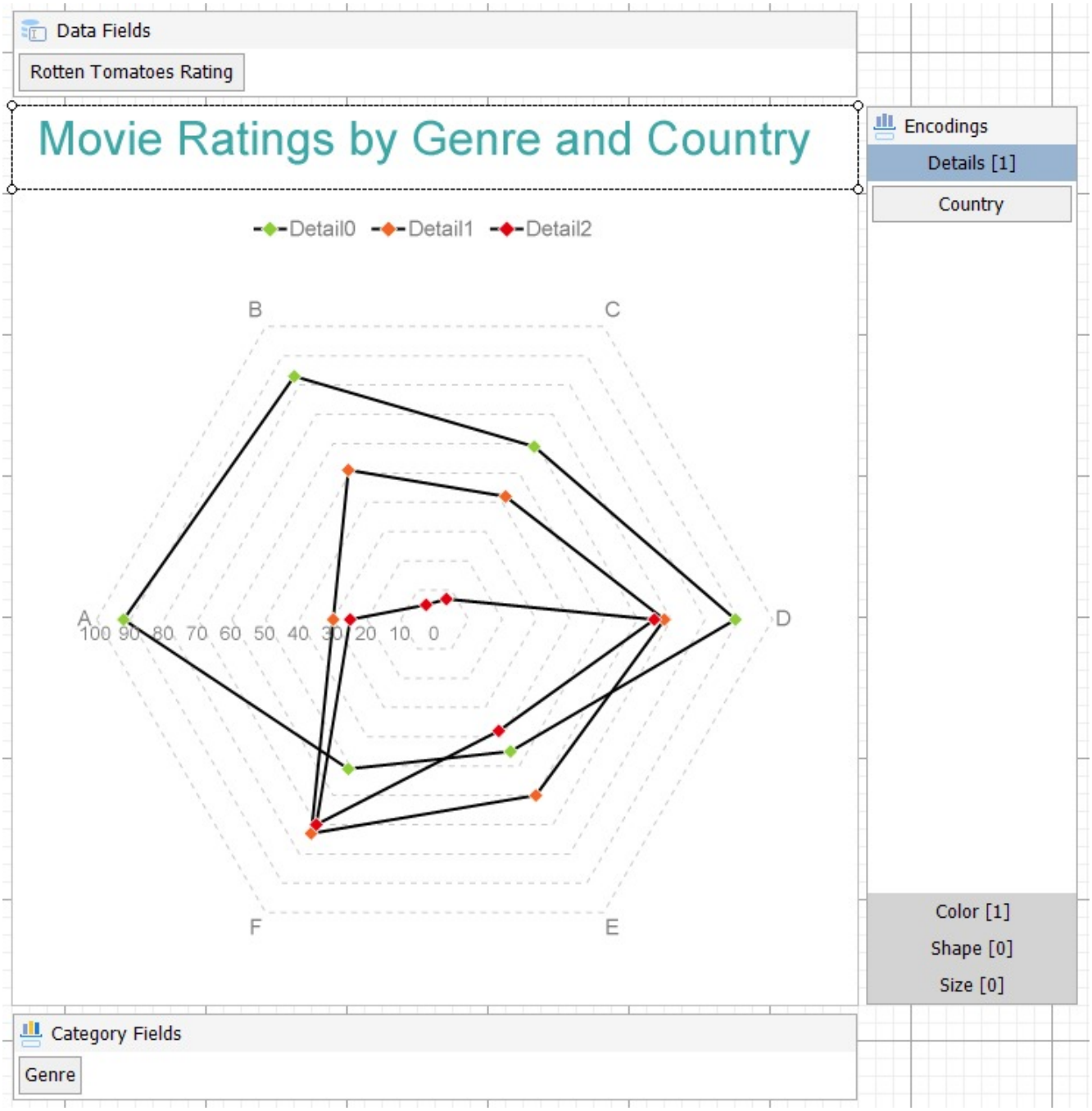
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select **Custom** from the drop-down and add the following colors.
    - #8fcd37
    - #f26324
3. Click **OK** to complete setting up the custom palette.

Legend - Color

1. To open the smart panel for the legend, click the 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: Gray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Movie Ratings by Genre and Country'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

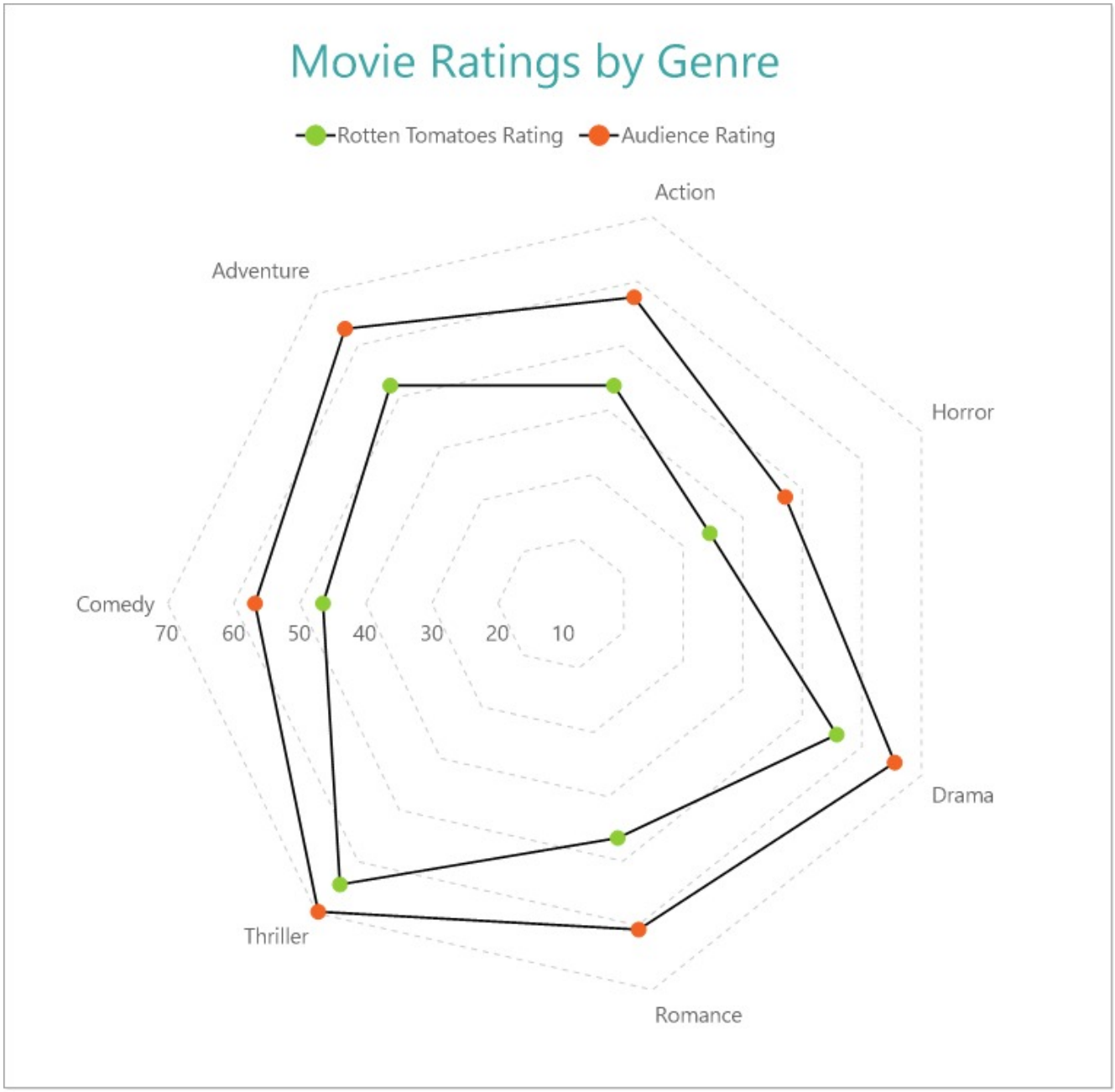


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Multiple Values Radar Line Chart

This walkthrough creates a Multiple Values Radar Line Chart. The chart shows the average rotten tomatoes rating and audience rating for multiple movie genres. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 87,
    "AudienceRating": 81,
    "Budget": 8,
    "Country": "UK"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 9,
    "AudienceRating": 44,
    "Budget": 105,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 30,
    "AudienceRating": 52,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 93,
    "AudienceRating": 84,
    "Budget": 18,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 55,
    "AudienceRating": 70,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 39,
    "AudienceRating": 63,
    "Budget": 200,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 40,
    "AudienceRating": 71,
    "Budget": 30,
```

```
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 48,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 93,
  "AudienceRating": 93,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 5,
  "AudienceRating": 51,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 89,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 13,
  "AudienceRating": 40,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 89,
  "AudienceRating": 64,
  "Budget": 7,
  "Country": "South Korea"
},
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 71,
  "Budget": 19,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 4,
  "AudienceRating": 46,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 54,
  "AudienceRating": 84,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 89,
  "AudienceRating": 56,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 43,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 72,
  "Budget": 200,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 37,
  "Budget": 40,
  "Country": "UK"
},
{
```

```
  "Genre": "Adventure",
  "RottenTomatoesRating": 30,
  "AudienceRating": 46,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 6,
  "AudienceRating": 35,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 64,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 21,
  "AudienceRating": 35,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 94,
  "AudienceRating": 78,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 66,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 23,
```

```
  "AudienceRating": 31,
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 55,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 34,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 55,
  "AudienceRating": 69,
  "Budget": 78,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 83,
  "AudienceRating": 92,
  "Budget": 237,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 67,
  "AudienceRating": 74,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 63,
  "AudienceRating": 59,
```

```
  "Budget": 70,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 7,
  "AudienceRating": 32,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 44,
  "AudienceRating": 38,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 79,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 9,
  "AudienceRating": 33,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 35,
  "AudienceRating": 50,
  "Budget": 70,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 65,
  "AudienceRating": 57,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 19,
  "AudienceRating": 50,
  "Budget": 17,
  "Country": "USA"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 25,
  "AudienceRating": 63,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 84,
  "AudienceRating": 80,
  "Budget": 4,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 52,
  "Budget": 150,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 41,
  "AudienceRating": 56,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 88,
  "AudienceRating": 86,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 69,
  "AudienceRating": 66,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 65,
  "Budget": 68,
  "Country": "South Korea"
},
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 11,
  "AudienceRating": 56,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 90,
  "AudienceRating": 77,
  "Budget": 33,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 47,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 61,
  "AudienceRating": 62,
  "Budget": 26,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 48,
  "Budget": 42,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 86,
  "AudienceRating": 63,
  "Budget": 2,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 36,
  "AudienceRating": 66,
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 78,
  "AudienceRating": 64,
  "Budget": 37,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 75,
  "Budget": 140,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 23,
  "AudienceRating": 42,
  "Budget": 26,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 13,
  "AudienceRating": 43,
  "Budget": 85,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 86,
  "AudienceRating": 61,
  "Budget": 10,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 62,
  "AudienceRating": 84,
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 50,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 81,
```

```
  "AudienceRating": 77,
  "Budget": 6,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 48,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 48,
  "Budget": 125,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 67,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 13,
  "AudienceRating": 56,
  "Budget": 140,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 55,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 34,
  "Budget": 90,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 70,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 84,
    "AudienceRating": 63,
    "Budget": 60,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 19,
    "AudienceRating": 45,
    "Budget": 30,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 20,
    "AudienceRating": 56,
    "Budget": 15,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 12,
    "AudienceRating": 47,
    "Budget": 60,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 44,
    "AudienceRating": 50,
    "Budget": 163,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 62,
    "AudienceRating": 54,
    "Budget": 13,
    "Country": "USA"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 78,
    "AudienceRating": 81,
    "Budget": 50,
    "Country": "USA"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 80,
  "AudienceRating": 51,
  "Budget": 7,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 1,
  "AudienceRating": 63,
  "Budget": 6,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 17,
  "AudienceRating": 35,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 66,
  "AudienceRating": 58,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 29,
  "AudienceRating": 66,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 48,
  "Budget": 21,
  "Country": "Germany"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 69,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 71,
  "AudienceRating": 72,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 52,
  "AudienceRating": 45,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 49,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 47,
  "AudienceRating": 63,
  "Budget": 21,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 12,
  "AudienceRating": 31,
  "Budget": 58,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 47,
  "Budget": 69,
  "Country": "UK"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 2,
  "AudienceRating": 28,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 91,
  "AudienceRating": 81,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 84,
  "AudienceRating": 81,
  "Budget": 37,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 59,
  "AudienceRating": 37,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 48,
  "AudienceRating": 47,
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 78,
  "AudienceRating": 75,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 92,
  "AudienceRating": 61,
```

```
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 44,
  "Budget": 75,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 15,
  "AudienceRating": 28,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 7,
  "AudienceRating": 38,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 93,
  "AudienceRating": 79,
  "Budget": 15,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 45,
  "AudienceRating": 38,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 56,
  "Budget": 50,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 64,
    "AudienceRating": 38,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 6,
    "AudienceRating": 28,
    "Budget": 20,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 27,
    "AudienceRating": 72,
    "Budget": 80,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 85,
    "AudienceRating": 75,
    "Budget": 8,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 36,
    "AudienceRating": 46,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 93,
    "AudienceRating": 80,
    "Budget": 85,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 2,
    "AudienceRating": 38,
    "Budget": 20,
    "Country": "UK"
  },
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 23,
  "AudienceRating": 56,
  "Budget": 175,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 55,
  "Budget": 21,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 74,
  "AudienceRating": 53,
  "Budget": 5,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 62,
  "AudienceRating": 37,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 55,
  "Budget": 31,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 62,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 25,
  "AudienceRating": 43,
  "Budget": 18,
  "Country": "USA"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 35,
  "AudienceRating": 55,
  "Budget": 130,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 76,
  "Budget": 85,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 83,
  "Budget": 125,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 41,
  "AudienceRating": 50,
  "Budget": 24,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 61,
  "AudienceRating": 56,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 22,
  "AudienceRating": 52,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 40,
  "AudienceRating": 51,
  "Budget": 1,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
```

```
  "AudienceRating": 59,
  "Budget": 13,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 11,
  "AudienceRating": 53,
  "Budget": 73,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 71,
  "AudienceRating": 71,
  "Budget": 24,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 73,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 75,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 52,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 25,
  "AudienceRating": 52,
  "Budget": 19,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 71,
  "AudienceRating": 68,
```

```
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 75,
  "AudienceRating": 68,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 57,
  "Budget": 52,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 58,
  "Budget": 75,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 8,
  "AudienceRating": 36,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 50,
  "Budget": 150,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 62,
  "Budget": 175,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 29,
  "AudienceRating": 43,
  "Budget": 13,
  "Country": "South Korea"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 73,
  "AudienceRating": 63,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 51,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 56,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
  "AudienceRating": 47,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 56,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 3,
  "AudienceRating": 61,
  "Budget": 25,
  "Country": "Germany"
},
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 80,
  "AudienceRating": 90,
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 48,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 60,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 75,
  "AudienceRating": 40,
  "Budget": 0,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 82,
  "AudienceRating": 86,
  "Budget": 53,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 10,
  "AudienceRating": 59,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 31,
  "Budget": 112,
  "Country": "UK"
},
{
  "Genre": "Drama",
```

```
  "RottenTomatoesRating": 58,
  "AudienceRating": 85,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 35,
  "AudienceRating": 44,
  "Budget": 36,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 26,
  "AudienceRating": 64,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 20,
  "AudienceRating": 45,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 67,
  "Budget": 150,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 67,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 44,
  "AudienceRating": 66,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 54,
```

```
  "AudienceRating": 65,
  "Budget": 12,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 83,
  "AudienceRating": 75,
  "Budget": 250,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 78,
  "AudienceRating": 82,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 42,
  "AudienceRating": 60,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 87,
  "AudienceRating": 70,
  "Budget": 83,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 54,
  "AudienceRating": 68,
  "Budget": 7,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 65,
  "AudienceRating": 76,
  "Budget": 11,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 15,
    "AudienceRating": 61,
    "Budget": 18,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 69,
    "AudienceRating": 72,
    "Budget": 35,
    "Country": "USA"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 44,
    "AudienceRating": 45,
    "Budget": 10,
    "Country": "South Korea"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 64,
    "AudienceRating": 57,
    "Budget": 36,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 46,
    "AudienceRating": 57,
    "Budget": 35,
    "Country": "South Korea"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 93,
    "AudienceRating": 84,
    "Budget": 150,
    "Country": "USA"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 69,
    "AudienceRating": 69,
    "Budget": 150,
    "Country": "Germany"
```

```
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 32,
  "AudienceRating": 57,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 38,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 71,
  "AudienceRating": 57,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 83,
  "AudienceRating": 72,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 73,
  "Budget": 85,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 38,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 44,
  "Budget": 55,
  "Country": "USA"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 59,
  "Budget": 75,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 72,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 38,
  "AudienceRating": 55,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 86,
  "AudienceRating": 93,
  "Budget": 160,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 59,
  "Budget": 185,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 88,
  "AudienceRating": 87,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 39,
  "AudienceRating": 51,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Horror",
```

```
  "RottenTomatoesRating": 67,
  "AudienceRating": 65,
  "Budget": 2,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 82,
  "AudienceRating": 90,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 94,
  "AudienceRating": 91,
  "Budget": 186,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 74,
  "AudienceRating": 80,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 63,
  "Budget": 85,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 42,
  "AudienceRating": 84,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 4,
  "AudienceRating": 59,
  "Budget": 79,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 85,
  "AudienceRating": 77,
```

```
  "Budget": 0,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 42,
  "AudienceRating": 37,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 55,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 13,
  "AudienceRating": 24,
  "Budget": 47,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 61,
  "AudienceRating": 55,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 50,
  "Budget": 83,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 94,
  "AudienceRating": 89,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 19,
  "AudienceRating": 63,
  "Budget": 80,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 45,
    "AudienceRating": 58,
    "Budget": 12,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 76,
    "AudienceRating": 83,
    "Budget": 30,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 25,
    "AudienceRating": 48,
    "Budget": 70,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 11,
    "AudienceRating": 45,
    "Budget": 75,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 78,
    "AudienceRating": 26,
    "Budget": 10,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 53,
    "AudienceRating": 52,
    "Budget": 117,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 91,
    "AudienceRating": 83,
    "Budget": 33,
    "Country": "Germany"
  },
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 50,
  "Budget": 50,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 26,
  "AudienceRating": 38,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 34,
  "AudienceRating": 46,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 49,
  "Budget": 19,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 41,
  "Budget": 58,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 19,
  "AudienceRating": 36,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 62,
  "Budget": 30,
  "Country": "UK"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 8,
  "AudienceRating": 55,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 28,
  "AudienceRating": 62,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 69,
  "AudienceRating": 73,
  "Budget": 27,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 48,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 40,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 79,
  "AudienceRating": 86,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 48,
  "AudienceRating": 55,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 18,
```

```
  "AudienceRating": 40,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 33,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 73,
  "AudienceRating": 66,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 29,
  "AudienceRating": 71,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 22,
  "AudienceRating": 51,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 61,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 76,
  "Budget": 66,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 86,
  "AudienceRating": 73,
```

```
  "Budget": 4,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 63,
  "AudienceRating": 77,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 46,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 36,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 19,
  "AudienceRating": 42,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
  "AudienceRating": 31,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 90,
  "AudienceRating": 72,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 93,
  "AudienceRating": 84,
  "Budget": 17,
  "Country": "Germany"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 34,
  "AudienceRating": 68,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 70,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 93,
  "AudienceRating": 86,
  "Budget": 145,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 95,
  "AudienceRating": 89,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 38,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 54,
  "AudienceRating": 54,
  "Budget": 40,
  "Country": "South Korea"
},
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 61,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 56,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 47,
  "AudienceRating": 54,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 63,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 51,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 43,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 73,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Horror",
```

```
  "RottenTomatoesRating": 9,
  "AudienceRating": 29,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 83,
  "AudienceRating": 84,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 49,
  "AudienceRating": 61,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 77,
  "AudienceRating": 54,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 31,
  "AudienceRating": 72,
  "Budget": 130,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 23,
  "AudienceRating": 37,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 70,
  "AudienceRating": 70,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 8,
```

```
  "AudienceRating": 48,
  "Budget": 56,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 29,
  "AudienceRating": 59,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 21,
  "AudienceRating": 27,
  "Budget": 3,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 73,
  "AudienceRating": 67,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 60,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 31,
  "AudienceRating": 56,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 51,
  "AudienceRating": 50,
  "Budget": 37,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 37,
  "AudienceRating": 40,
  "Budget": 80,
```

```
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 25,
    "AudienceRating": 57,
    "Budget": 50,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 95,
    "AudienceRating": 84,
    "Budget": 25,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 39,
    "AudienceRating": 64,
    "Budget": 28,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 49,
    "AudienceRating": 57,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 9,
    "AudienceRating": 56,
    "Budget": 60,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 34,
    "AudienceRating": 66,
    "Budget": 5,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 51,
    "AudienceRating": 40,
    "Budget": 18,
    "Country": "USA"
```

```
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 20,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 70,
  "AudienceRating": 74,
  "Budget": 85,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 5,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 37,
  "AudienceRating": 54,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 0,
  "AudienceRating": 36,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 55,
  "AudienceRating": 65,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 0,
  "Country": "USA"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 79,
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 47,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 21,
  "AudienceRating": 82,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 82,
  "AudienceRating": 56,
  "Budget": 1,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 0,
  "AudienceRating": 0,
  "Budget": 3,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 68,
  "AudienceRating": 58,
  "Budget": 5,
  "Country": "South Korea"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 11,
  "AudienceRating": 41,
  "Budget": 26,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 50,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 74,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 95,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 68,
  "AudienceRating": 74,
  "Budget": 26,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 74,
  "AudienceRating": 45,
  "Budget": 24,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 60,
  "AudienceRating": 72,
```

```
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 34,
  "AudienceRating": 61,
  "Budget": 250,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 45,
  "AudienceRating": 74,
  "Budget": 300,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 88,
  "AudienceRating": 62,
  "Budget": 3,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 7,
  "AudienceRating": 31,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 8,
  "AudienceRating": 55,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 17,
  "AudienceRating": 37,
  "Budget": 60,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 36,
    "AudienceRating": 71,
    "Budget": 200,
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 8,
    "AudienceRating": 46,
    "Budget": 18,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 27,
    "AudienceRating": 47,
    "Budget": 35,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 22,
    "AudienceRating": 47,
    "Budget": 38,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 64,
    "AudienceRating": 62,
    "Budget": 230,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 59,
    "AudienceRating": 46,
    "Budget": 12,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 42,
    "AudienceRating": 50,
    "Budget": 50,
    "Country": "South Korea"
  },
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 85,
  "AudienceRating": 61,
  "Budget": 12,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 70,
  "Budget": 48,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 81,
  "Budget": 110,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 72,
  "Budget": 58,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 11,
  "AudienceRating": 41,
  "Budget": 42,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 57,
  "AudienceRating": 59,
  "Budget": 4,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 28,
  "AudienceRating": 70,
  "Budget": 16,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 43,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 83,
  "AudienceRating": 87,
  "Budget": 93,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 59,
  "Budget": 200,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 77,
  "AudienceRating": 75,
  "Budget": 28,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 20,
  "AudienceRating": 68,
  "Budget": 140,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 62,
  "AudienceRating": 62,
  "Budget": 110,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 30,
```

```
  "AudienceRating": 39,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 18,
  "AudienceRating": 70,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 40,
  "AudienceRating": 67,
  "Budget": 11,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 81,
  "AudienceRating": 83,
  "Budget": 60,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 58,
  "AudienceRating": 57,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 10,
  "AudienceRating": 32,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 41,
  "Budget": 3,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 77,
```

```
  "Budget": 55,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 49,
  "AudienceRating": 81,
  "Budget": 58,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 49,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 46,
  "AudienceRating": 61,
  "Budget": 19,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 80,
  "AudienceRating": 80,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 16,
  "AudienceRating": 25,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 57,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 70,
  "AudienceRating": 81,
  "Budget": 90,
  "Country": "UK"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 60,
  "AudienceRating": 79,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 66,
  "Budget": 39,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 82,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 7,
  "AudienceRating": 40,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 68,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 19,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 50,
  "AudienceRating": 41,
  "Budget": 7,
  "Country": "UK"
},
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 81,
  "AudienceRating": 47,
  "Budget": 15,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 0,
  "AudienceRating": 0,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 22,
  "AudienceRating": 38,
  "Budget": 13,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 45,
  "AudienceRating": 42,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 79,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 92,
  "AudienceRating": 81,
  "Budget": 32,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 61,
  "Budget": 120,
  "Country": "South Korea"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 61,
  "AudienceRating": 54,
  "Budget": 258,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 40,
  "Budget": 27,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 94,
  "AudienceRating": 91,
  "Budget": 140,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 76,
  "AudienceRating": 86,
  "Budget": 70,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 54,
  "AudienceRating": 68,
  "Budget": 65,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 26,
  "AudienceRating": 81,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 67,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 49,
```

```
  "AudienceRating": 20,
  "Budget": 22,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 65,
  "AudienceRating": 27,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 28,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 37,
  "AudienceRating": 62,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 23,
  "AudienceRating": 48,
  "Budget": 82,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 59,
  "Budget": 5,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 82,
  "AudienceRating": 78,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 88,
  "AudienceRating": 87,
  "Budget": 20,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 15,
    "AudienceRating": 36,
    "Budget": 65,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 39,
    "AudienceRating": 42,
    "Budget": 80,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 36,
    "AudienceRating": 44,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 28,
    "AudienceRating": 46,
    "Budget": 25,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 92,
    "AudienceRating": 85,
    "Budget": 5,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 58,
    "AudienceRating": 83,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 27,
    "AudienceRating": 57,
    "Budget": 32,
    "Country": "USA"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 48,
  "AudienceRating": 48,
  "Budget": 29,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 64,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 71,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 72,
  "AudienceRating": 67,
  "Budget": 51,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 74,
  "AudienceRating": 78,
  "Budget": 130,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 97,
  "AudienceRating": 91,
  "Budget": 16,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 20,
  "AudienceRating": 47,
  "Budget": 35,
  "Country": "UK"
},
{
```

```
  "Genre": "Thriller",
  "RottenTomatoesRating": 78,
  "AudienceRating": 74,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 62,
  "AudienceRating": 57,
  "Budget": 21,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 62,
  "Budget": 41,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 48,
  "AudienceRating": 68,
  "Budget": 80,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 7,
  "AudienceRating": 41,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 93,
  "AudienceRating": 91,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 27,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Drama",
```

```
  "RottenTomatoesRating": 63,
  "AudienceRating": 84,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 65,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 81,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 52,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 78,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 49,
  "AudienceRating": 63,
  "Budget": 155,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 66,
  "AudienceRating": 55,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 56,
  "AudienceRating": 65,
```

```
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 72,
  "AudienceRating": 54,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 67,
  "AudienceRating": 79,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 94,
  "AudienceRating": 96,
  "Budget": 185,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 19,
  "AudienceRating": 34,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 76,
  "AudienceRating": 70,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 89,
  "AudienceRating": 88,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 23,
  "AudienceRating": 31,
  "Budget": 70,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 60,
    "AudienceRating": 68,
    "Budget": 14,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 39,
    "AudienceRating": 43,
    "Budget": 25,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 40,
    "AudienceRating": 58,
    "Budget": 82,
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 22,
    "AudienceRating": 51,
    "Budget": 12,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 91,
    "AudienceRating": 88,
    "Budget": 25,
    "Country": "South Korea"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 29,
    "AudienceRating": 52,
    "Budget": 40,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 64,
    "AudienceRating": 65,
    "Budget": 55,
    "Country": "UK"
  },
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
  "AudienceRating": 75,
  "Budget": 22,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 87,
  "AudienceRating": 89,
  "Budget": 90,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 55,
  "Budget": 180,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 44,
  "AudienceRating": 47,
  "Budget": 120,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 35,
  "AudienceRating": 58,
  "Budget": 80,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 19,
  "AudienceRating": 29,
  "Budget": 48,
  "Country": "Germany"
},
{
```

```
  "Genre": "Drama",
  "RottenTomatoesRating": 17,
  "AudienceRating": 51,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 3,
  "AudienceRating": 22,
  "Budget": 5,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 30,
  "AudienceRating": 41,
  "Budget": 60,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 75,
  "AudienceRating": 91,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 11,
  "AudienceRating": 50,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 54,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 97,
  "AudienceRating": 83,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 85,
```

```
  "AudienceRating": 76,
  "Budget": 13,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 138,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 42,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 39,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 47,
  "Budget": 19,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 54,
  "Budget": 13,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 61,
  "AudienceRating": 47,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 75,
```

```
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 94,
  "AudienceRating": 72,
  "Budget": 4,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 52,
  "AudienceRating": 78,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 66,
  "AudienceRating": 85,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 6,
  "AudienceRating": 42,
  "Budget": 150,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 73,
  "AudienceRating": 32,
  "Budget": 2,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 19,
  "AudienceRating": 66,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 84,
  "AudienceRating": 82,
  "Budget": 40,
  "Country": "South Korea"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 39,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 55,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 38,
  "Budget": 62,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 32,
  "AudienceRating": 57,
  "Budget": 65,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 43,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 72,
  "AudienceRating": 64,
  "Budget": 18,
  "Country": "USA"
},
```

```
{
  "Genre": "Action",
  "RottenTomatoesRating": 14,
  "AudienceRating": 40,
  "Budget": 145,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 97,
  "AudienceRating": 87,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 33,
  "AudienceRating": 52,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 65,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 57,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 12,
  "AudienceRating": 47,
  "Budget": 59,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 74,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Horror",
```

```
  "RottenTomatoesRating": 9,
  "AudienceRating": 53,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 20,
  "AudienceRating": 43,
  "Budget": 37,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 75,
  "AudienceRating": 65,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 4,
  "AudienceRating": 29,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 46,
  "AudienceRating": 32,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 50,
  "AudienceRating": 48,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 56,
  "AudienceRating": 78,
  "Budget": 11,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 90,
```

```
    "AudienceRating": 78,
    "Budget": 75,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 22,
    "AudienceRating": 43,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 56,
    "AudienceRating": 59,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 18,
    "AudienceRating": 61,
    "Budget": 15,
    "Country": "UK"
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Ratings'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Click **OK** to save the changes.

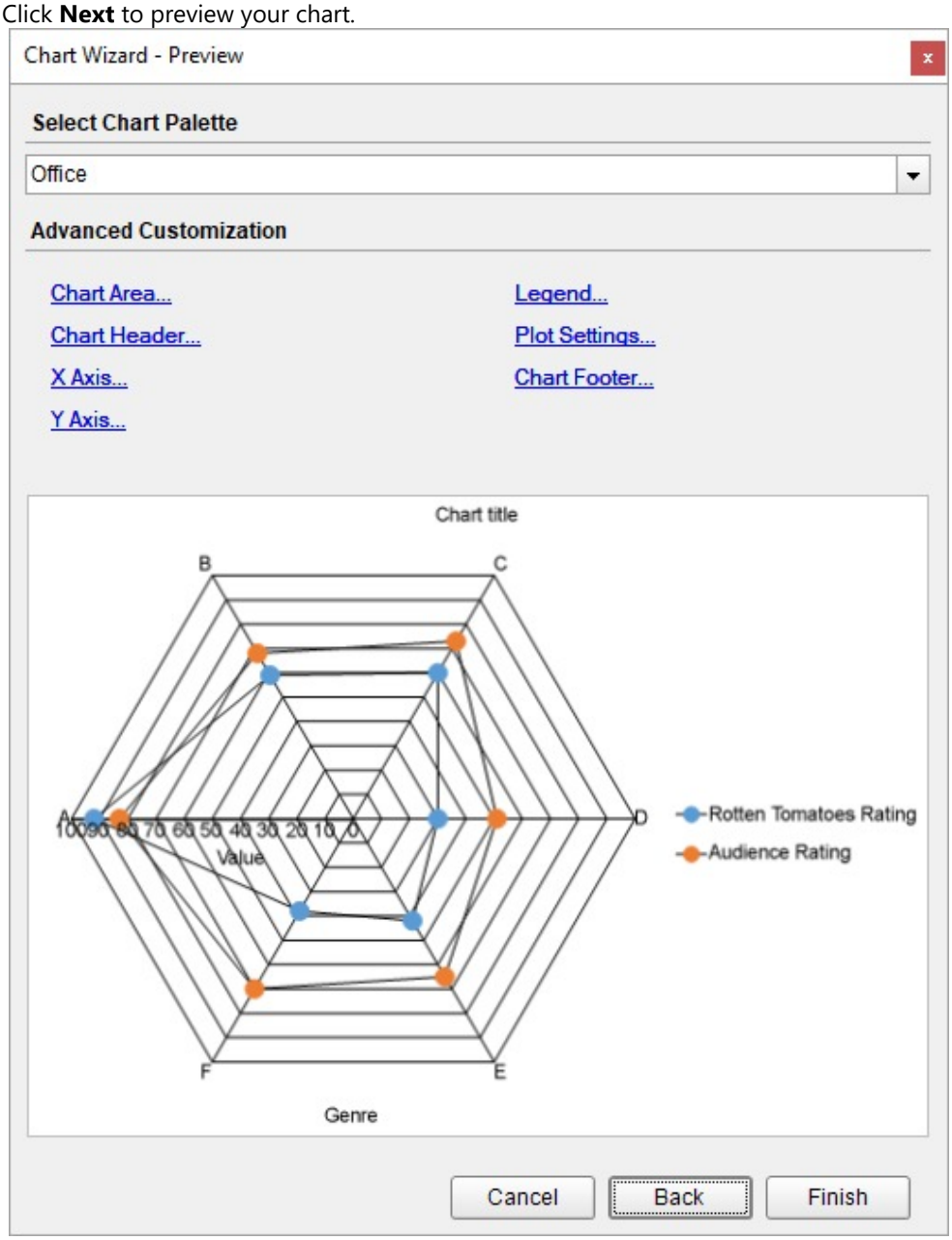
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Ratings' and the **Chart Type** as 'Radar Line'.
3. Click **Next** to proceed. Here, we will define two data values to display the average rotten tomatoes rating and audience rating in the chart.
4. Under **Choose Data Values**, add two data values, and set their properties as below.

Field	Aggregate
=[RottenTomatoesRating]	Average
=[AudienceRating]	Average

5. In **Choose Data Category**, set **Field** to `=[Genre]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Line Style > Style**: 1.5pt
    - **Symbol Settings > Shape**: Dot
3. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 10
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select **Custom** from the drop-down and add the following colors.
    - #8fcd37
    - #f26324
3. Click **OK** to complete setting up the custom palette.

Legend - Color

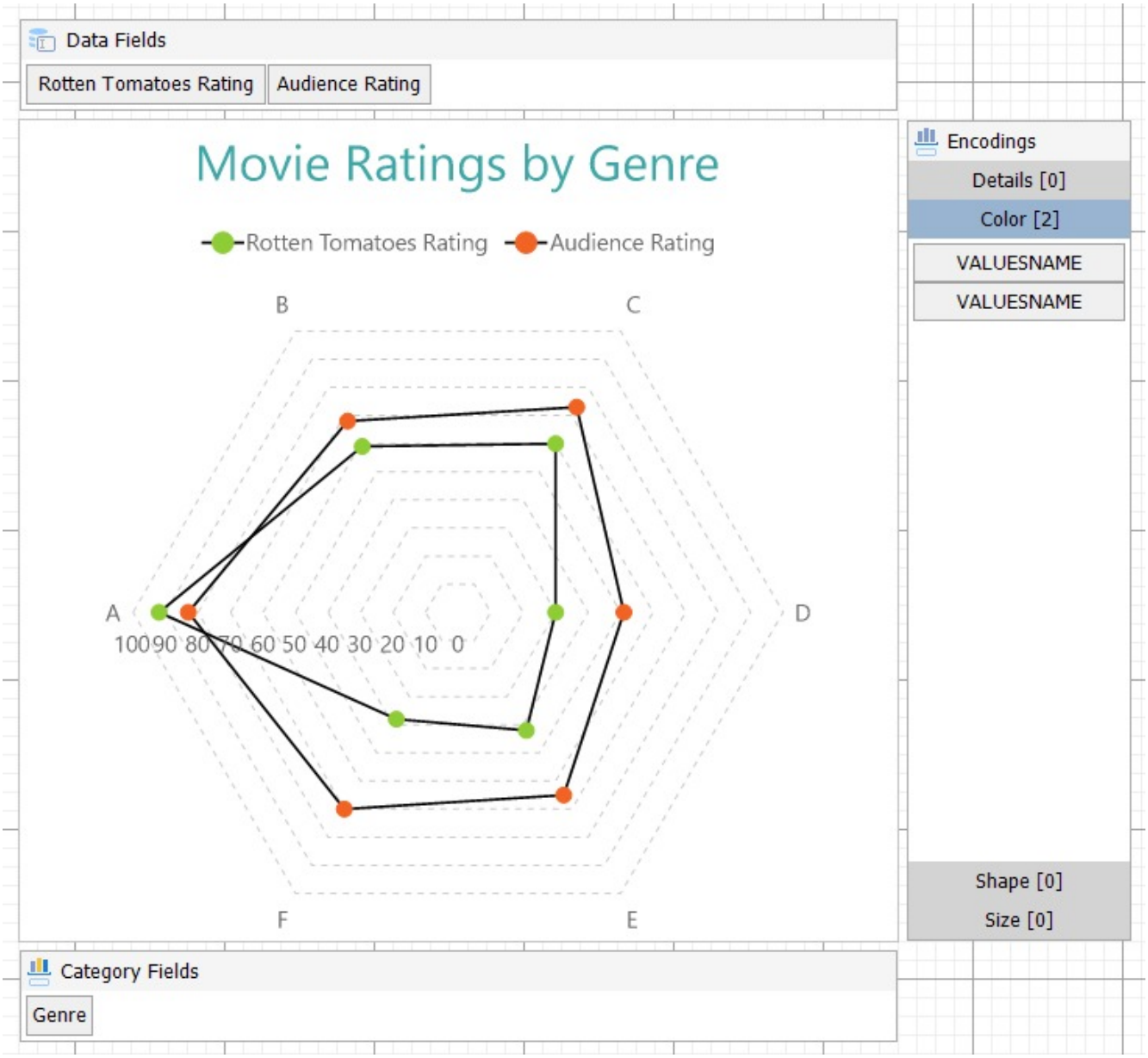
1. To open the smart panel for the legend, click the 'Legend - Color' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 12pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.

- **Position**: Top
- **Orientation**: Horizontal

4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Movie Ratings by Genre'.
3. Go to the **Font** page and set the properties as below.
   - **Size**: 24pt
   - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the

> chart on the preview.

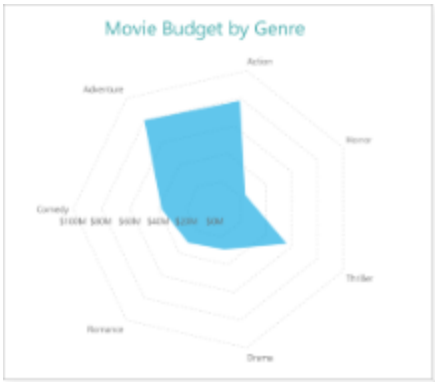
5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Radar Area Chart

The Radar area plot arranges categories on the circumference of a circle and connects the corresponding points with straight lines. The data values are plotted radially and are connected with straight lines resulting in a polygon shape. Radar Area plots are generally used to display ordinal variables across different category ranges.

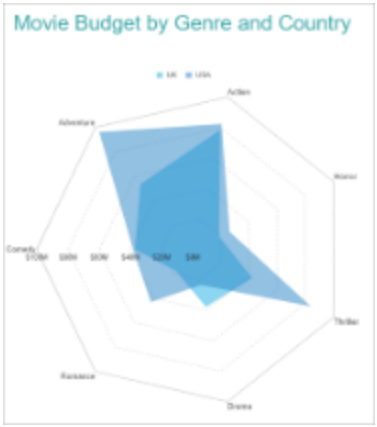
Radar Area Plot with Single Value

A single value radar area chart shows measurements of one variable. The Create Radar Area Chart with Single Value walkthrough showcases the average budget for different movie genres.



Radar Area Plot with Multiple Values

A multiple values radar chart lets you split data values into subcategories for finer analysis. The Create Radar Area Chart with Multiple Values walkthrough showcases the average movie budget in a country for different movie genres.



Radial Area Plot Properties

The plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart

Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the plot.
  - Center: Positions the data label text on the center of the chart.
  - Inside: Positions the data label text inside the chart.
  - Outside: Positions the data label text outside the chart.
  - Auto: The default setting, same as Outside for chart.

LineStyle

The line style for the borders. Includes LineColor, LineStyle, and LineWidth properties for the customization.

- **LineColor**: Specify the color of the border around the filled area.
- **LineStyle:** Specify the line style of the border around the filled area as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the filled area.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to:

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for area plots.

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default:** Indicates a straight line.
- **Spline:** Indicates a bezier curve.
- **Step Center, Step Left, and Step Right:** Indicates a stepped line with different step directions.

Opacity

The Opacity determines the opacity of areas filled with color. The Opacity is specified in percentage. 0% means that they are entirely transparent, and 100% means they are opaque.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

StartAngle

Indicates the arc angle that defines the clockwise rotation of the plot. A full rotation makes 360 degrees.

UnpivotData

Determines whether to display multiple data fields as a single data field. By default, the property is set to 'False'.

Encodings

Category Encoding

The Category Encoding of a plot is a set of properties that determine the period over which the plot generates connected data points representing those above Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Color Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Area plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the area subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked plot.
  - Cluster: You can use this value to configure area subsections that overlap each other.
  - None: Equals to Cluster.

- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Area plots, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Area plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

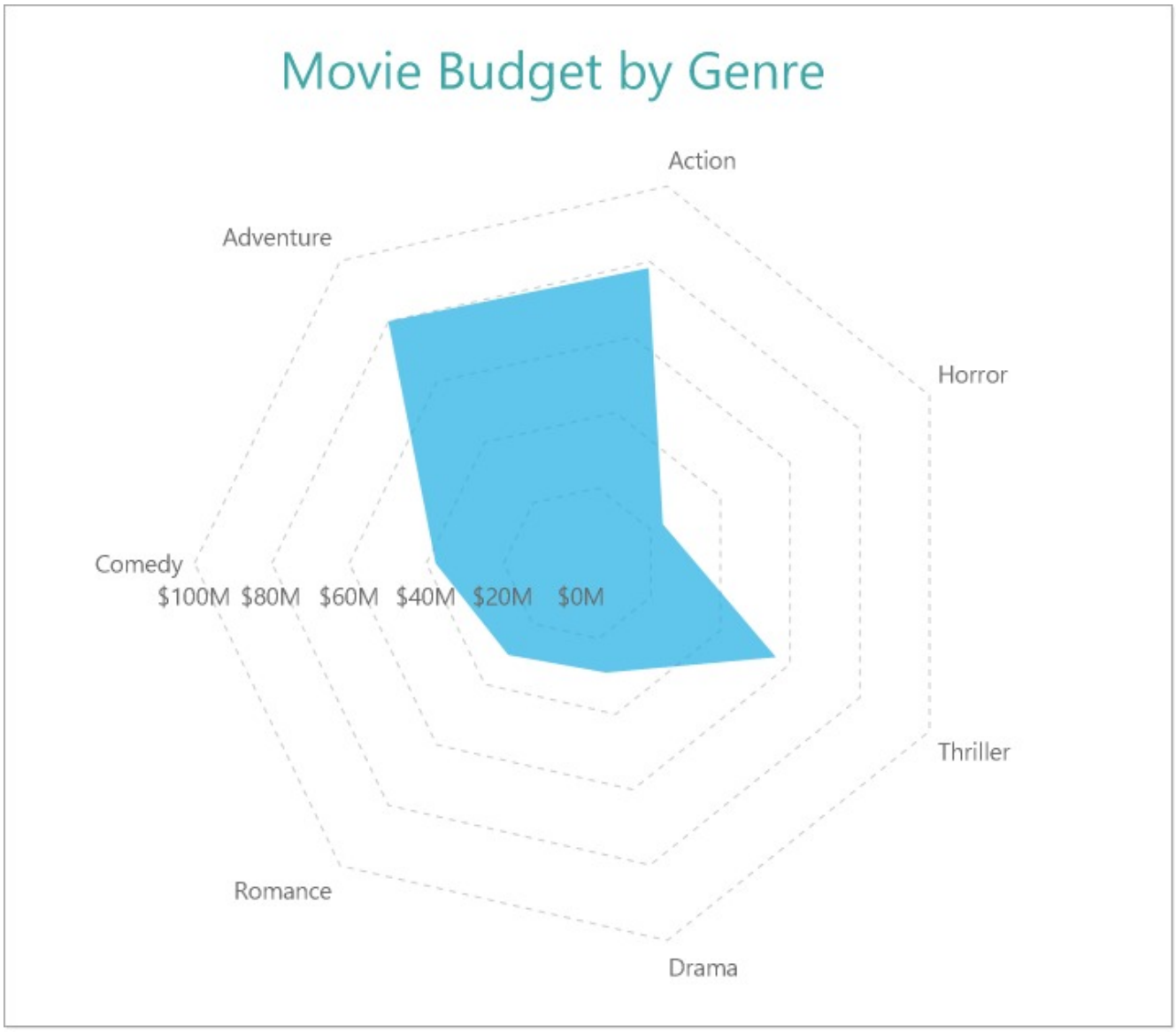
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Radar Area Chart with Single Value

This walkthrough creates a Radar Area Chart with Single Value. The chart shows the average budget for certain movie genres. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data

```
[
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 87,
  "AudienceRating": 81,
  "Budget": 8,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 9,
  "AudienceRating": 44,
  "Budget": 105,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 30,
  "AudienceRating": 52,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 93,
  "AudienceRating": 84,
  "Budget": 18,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 55,
  "AudienceRating": 70,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 39,
  "AudienceRating": 63,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 71,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Horror",
```

```
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 48,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 93,
  "AudienceRating": 93,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 5,
  "AudienceRating": 51,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 89,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 13,
  "AudienceRating": 40,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 89,
  "AudienceRating": 64,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
```

```
  "AudienceRating": 71,
  "Budget": 19,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 4,
  "AudienceRating": 46,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 54,
  "AudienceRating": 84,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 89,
  "AudienceRating": 56,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 43,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 72,
  "Budget": 200,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 37,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 30,
  "AudienceRating": 46,
  "Budget": 45,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 6,
    "AudienceRating": 35,
    "Budget": 15,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 33,
    "AudienceRating": 64,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 21,
    "AudienceRating": 35,
    "Budget": 40,
    "Country": "UK"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 79,
    "AudienceRating": 87,
    "Budget": 100,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 94,
    "AudienceRating": 78,
    "Budget": 8,
    "Country": "South Korea"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 46,
    "AudienceRating": 66,
    "Budget": 30,
    "Country": "USA"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 23,
    "AudienceRating": 31,
    "Budget": 5,
    "Country": "UK"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 55,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 34,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 55,
  "AudienceRating": 69,
  "Budget": 78,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 83,
  "AudienceRating": 92,
  "Budget": 237,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 67,
  "AudienceRating": 74,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 63,
  "AudienceRating": 59,
  "Budget": 70,
  "Country": "USA"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 7,
  "AudienceRating": 32,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 44,
  "AudienceRating": 38,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 79,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 9,
  "AudienceRating": 33,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 35,
  "AudienceRating": 50,
  "Budget": 70,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 65,
  "AudienceRating": 57,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 19,
  "AudienceRating": 50,
  "Budget": 17,
  "Country": "USA"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 25,
  "AudienceRating": 63,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 84,
  "AudienceRating": 80,
  "Budget": 4,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 52,
  "Budget": 150,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 41,
  "AudienceRating": 56,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 88,
  "AudienceRating": 86,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 69,
  "AudienceRating": 66,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 65,
  "Budget": 68,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 11,
  "AudienceRating": 56,
```

```
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 90,
  "AudienceRating": 77,
  "Budget": 33,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 47,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 61,
  "AudienceRating": 62,
  "Budget": 26,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 48,
  "Budget": 42,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 86,
  "AudienceRating": 63,
  "Budget": 2,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 36,
  "AudienceRating": 66,
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 64,
  "Budget": 37,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 78,
    "AudienceRating": 75,
    "Budget": 140,
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 23,
    "AudienceRating": 42,
    "Budget": 26,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 13,
    "AudienceRating": 43,
    "Budget": 85,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 86,
    "AudienceRating": 61,
    "Budget": 10,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 62,
    "AudienceRating": 84,
    "Budget": 55,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 38,
    "AudienceRating": 50,
    "Budget": 40,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 81,
    "AudienceRating": 77,
    "Budget": 6,
    "Country": "USA"
  },
```

```
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 48,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 48,
  "Budget": 125,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 67,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 13,
  "AudienceRating": 56,
  "Budget": 140,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 55,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 34,
  "Budget": 90,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 70,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Thriller",
  "RottenTomatoesRating": 84,
  "AudienceRating": 63,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 19,
  "AudienceRating": 45,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 20,
  "AudienceRating": 56,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 12,
  "AudienceRating": 47,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 44,
  "AudienceRating": 50,
  "Budget": 163,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 62,
  "AudienceRating": 54,
  "Budget": 13,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 78,
  "AudienceRating": 81,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 80,
```

```
  "AudienceRating": 51,
  "Budget": 7,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 1,
  "AudienceRating": 63,
  "Budget": 6,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 17,
  "AudienceRating": 35,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 66,
  "AudienceRating": 58,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 29,
  "AudienceRating": 66,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 48,
  "Budget": 21,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 69,
```

```
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 71,
  "AudienceRating": 72,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 52,
  "AudienceRating": 45,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 49,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 47,
  "AudienceRating": 63,
  "Budget": 21,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 12,
  "AudienceRating": 31,
  "Budget": 58,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 47,
  "Budget": 69,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
  "AudienceRating": 28,
  "Budget": 20,
  "Country": "UK"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 91,
  "AudienceRating": 81,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 84,
  "AudienceRating": 81,
  "Budget": 37,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 59,
  "AudienceRating": 37,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 48,
  "AudienceRating": 47,
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 78,
  "AudienceRating": 75,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 92,
  "AudienceRating": 61,
  "Budget": 30,
  "Country": "UK"
},
```

```
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 44,
  "Budget": 75,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 15,
  "AudienceRating": 28,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 7,
  "AudienceRating": 38,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 93,
  "AudienceRating": 79,
  "Budget": 15,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 45,
  "AudienceRating": 38,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 56,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
```

```
  "RottenTomatoesRating": 64,
  "AudienceRating": 38,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 6,
  "AudienceRating": 28,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 72,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 75,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 36,
  "AudienceRating": 46,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 93,
  "AudienceRating": 80,
  "Budget": 85,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
  "AudienceRating": 38,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 23,
```

```
  "AudienceRating": 56,
  "Budget": 175,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 55,
  "Budget": 21,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 74,
  "AudienceRating": 53,
  "Budget": 5,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 62,
  "AudienceRating": 37,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 55,
  "Budget": 31,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 62,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 25,
  "AudienceRating": 43,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 35,
  "AudienceRating": 55,
  "Budget": 130,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 28,
    "AudienceRating": 76,
    "Budget": 85,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 78,
    "AudienceRating": 83,
    "Budget": 125,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 41,
    "AudienceRating": 50,
    "Budget": 24,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 61,
    "AudienceRating": 56,
    "Budget": 40,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 22,
    "AudienceRating": 52,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 40,
    "AudienceRating": 51,
    "Budget": 1,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 13,
    "AudienceRating": 59,
    "Budget": 13,
    "Country": "USA"
```

```
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 11,
  "AudienceRating": 53,
  "Budget": 73,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 71,
  "AudienceRating": 71,
  "Budget": 24,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 73,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 75,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 52,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 25,
  "AudienceRating": 52,
  "Budget": 19,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 71,
  "AudienceRating": 68,
  "Budget": 35,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Horror",
  "RottenTomatoesRating": 75,
  "AudienceRating": 68,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 57,
  "Budget": 52,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 58,
  "Budget": 75,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 8,
  "AudienceRating": 36,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 50,
  "Budget": 150,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 62,
  "Budget": 175,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 29,
  "AudienceRating": 43,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 73,
  "AudienceRating": 63,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 51,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 56,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
  "AudienceRating": 47,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 56,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 3,
  "AudienceRating": 61,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 80,
  "AudienceRating": 90,
```

```
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 48,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 60,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 75,
  "AudienceRating": 40,
  "Budget": 0,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 82,
  "AudienceRating": 86,
  "Budget": 53,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 10,
  "AudienceRating": 59,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 31,
  "Budget": 112,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 58,
  "AudienceRating": 85,
  "Budget": 16,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 35,
    "AudienceRating": 44,
    "Budget": 36,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 26,
    "AudienceRating": 64,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 20,
    "AudienceRating": 45,
    "Budget": 15,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 40,
    "AudienceRating": 67,
    "Budget": 150,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 71,
    "AudienceRating": 67,
    "Budget": 30,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 44,
    "AudienceRating": 66,
    "Budget": 35,
    "Country": "South Korea"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 54,
    "AudienceRating": 65,
    "Budget": 12,
    "Country": "South Korea"
  },
```

```
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 83,
  "AudienceRating": 75,
  "Budget": 250,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 78,
  "AudienceRating": 82,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 42,
  "AudienceRating": 60,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 87,
  "AudienceRating": 70,
  "Budget": 83,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 54,
  "AudienceRating": 68,
  "Budget": 7,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 65,
  "AudienceRating": 76,
  "Budget": 11,
  "Country": "UK"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 15,
  "AudienceRating": 61,
  "Budget": 18,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 69,
  "AudienceRating": 72,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 44,
  "AudienceRating": 45,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 64,
  "AudienceRating": 57,
  "Budget": 36,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 46,
  "AudienceRating": 57,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 93,
  "AudienceRating": 84,
  "Budget": 150,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 69,
  "AudienceRating": 69,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 32,
```

```
  "AudienceRating": 57,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 38,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 71,
  "AudienceRating": 57,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 83,
  "AudienceRating": 72,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 73,
  "Budget": 85,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 38,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 44,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 59,
```

```
  "Budget": 75,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 72,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 38,
  "AudienceRating": 55,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 86,
  "AudienceRating": 93,
  "Budget": 160,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 59,
  "Budget": 185,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 88,
  "AudienceRating": 87,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 39,
  "AudienceRating": 51,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 67,
  "AudienceRating": 65,
  "Budget": 2,
  "Country": "Germany"
```

```
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 82,
  "AudienceRating": 90,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 94,
  "AudienceRating": 91,
  "Budget": 186,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 74,
  "AudienceRating": 80,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 63,
  "Budget": 85,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 42,
  "AudienceRating": 84,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 4,
  "AudienceRating": 59,
  "Budget": 79,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 85,
  "AudienceRating": 77,
  "Budget": 0,
  "Country": "USA"
},
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 42,
  "AudienceRating": 37,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 55,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 13,
  "AudienceRating": 24,
  "Budget": 47,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 61,
  "AudienceRating": 55,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 50,
  "Budget": 83,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 94,
  "AudienceRating": 89,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 19,
  "AudienceRating": 63,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 45,
  "AudienceRating": 58,
  "Budget": 12,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 76,
  "AudienceRating": 83,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 25,
  "AudienceRating": 48,
  "Budget": 70,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 11,
  "AudienceRating": 45,
  "Budget": 75,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 78,
  "AudienceRating": 26,
  "Budget": 10,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 52,
  "Budget": 117,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 91,
  "AudienceRating": 83,
  "Budget": 33,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
```

```
  "AudienceRating": 50,
  "Budget": 50,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 26,
  "AudienceRating": 38,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 34,
  "AudienceRating": 46,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 49,
  "Budget": 19,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 41,
  "Budget": 58,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 19,
  "AudienceRating": 36,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 62,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 8,
  "AudienceRating": 55,
  "Budget": 35,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 28,
    "AudienceRating": 62,
    "Budget": 38,
    "Country": "South Korea"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 69,
    "AudienceRating": 73,
    "Budget": 27,
    "Country": "South Korea"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 27,
    "AudienceRating": 48,
    "Budget": 35,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 9,
    "AudienceRating": 40,
    "Budget": 100,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 79,
    "AudienceRating": 86,
    "Budget": 110,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 48,
    "AudienceRating": 55,
    "Budget": 30,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 18,
    "AudienceRating": 40,
    "Budget": 18,
    "Country": "South Korea"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 33,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 73,
  "AudienceRating": 66,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 29,
  "AudienceRating": 71,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 22,
  "AudienceRating": 51,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 61,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 76,
  "Budget": 66,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 86,
  "AudienceRating": 73,
  "Budget": 4,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 63,
  "AudienceRating": 77,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 46,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 36,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 19,
  "AudienceRating": 42,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
  "AudienceRating": 31,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 90,
  "AudienceRating": 72,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 93,
  "AudienceRating": 84,
  "Budget": 17,
  "Country": "Germany"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 34,
  "AudienceRating": 68,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 70,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 93,
  "AudienceRating": 86,
  "Budget": 145,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 95,
  "AudienceRating": 89,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 38,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 54,
  "AudienceRating": 54,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 61,
```

```
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 56,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 47,
  "AudienceRating": 54,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 63,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 51,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 43,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 73,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 9,
  "AudienceRating": 29,
  "Budget": 25,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 83,
    "AudienceRating": 84,
    "Budget": 10,
    "Country": "USA"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 49,
    "AudienceRating": 61,
    "Budget": 20,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 77,
    "AudienceRating": 54,
    "Budget": 35,
    "Country": "USA"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 31,
    "AudienceRating": 72,
    "Budget": 130,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 23,
    "AudienceRating": 37,
    "Budget": 21,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 70,
    "AudienceRating": 70,
    "Budget": 15,
    "Country": "UK"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 8,
    "AudienceRating": 48,
    "Budget": 56,
    "Country": "Germany"
  },
```

```
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 29,
  "AudienceRating": 59,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 21,
  "AudienceRating": 27,
  "Budget": 3,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 73,
  "AudienceRating": 67,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 60,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 31,
  "AudienceRating": 56,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 51,
  "AudienceRating": 50,
  "Budget": 37,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 37,
  "AudienceRating": 40,
  "Budget": 80,
  "Country": "Germany"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 25,
  "AudienceRating": 57,
  "Budget": 50,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 95,
  "AudienceRating": 84,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 64,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 49,
  "AudienceRating": 57,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 56,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 34,
  "AudienceRating": 66,
  "Budget": 5,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 51,
  "AudienceRating": 40,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 20,
```

```
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 70,
  "AudienceRating": 74,
  "Budget": 85,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 5,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 37,
  "AudienceRating": 54,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 0,
  "AudienceRating": 36,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 55,
  "AudienceRating": 65,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 0,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 79,
```

```
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 47,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 21,
  "AudienceRating": 82,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 82,
  "AudienceRating": 56,
  "Budget": 1,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 0,
  "AudienceRating": 0,
  "Budget": 3,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 68,
  "AudienceRating": 58,
  "Budget": 5,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 11,
  "AudienceRating": 41,
  "Budget": 26,
  "Country": "UK"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 50,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 74,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 95,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 68,
  "AudienceRating": 74,
  "Budget": 26,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 74,
  "AudienceRating": 45,
  "Budget": 24,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 60,
  "AudienceRating": 72,
  "Budget": 50,
  "Country": "Germany"
},
```

```
{
  "Genre": "Action",
  "RottenTomatoesRating": 34,
  "AudienceRating": 61,
  "Budget": 250,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 45,
  "AudienceRating": 74,
  "Budget": 300,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 88,
  "AudienceRating": 62,
  "Budget": 3,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 7,
  "AudienceRating": 31,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 8,
  "AudienceRating": 55,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 17,
  "AudienceRating": 37,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 36,
  "AudienceRating": 71,
  "Budget": 200,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 8,
  "AudienceRating": 46,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 47,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 47,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 62,
  "Budget": 230,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 59,
  "AudienceRating": 46,
  "Budget": 12,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 50,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 85,
```

```
  "AudienceRating": 61,
  "Budget": 12,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 70,
  "Budget": 48,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 81,
  "Budget": 110,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 72,
  "Budget": 58,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 11,
  "AudienceRating": 41,
  "Budget": 42,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 57,
  "AudienceRating": 59,
  "Budget": 4,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 28,
  "AudienceRating": 70,
  "Budget": 16,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 43,
  "Budget": 32,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 24,
    "AudienceRating": 53,
    "Budget": 60,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 83,
    "AudienceRating": 87,
    "Budget": 93,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 42,
    "AudienceRating": 59,
    "Budget": 200,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 77,
    "AudienceRating": 75,
    "Budget": 28,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 20,
    "AudienceRating": 68,
    "Budget": 140,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 62,
    "AudienceRating": 62,
    "Budget": 110,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 30,
    "AudienceRating": 39,
    "Budget": 30,
    "Country": "Germany"
```

```
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 18,
  "AudienceRating": 70,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 40,
  "AudienceRating": 67,
  "Budget": 11,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 81,
  "AudienceRating": 83,
  "Budget": 60,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 58,
  "AudienceRating": 57,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 10,
  "AudienceRating": 32,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 41,
  "Budget": 3,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 77,
  "Budget": 55,
  "Country": "Germany"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 49,
  "AudienceRating": 81,
  "Budget": 58,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 49,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 46,
  "AudienceRating": 61,
  "Budget": 19,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 80,
  "AudienceRating": 80,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 16,
  "AudienceRating": 25,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 57,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 70,
  "AudienceRating": 81,
  "Budget": 90,
  "Country": "UK"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 60,
  "AudienceRating": 79,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 66,
  "Budget": 39,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 82,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 7,
  "AudienceRating": 40,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 68,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 19,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 50,
  "AudienceRating": 41,
  "Budget": 7,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 81,
  "AudienceRating": 47,
```

```
  "Budget": 15,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 0,
  "AudienceRating": 0,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 22,
  "AudienceRating": 38,
  "Budget": 13,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 45,
  "AudienceRating": 42,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 79,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 92,
  "AudienceRating": 81,
  "Budget": 32,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 61,
  "Budget": 120,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 61,
  "AudienceRating": 54,
  "Budget": 258,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 22,
    "AudienceRating": 40,
    "Budget": 27,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 94,
    "AudienceRating": 91,
    "Budget": 140,
    "Country": "UK"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 76,
    "AudienceRating": 86,
    "Budget": 70,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 54,
    "AudienceRating": 68,
    "Budget": 65,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 26,
    "AudienceRating": 81,
    "Budget": 18,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 46,
    "AudienceRating": 67,
    "Budget": 30,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 49,
    "AudienceRating": 20,
    "Budget": 22,
    "Country": "South Korea"
  },
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 65,
  "AudienceRating": 27,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 28,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 37,
  "AudienceRating": 62,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 23,
  "AudienceRating": 48,
  "Budget": 82,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 59,
  "Budget": 5,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 82,
  "AudienceRating": 78,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 88,
  "AudienceRating": 87,
  "Budget": 20,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 36,
  "Budget": 65,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 39,
  "AudienceRating": 42,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 36,
  "AudienceRating": 44,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 28,
  "AudienceRating": 46,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 92,
  "AudienceRating": 85,
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 83,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 57,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 48,
```

```
  "AudienceRating": 48,
  "Budget": 29,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 64,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 71,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 72,
  "AudienceRating": 67,
  "Budget": 51,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 74,
  "AudienceRating": 78,
  "Budget": 130,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 97,
  "AudienceRating": 91,
  "Budget": 16,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 20,
  "AudienceRating": 47,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 78,
  "AudienceRating": 74,
```

```
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 62,
  "AudienceRating": 57,
  "Budget": 21,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 62,
  "Budget": 41,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 48,
  "AudienceRating": 68,
  "Budget": 80,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 7,
  "AudienceRating": 41,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 93,
  "AudienceRating": 91,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 27,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 63,
  "AudienceRating": 84,
  "Budget": 13,
  "Country": "South Korea"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 65,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 81,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 52,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 78,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 49,
  "AudienceRating": 63,
  "Budget": 155,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 66,
  "AudienceRating": 55,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 56,
  "AudienceRating": 65,
  "Budget": 25,
  "Country": "South Korea"
},
```

```
{
  "Genre": "Horror",
  "RottenTomatoesRating": 72,
  "AudienceRating": 54,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 67,
  "AudienceRating": 79,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 94,
  "AudienceRating": 96,
  "Budget": 185,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 19,
  "AudienceRating": 34,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 76,
  "AudienceRating": 70,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 89,
  "AudienceRating": 88,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 23,
  "AudienceRating": 31,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Drama",
```

```
  "RottenTomatoesRating": 60,
  "AudienceRating": 68,
  "Budget": 14,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 39,
  "AudienceRating": 43,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 58,
  "Budget": 82,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 22,
  "AudienceRating": 51,
  "Budget": 12,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 91,
  "AudienceRating": 88,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 29,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 65,
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
```

```
  "AudienceRating": 75,
  "Budget": 22,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 87,
  "AudienceRating": 89,
  "Budget": 90,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 55,
  "Budget": 180,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 44,
  "AudienceRating": 47,
  "Budget": 120,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 35,
  "AudienceRating": 58,
  "Budget": 80,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 19,
  "AudienceRating": 29,
  "Budget": 48,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 17,
  "AudienceRating": 51,
  "Budget": 30,
```

```
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 3,
    "AudienceRating": 22,
    "Budget": 5,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 30,
    "AudienceRating": 41,
    "Budget": 60,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 75,
    "AudienceRating": 91,
    "Budget": 25,
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 11,
    "AudienceRating": 50,
    "Budget": 15,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 39,
    "AudienceRating": 54,
    "Budget": 25,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 97,
    "AudienceRating": 83,
    "Budget": 15,
    "Country": "USA"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 85,
    "AudienceRating": 76,
    "Budget": 13,
    "Country": "UK"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 138,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 42,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 39,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 47,
  "Budget": 19,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 54,
  "Budget": 13,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 61,
  "AudienceRating": 47,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 75,
  "Budget": 40,
  "Country": "USA"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 94,
  "AudienceRating": 72,
  "Budget": 4,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 52,
  "AudienceRating": 78,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 66,
  "AudienceRating": 85,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 6,
  "AudienceRating": 42,
  "Budget": 150,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 73,
  "AudienceRating": 32,
  "Budget": 2,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 19,
  "AudienceRating": 66,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 84,
  "AudienceRating": 82,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
```

```
    "RottenTomatoesRating": 39,
    "AudienceRating": 39,
    "Budget": 30,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 47,
    "AudienceRating": 55,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 14,
    "AudienceRating": 38,
    "Budget": 62,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 32,
    "AudienceRating": 57,
    "Budget": 65,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 53,
    "AudienceRating": 52,
    "Budget": 40,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 52,
    "AudienceRating": 43,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 72,
    "AudienceRating": 64,
    "Budget": 18,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 14,
    "AudienceRating": 40,
```

```
  "Budget": 145,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 97,
  "AudienceRating": 87,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 33,
  "AudienceRating": 52,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 65,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 57,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 12,
  "AudienceRating": 47,
  "Budget": 59,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 74,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 9,
  "AudienceRating": 53,
  "Budget": 40,
```

```
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 20,
    "AudienceRating": 43,
    "Budget": 37,
    "Country": "UK"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 75,
    "AudienceRating": 65,
    "Budget": 25,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 4,
    "AudienceRating": 29,
    "Budget": 16,
    "Country": "USA"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 46,
    "AudienceRating": 32,
    "Budget": 25,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 50,
    "AudienceRating": 48,
    "Budget": 45,
    "Country": "South Korea"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 56,
    "AudienceRating": 78,
    "Budget": 11,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 90,
    "AudienceRating": 78,
    "Budget": 75,
    "Country": "USA"
  },
```

```
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 22,
    "AudienceRating": 43,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 56,
    "AudienceRating": 59,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 18,
    "AudienceRating": 61,
    "Budget": 15,
    "Country": "UK"
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Ratings'.
2. Go to the **Query** page and enter the following query to fetch the required fields.

Query
`$.[*]`

3. Go to the **Filters** page, add a new filter value, and set its properties as below.

Expression	Operator	Values
`=[Country]`	In	`="UK"` `="USA"`

4. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Ratings' and the **Chart Type** as 'Radar Area'.
3. Click **Next** to proceed. Here, we will define a data series value to display the average movie budget in the chart.

4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[Budget]`	Average

5. In **Choose Data Category**, set **Field** to `=[Genre]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.

You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** page and set the **Format** to '$'0'M'.
4. Navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 20
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Click **OK** to complete setting up the Y-axis.

X-Axis

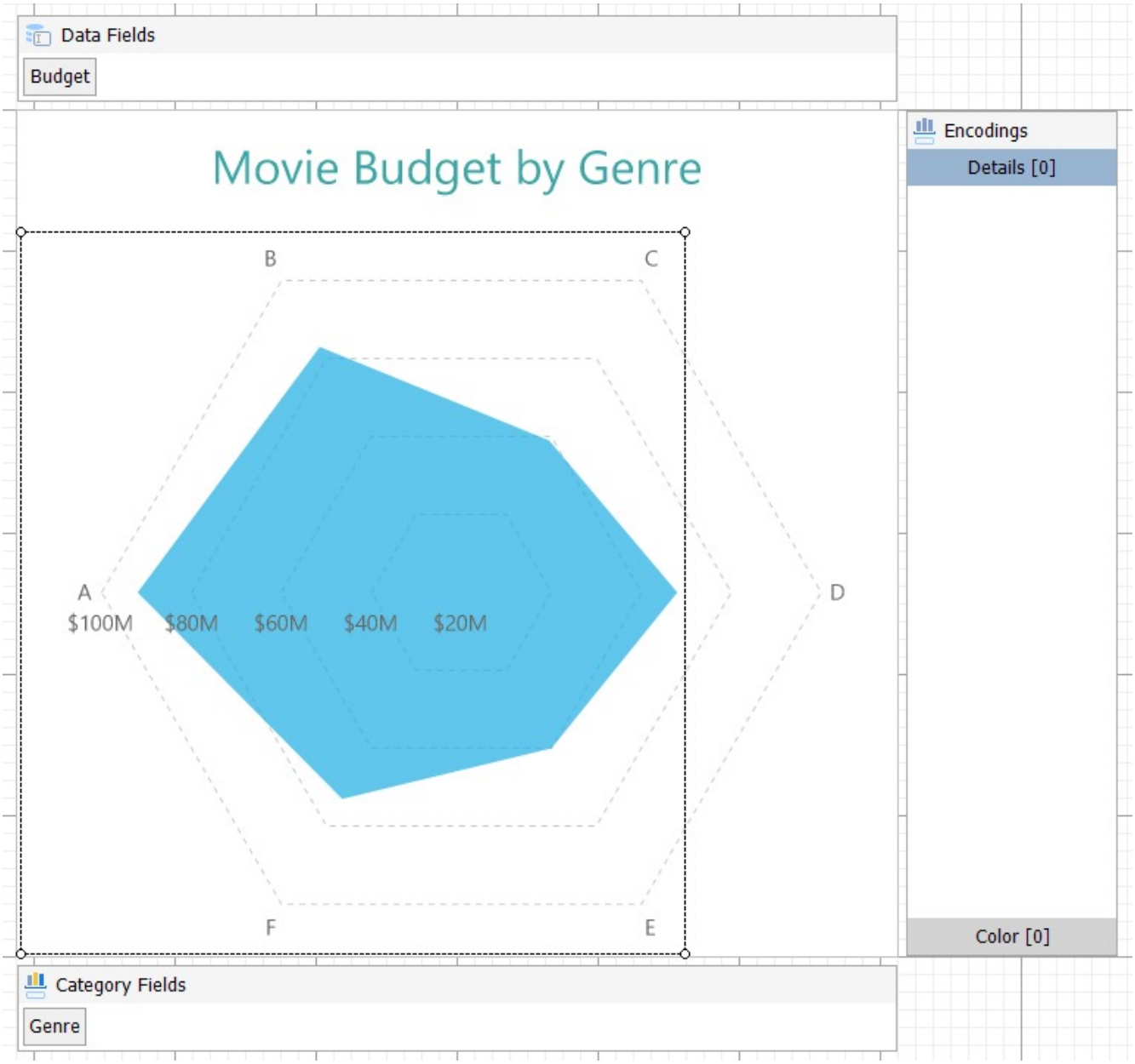
1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select 'Blue2' from the drop-down.
3. Click **OK** to complete setting up the chart palette.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Movie Budget by Genre'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

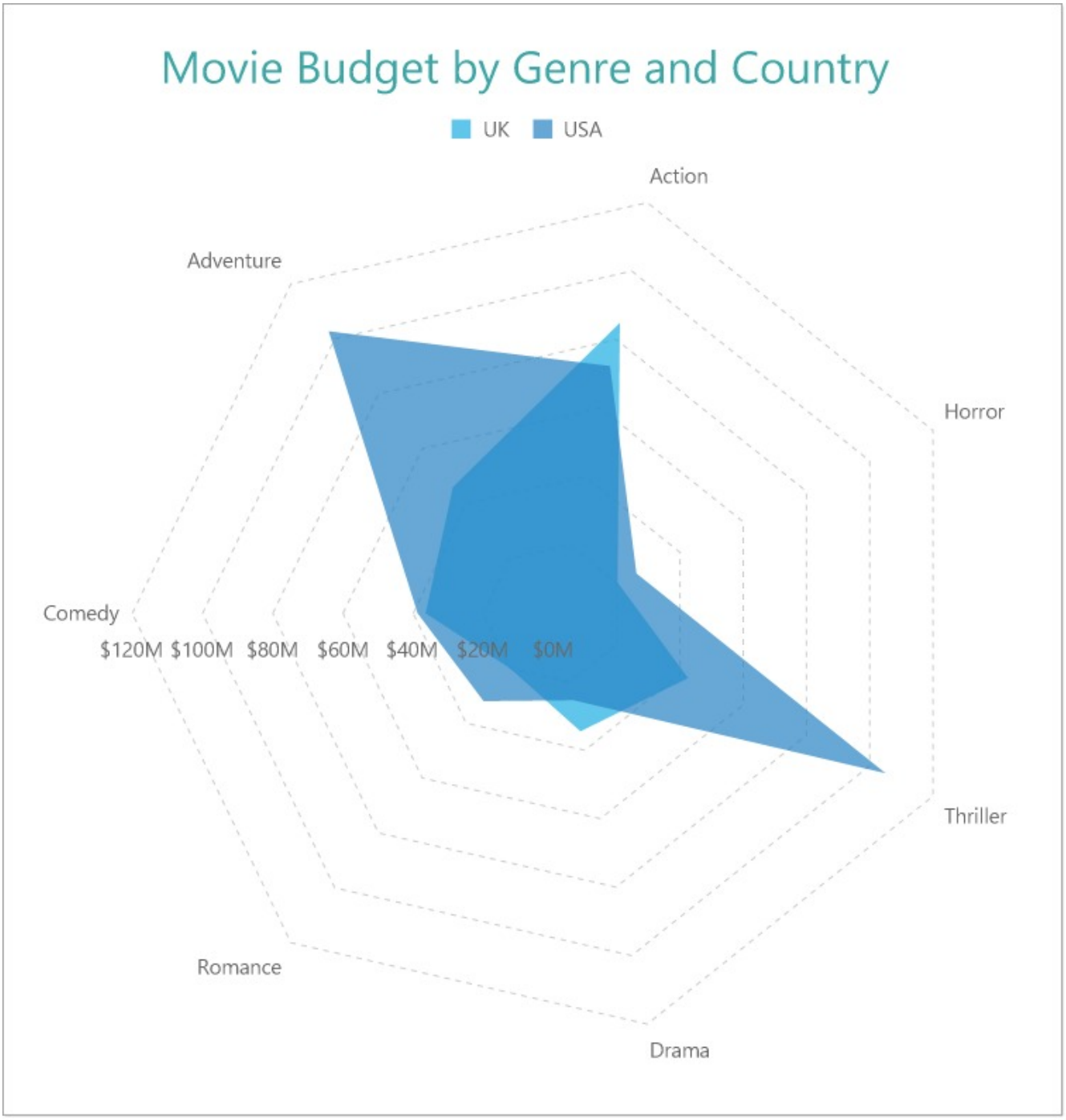


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Radar Area Chart with Multiple Values

This walkthrough creates a Radar Area Chart with Multiple Values. The chart shows the average budget for certain movie genres in different countries. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.

3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 87,
    "AudienceRating": 81,
    "Budget": 8,
    "Country": "UK"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 9,
    "AudienceRating": 44,
    "Budget": 105,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 30,
    "AudienceRating": 52,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 93,
    "AudienceRating": 84,
    "Budget": 18,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 55,
    "AudienceRating": 70,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 39,
    "AudienceRating": 63,
    "Budget": 200,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 40,
  "AudienceRating": 71,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 48,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 93,
  "AudienceRating": 93,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 5,
  "AudienceRating": 51,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 89,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 13,
  "AudienceRating": 40,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 89,
  "AudienceRating": 64,
```

```
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 71,
  "Budget": 19,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 4,
  "AudienceRating": 46,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 54,
  "AudienceRating": 84,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 89,
  "AudienceRating": 56,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 43,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 72,
  "Budget": 200,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 37,
  "Budget": 40,
```

```
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 30,
  "AudienceRating": 46,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 6,
  "AudienceRating": 35,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 64,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 21,
  "AudienceRating": 35,
  "Budget": 40,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 94,
  "AudienceRating": 78,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 66,
  "Budget": 30,
  "Country": "USA"
},
```

```
{
  "Genre": "Horror",
  "RottenTomatoesRating": 23,
  "AudienceRating": 31,
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 55,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 34,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 55,
  "AudienceRating": 69,
  "Budget": 78,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 83,
  "AudienceRating": 92,
  "Budget": 237,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 67,
  "AudienceRating": 74,
  "Budget": 21,
  "Country": "USA"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 63,
  "AudienceRating": 59,
  "Budget": 70,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 7,
  "AudienceRating": 32,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 44,
  "AudienceRating": 38,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 79,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 9,
  "AudienceRating": 33,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 35,
  "AudienceRating": 50,
  "Budget": 70,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 65,
  "AudienceRating": 57,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 19,
```

```
  "AudienceRating": 50,
  "Budget": 17,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 25,
  "AudienceRating": 63,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 84,
  "AudienceRating": 80,
  "Budget": 4,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 52,
  "Budget": 150,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 41,
  "AudienceRating": 56,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 88,
  "AudienceRating": 86,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 69,
  "AudienceRating": 66,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 65,
```

```
  "Budget": 68,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 11,
  "AudienceRating": 56,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 90,
  "AudienceRating": 77,
  "Budget": 33,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 47,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 61,
  "AudienceRating": 62,
  "Budget": 26,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 48,
  "Budget": 42,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 86,
  "AudienceRating": 63,
  "Budget": 2,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 36,
  "AudienceRating": 66,
  "Budget": 55,
  "Country": "UK"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 64,
  "Budget": 37,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 75,
  "Budget": 140,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 23,
  "AudienceRating": 42,
  "Budget": 26,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 13,
  "AudienceRating": 43,
  "Budget": 85,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 86,
  "AudienceRating": 61,
  "Budget": 10,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 62,
  "AudienceRating": 84,
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 50,
  "Budget": 40,
  "Country": "Germany"
},
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 81,
  "AudienceRating": 77,
  "Budget": 6,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 52,
  "AudienceRating": 48,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 48,
  "Budget": 125,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 77,
  "AudienceRating": 67,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 13,
  "AudienceRating": 56,
  "Budget": 140,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 55,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 34,
  "Budget": 90,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 70,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 84,
  "AudienceRating": 63,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 19,
  "AudienceRating": 45,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 20,
  "AudienceRating": 56,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 12,
  "AudienceRating": 47,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 44,
  "AudienceRating": 50,
  "Budget": 163,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 62,
  "AudienceRating": 54,
  "Budget": 13,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 78,
```

```
  "AudienceRating": 81,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 80,
  "AudienceRating": 51,
  "Budget": 7,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 1,
  "AudienceRating": 63,
  "Budget": 6,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 17,
  "AudienceRating": 35,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 66,
  "AudienceRating": 58,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 29,
  "AudienceRating": 66,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 48,
  "Budget": 21,
```

```
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 42,
    "AudienceRating": 69,
    "Budget": 45,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 71,
    "AudienceRating": 72,
    "Budget": 7,
    "Country": "South Korea"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 52,
    "AudienceRating": 45,
    "Budget": 10,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 53,
    "AudienceRating": 49,
    "Budget": 15,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 47,
    "AudienceRating": 63,
    "Budget": 21,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 12,
    "AudienceRating": 31,
    "Budget": 58,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 43,
    "AudienceRating": 47,
    "Budget": 69,
    "Country": "UK"
```

```
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
  "AudienceRating": 28,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 91,
  "AudienceRating": 81,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 84,
  "AudienceRating": 81,
  "Budget": 37,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 59,
  "AudienceRating": 37,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 48,
  "AudienceRating": 47,
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 78,
  "AudienceRating": 75,
  "Budget": 25,
  "Country": "USA"
},
{
```

```
  "Genre": "Horror",
  "RottenTomatoesRating": 92,
  "AudienceRating": 61,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 44,
  "Budget": 75,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 15,
  "AudienceRating": 28,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 7,
  "AudienceRating": 38,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 49,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 93,
  "AudienceRating": 79,
  "Budget": 15,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 45,
  "AudienceRating": 38,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
```

```
    "RottenTomatoesRating": 40,
    "AudienceRating": 56,
    "Budget": 50,
    "Country": "South Korea"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 64,
    "AudienceRating": 38,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 6,
    "AudienceRating": 28,
    "Budget": 20,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 27,
    "AudienceRating": 72,
    "Budget": 80,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 85,
    "AudienceRating": 75,
    "Budget": 8,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 36,
    "AudienceRating": 46,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 93,
    "AudienceRating": 80,
    "Budget": 85,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 2,
    "AudienceRating": 38,
```

```
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 23,
  "AudienceRating": 56,
  "Budget": 175,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 55,
  "Budget": 21,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 74,
  "AudienceRating": 53,
  "Budget": 5,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 62,
  "AudienceRating": 37,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 55,
  "Budget": 31,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 62,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 25,
  "AudienceRating": 43,
  "Budget": 18,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 35,
    "AudienceRating": 55,
    "Budget": 130,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 28,
    "AudienceRating": 76,
    "Budget": 85,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 78,
    "AudienceRating": 83,
    "Budget": 125,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 41,
    "AudienceRating": 50,
    "Budget": 24,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 61,
    "AudienceRating": 56,
    "Budget": 40,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 22,
    "AudienceRating": 52,
    "Budget": 20,
    "Country": "Germany"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 40,
    "AudienceRating": 51,
    "Budget": 1,
    "Country": "Germany"
  },
```

```
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 59,
  "Budget": 13,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 11,
  "AudienceRating": 53,
  "Budget": 73,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 71,
  "AudienceRating": 71,
  "Budget": 24,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 73,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 75,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 26,
  "AudienceRating": 52,
  "Budget": 80,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 25,
  "AudienceRating": 52,
  "Budget": 19,
  "Country": "UK"
},
{
```

```
  "Genre": "Romance",
  "RottenTomatoesRating": 71,
  "AudienceRating": 68,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 75,
  "AudienceRating": 68,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 57,
  "Budget": 52,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 68,
  "AudienceRating": 58,
  "Budget": 75,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 8,
  "AudienceRating": 36,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 50,
  "Budget": 150,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 62,
  "Budget": 175,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 29,
```

```
  "AudienceRating": 43,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 73,
  "AudienceRating": 63,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 51,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 28,
  "AudienceRating": 56,
  "Budget": 110,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 85,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
  "AudienceRating": 47,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 56,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 3,
  "AudienceRating": 61,
```

```
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 80,
  "AudienceRating": 90,
  "Budget": 33,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 48,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 60,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 75,
  "AudienceRating": 40,
  "Budget": 0,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 82,
  "AudienceRating": 86,
  "Budget": 53,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 10,
  "AudienceRating": 59,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 31,
  "Budget": 112,
  "Country": "UK"
```

```
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 58,
  "AudienceRating": 85,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 35,
  "AudienceRating": 44,
  "Budget": 36,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 26,
  "AudienceRating": 64,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 20,
  "AudienceRating": 45,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 67,
  "Budget": 150,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 67,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 44,
  "AudienceRating": 66,
  "Budget": 35,
  "Country": "South Korea"
},
```

```
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 54,
  "AudienceRating": 65,
  "Budget": 12,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 83,
  "AudienceRating": 75,
  "Budget": 250,
  "Country": "Germany"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 78,
  "AudienceRating": 82,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 42,
  "AudienceRating": 60,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 87,
  "AudienceRating": 70,
  "Budget": 83,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 54,
  "AudienceRating": 68,
  "Budget": 7,
  "Country": "USA"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 65,
  "AudienceRating": 76,
  "Budget": 11,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 15,
  "AudienceRating": 61,
  "Budget": 18,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 69,
  "AudienceRating": 72,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 44,
  "AudienceRating": 45,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 64,
  "AudienceRating": 57,
  "Budget": 36,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 46,
  "AudienceRating": 57,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 93,
  "AudienceRating": 84,
  "Budget": 150,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 69,
```

```
  "AudienceRating": 69,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 32,
  "AudienceRating": 57,
  "Budget": 60,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 38,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 71,
  "AudienceRating": 57,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 83,
  "AudienceRating": 72,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 73,
  "Budget": 85,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 38,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 44,
  "Budget": 55,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 36,
    "AudienceRating": 59,
    "Budget": 75,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 43,
    "AudienceRating": 72,
    "Budget": 10,
    "Country": "South Korea"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 38,
    "AudienceRating": 55,
    "Budget": 40,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 86,
    "AudienceRating": 93,
    "Budget": 160,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 77,
    "AudienceRating": 59,
    "Budget": 185,
    "Country": "USA"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 88,
    "AudienceRating": 87,
    "Budget": 70,
    "Country": "UK"
  },
  {
    "Genre": "Adventure",
    "RottenTomatoesRating": 39,
    "AudienceRating": 51,
    "Budget": 60,
    "Country": "UK"
```

```
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 67,
  "AudienceRating": 65,
  "Budget": 2,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 82,
  "AudienceRating": 90,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 94,
  "AudienceRating": 91,
  "Budget": 186,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 74,
  "AudienceRating": 80,
  "Budget": 200,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 63,
  "Budget": 85,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 42,
  "AudienceRating": 84,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 4,
  "AudienceRating": 59,
  "Budget": 79,
  "Country": "South Korea"
},
{
```

```
  "Genre": "Romance",
  "RottenTomatoesRating": 85,
  "AudienceRating": 77,
  "Budget": 0,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 42,
  "AudienceRating": 37,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 38,
  "AudienceRating": 55,
  "Budget": 45,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 13,
  "AudienceRating": 24,
  "Budget": 47,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 61,
  "AudienceRating": 55,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 50,
  "Budget": 83,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 94,
  "AudienceRating": 89,
  "Budget": 8,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
```

```
    "RottenTomatoesRating": 19,
    "AudienceRating": 63,
    "Budget": 80,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 45,
    "AudienceRating": 58,
    "Budget": 12,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 76,
    "AudienceRating": 83,
    "Budget": 30,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 25,
    "AudienceRating": 48,
    "Budget": 70,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 11,
    "AudienceRating": 45,
    "Budget": 75,
    "Country": "UK"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 78,
    "AudienceRating": 26,
    "Budget": 10,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 53,
    "AudienceRating": 52,
    "Budget": 117,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 91,
    "AudienceRating": 83,
```

```
  "Budget": 33,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 33,
  "AudienceRating": 50,
  "Budget": 50,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 26,
  "AudienceRating": 38,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 34,
  "AudienceRating": 46,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 49,
  "Budget": 19,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 41,
  "Budget": 58,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 19,
  "AudienceRating": 36,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 40,
  "AudienceRating": 62,
  "Budget": 30,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 8,
    "AudienceRating": 55,
    "Budget": 35,
    "Country": "South Korea"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 28,
    "AudienceRating": 62,
    "Budget": 38,
    "Country": "South Korea"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 69,
    "AudienceRating": 73,
    "Budget": 27,
    "Country": "South Korea"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 27,
    "AudienceRating": 48,
    "Budget": 35,
    "Country": "Germany"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 9,
    "AudienceRating": 40,
    "Budget": 100,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 79,
    "AudienceRating": 86,
    "Budget": 110,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 48,
    "AudienceRating": 55,
    "Budget": 30,
    "Country": "Germany"
  },
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 18,
  "AudienceRating": 40,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 33,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 73,
  "AudienceRating": 66,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 29,
  "AudienceRating": 71,
  "Budget": 30,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 22,
  "AudienceRating": 51,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 13,
  "AudienceRating": 61,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 53,
  "AudienceRating": 76,
  "Budget": 66,
  "Country": "USA"
},
{
```

```
  "Genre": "Thriller",
  "RottenTomatoesRating": 86,
  "AudienceRating": 73,
  "Budget": 4,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 63,
  "AudienceRating": 77,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 46,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 36,
  "Budget": 35,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 19,
  "AudienceRating": 42,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 2,
  "AudienceRating": 31,
  "Budget": 30,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 90,
  "AudienceRating": 72,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 93,
```

```
  "AudienceRating": 84,
  "Budget": 17,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 34,
  "AudienceRating": 68,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 70,
  "Budget": 60,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 93,
  "AudienceRating": 86,
  "Budget": 145,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 95,
  "AudienceRating": 89,
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 38,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 54,
  "AudienceRating": 54,
```

```
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 79,
  "AudienceRating": 61,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 56,
  "AudienceRating": 75,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 47,
  "AudienceRating": 54,
  "Budget": 55,
  "Country": "USA"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 63,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 51,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 43,
  "Budget": 17,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 47,
  "AudienceRating": 73,
  "Budget": 28,
  "Country": "USA"
```

```
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 9,
  "AudienceRating": 29,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 83,
  "AudienceRating": 84,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 49,
  "AudienceRating": 61,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 77,
  "AudienceRating": 54,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 31,
  "AudienceRating": 72,
  "Budget": 130,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 23,
  "AudienceRating": 37,
  "Budget": 21,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 70,
  "AudienceRating": 70,
  "Budget": 15,
  "Country": "UK"
},
```

```
{
  "Genre": "Romance",
  "RottenTomatoesRating": 8,
  "AudienceRating": 48,
  "Budget": 56,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 29,
  "AudienceRating": 59,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 21,
  "AudienceRating": 27,
  "Budget": 3,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 73,
  "AudienceRating": 67,
  "Budget": 10,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 60,
  "Budget": 150,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 31,
  "AudienceRating": 56,
  "Budget": 30,
  "Country": "USA"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 51,
  "AudienceRating": 50,
  "Budget": 37,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
```

```
  "RottenTomatoesRating": 37,
  "AudienceRating": 40,
  "Budget": 80,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 25,
  "AudienceRating": 57,
  "Budget": 50,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 95,
  "AudienceRating": 84,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 64,
  "Budget": 28,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 49,
  "AudienceRating": 57,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 9,
  "AudienceRating": 56,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 34,
  "AudienceRating": 66,
  "Budget": 5,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 51,
```

```
  "AudienceRating": 40,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 20,
  "AudienceRating": 50,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 70,
  "AudienceRating": 74,
  "Budget": 85,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 5,
  "AudienceRating": 49,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 37,
  "AudienceRating": 54,
  "Budget": 15,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 0,
  "AudienceRating": 36,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 55,
  "AudienceRating": 65,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 49,
  "Budget": 0,
```

```
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 68,
    "AudienceRating": 79,
    "Budget": 5,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 15,
    "AudienceRating": 47,
    "Budget": 10,
    "Country": "Germany"
  },
  {
    "Genre": "Romance",
    "RottenTomatoesRating": 21,
    "AudienceRating": 82,
    "Budget": 30,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 28,
    "AudienceRating": 49,
    "Budget": 40,
    "Country": "USA"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 82,
    "AudienceRating": 56,
    "Budget": 1,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 0,
    "AudienceRating": 0,
    "Budget": 3,
    "Country": "UK"
  },
  {
    "Genre": "Horror",
    "RottenTomatoesRating": 68,
    "AudienceRating": 58,
    "Budget": 5,
    "Country": "South Korea"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 11,
  "AudienceRating": 41,
  "Budget": 26,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 70,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 50,
  "Budget": 26,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 74,
  "Budget": 15,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 50,
  "AudienceRating": 57,
  "Budget": 95,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 68,
  "AudienceRating": 74,
  "Budget": 26,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 74,
  "AudienceRating": 45,
  "Budget": 24,
  "Country": "Germany"
},
{
```

```
  "Genre": "Comedy",
  "RottenTomatoesRating": 60,
  "AudienceRating": 72,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 34,
  "AudienceRating": 61,
  "Budget": 250,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 45,
  "AudienceRating": 74,
  "Budget": 300,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 88,
  "AudienceRating": 62,
  "Budget": 3,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 7,
  "AudienceRating": 31,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 8,
  "AudienceRating": 55,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Action",
```

```
  "RottenTomatoesRating": 17,
  "AudienceRating": 37,
  "Budget": 60,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 71,
  "Budget": 200,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 8,
  "AudienceRating": 46,
  "Budget": 18,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
  "AudienceRating": 47,
  "Budget": 35,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 47,
  "Budget": 38,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 62,
  "Budget": 230,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 59,
  "AudienceRating": 46,
  "Budget": 12,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 50,
```

```
  "Budget": 50,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 85,
  "AudienceRating": 61,
  "Budget": 12,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 36,
  "AudienceRating": 70,
  "Budget": 48,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 81,
  "Budget": 110,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 71,
  "AudienceRating": 72,
  "Budget": 58,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 11,
  "AudienceRating": 41,
  "Budget": 42,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 57,
  "AudienceRating": 59,
  "Budget": 4,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 28,
  "AudienceRating": 70,
  "Budget": 16,
```

```
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 22,
    "AudienceRating": 43,
    "Budget": 32,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 24,
    "AudienceRating": 53,
    "Budget": 60,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 83,
    "AudienceRating": 87,
    "Budget": 93,
    "Country": "Germany"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 42,
    "AudienceRating": 59,
    "Budget": 200,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 77,
    "AudienceRating": 75,
    "Budget": 28,
    "Country": "South Korea"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 20,
    "AudienceRating": 68,
    "Budget": 140,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 62,
    "AudienceRating": 62,
    "Budget": 110,
    "Country": "UK"
  },
```

```
{
  "Genre": "Action",
  "RottenTomatoesRating": 30,
  "AudienceRating": 39,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 18,
  "AudienceRating": 70,
  "Budget": 10,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 40,
  "AudienceRating": 67,
  "Budget": 11,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 81,
  "AudienceRating": 83,
  "Budget": 60,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 58,
  "AudienceRating": 57,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 10,
  "AudienceRating": 32,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 21,
  "AudienceRating": 41,
  "Budget": 3,
  "Country": "UK"
},
{
```

```
  "Genre": "Drama",
  "RottenTomatoesRating": 27,
  "AudienceRating": 77,
  "Budget": 55,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 49,
  "AudienceRating": 81,
  "Budget": 58,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 49,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 46,
  "AudienceRating": 61,
  "Budget": 19,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 80,
  "AudienceRating": 80,
  "Budget": 7,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 16,
  "AudienceRating": 25,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 57,
  "AudienceRating": 60,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 70,
```

```
  "AudienceRating": 81,
  "Budget": 90,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 60,
  "AudienceRating": 79,
  "Budget": 125,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 66,
  "Budget": 39,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 82,
  "Budget": 61,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 7,
  "AudienceRating": 40,
  "Budget": 8,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 68,
  "AudienceRating": 73,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 19,
  "Budget": 10,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 50,
  "AudienceRating": 41,
```

```
  "Budget": 7,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 81,
  "AudienceRating": 47,
  "Budget": 15,
  "Country": "South Korea"
},
{
  "Genre": "Romance",
  "RottenTomatoesRating": 0,
  "AudienceRating": 0,
  "Budget": 35,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 22,
  "AudienceRating": 38,
  "Budget": 13,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 45,
  "AudienceRating": 42,
  "Budget": 0,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 79,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 92,
  "AudienceRating": 81,
  "Budget": 32,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 38,
  "AudienceRating": 61,
  "Budget": 120,
  "Country": "South Korea"
```

```
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 61,
  "AudienceRating": 54,
  "Budget": 258,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 22,
  "AudienceRating": 40,
  "Budget": 27,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 94,
  "AudienceRating": 91,
  "Budget": 140,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 76,
  "AudienceRating": 86,
  "Budget": 70,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 54,
  "AudienceRating": 68,
  "Budget": 65,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 26,
  "AudienceRating": 81,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 46,
  "AudienceRating": 67,
  "Budget": 30,
  "Country": "UK"
},
```

```
{
  "Genre": "Drama",
  "RottenTomatoesRating": 49,
  "AudienceRating": 20,
  "Budget": 22,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 65,
  "AudienceRating": 27,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 16,
  "AudienceRating": 28,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 37,
  "AudienceRating": 62,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 23,
  "AudienceRating": 48,
  "Budget": 82,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 72,
  "AudienceRating": 59,
  "Budget": 5,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 82,
  "AudienceRating": 78,
  "Budget": 50,
  "Country": "USA"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 88,
  "AudienceRating": 87,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 15,
  "AudienceRating": 36,
  "Budget": 65,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 39,
  "AudienceRating": 42,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 36,
  "AudienceRating": 44,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 28,
  "AudienceRating": 46,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 92,
  "AudienceRating": 85,
  "Budget": 5,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 83,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 27,
```

```
  "AudienceRating": 57,
  "Budget": 32,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 48,
  "AudienceRating": 48,
  "Budget": 29,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 33,
  "AudienceRating": 64,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 71,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 72,
  "AudienceRating": 67,
  "Budget": 51,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 74,
  "AudienceRating": 78,
  "Budget": 130,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 97,
  "AudienceRating": 91,
  "Budget": 16,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 20,
  "AudienceRating": 47,
  "Budget": 35,
```

```
    "Country": "UK"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 78,
    "AudienceRating": 74,
    "Budget": 20,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 62,
    "AudienceRating": 57,
    "Budget": 21,
    "Country": "UK"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 38,
    "AudienceRating": 62,
    "Budget": 41,
    "Country": "USA"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 48,
    "AudienceRating": 68,
    "Budget": 80,
    "Country": "UK"
  },
  {
    "Genre": "Action",
    "RottenTomatoesRating": 7,
    "AudienceRating": 41,
    "Budget": 40,
    "Country": "USA"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 93,
    "AudienceRating": 91,
    "Budget": 110,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 46,
    "AudienceRating": 27,
    "Budget": 30,
    "Country": "UK"
```

```
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 63,
  "AudienceRating": 84,
  "Budget": 13,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 43,
  "AudienceRating": 65,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 81,
  "Budget": 45,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 24,
  "AudienceRating": 53,
  "Budget": 52,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 78,
  "Budget": 200,
  "Country": "South Korea"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 49,
  "AudienceRating": 63,
  "Budget": 155,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 66,
  "AudienceRating": 55,
  "Budget": 15,
  "Country": "USA"
},
{
```

```
  "Genre": "Drama",
  "RottenTomatoesRating": 56,
  "AudienceRating": 65,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 72,
  "AudienceRating": 54,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 67,
  "AudienceRating": 79,
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 94,
  "AudienceRating": 96,
  "Budget": 185,
  "Country": "USA"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 19,
  "AudienceRating": 34,
  "Budget": 100,
  "Country": "South Korea"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 76,
  "AudienceRating": 70,
  "Budget": 20,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 89,
  "AudienceRating": 88,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Comedy",
```

```
  "RottenTomatoesRating": 23,
  "AudienceRating": 31,
  "Budget": 70,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 60,
  "AudienceRating": 68,
  "Budget": 14,
  "Country": "South Korea"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 39,
  "AudienceRating": 43,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 40,
  "AudienceRating": 58,
  "Budget": 82,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 22,
  "AudienceRating": 51,
  "Budget": 12,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 91,
  "AudienceRating": 88,
  "Budget": 25,
  "Country": "South Korea"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 29,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 64,
  "AudienceRating": 65,
```

```
  "Budget": 55,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 27,
  "AudienceRating": 75,
  "Budget": 22,
  "Country": "Germany"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 87,
  "AudienceRating": 89,
  "Budget": 90,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 42,
  "AudienceRating": 55,
  "Budget": 180,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 44,
  "AudienceRating": 47,
  "Budget": 120,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 79,
  "AudienceRating": 87,
  "Budget": 35,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 35,
  "AudienceRating": 58,
  "Budget": 80,
  "Country": "UK"
},
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 19,
  "AudienceRating": 29,
  "Budget": 48,
```

```
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 17,
  "AudienceRating": 51,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 3,
  "AudienceRating": 22,
  "Budget": 5,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 30,
  "AudienceRating": 41,
  "Budget": 60,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 75,
  "AudienceRating": 91,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 11,
  "AudienceRating": 50,
  "Budget": 15,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 54,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 97,
  "AudienceRating": 83,
  "Budget": 15,
  "Country": "USA"
},
```

```
{
  "Genre": "Thriller",
  "RottenTomatoesRating": 85,
  "AudienceRating": 76,
  "Budget": 13,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 67,
  "AudienceRating": 75,
  "Budget": 138,
  "Country": "UK"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 78,
  "AudienceRating": 42,
  "Budget": 22,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 58,
  "AudienceRating": 39,
  "Budget": 50,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 56,
  "AudienceRating": 47,
  "Budget": 19,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 53,
  "AudienceRating": 54,
  "Budget": 13,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 61,
  "AudienceRating": 47,
  "Budget": 10,
  "Country": "USA"
},
{
```

```
  "Genre": "Action",
  "RottenTomatoesRating": 66,
  "AudienceRating": 75,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 94,
  "AudienceRating": 72,
  "Budget": 4,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 52,
  "AudienceRating": 78,
  "Budget": 80,
  "Country": "USA"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 66,
  "AudienceRating": 85,
  "Budget": 20,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 6,
  "AudienceRating": 42,
  "Budget": 150,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 73,
  "AudienceRating": 32,
  "Budget": 2,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 19,
  "AudienceRating": 66,
  "Budget": 20,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 84,
```

```
  "AudienceRating": 82,
  "Budget": 40,
  "Country": "South Korea"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 39,
  "AudienceRating": 39,
  "Budget": 30,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 47,
  "AudienceRating": 55,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 14,
  "AudienceRating": 38,
  "Budget": 62,
  "Country": "UK"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 32,
  "AudienceRating": 57,
  "Budget": 65,
  "Country": "UK"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 53,
  "AudienceRating": 52,
  "Budget": 40,
  "Country": "USA"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 52,
  "AudienceRating": 43,
  "Budget": 25,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 72,
  "AudienceRating": 64,
```

```
  "Budget": 18,
  "Country": "USA"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 14,
  "AudienceRating": 40,
  "Budget": 145,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 97,
  "AudienceRating": 87,
  "Budget": 45,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 33,
  "AudienceRating": 52,
  "Budget": 20,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 41,
  "AudienceRating": 65,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Action",
  "RottenTomatoesRating": 78,
  "AudienceRating": 57,
  "Budget": 100,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 12,
  "AudienceRating": 47,
  "Budget": 59,
  "Country": "Germany"
},
{
  "Genre": "Comedy",
  "RottenTomatoesRating": 43,
  "AudienceRating": 74,
  "Budget": 40,
  "Country": "Germany"
```

```
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 9,
  "AudienceRating": 53,
  "Budget": 40,
  "Country": "Germany"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 20,
  "AudienceRating": 43,
  "Budget": 37,
  "Country": "UK"
},
{
  "Genre": "Adventure",
  "RottenTomatoesRating": 75,
  "AudienceRating": 65,
  "Budget": 25,
  "Country": "UK"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 4,
  "AudienceRating": 29,
  "Budget": 16,
  "Country": "USA"
},
{
  "Genre": "Horror",
  "RottenTomatoesRating": 46,
  "AudienceRating": 32,
  "Budget": 25,
  "Country": "Germany"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 50,
  "AudienceRating": 48,
  "Budget": 45,
  "Country": "South Korea"
},
{
  "Genre": "Drama",
  "RottenTomatoesRating": 56,
  "AudienceRating": 78,
  "Budget": 11,
  "Country": "South Korea"
},
```

```
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 90,
    "AudienceRating": 78,
    "Budget": 75,
    "Country": "USA"
  },
  {
    "Genre": "Comedy",
    "RottenTomatoesRating": 22,
    "AudienceRating": 43,
    "Budget": 25,
    "Country": "USA"
  },
  {
    "Genre": "Drama",
    "RottenTomatoesRating": 56,
    "AudienceRating": 59,
    "Budget": 60,
    "Country": "Germany"
  },
  {
    "Genre": "Thriller",
    "RottenTomatoesRating": 18,
    "AudienceRating": 61,
    "Budget": 15,
    "Country": "UK"
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Ratings'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Go to the **Filters** page, add a new filter value, and set its properties as below.

Expression	Operator	Values
`=[Country]`	In	`="UK"` `="USA"`

4. Click **OK** to save the changes.

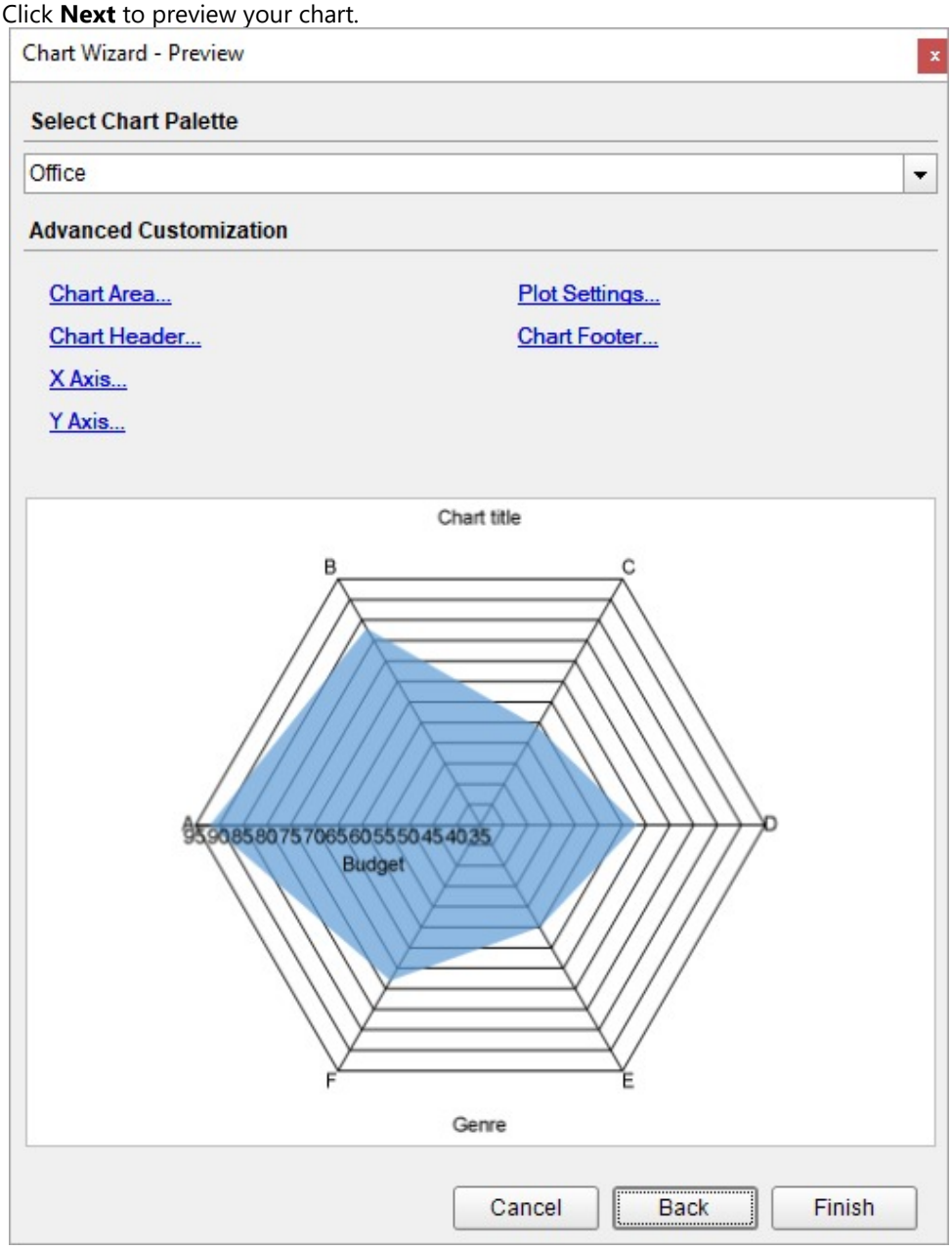
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'Ratings' and the **Chart Type** as 'Radar Area'.
3. Click **Next** to proceed. Here, we will define a data series value to display the average movie budget in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[Budget]`	Average

5. In **Choose Data Category**, set **Field** to `=[Genre]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or,

you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Encodings** page > **Detail** tab, add a new value, and set its properties as below.
    - Set **Expression** to `=[Country]` to display average movie budgets for different countries.
    - Under **Grouping**, set **Group** to 'Cluster'.
3. Go to the **Color** tab, add a new value and, set the **Expression** to `=[Country]`.
4. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to '$'0'M'.
4. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 20
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the X-axis.

Chart Palette

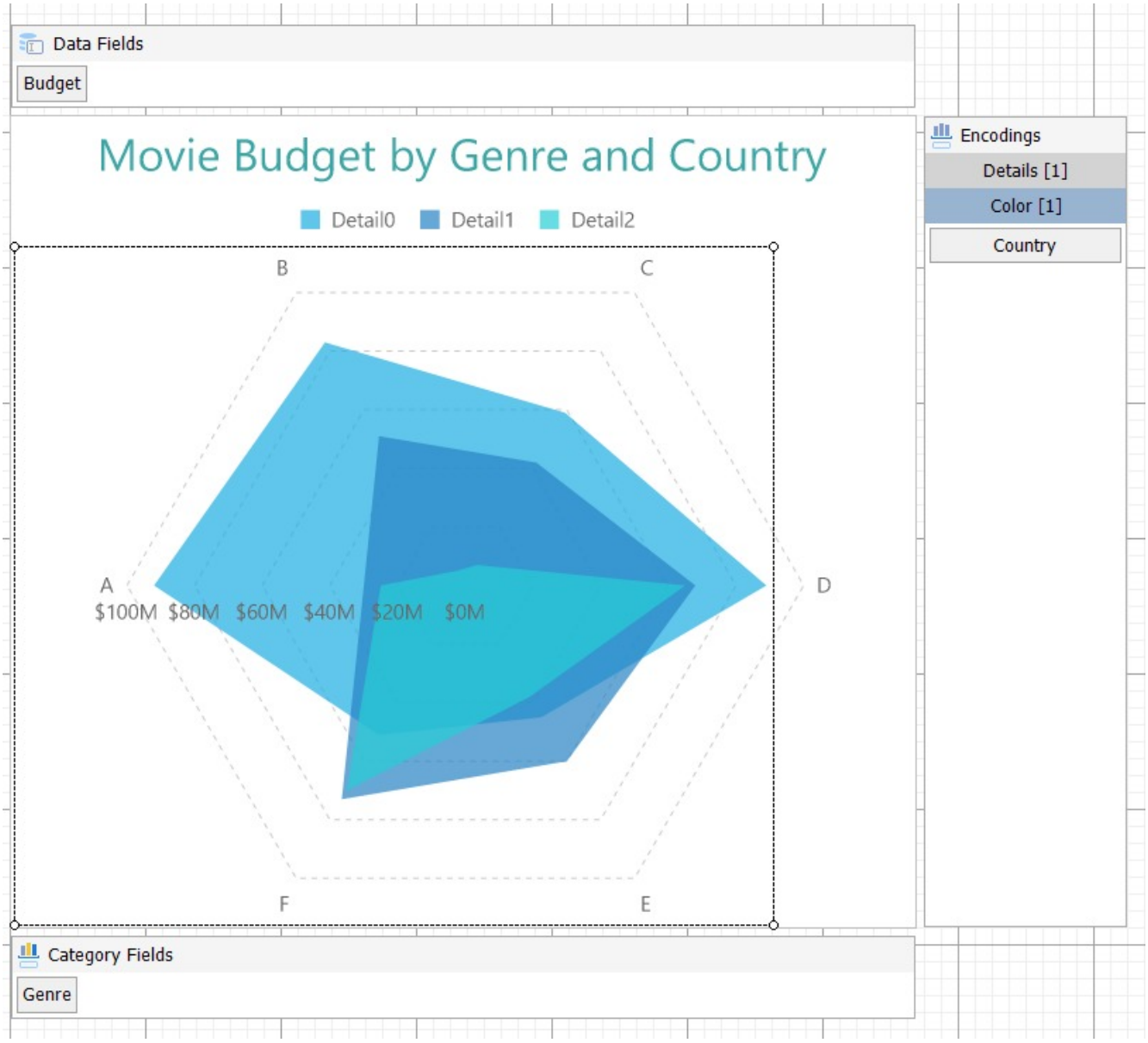
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select 'Blue2' from the drop-down.
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Movie Budget by Genre and Country'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Spiral Chart

The Spiral plot originates at the center of a circle and then progresses outward, arranging the categories across a radial line and encoding the data values into circular bars along the spiral path. The Spiral plot is suitable for comparing data values across categories using a spiral shape. The categories can be further split to represent data values as clusters or stacks. Spiral Plots are highly useful in showing large data sets, usually in displaying trends over an extended period. This makes Spiral Plots a good option for displaying periodic patterns.

Simple Spiral Chart

The simplest type of a spiral chart is quite similar to a bar chart showcased on a polar coordinate system. The Create Stacked Spiral Chart walkthrough showcases plotting the Sales Amount per Product Category.

Stacked Spiral Chart

A Stacked Spiral chart breaks down values of data into subcategories by placing corresponding circular subsections next to each other within the containing category. The Create Stacked Spiral Chart walkthrough showcases plotting the Sales Amount for each Sales Channel per Product Category.

Stacked Percentage Spiral Chart

A Stacked Percentage Spiral chart combines the stacked spiral plot and the Percentage axis scale. The chart indicates the contribution of each data values' sub-category to a containing category's total. The Create Stacked Percentage Spiral Chart walkthrough showcases plotting the share of Sales Channels' Sales Amount per Product Category.

Spiral Plot Properties

The plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.

- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the plot.
  - Center: Positions the data label text on the center of the chart.
  - Inside: Positions the data label text inside the chart.
  - Outside: Positions the data label text outside the chart.
  - Auto: The default setting, same as Outside for chart.

LineStyle

The line style for the borders. Includes LineColor, LineStyle, and LineWidth properties for the customization.

- **LineColor**: Specify the color of the border around the bars.
- **LineStyle:** Specify the line style of the border around the bars as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the bars.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to:

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Spiral charts.

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default:** Indicates a straight line.
- **Spline:** Indicates a bezier curve.
- **Step Center, Step Left, and Step Right**: Indicates a stepped line with different step directions.

Opacity

The Opacity determines the opacity of areas filled with color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

StartAngle

Indicates the arc angle that defines the clockwise rotation of the plot. A full rotation makes 360 degrees. The Create Simple Spiral Chart walkthrough shows the chart **StartAngle** set to 90 degrees.

UnpivotData

Determines whether to display multiple data fields as a single data field. By default, the property is set to 'False'.

Encodings

Category Encoding

The Category Encoding of a plot is a set of properties that determine the period over which the plot generates connected data points representing those above Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Colors Encoding

The Colors Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Spiral charts take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the area subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked plot.
  - Cluster: You can use this value to configure area subsections that overlap each other.
  - None: Equals to Cluster.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Spiral charts, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Spiral charts take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

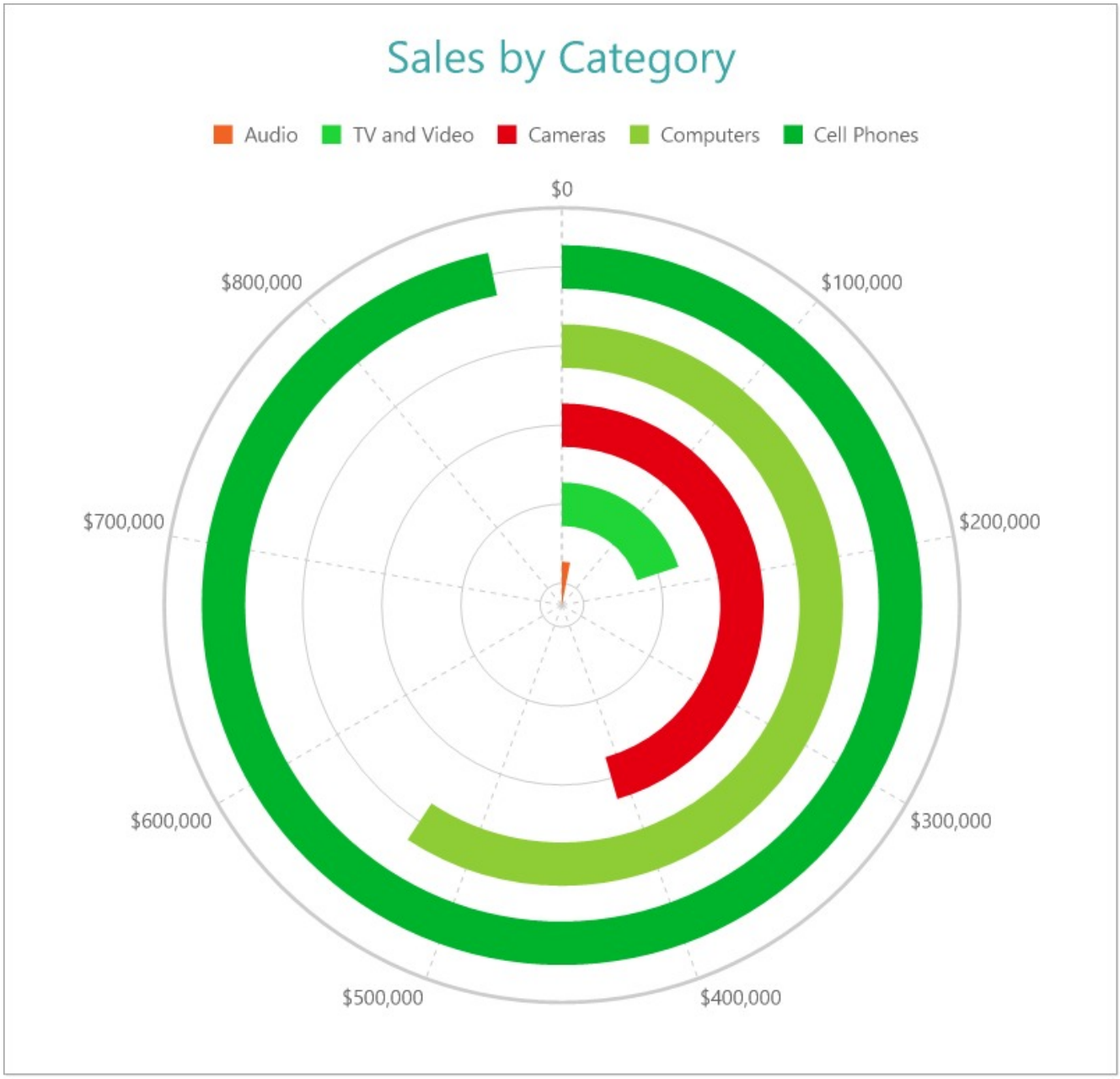
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Simple Spiral Chart

This walkthrough creates a Simple Spiral Chart. The chart shows the total sales for different product categories. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. On the same page, add a calculated field:

Field Name	Value
ProductCategory	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

4. Click **OK** to save the changes.

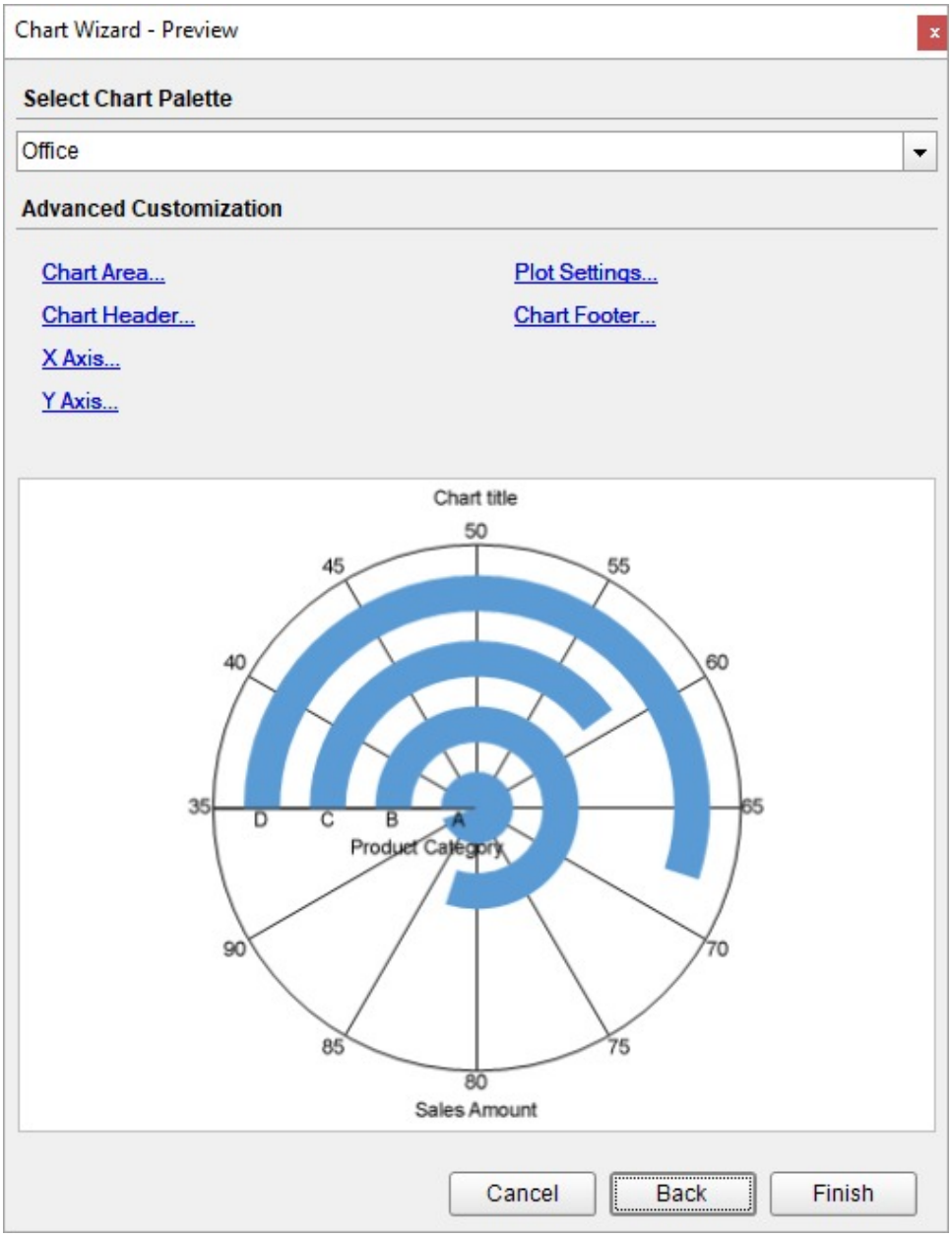
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Spiral'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Spiral chart.
4. In **Choose Data values** section, we will define a data value to display the sales amount for different product categories in the chart.
5. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

6. In **Choose Data Category**, set **Field** to `=[ProductCategory]`. We will add more customizations to the category in later steps.
7. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** tab. Here, we will sort the product categories in increasing order of their sales amounts. So fill in the following settings:

Field	Settings
Sorting field	`=[SalesAmount]`
Sorting direction	Ascending
Sorting aggregate	Sum

3. Go to the **Encodings** page > **Color** tab, add a new value and, set the **Expression** to `=[ProductCategory]`.
4. Click **OK** to complete setting up the plot.
5. With 'Plot-Plot1' selected, go to the Properties window and set the **StartAngle** property to '90' to set the start angle of the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal places)'.
4. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
    - **Show Line**: Check-on
    - **Appearance > Color**: #cccccc
    - **Appearance > Width**: 2pt
    - **Appearance > Style**: Solid
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 100000
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option to hide the data labels.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 1
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Solid
6. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and add the following colors.
    - #f26324

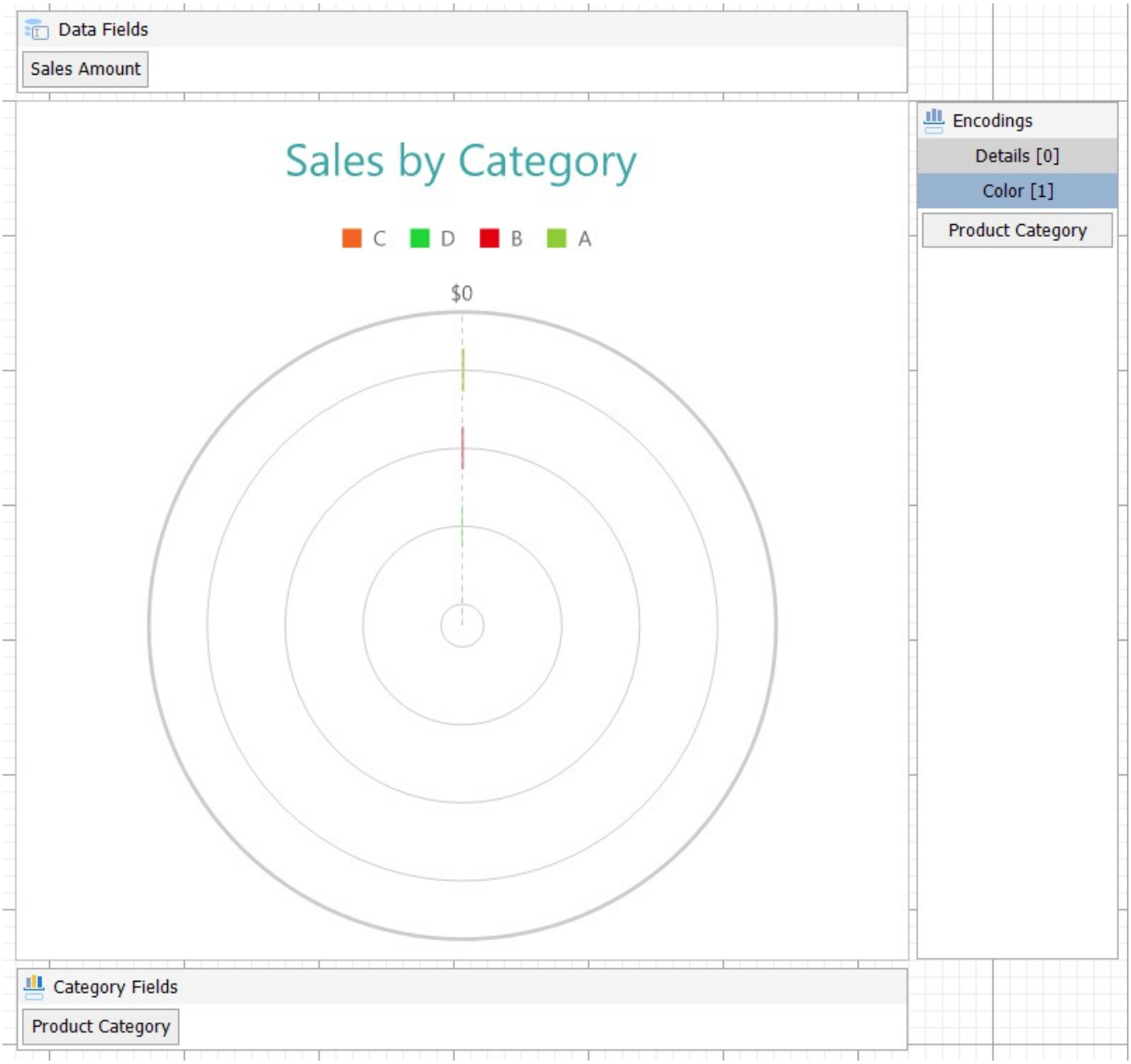
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
3. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

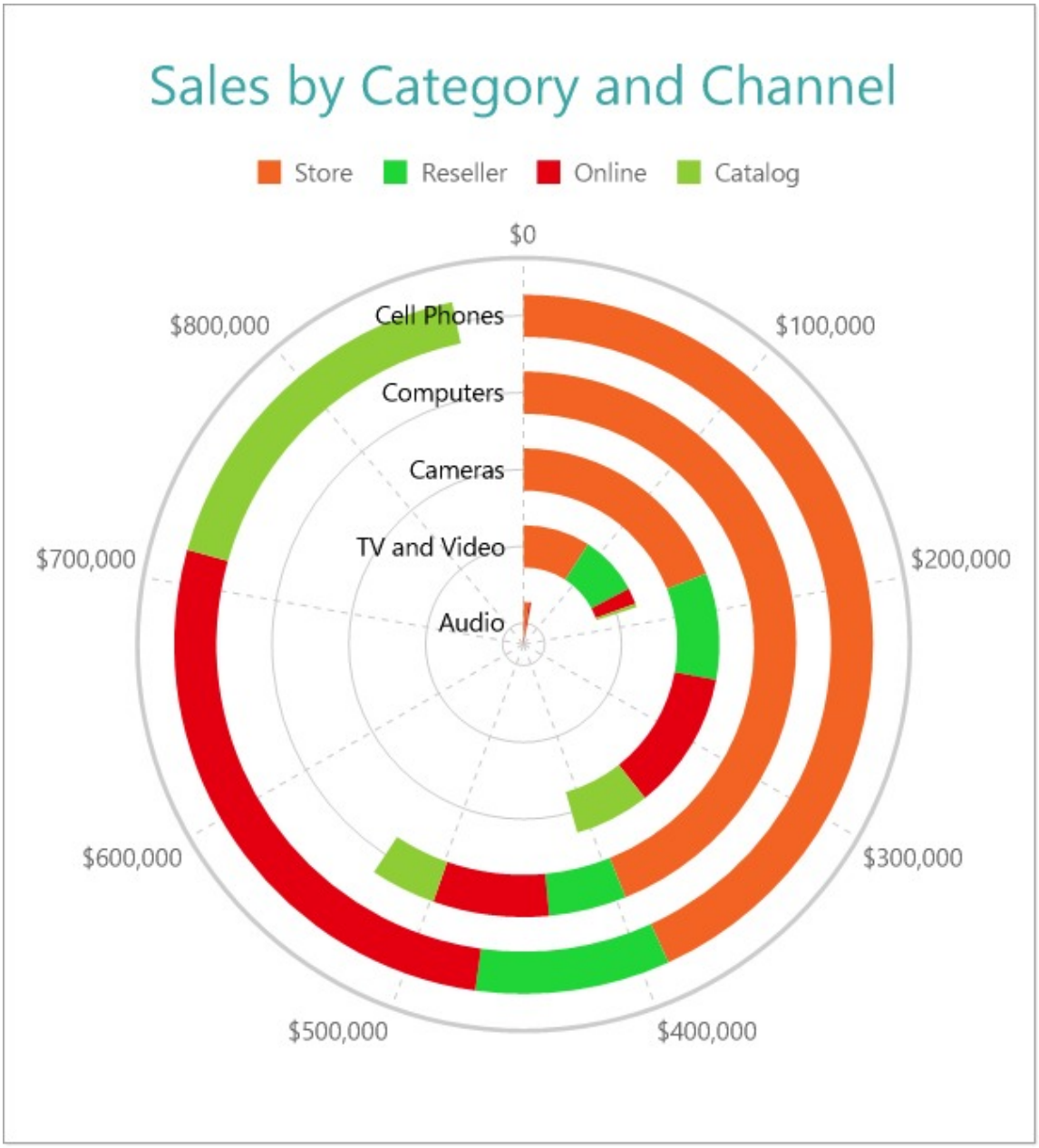


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Spiral Chart

This walkthrough creates a Stacked Spiral Chart. The chart shows the sales by product categories for each sales channel. In a stacked spiral chart, the data values are broken down into subcategories and are stacked next to each other. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. On the same page, add two calculated fields:

Field Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

4. Click **OK** to save the changes.

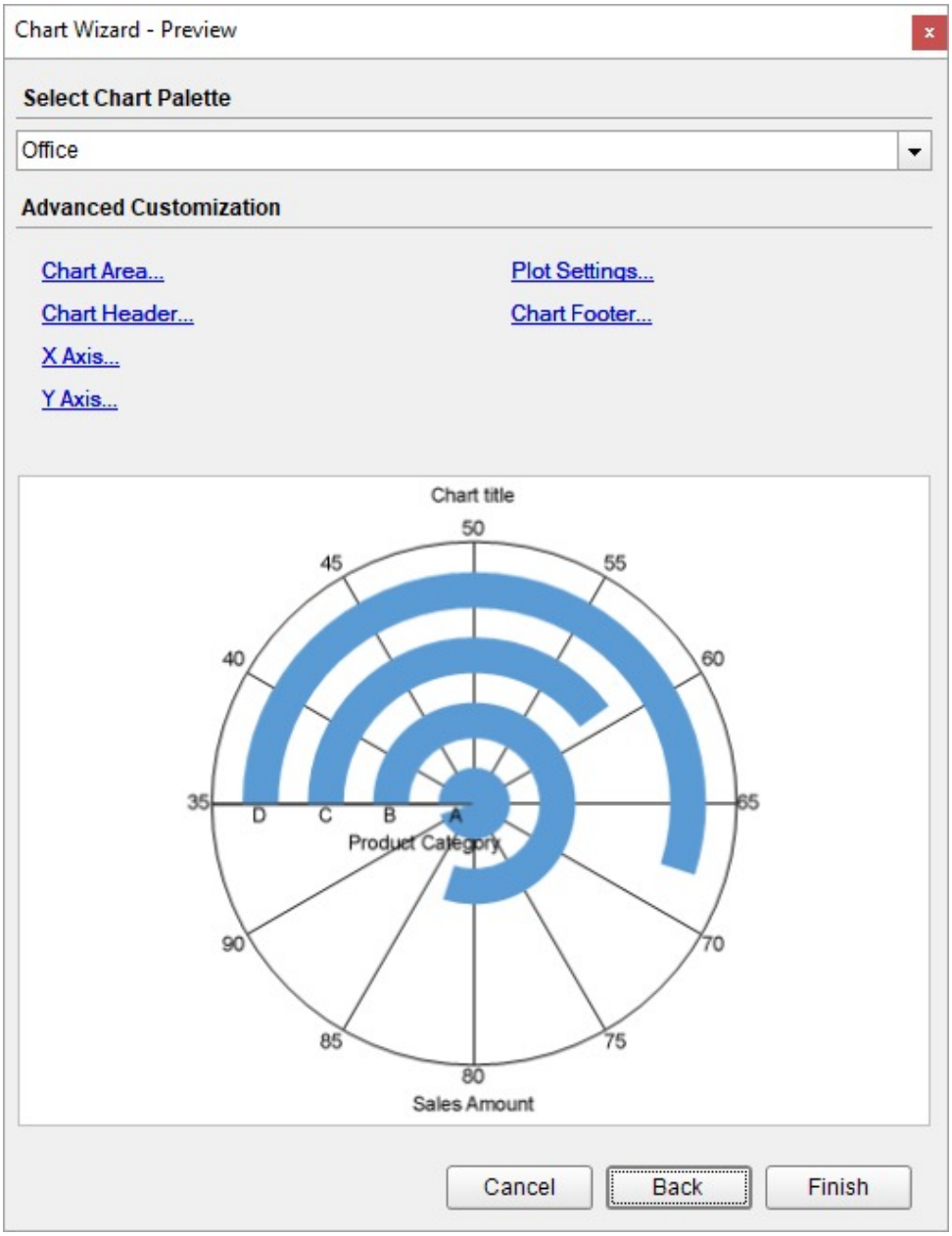
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Spiral'.
3. Click **Next** to proceed. We will define a data value to display the sales amount for different product categories in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Category**, set **Field** to `=[Product Category]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the product categories in increasing order of their sales amounts. So fill in the following settings:

Field	Settings
Sorting field	`=[SalesAmount]`
Sorting direction	Ascending
Sorting aggregate	Sum

3. Go to the **Encodings** page > **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Sales Channel]` to display sales amounts for online, store, reseller, and catalog.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to display the subcategories stacked next to each other in the spiral chart.
4. Go to the **Color** tab, add a new value and, set the **Expression** to `=[Sales Channel]`.
5. Click **OK** to complete setting up the plot.
6. With 'Plot-Plot1' selected, go to the Properties window and set the **StartAngle** property to '90' to set the start angle of the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to remove the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Currency (with 0 decimal places)'.
4. Navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
    - **Show Line**: Check-on
    - **Appearance > Color**: #cccccc
    - **Appearance > Width**: 2pt
    - **Appearance > Style**: Solid
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 100000
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-90'.
4. Navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 1
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Solid

7. Click **OK** to complete setting up the X-axis.

Chart Palette

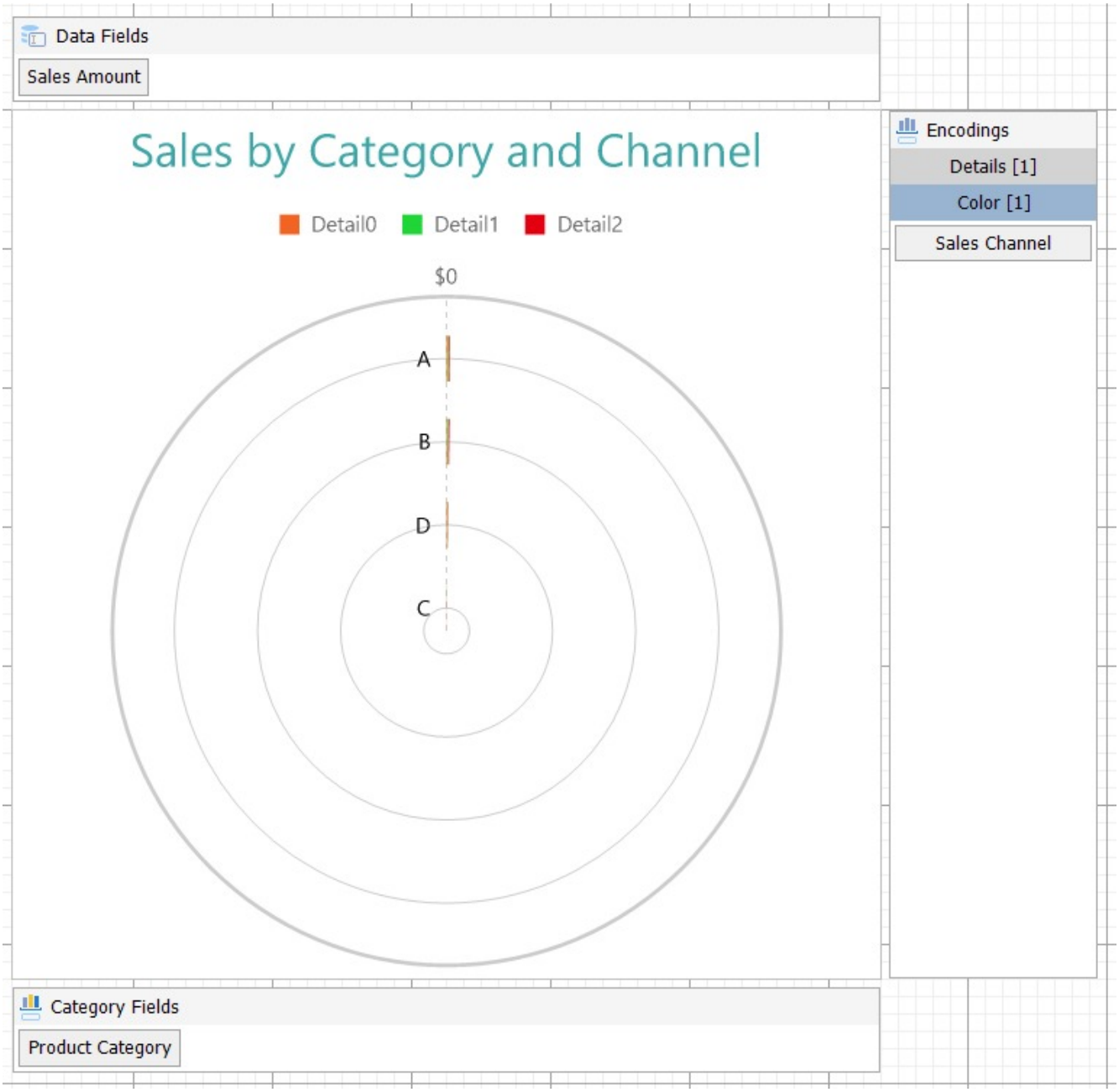
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Click **OK** to complete setting up the chart.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category and Channel'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

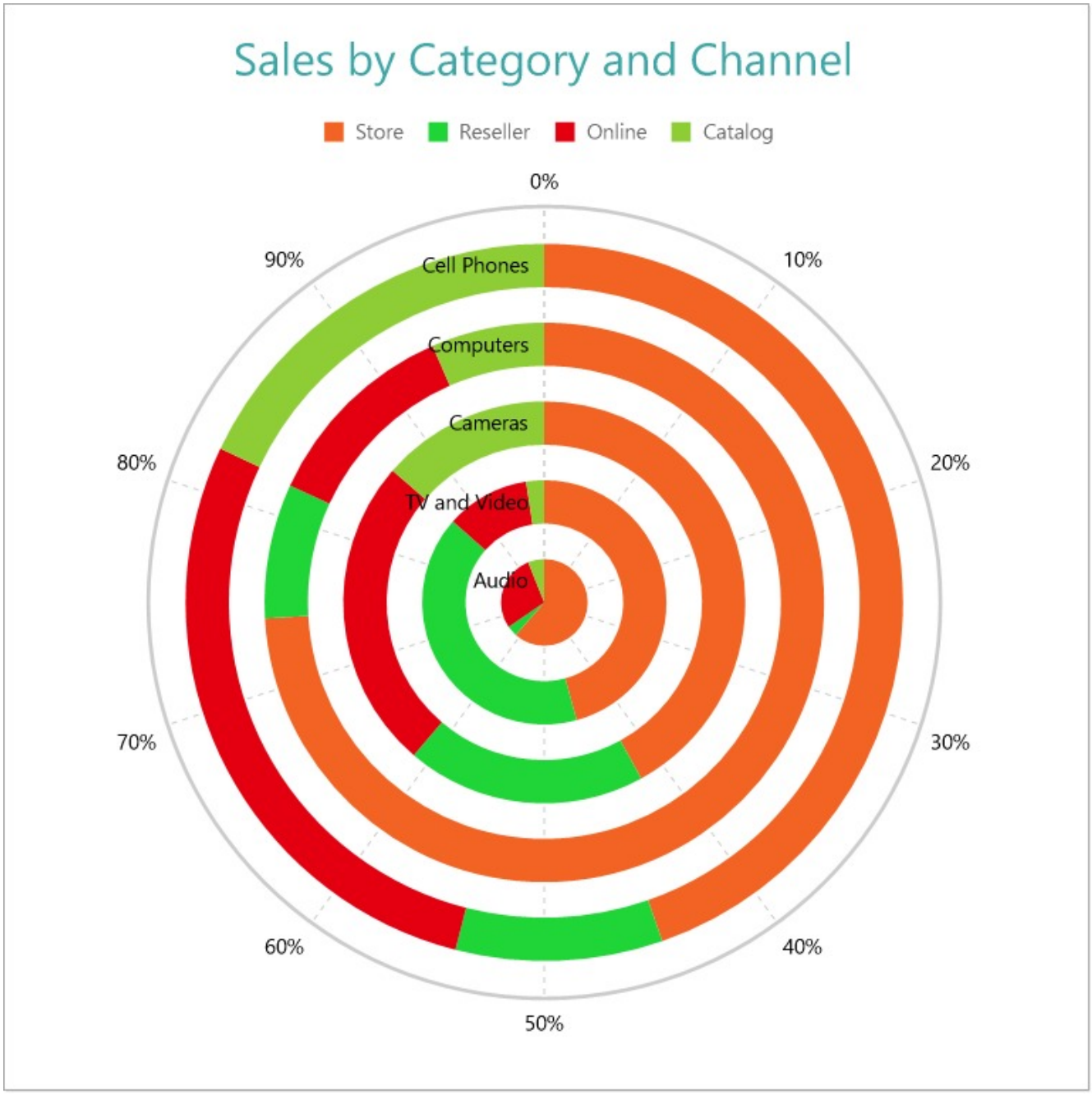


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Percentage Spiral Chart

This walkthrough creates a Stacked Percentage Spiral Chart. The chart shows the sales by product categories for each sales channel. The stacked percentage spiral chart is used to display the percentage values that each subcategory contributes within a category. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:

https://demodata.mescius.io/contoso/odata/v1/FactSales
For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. On the same page, add two calculated fields:

Field Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

4. Click **OK** to save the changes.

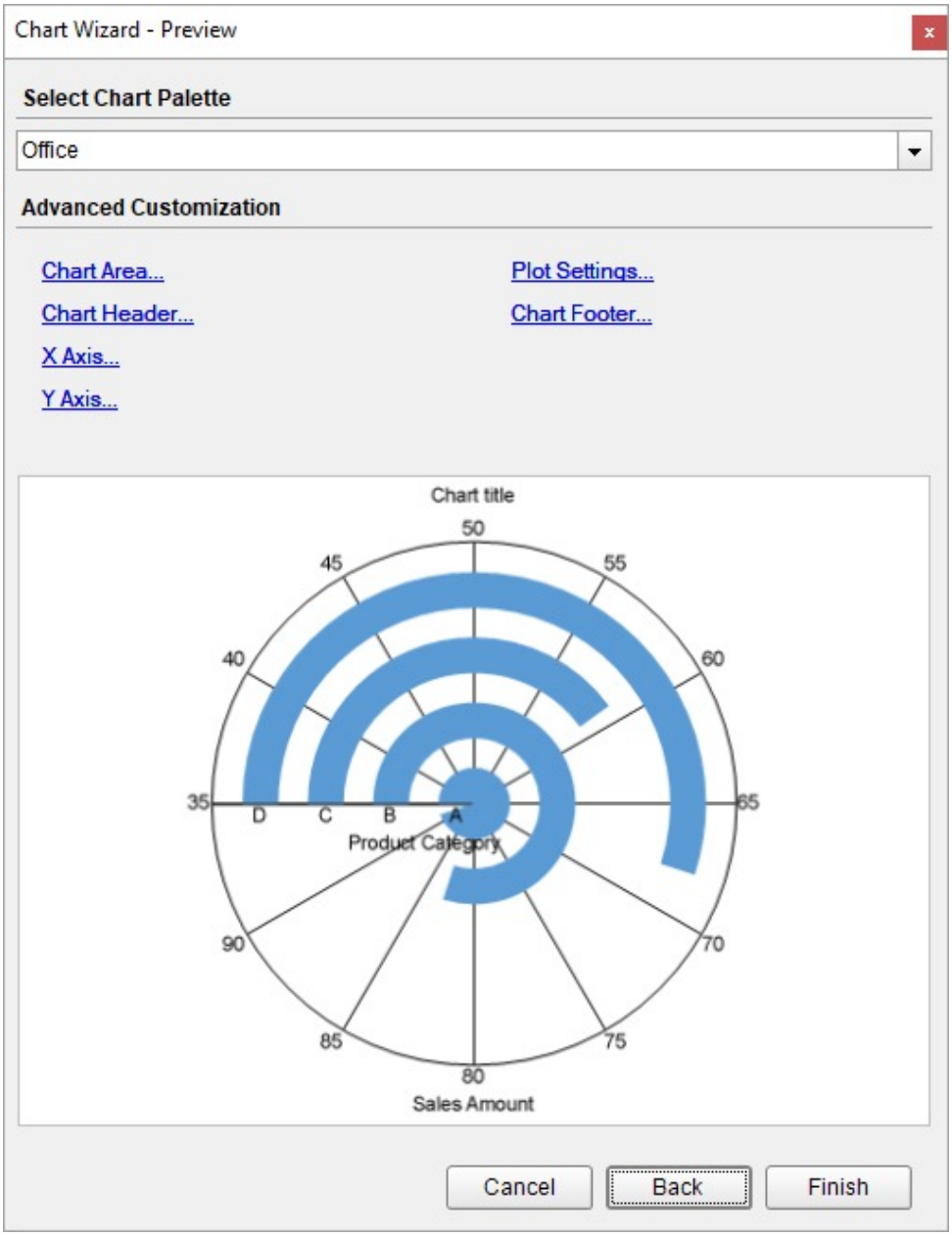
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Spiral'.
3. Click **Next** to proceed. We will define a data value to display the sales amount for different product categories in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Category**, set **Field** to `=[Product Category]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the product categories in increasing order of their sales amounts. So fill in the following settings:

Field	Settings
Sorting field	`=[SalesAmount]`
Sorting direction	Ascending
Sorting aggregate	Sum

3. Go to the **Encodings** page > **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Sales Channel]` to display sales amounts for online, store, reseller, and catalog.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to display the subcategories stacked next to each other in the spiral chart.
4. Go to the **Color** tab, add a new value and, set the **Expression** to `=[Sales Channel]`.
5. Click **OK** to complete setting up the plot.
6. With 'Plot-Plot1' selected, go to the Properties window and set the **StartAngle** property to '90' to set the start angle of the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page >**Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
4. Go to the **Line** page and set the following properties.
    - **Show Line**: Check-on
    - **Appearance > Color**: #cccccc
    - **Appearance > Width**: 2pt
    - **Appearance > Style**: Solid
5. Go to the **Major Gridline** page and set the following properties.
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
6. Go to the **Scale** page and set the **Scale Type** to 'Percentage'.
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-90'.
4. Navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 1
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Solid
7. Click **OK** to complete setting up the X-axis.

Chart Palette

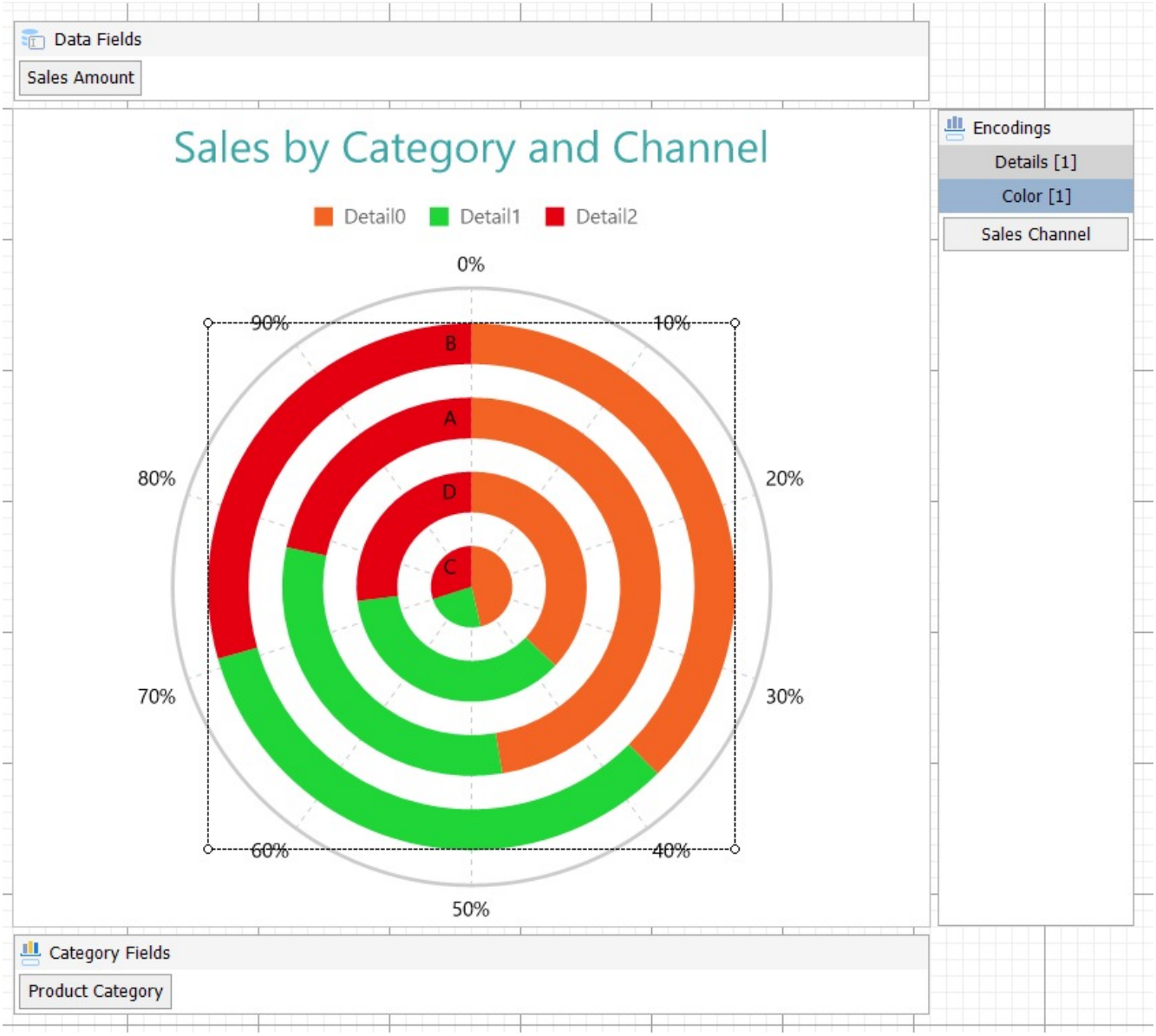
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category and Channel'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, **F5** to preview the report.

Polar Chart

The Polar plot arranges categories along the circumference of a circle and encodes data values into sectors along radial lines. The Polar plot is suitable for comparing data values across categories using a circular shape. The categories can be further split to represent data values as clusters or stacks.

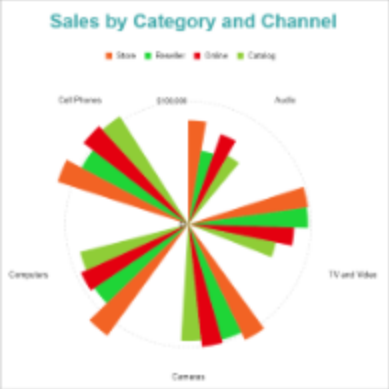
There are different types of Polar charts, which are elaborated below:

Simple Polar Chart

A Simple Polar chart displays one value for each category. See Create Simple Polar Chart walkthrough showcases plotting the Sales Amount per Product Category.

Clustered Polar Chart

A Clustered Polar chart splits the values into subcategories by placing corresponding circular sectors adjacent to each other within the containing category. The Create Clustered Polar Chart walkthrough showcases plotting the Sales Amount for each Sales Channel per Product Category.

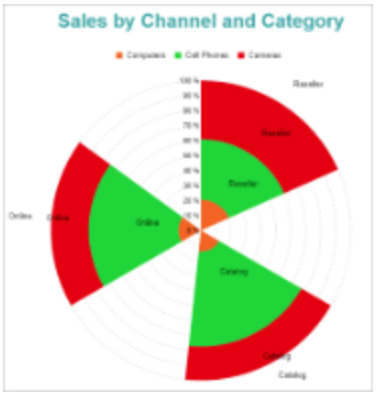


Stacked Polar Chart

A Stacked Polar chart is a good option to split the data values into subcategories by dividing corresponding circular sectors into subsections. The Create Stacked Polar Chart walkthrough showcases plotting the Sales Amount for each Product Category per Sales Channel.

Stacked Percentage Polar Chart

A Stacked Percentage Polar chart combines the stacked polar plot and the Percentage axis scale. This chart type indicates the contribution of each data values' sub-category to a total of the containing category. The Create Stacked Percentage Polar Chart walkthrough showcases plotting the percentage share of the Sales Amount of product categories for each Sales Channel.



Polar Plot Properties

The plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of the labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the plot.
  - Center: Positions the data label text on the center of the chart.
  - Inside: Positions the data label text inside the chart.
  - Outside: Positions the data label text outside the chart.
  - Auto: The default setting, same as Outside for chart.

LineStyle

The line style for the borders. Includes LineColor, LineStyle, and LineWidth properties for the customization.

- **LineColor**: Specify the color of the border around the sectors.
- **LineStyle:** Specify the line style of the border around the sectors as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the sectors.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to:

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Polar charts.

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default:** Indicates a straight line.
- **Spline:** Indicates a bezier curve.
- **Step Center, Step Left, and Step Right:** Indicate a stepped line with different step directions.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

StartAngle

Indicates the arc angle that defines the clockwise rotation of the plot. A full rotation makes 360 degrees.

UnpivotData

Determines whether to display multiple data fields as a single data field. By default, the property is set to 'False'.

Encodings

Category Encoding

The Category Encoding of a plot is a set of properties that determine the period over which the plot generates connected data points representing those above Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Colors Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Polar charts take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the area subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked plot.
  - Cluster: You can use this value to configure area subsections that overlap each other.
  - None: Equals to Cluster.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Polar charts, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Polar charts take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

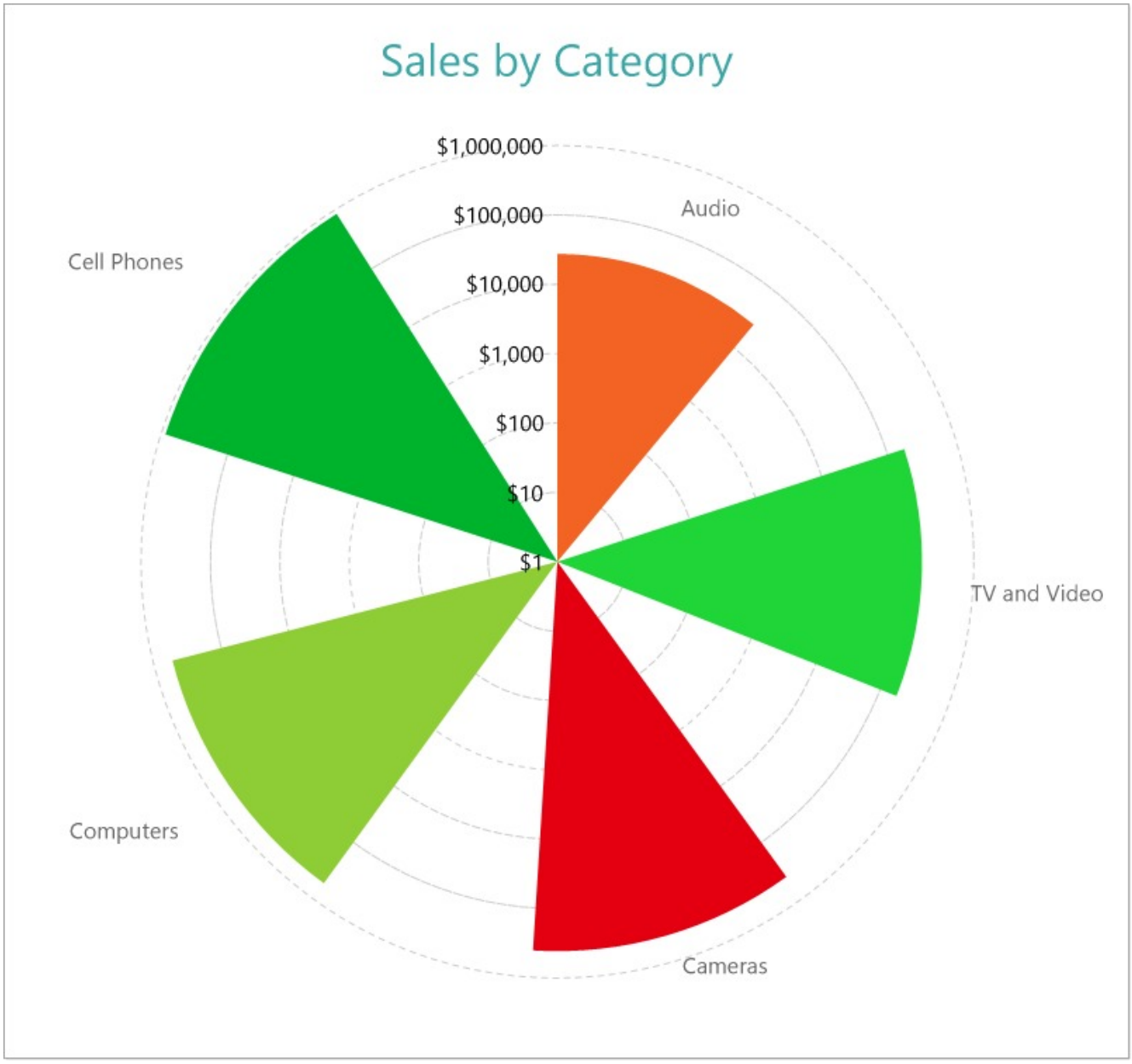
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Simple Polar Chart

This walkthrough creates a Simple Polar Chart. The chart shows the total sales for different product categories. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. On the same page, add a calculated field:

Field Name	Value
ProductCategory	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

4. Click **OK** to save the changes.

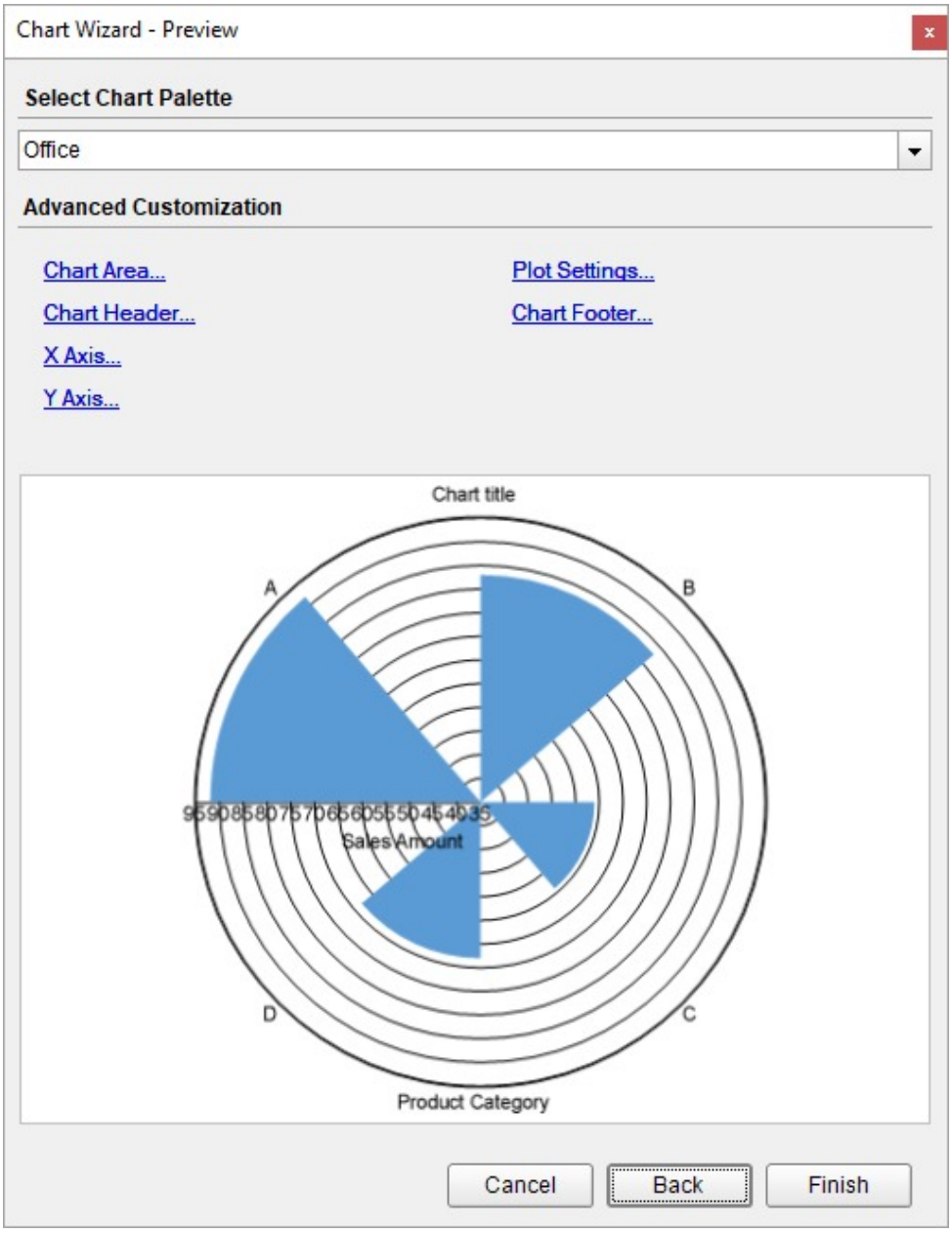
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Polar'.
3. Click **Next** to proceed. Here, we will define a data value to display the sales amount for different product categories in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Category**, set **Field** to `=[ProductCategory]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** tab. Here, we will sort the product categories in increasing order of their sales amounts. So fill in the following settings:

Field	Settings
Sorting field	`=[SalesAmount]`
Sorting direction	Ascending
Sorting aggregate	Sum

3. Go to the **Encodings** page > **Color** tab, add a new value and, set the **Expression** to `=[ProductCategory]`.
4. Go to the **Labels** page > **General** tab and set the following properties.
    - **Template**: `{categoryField.value}`
    - **Text Position**: Outside
    - **Offset**: 30
5. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
6. Click **OK** to complete setting up the plot.
7. With 'Plot-Plot1' selected, go to the Properties window and set the **StartAngle** property to '90' to set the start angle of the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the following properties.
    - **Format**: Currency (with 0 decimal places)
    - **Angle**: -90
4. Then, navigate to the Appearance tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Logarithmic
    - **Logarithmic base**: 10
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option to hide the data labels.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.

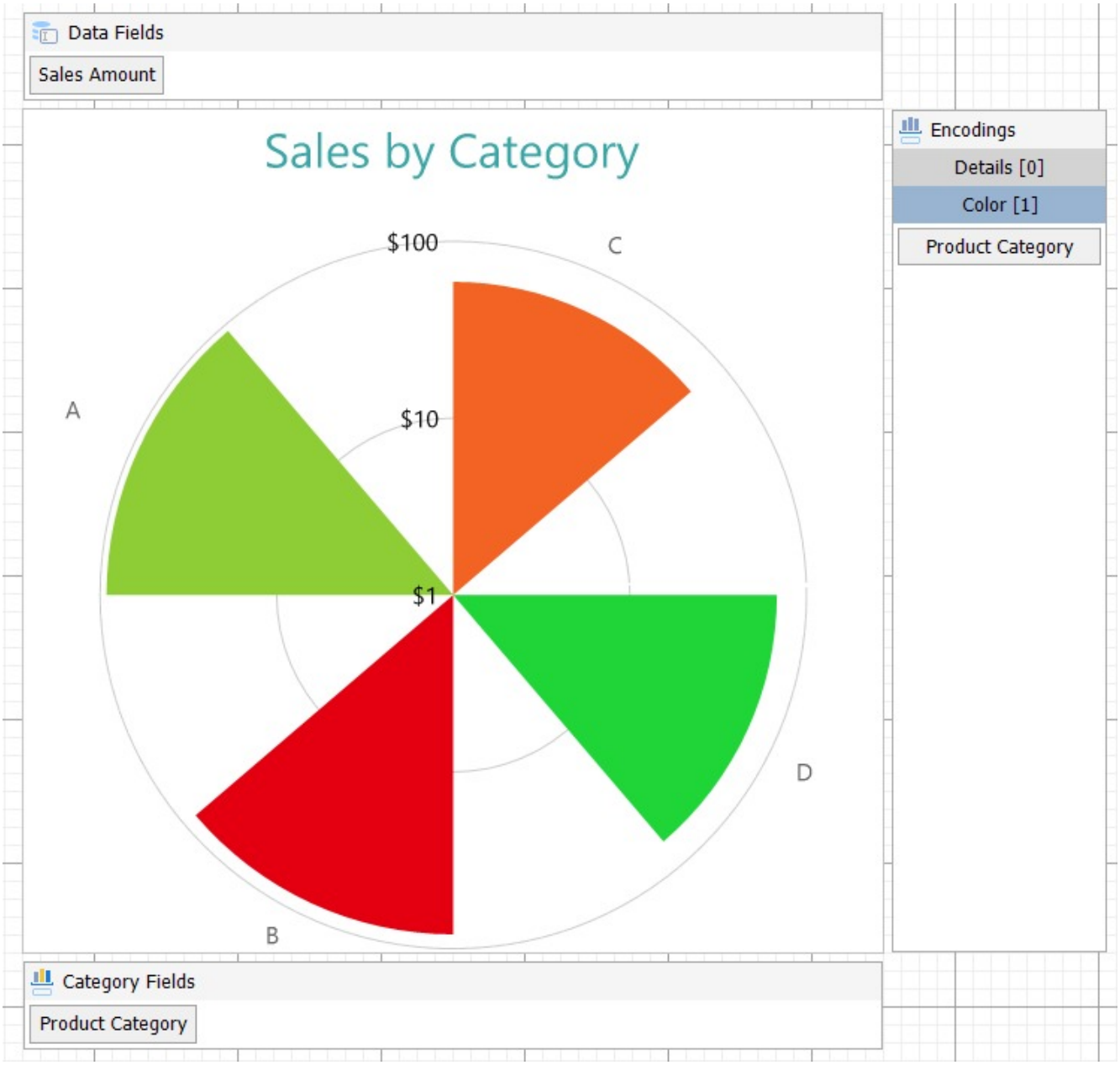
2. Go to the **Palette** page and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and check the **Hide Legend** option.
3. Click **OK** to save the settings.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

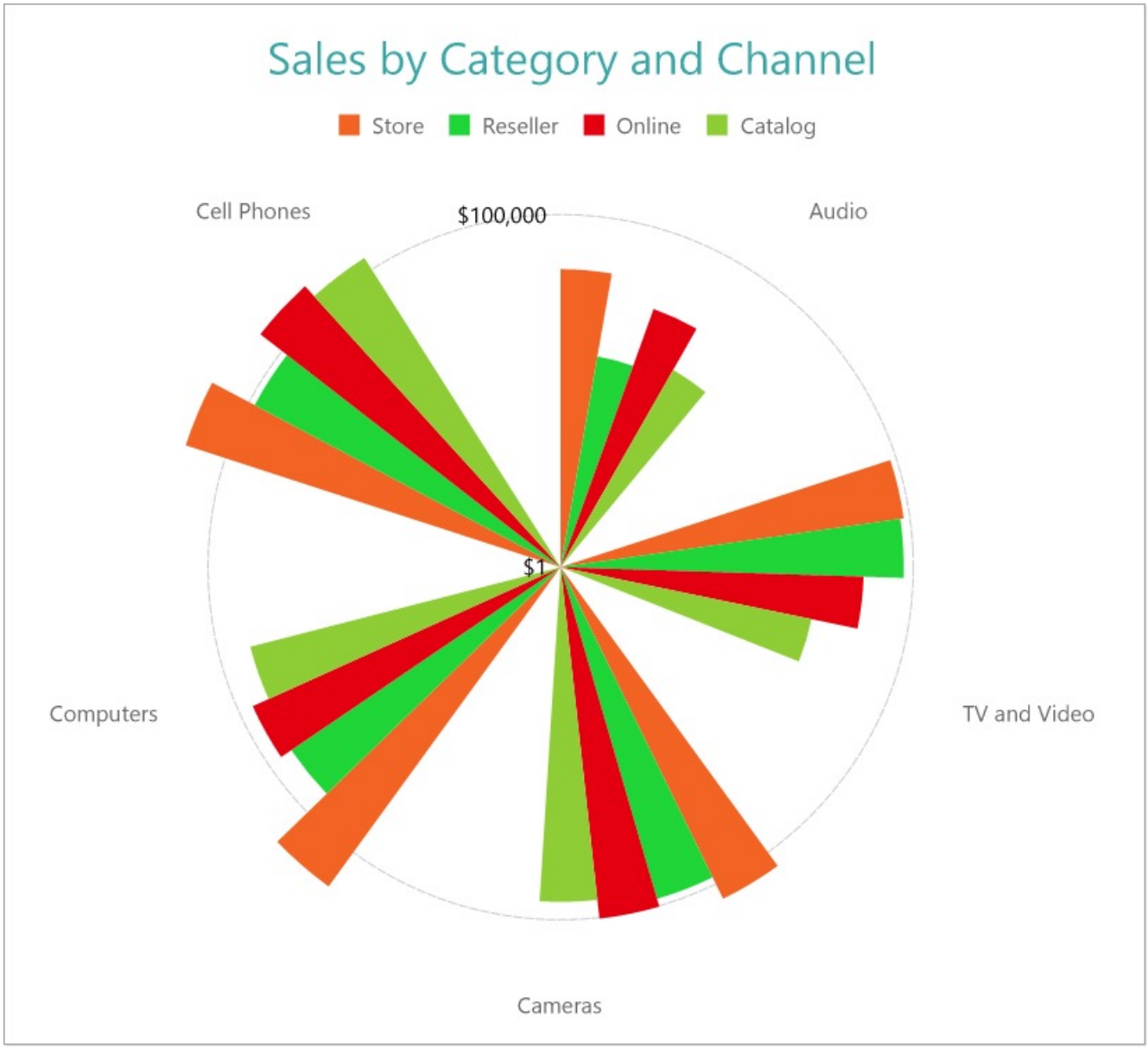


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Clustered Polar Chart

This walkthrough creates a Clustered Polar Chart. The chart displays the sales amount by product categories for each sales channel. In a clustered polar chart, the data values are broken down into subcategories and are placed next to each other. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource**

icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. On the same page, add two calculated fields:

Field Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

4. Click **OK** to save the changes.

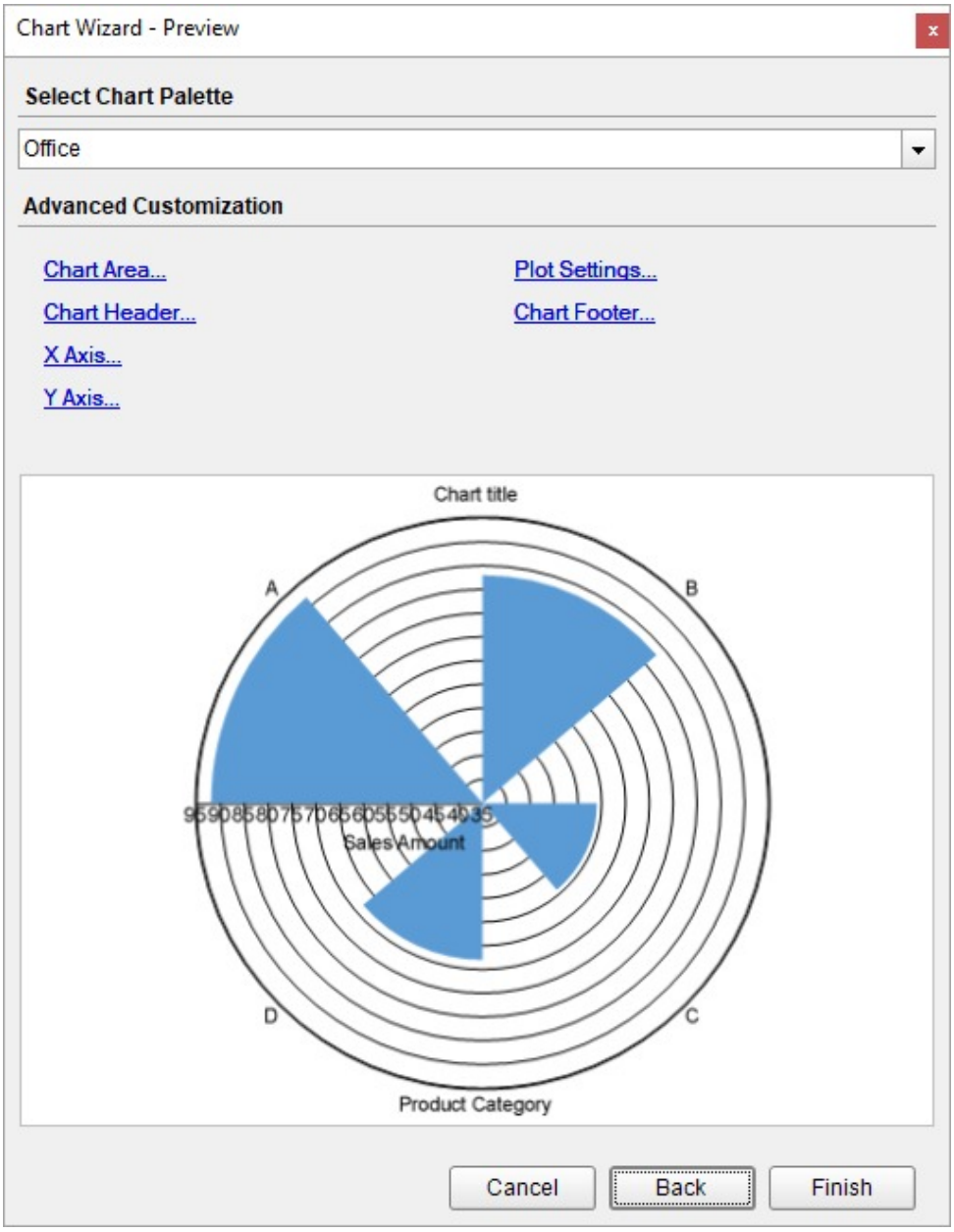
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Polar'.
3. Click **Next** to proceed. Here, we will define a data value to display the sales amount for different product categories in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Category**, set **Field** to `=[ProductCategory]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the product categories in increasing order of their sales amounts. So fill in the following settings:

Field	Settings
Sorting field	`=[SalesAmount]`
Sorting direction	Ascending
Sorting aggregate	Sum

3. Go to the **Encodings** page > **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Sales Channel]` to display sales amount for online, store, reseller, and catalog.
    2. Under **Grouping**, set **Group** to 'Cluster' since we want to display the subcategories in a cluster.
4. Go to the **Encodings** page > **Color** tab, add a new value and, set the **Expression** to `=[Sales Channel]`.
5. Click **OK** to save the changes.
6. With 'Plot-Plot1' selected, go to the Properties window and set the **StartAngle** property to '90' to set the start angle of the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the following properties.
    - **Format**: Currency (with 0 decimal places)
    - **Angle**: -90
4. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 100000
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Go to the **Scale** page and set the following properties.
    - **Scale Type**: Logarithmic
    - **Logarithmic base**: 10
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and add the following colors.

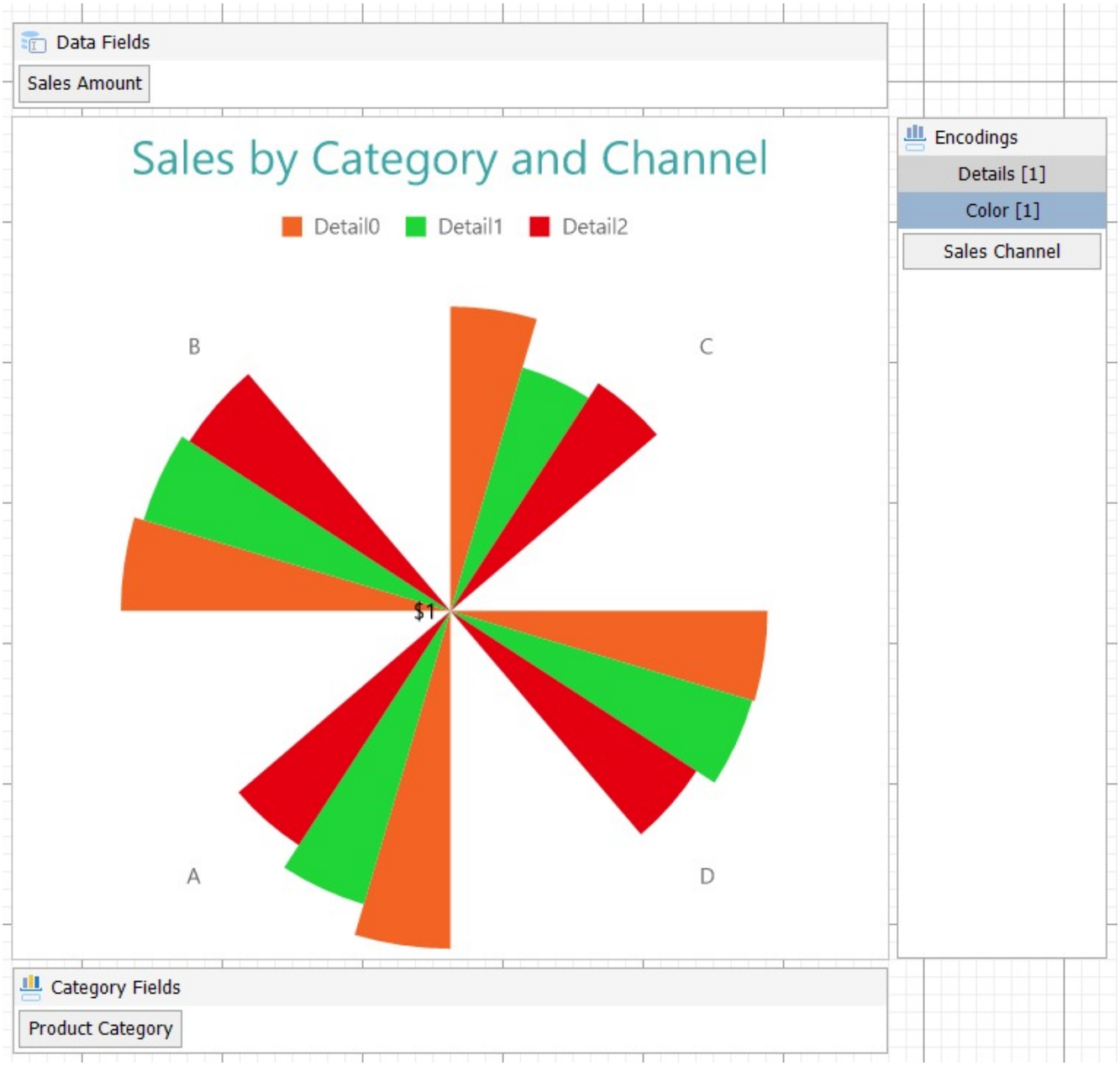
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category and Channel'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

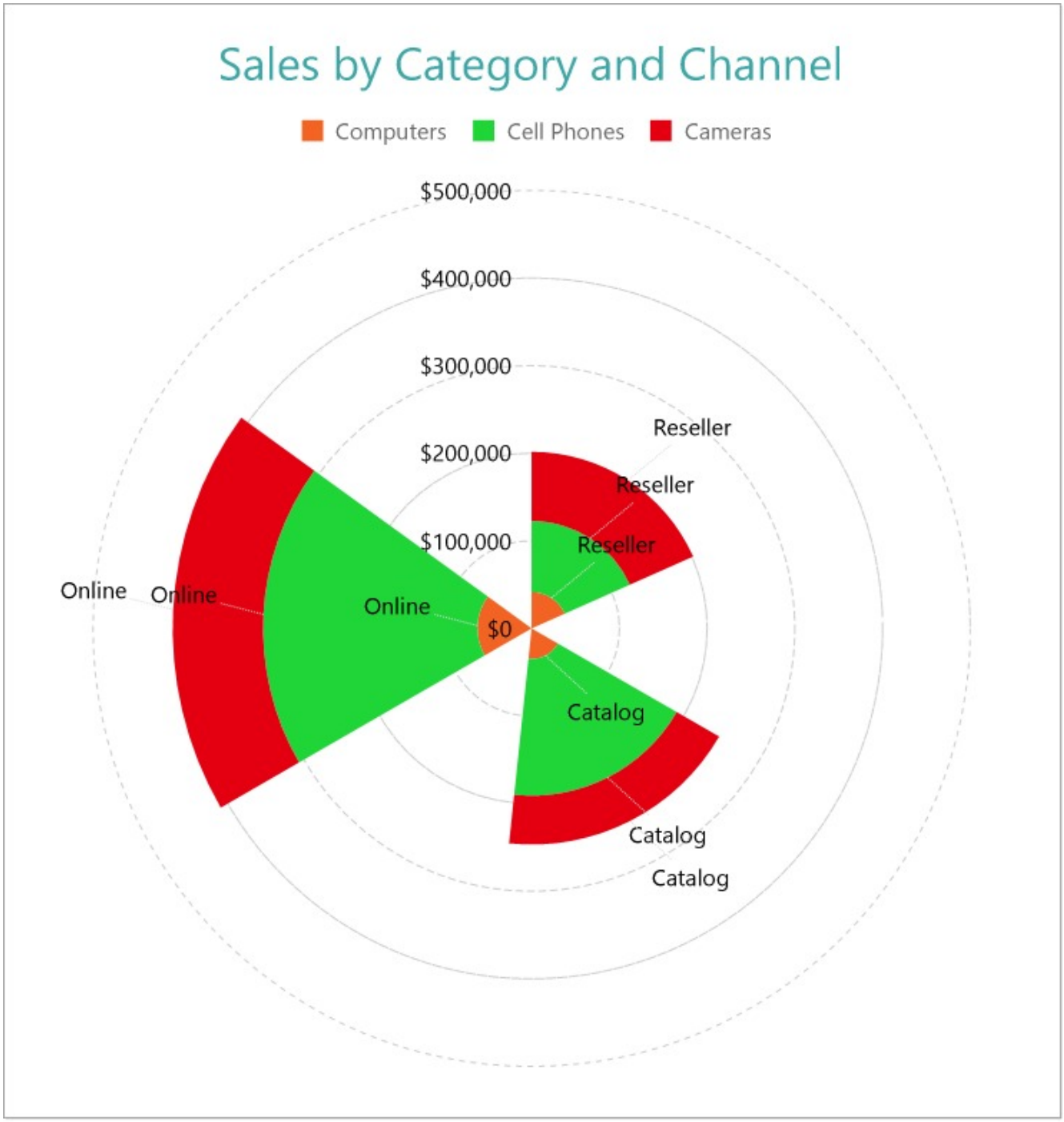


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Polar Chart

This walkthrough creates a Stacked Polar Chart. The chart displays the sales amount by product categories for each sales channel. In a stacked polar chart, the data values are broken down into subcategories and are stacked on top of each other. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.

2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource**  icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. On the same page, add two calculated fields:

Field Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

4. Go to the **Filters** page, and add these filters:

Expression	Operator	Value
`=[ChannelKey]`	NotEqual	1
`=[ProductKey]`	GreaterThan	338

5. Click **OK** to save the changes.

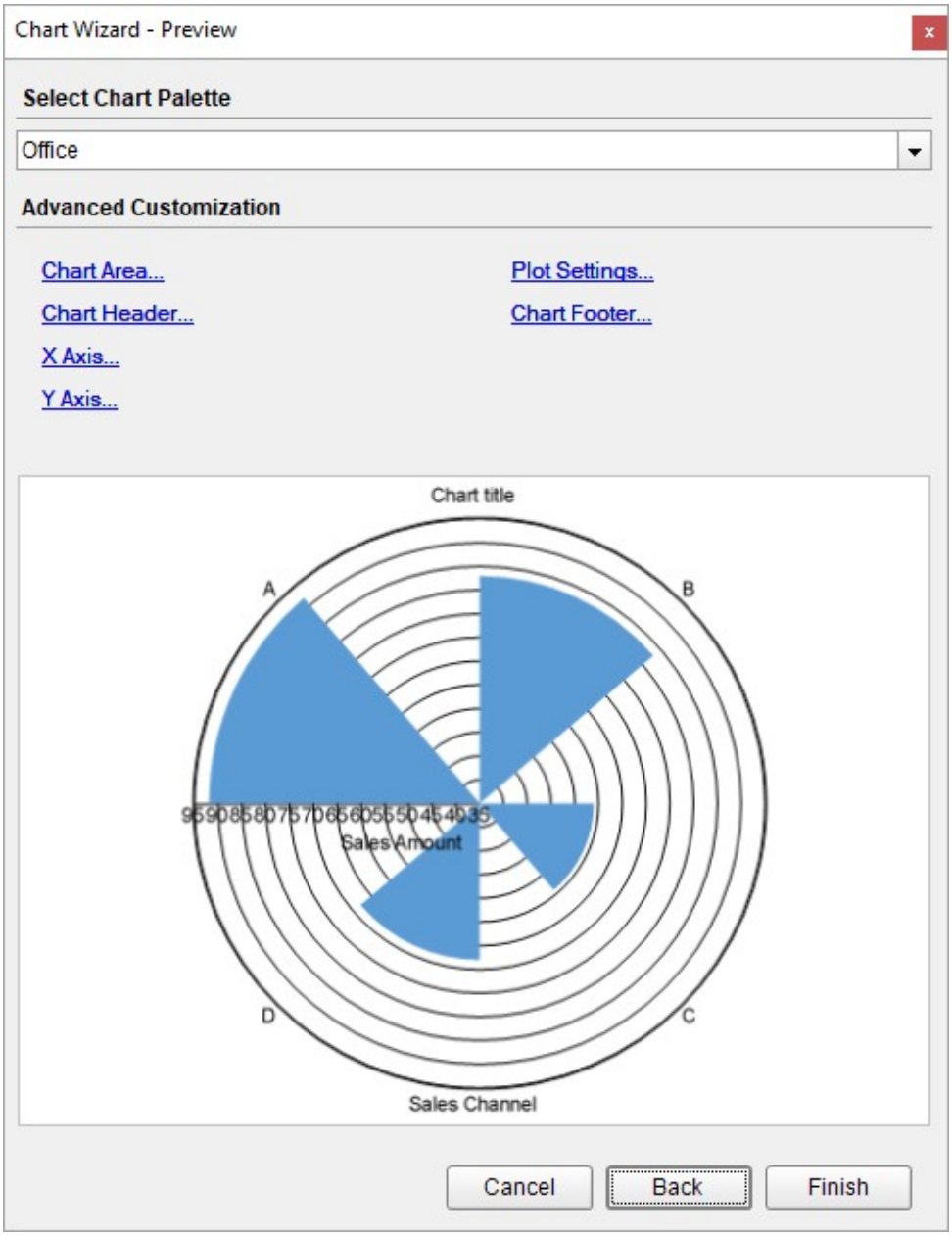
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Polar'.
3. Click **Next** to proceed. Here, we will define a data value to display the sales amount for different sales channels in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Category**, set **Field** to `=[Sales Channel]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the sales channels in increasing order of their sales amounts. So fill in the following settings:

Field	Settings
Sorting field	`=[SalesAmount]`
Sorting direction	Ascending
Sorting aggregate	Sum

3. Go to the **Encodings** page > **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Product Category]` to display the sales amount for cell phones, computers, cameras, tv and video, and audio.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to display the subcategories in a stack.
4. Then, navigate to the **Color** tab, add a new value and, set the **Expression** to `=[Product Category]`.
5. Go to the **Labels** page > **General** tab and set the following properties.
    - **Template**: `{categoryField.value}`
    - **Text Position**: Outside
    - **Offset**: 30
6. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
    - **Connecting Line > Style**: Dotted
    - **Connecting Line > Color**: Gainsboro
    - **Connecting Line > Width**: 0.25pt
7. Click **OK** to save the changes.
8. With 'Plot-Plot1' selected, go to the Properties window and set the **StartAngle** property to '90' to set the start angle of the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the following properties.
    - **Format**: Currency (with 0 decimal places)
    - **Angle**: -90
4. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid Interval**: 100000
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option to hide the labels.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.

6. Click **OK** to complete setting up the X-axis.

Chart Palette

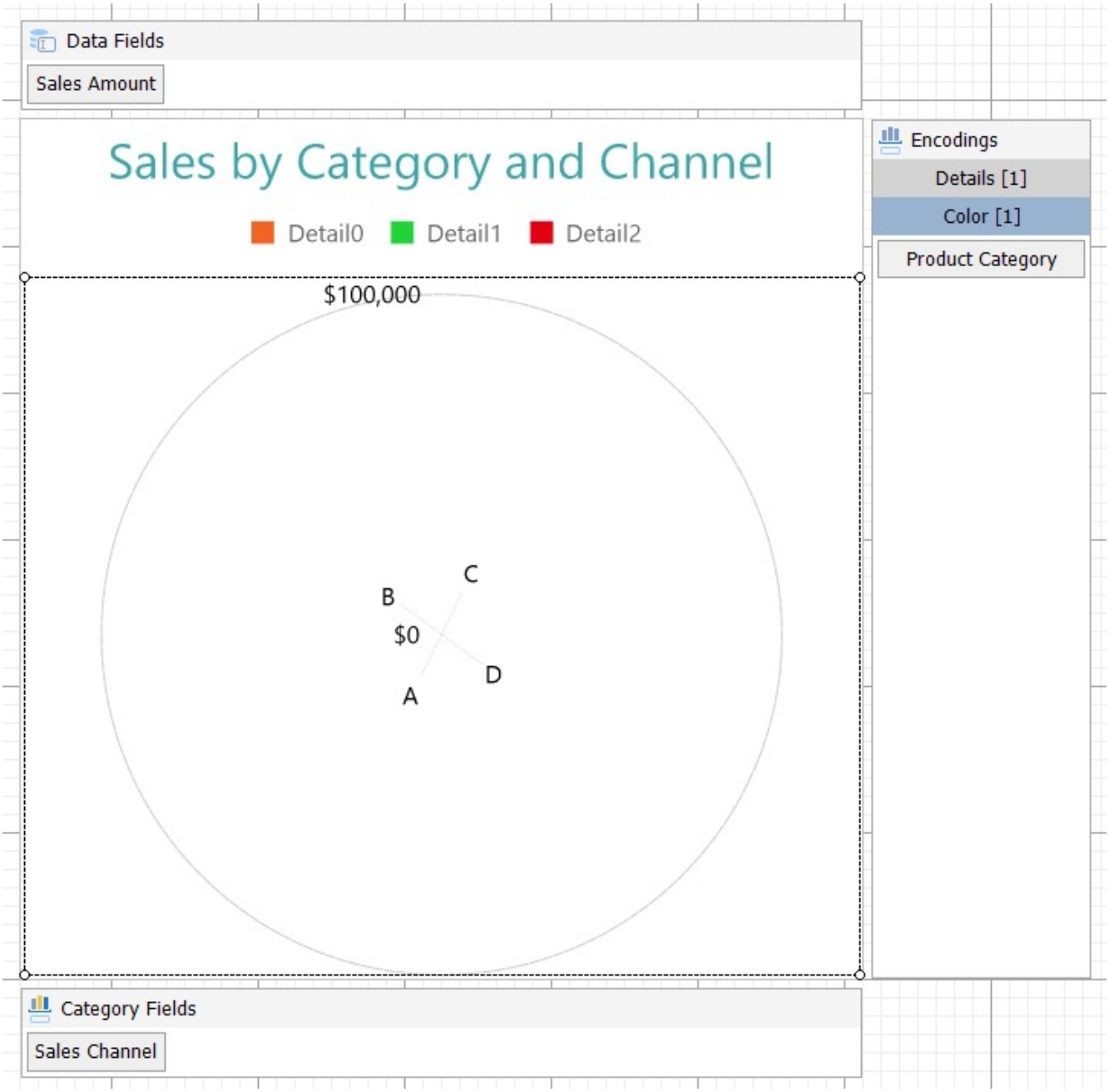
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Click **OK** to complete setting up the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Category and Channel'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

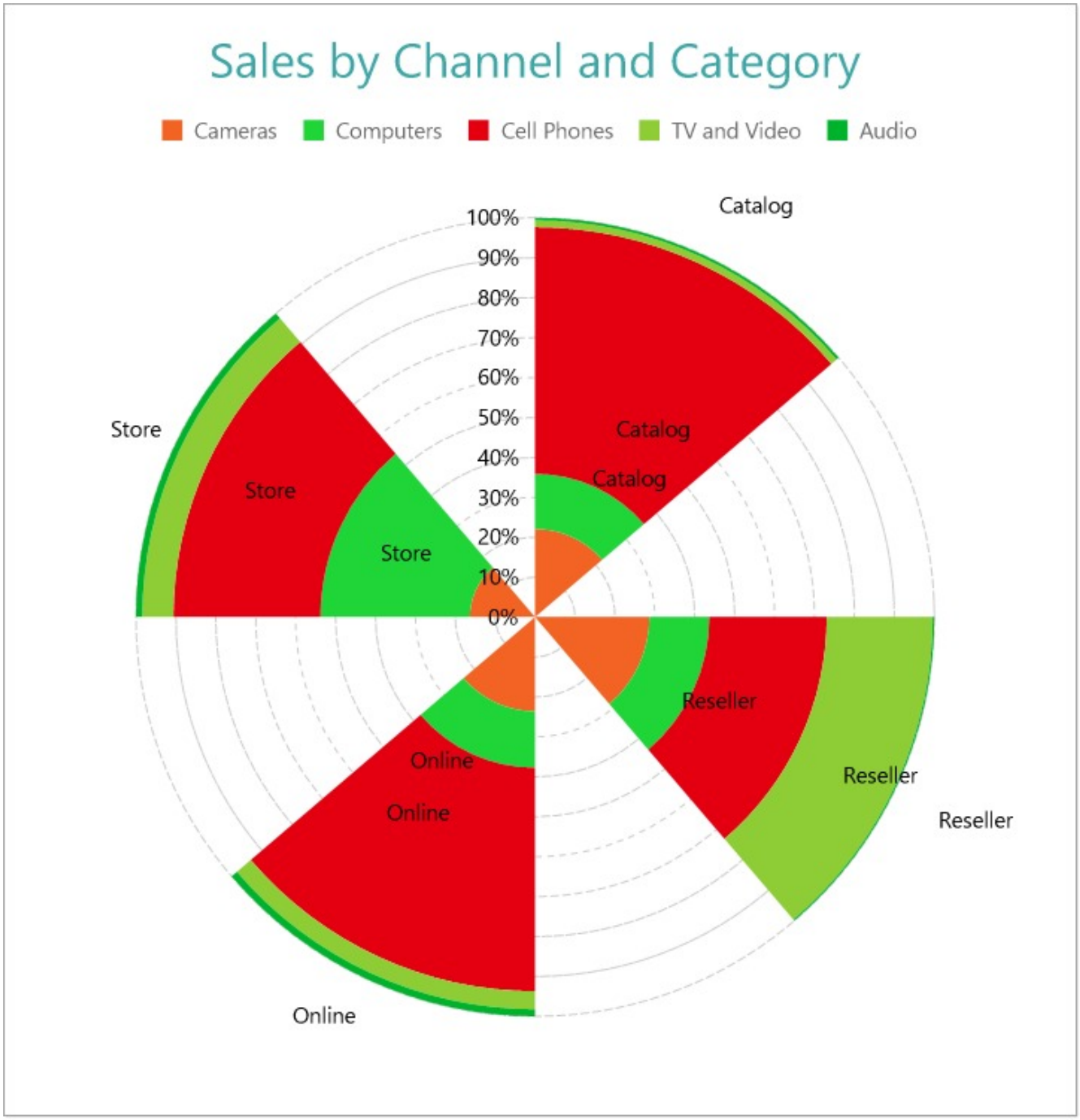


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Stacked Percentage Polar Chart

This walkthrough creates a Stacked Percentage Polar Chart. The chart displays the sales amount by sales channels for each product category. The stacked percentage polar chart is used to display the percentage values that each subcategory contributes within a category. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.

3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. On the same page, add two calculated fields:

Field Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`
Sales Channel	`=Switch([ChannelKey] = 1, "Store", [ChannelKey] = 2, "Online", [ChannelKey] = 3, "Catalog", [ChannelKey] = 4, "Reseller")`

4. Click **OK** to save the changes.

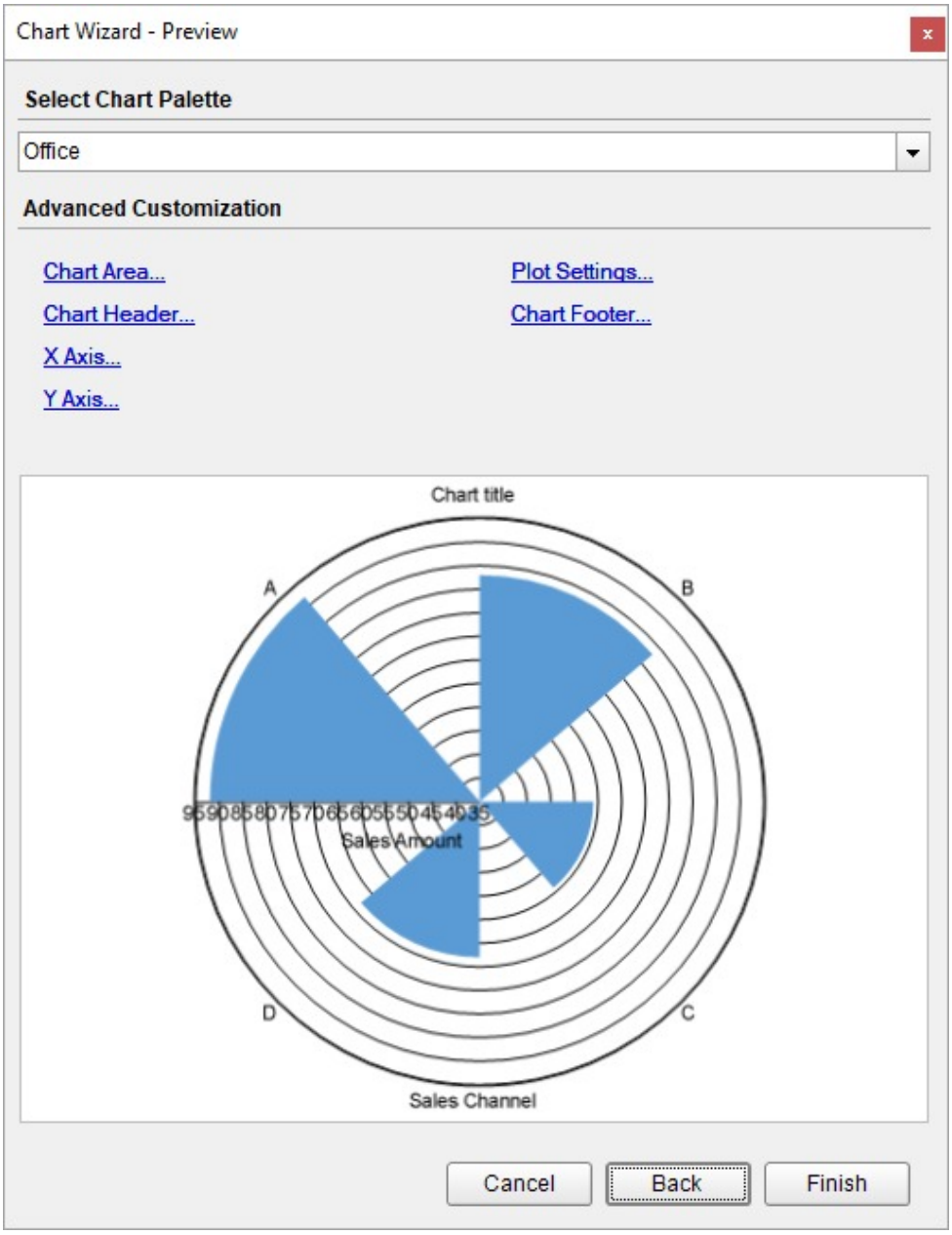
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Polar'.
3. Click **Next** to proceed. Here, we will define a data value to display the sales amount for different sales channels in the chart.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Field	Aggregate
`=[SalesAmount]`	Sum

5. In **Choose Data Category**, set **Field** to `=[Sales Channel]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page. Here, we will sort the sales channels in increasing order of their sales amounts. So fill in the following settings:

Field	Settings
Sorting field	`=[SalesAmount]`
Sorting direction	Ascending
Sorting aggregate	Sum

3. Go to the **Encodings** page > **Detail** tab, add a new value, and set its properties as below.
    1. Set **Expression** to `=[Product Category]` to display the sales amount for cell phones, computers, cameras, tv and video, and audio.
    2. Under **Grouping**, set **Group** to 'Stack' since we want to display the subcategories in a stack.
4. Then, navigate to the **Color** tab, add a new value and, set the **Expression** to `=[Product Category]`.
5. Go to the **Labels** page > **General** tab and set the following properties.
    - **Template**: `{categoryField.value}`
    - **Text Position**: Outside
    - **Offset**: 30
6. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
7. Click **OK** to save the changes.
8. With 'Plot-Plot1' selected, go to the Properties window and set the **StartAngle** property to '90' to set the start angle of the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Angle** to '-90'.
4. Then, navigate to the **Appearance** tab and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: Black
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Style**: Dashed
7. Go to the **Scale** page and set the **Scale Type** to 'Percentage'.
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option to hide the labels.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property**

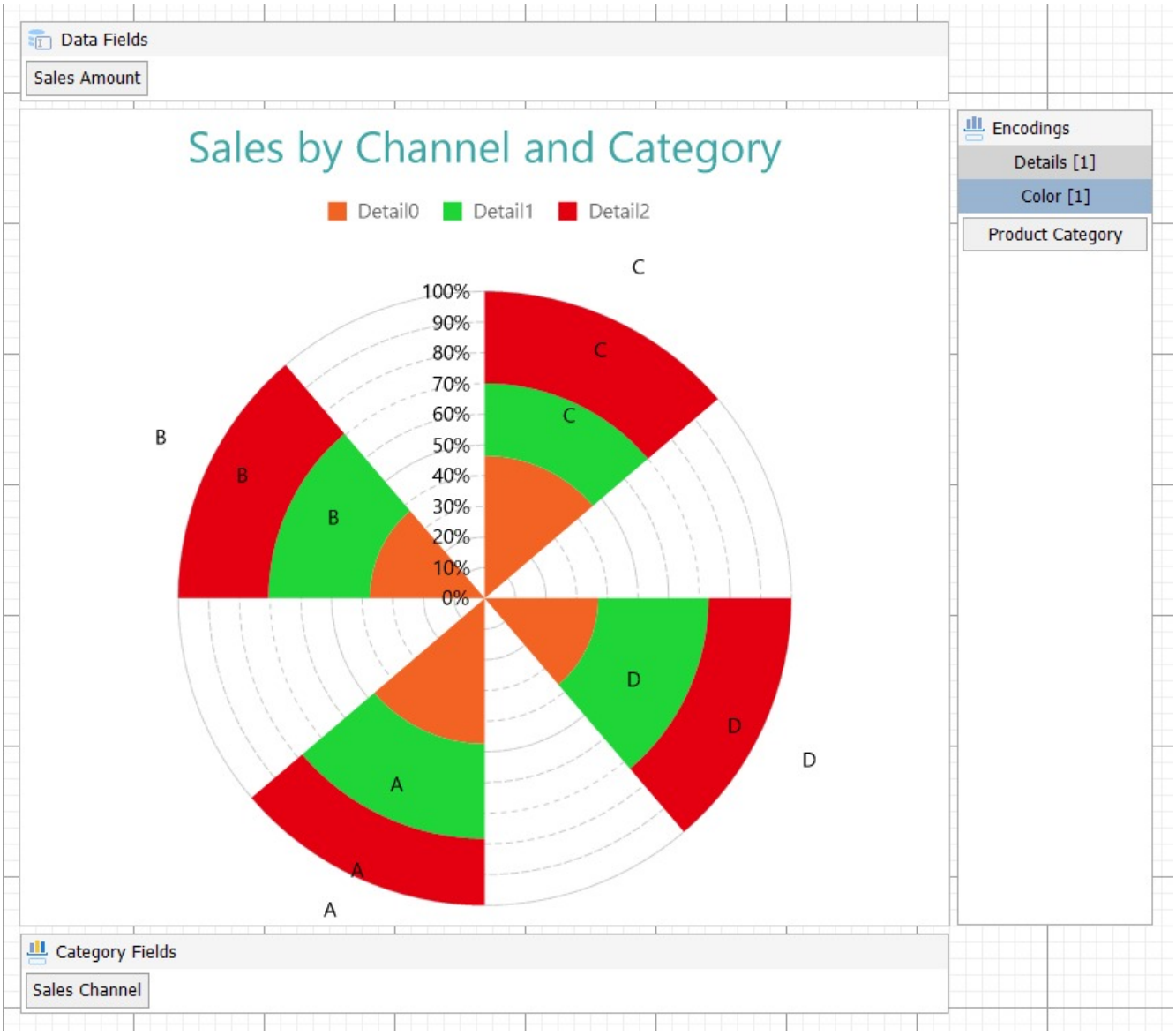
Dialog.
2. Go to the **Palette** page and add the following colors.
    - #f26324
    - #1fd537
    - #e40010
    - #8fcd37
    - #00b32c
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size:** 12pt
    - **Font > Color**: DimGray
4. Click **OK** to save the chart legend.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Sales by Channel and Category'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



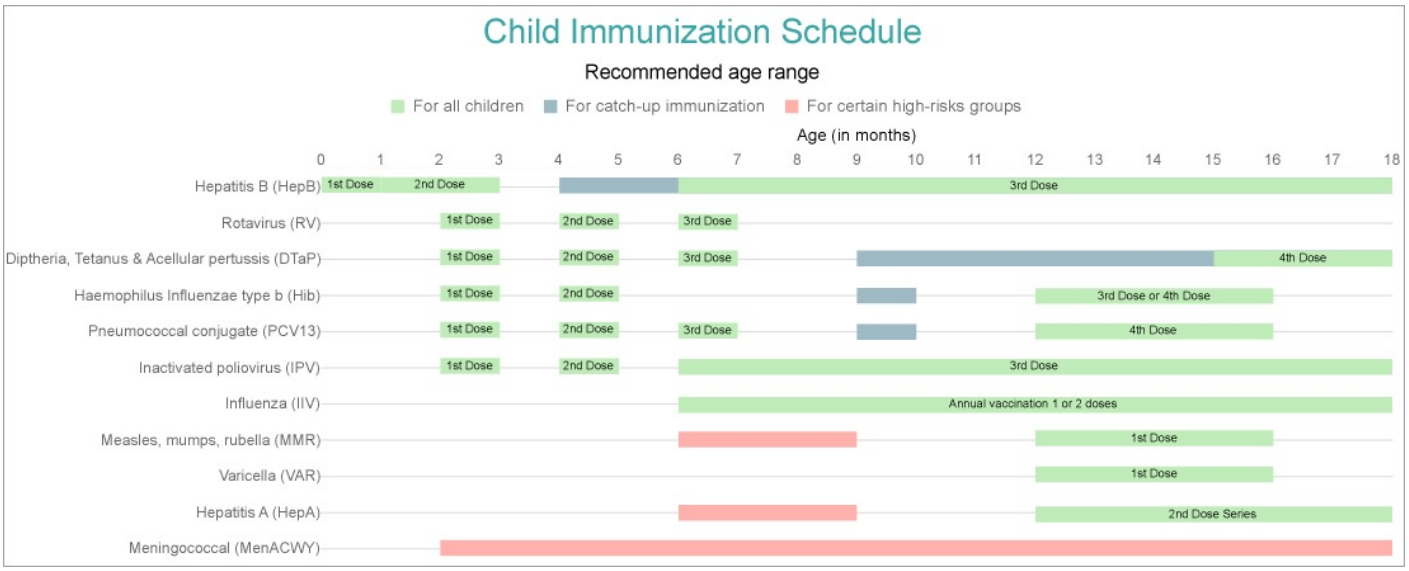
Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

5. Once you are done with configuring and customizing the chart, **F5** to preview the report.

Gantt Chart

A Gantt chart is a common chart type used in project management. Gantt plots provide the most useful ways of displaying activities (tasks or events) against time periods. Gantt charts can also be used to schedule big projects by breaking them into tasks and subtasks. The Gantt plot helps lay the tasks and subtasks on a timeline. In the Gantt chart, each task is listed on the chart's one side and each has a horizontal bar opposite it corresponding to the task length. By using the Gantt chart, you can find out how long each task will take and which tasks can overlap.

See Create Gantt Chart walkthrough to learn how the immunization schedule can be laid out using the Gantt chart.



A Gantt chart is made of essential elements, such as:

- Task: An activity involved in the project.
- Start: The date or time when the task execution begins.
- End: The date or time when the task execution ends.

Gantt Plot Properties

The plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the plot edge with the data label. Customize the appearance of

the connecting line using the following properties:
    - LineColor: Specify the color of the connecting line.
    - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
    - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
    - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
    - Auto: Hides labels that overlap.
    - Show: Shows the labels even if the labels overlap.
    - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the plot.
    - Center: Positions the data label text on the center of the chart.
    - Inside: Positions the data label text inside the chart.
    - Outside: Positions the data label text outside the chart.
    - Auto: The default setting, same as Outside for chart.

LineStyle

The line style for the border of the bars. Includes the following properties for the customization.

- **LineColor**: Specify the color of the border around the bars.
- **LineStyle:** Specify the line style of the border around the bars as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the bars.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

BarLineStyle

The BarLineStyle settings are applied when the **ShowBarLines** property is set to True.

- **LineColor:** Specify the color of the bar line.
- **LineStyle:** Specify the bar line style as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the bar line width.

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to:

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for charts.

Offset

Indicates the plot offset in percentage.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

ShowBarLines

Setting this to True shows the bar lines in the chart.

Design

BarSettings

The BarSettings specifies the bar-style settings.

- **NeckHeight**: Determines the bar neck height in percent.
- **BottomWidth**: Determines the Bottom width in percent.
- **Overlap**: Determines the Percentage bar overlap.
- **TopWidth**: Determines the Top width in percent.
- **Width**: Determines the bar width in percent.

Encodings

Category Encoding

The Category Encoding of a plot is a set of properties that determine the period over which the plot generates connected data points representing those above Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category. See the Create Gantt Chart walkthrough where **Values** is set to `[Vaccine]`.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Color Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the charts take the first item from the collection. See the Create Gantt Chart walkthrough where **Values** is set to `[Range]`.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This property determines whether to exclude the null details values in the dataset field.
- **Group**: This property determines the area subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked plot.
  - Cluster: You can use this value to configure subsections that overlap each other.
  - None: Equals to Cluster.
    See the Create Gantt Chart walkthrough where **Group** is set to 'Cluster'.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed. See the Create Gantt Chart walkthrough where **SortingField** is the `[Start]` field.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory. See the Create Gantt Chart walkthrough where **Values** is set to `[Dose]`.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Gantt charts, 'Complex' is acceptable.

Subfields

The Subfields property is the collection and takes a start field that goes to the lower value and an end field that goes to the upper value.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

See the Create Gantt Chart walkthrough where Type is set to 'Complex' and Subfields are set to `[Start]` and `[End]` fields.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

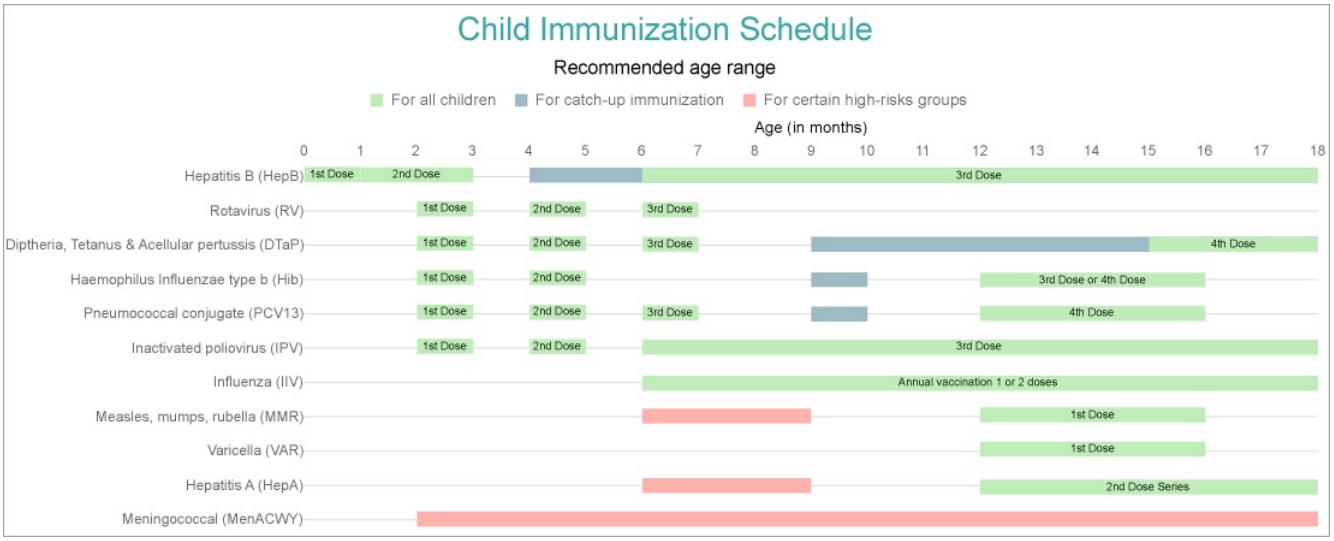
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Gantt Chart

This walkthrough creates a Gantt Chart to layout recommended child immunization schedule based on age in the United States. The schedule shows recommended vaccines at an acceptable age range, the number of doses, and highlights the recommendations for all children, catch-up vaccinations in case of missed vaccines, and other high-risk groups.



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data

```
[
 {
   "Vaccine": "Hepatitis B (HepB)",
   "Start": 0,
   "End": 1,
   "Dose": "1st Dose",
   "Range": "For all children"
 },
 {
   "Vaccine": "Hepatitis B (HepB)",
   "Start": 1,
   "End": 3,
   "Dose": "2nd Dose",
   "Range": "For all children"
 },
 {
   "Vaccine": "Hepatitis B (HepB)",
   "Start": 4,
   "End": 5,
   "Dose": "",
```

```
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Hepatitis B (HepB)",
  "Start": 6,
  "End": 18,
  "Dose": "3rd Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Hepatitis B (HepB)",
  "Start": 19,
  "End": 216,
  "Dose": "",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Rotavirus (RV)",
  "Start": 2,
  "End": 3,
  "Dose": "1st Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Rotavirus (RV)",
  "Start": 4,
  "End": 5,
  "Dose": "2nd Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Rotavirus (RV)",
  "Start": 6,
  "End": 7,
  "Dose": "3rd Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Diptheria, Tetanus & Acellular pertussis (DTaP)",
  "Start": 2,
  "End": 3,
  "Dose": "1st Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Diptheria, Tetanus & Acellular pertussis (DTaP)",
  "Start": 4,
  "End": 5,
  "Dose": "2nd Dose",
  "Range": "For all children"
},
```

```
{
  "Vaccine": "Diptheria, Tetanus & Acellular pertussis (DTaP)",
  "Start": 6,
  "End": 7,
  "Dose": "3rd Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Diptheria, Tetanus & Acellular pertussis (DTaP)",
  "Start": 9,
  "End": 12,
  "Dose": " ",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Diptheria, Tetanus & Acellular pertussis (DTaP)",
  "Start": 15,
  "End": 18,
  "Dose": "4th Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Diptheria, Tetanus & Acellular pertussis (DTaP)",
  "Start": 18.5,
  "End": 48,
  "Dose": " ",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Diptheria, Tetanus & Acellular pertussis (DTaP)",
  "Start": 48,
  "End": 84,
  "Dose": "5th Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Haemophilus Influenzae type b (Hib)",
  "Start": 2,
  "End": 3,
  "Dose": "1st Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Haemophilus Influenzae type b (Hib)",
  "Start": 4,
  "End": 5,
  "Dose": "2nd Dose",
  "Range": "For all children"
},
{
```

```
  "Vaccine": "Haemophilus Influenzae type b (Hib)",
  "Start": 9,
  "End": 10,
  "Dose": "  ",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Haemophilus Influenzae type b (Hib)",
  "Start": 12,
  "End": 16,
  "Dose": "3rd Dose or 4th Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Haemophilus Influenzae type b (Hib)",
  "Start": 18,
  "End": 36,
  "Dose": "",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Haemophilus Influenzae type b (Hib)",
  "Start": 48,
  "End": 84,
  "Dose": " ",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Haemophilus Influenzae type b (Hib)",
  "Start": 84,
  "End": 216,
  "Dose": " ",
  "Range": "For certain high-risks groups"
},
{
  "Vaccine": "Pneumococcal conjugate (PCV13)",
  "Start": 2,
  "End": 3,
  "Dose": "1st Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Pneumococcal conjugate (PCV13)",
  "Start": 4,
  "End": 5,
  "Dose": "2nd Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Pneumococcal conjugate (PCV13)",
  "Start": 6,
```

```
  "End": 7,
  "Dose": "3rd Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Pneumococcal conjugate (PCV13)",
  "Start": 9,
  "End": 10,
  "Dose": "  ",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Pneumococcal conjugate (PCV13)",
  "Start": 12,
  "End": 16,
  "Dose": "4th Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Pneumococcal conjugate (PCV13)",
  "Start": 19,
  "End": 84,
  "Dose": "",
  "Range": "For catch-up immunization"
},
{
  "Vaccine": "Pneumococcal conjugate (PCV13)",
  "Start": 84,
  "End": 216,
  "Dose": " ",
  "Range": "For certain high-risks groups"
},
{
  "Vaccine": "Inactivated poliovirus (IPV)",
  "Start": 2,
  "End": 3,
  "Dose": "1st Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Inactivated poliovirus (IPV)",
  "Start": 4,
  "End": 5,
  "Dose": "2nd Dose",
  "Range": "For all children"
},
{
  "Vaccine": "Inactivated poliovirus (IPV)",
  "Start": 6,
  "End": 18,
```

```
  "Dose": "3rd Dose",
  "Range": "For all children"
 },
 {
  "Vaccine": "Inactivated poliovirus (IPV)",
  "Start": 18.1,
  "End": 48,
  "Dose": "",
  "Range": "For catch-up immunization"
 },
 {
  "Vaccine": "Inactivated poliovirus (IPV)",
  "Start": 48,
  "End": 84,
  "Dose": "4th Dose",
  "Range": "For all children"
 },
 {
  "Vaccine": "Inactivated poliovirus (IPV)",
  "Start": 84,
  "End": 204,
  "Dose": "",
  "Range": "For catch-up immunization"
 },
 {
  "Vaccine": "Influenza (IIV)",
  "Start": 6,
  "End": 18,
  "Dose": "Annual vaccination 1 or 2 doses ",
  "Range": "For all children"
 },
 {
  "Vaccine": "Influenza (IIV)",
  "Start": 18,
  "End": 19,
  "Dose": " ",
  "Range": "For all children"
},
 {
  "Vaccine": "Influenza (IIV)",
  "Start": 19,
  "End": 84,
  "Dose": "Annual vaccination 1 or 2 doses",
  "Range": "For all children"
 },
 {
  "Vaccine": "Influenza (IIV)",
  "Start": 96,
  "End": 216,
  "Dose": "Annual vaccination 1 dose only",
  "Range": "For all children"
```

```
 },
 {
  "Vaccine": "Measles, mumps, rubella (MMR)",
  "Start": 6,
  "End": 9,
  "Dose": " ",
  "Range": "For certain high-risks groups"
},
 {
   "Vaccine": "Measles, mumps, rubella (MMR)",
   "Start": 12,
   "End": 16,
   "Dose": "1st Dose",
   "Range": "For all children"
 },
 {
   "Vaccine": "Measles, mumps, rubella (MMR)",
   "Start": 18.1,
   "End": 48,
   "Dose": "  ",
   "Range": "For catch-up immunization"
 },
 {
   "Vaccine": "Measles, mumps, rubella (MMR)",
   "Start": 48,
   "End": 84,
   "Dose": "2nd Dose",
   "Range": "For all children"
 },
 {
   "Vaccine": "Measles, mumps, rubella (MMR)",
   "Start": 84,
   "End": 216,
   "Dose": "",
   "Range": "For catch-up immunization"
 },
 {
   "Vaccine": "Varicella (VAR)",
   "Start": 12,
   "End": 16,
   "Dose": "1st Dose",
   "Range": "For all children"
 },
 {
   "Vaccine": "Varicella (VAR)",
   "Start": 18.1,
   "End": 48,
   "Dose": "  ",
   "Range": "For catch-up immunization"
 },
```

```
 {
   "Vaccine": "Varicella (VAR)",
   "Start": 48,
   "End": 84,
   "Dose": "2nd Dose",
   "Range": "For all children"
 },
 {
   "Vaccine": "Varicella (VAR)",
   "Start": 84,
   "End": 216,
   "Dose": "",
   "Range": "For catch-up immunization"
 },
 {
  "Vaccine": "Hepatitis A (HepA)",
  "Start": 6,
  "End": 9,
  "Dose": "  ",
  "Range": "For certain high-risks groups"
},
 {
   "Vaccine": "Hepatitis A (HepA)",
   "Start": 12,
   "End": 18,
   "Dose": "2nd Dose Series",
   "Range": "For all children"
 },
 {
   "Vaccine": "Hepatitis A (HepA)",
   "Start": 18,
   "End": 23,
   "Dose": " ",
   "Range": "For all children"
 },
 {
   "Vaccine": "Hepatitis A (HepA)",
   "Start": 24,
   "End": 216,
   "Dose": "",
   "Range": "For catch-up immunization"
 },

 {
  "Vaccine": "Meningococcal (MenACWY)",
  "Start": 2,
  "End": 132,
  "Dose": " ",
  "Range": "For certain high-risks groups"
},
 {
```

```
  "Vaccine": "Meningococcal (MenACWY)",
  "Start": 132,
  "End": 156,
  "Dose": "1st Dose",
  "Range": "For all children"
 },
 {
  "Vaccine": "Meningococcal (MenACWY)",
  "Start": 156,
  "End": 180,
  "Dose": "   ",
  "Range": "For catch-up immunization"
 },
 {
  "Vaccine": "Meningococcal (MenACWY)",
  "Start": 180,
  "End": 191,
  "Dose": "2nd Dose",
  "Range": "For all children"
 },
 {
  "Vaccine": "Meningococcal (MenACWY)",
  "Start": 191,
  "End": 216,
  "Dose": "  ",
  "Range": "For catch-up immunization"
 }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the Chart Wizard, from the **Select Data Set** dialog, select a suitable Dataset name from the dropdown menu.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'DataSet1' and the **Chart Type** as 'Gantt'.

3. Click **Next** to proceed.
4. Under **Choose Data Values**, add a new data value, and set its properties as below.

Start Field	End Field
`=[Start]`	`=[End]`

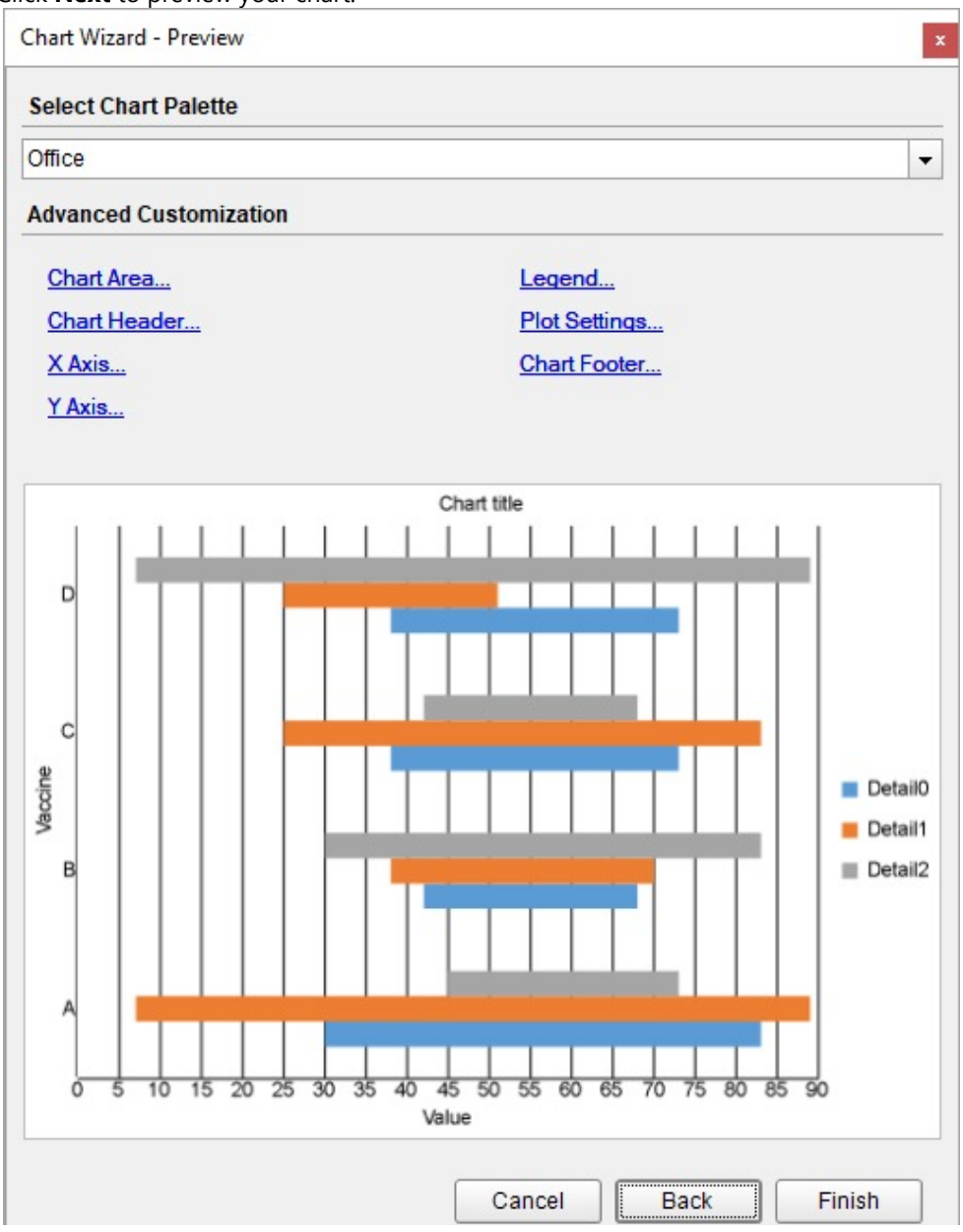
5. Under the Choose Data Categories option, set the following fields:

Field	SortDirection
`=[Vaccine]`	None

6. Under the Choose Data Subcategories option, set the following fields:

Field	SortDirection
`=[Dose]`	None

7. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Data Fields** page and ensure that **Type** is set to 'Complex'.
3. Go to the **Categories** page, with the Expression to `=[Vaccine]`, remove the **Sorting field**.
4. Go to the **Encodings** page > **Detail** tab and set the following properties:
    - Expression: `=[Dose]`
    - Group: Cluster
    - Sorting field: `=[Start]`
    - Exclude nulls: Check-on
5. Go to the **Color** tab and set the Expression to `=[Range]`.
6. Go to the **Labels** page > **General** and set following properties:
    - Template: `{detailFields.value}`
    - Text Position: Center
    - Overlapping Labels Mode: Auto
7. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and set the following properties.
    - Title: Age (in months)
    - Font > Size: 10pt
    - Padding > Top: 4pt
    - Padding > Bottom: 4pt
3. Go to the **Layout** page and set **Position** to 'Far'.
4. Go to the **Labels** page > **General**, check-on **Show Labels** and set **Format** to 'n0'.
5. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - Font > Size: 9pt
    - Font > Color: DimGray
6. Go to the **Line** page and check the **Show Line** option. Set the color to 'Gainsboro'.
7. Go to the **Major Gridline** page and set **Grid appearance > Show Grid**: Uncheck
8. Go to the **Scale** page and set the following properties.
    - Scale Type: Linear
    - Minimum Scale value: 0
    - Maximum Scale value: 18
9. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.

2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Layout** page and check the **Position** > **Reversed** option to reverse the axis to appear from top to bottom.
4. Go to the **Labels** page > **General** tab and check the **Show Labels** option to show the data labels.
5. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - Font > Size: 9pt
    - Font > Color: DimGray
6. Go to the **Line** page and uncheck the **Show Line** option.
7. Go to the **Minor Gridline** page and uncheck the **Show Grid** option to hide minor gridlines.
8. Go to the **Major Gridline** page, check the **Show Grid** option, and set the **Color** to 'GainsBoro'.
9. Click **OK** to complete setting up the X-axis.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and add these custom palette colors, #bfecb8, #9ebac4 and #ffb1ae.
3. Click **OK** to complete setting up the chart palette.

Legend - Color

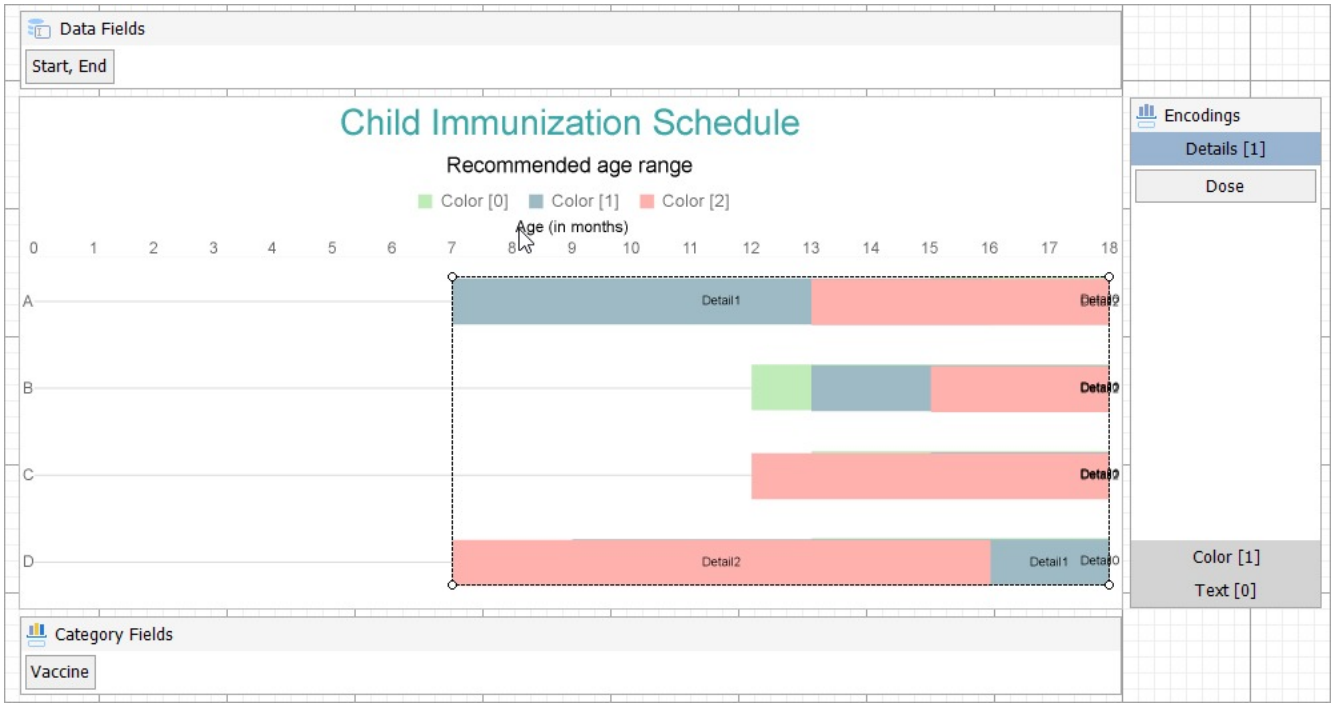
1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Title** page and set the title to 'Recommended age range' and **Font > Size** to '12pt'.
3. Go to the **Appearance** page and set the following properties.
    - **Font > Size**: 10pt
    - **Font > Color**: DimGray
4. Go to the **Layout** page and set the following properties.
    - **Position**: Top
    - **Orientation**: Horizontal
5. Click **OK** to complete setting up the Legend.

Chart

1. From the Report Explorer, select Chart.
2. In the Properties panel, go to **BarSettings** > **Overlap** and set it to some percentage, say '98%'.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Child Immunization Schedule'.
3. Set the Horizontal Alignment to Center and Vertical Alignment to Middle.
4. Go to the **Font** page and set the properties as below.
    - **Size**: 20pt
    - **Color**: #3da7a8
5. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

6. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Funnel and Pyramid Charts

Funnel Chart

A Funnel chart helps in visualizing sequential stages in a linear process such as the flow of users via a sales or business process. The chart takes its name from the funnel shape it has - which starts with a broad head and ends with a narrow neck. The number of users at different process stage are indicated by the width of the funnel.

Funnel charts can help identify potential problem areas in processes where it is noticeable at what stages and rate the values decrease. A real-time use case for Funnel Charts can be an order fulfillment process scenario, which tracks the number of orders getting across a stage, such as orders received, processed, approved, released, shipped, completed, and delivered. Each stage in such a process represents a percentage or proportion of the total. That's the reason why the chart takes a funnel shape with the first stage being the largest and the consecutive stage smaller than the predecessor.

A Funnel chart arranges related values stacked on top of one another.

The Create Funnel Chart walkthrough showcases plotting parameters like Visitors, Opportunities, Total Quality Organization, and Customers.

Pyramid Chart

A Pyramid chart is an easy-to-understand chart type, which can be used to depict directional workflows or hierarchies. The Pyramid chart is beneficial for organizing and visualizing data and explaining internal business management structures or workflows. A Pyramid chart can quickly communicate processes, hierarchies, and relationships between categories of information.

A Pyramid chart arranges each series vertically stacked one over the other. The Create Pyramid Chart walkthrough showcases plotting the Sales Amount for each Product Category per Sales Channel.

Funnel and Pyramid Plot Properties

The plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of the labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the plot.
  - Center: Positions the data label text on the center of the chart.
  - Inside: Positions the data label text inside the chart.
  - Outside: Positions the data label text outside the chart.
  - Auto: The default setting, same as Outside for chart.

LineStyle

The line style for the borders. Includes the following properties for the customization.

- **LineColor**: Specify the color of the border around the sectors.
- **LineStyle:** Specify the line style of the border around the sectors as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the sectors.

Name

The name of the plot. By default, a chart containing a single plot has the plot name 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to:

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for Pyramid charts.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

UnpivotData

Determines whether to display multiple data fields as a single data field. By default, the property is set to 'False'.

Design

BarSettings

The BarSettings specifies the bar-style settings.

- **NeckHeight**: Determines the bar neck height in percent.
- **BottomWidth**: Determines the Bottom width in percent.
- **Overlap**: Determines the Percentage bar overlap.
- **TopWidth**: Determines the Top width in percent.
- **Width**: Determines the bar width in percent.

Encodings

Color Encoding

The Colors Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the Pyramid charts take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This flag indicates whether dataset records with undefined details should be exempted from the visualization.
- **Group**: This property determines the area subsection arrangement of the plot.
  - Stack: You can use this value to configure a Stacked plot.
  - Cluster: You can use this value to configure area subsections that overlap each other.
  - None: Equals to Cluster.
    In Funnel and Pyramid plots, 'Stack' is the suitable option to represent the group.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: It defines the order in which the subcategories are displayed.
- **Values**: The Values could be one or more bound field references, and the bound DataSet records with the same values of these fields come under the same subcategory.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Pyramid and Funnel charts, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the Pyramid and Funnel charts take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional

info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Orders Count: {Order ID.value}
Orders Value: {valueField.value}

Value

A field, constant or expression to be displayed.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

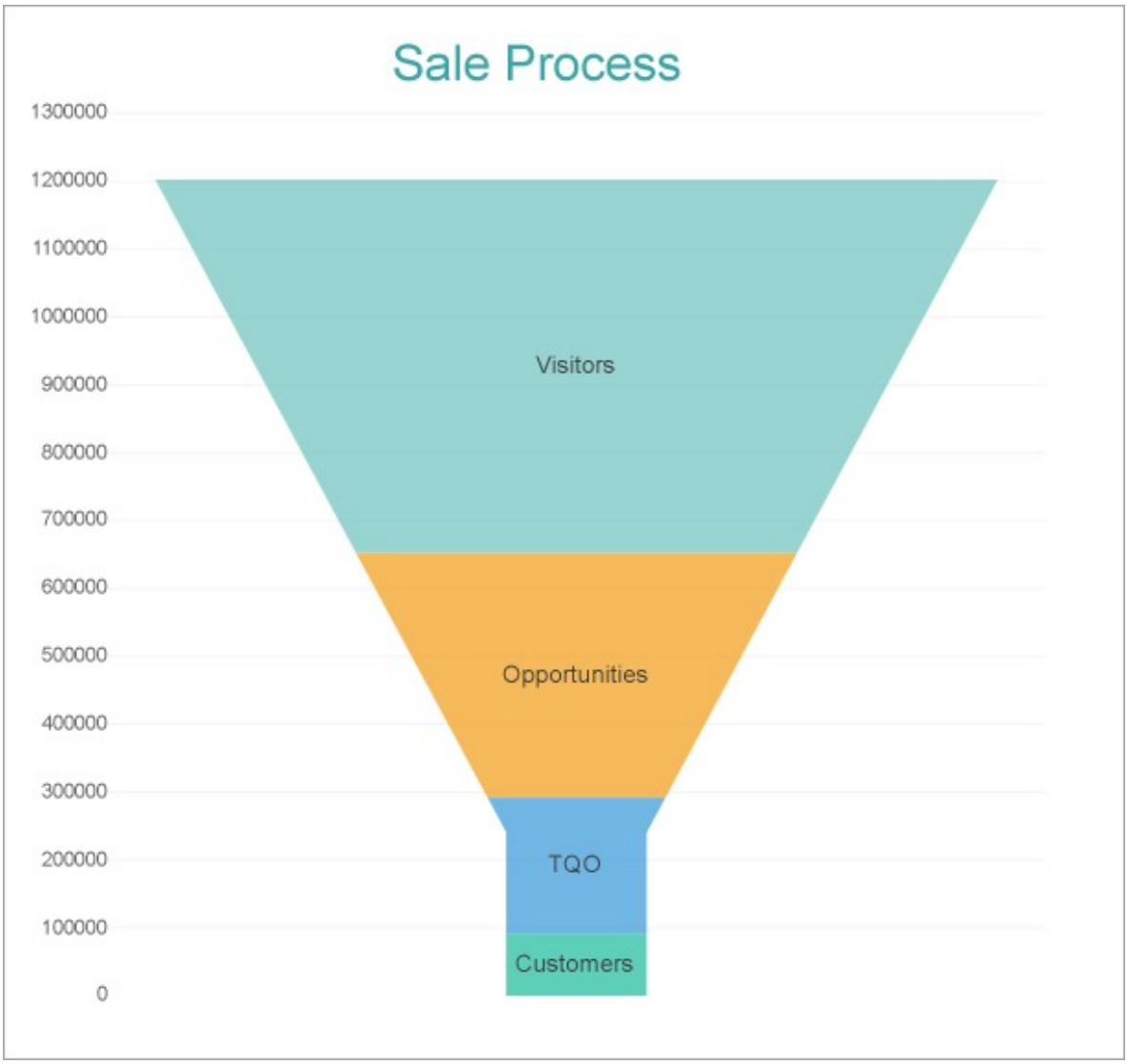
Value

A field, constant, or expression to be displayed.

Create Funnel Chart

This walkthrough creates a Funnel Chart, which shows how various related values are stacked or arranged on top of one another. For example, the chart represents a Sales Funnel, which depicts many factors like Visitor count, Opportunities, TQO (Total Quality Organization) and Customers in an organization. By Visitor count, we mean the total number of visitors at a website. By Opportunities, we imply the number of potential visitors that can be converted to successful customers. Further, TQO is an important aspect, which drives customer satisfaction and is essential for the growth and survival of the organization.

The final chart appears like this, which looks very similar to an ideal marketing sales funnel in its symmetrical form – a paradigm of efficiency that makes sense of a website's visitors, opportunities, TQO and customers.



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data

```
{
   "data":{
      "values":[
```

```
        {
            "Type":"Customers",
            "Value":91658
        },
        {
            "Type":"TQO",
            "Value":200400
        },
        {
            "Type":"Opportunities",
            "Value":360470
        },
        {
            "Type":"Visitors",
            "Value":550000
        }
    ]
  }
}
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource**  icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'SaleStages'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

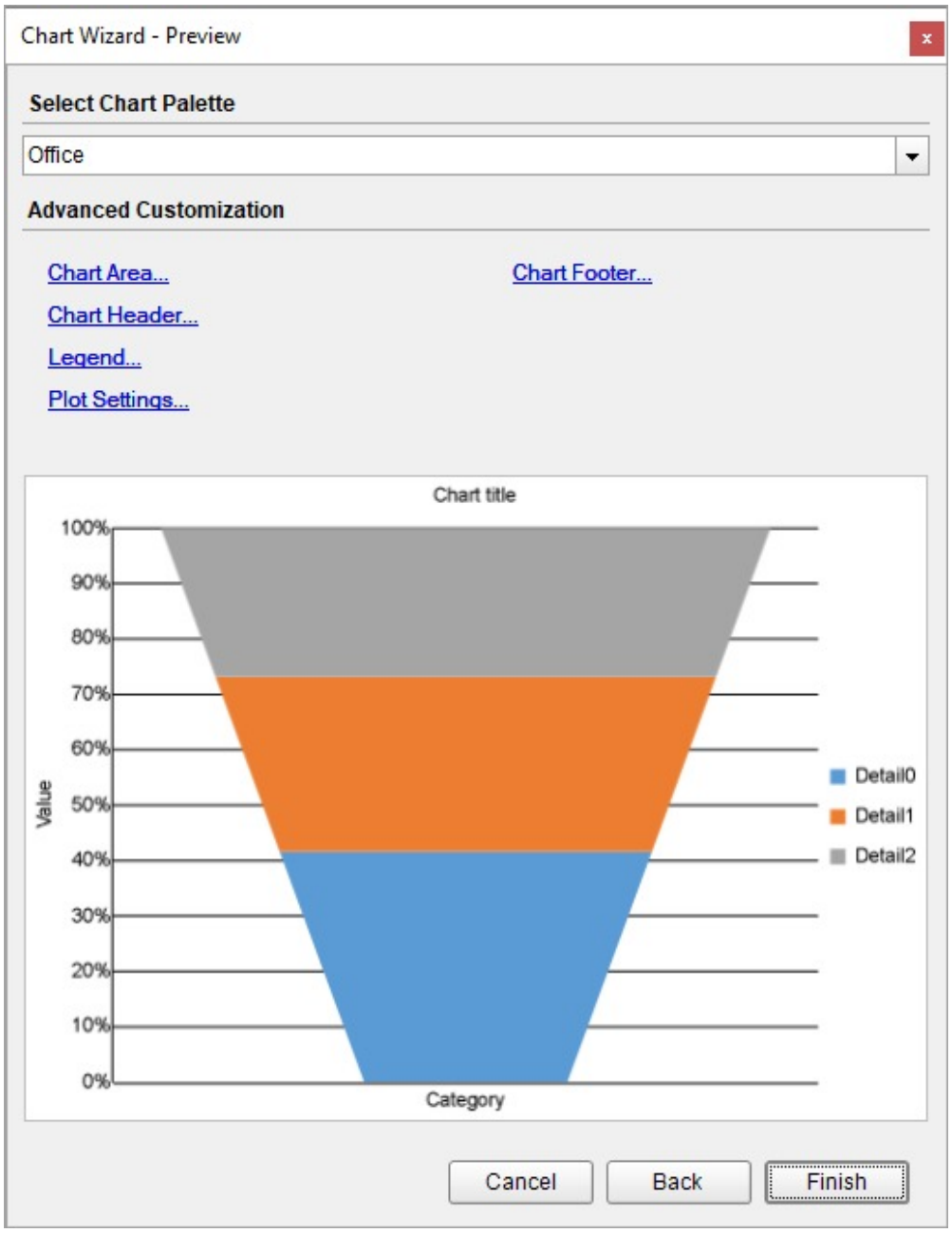
Query
`$.data.values[*]`

3. Click **OK** to save the changes and open the **DataSet** dialog.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'SaleStages' and the **Chart Type** as 'Funnel'.
3. Click **Next** to proceed. Here, we will define a data value to display.
4. Under **Choose Data Values**, add a new data value, and set the **Field** to `=[Value]`.
5. In **Choose Data Category,** set the **Field** to `=[Type]` and the **Break-down method** to 'Stacked'.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.

2. Go to the **Encodings** page > **Details** tab, ensure that the **Expression** is set to `=[Type]`. Remove any value from **Sorting field**.
3. Go to **Appearance** page and set the following properties.
    - Bar Settings > Bottom Width: 15%
    - Bar Settings > Neck Height: 20%.
    - Common > Opacity: 70%
4. Go to the **Labels** page > **General** tab and set following properties:
    - Template: `{detailFields.value}`
    - Text Position: Center
    - Overlapping Labels Mode: Auto
5. Go to the **Labels** page > **Appearance** tab and set following properties:
    - Font > Size: 10pt
    - Font > Color: DimGray
6. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the **Format** to 'Default'.
4. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 8pt
    - **Font > Color**: DimGray
5. Go to the **Line** page and uncheck the **Show Line** option.
6. Go to the **Major Gridline** page and set the following properties.
    - **Grid appearance > Show Grid**: Check-on
    - **Grid appearance > Width**: 1pt
    - **Grid appearance > Color**: WhiteSmoke
    - **Grid appearance > Style**: Solid
7. Go to the **Scale** page and set **Scale Type** as 'Linear'.
8. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option to hide the data labels.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Click **OK** to complete setting up the X-axis.

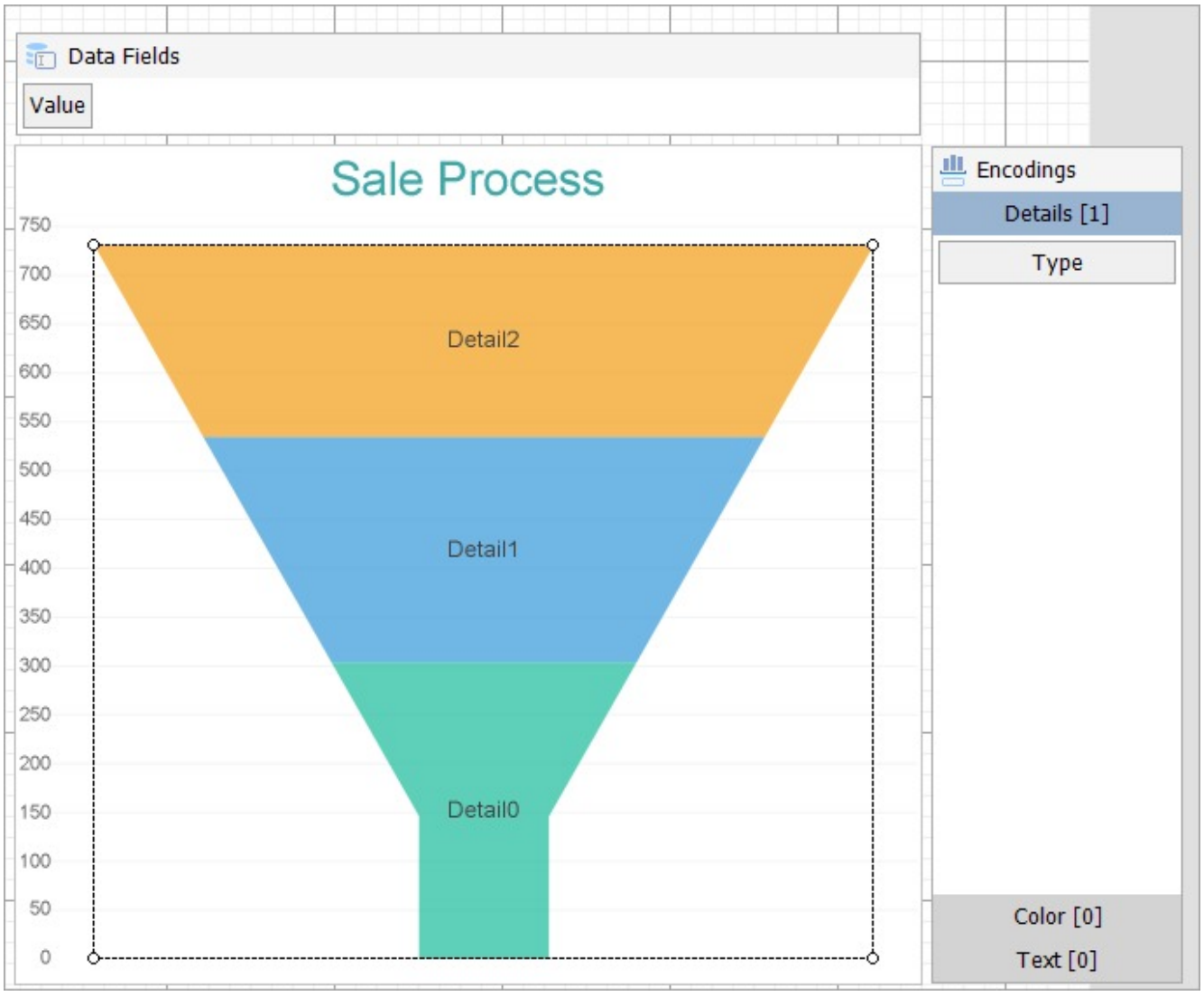
Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and add the 'Flatly' palette.
3. Click **OK** to complete setting up the chart palette.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**

2. Go to the **General** page and set **Title** to 'Sale Process'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 20pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

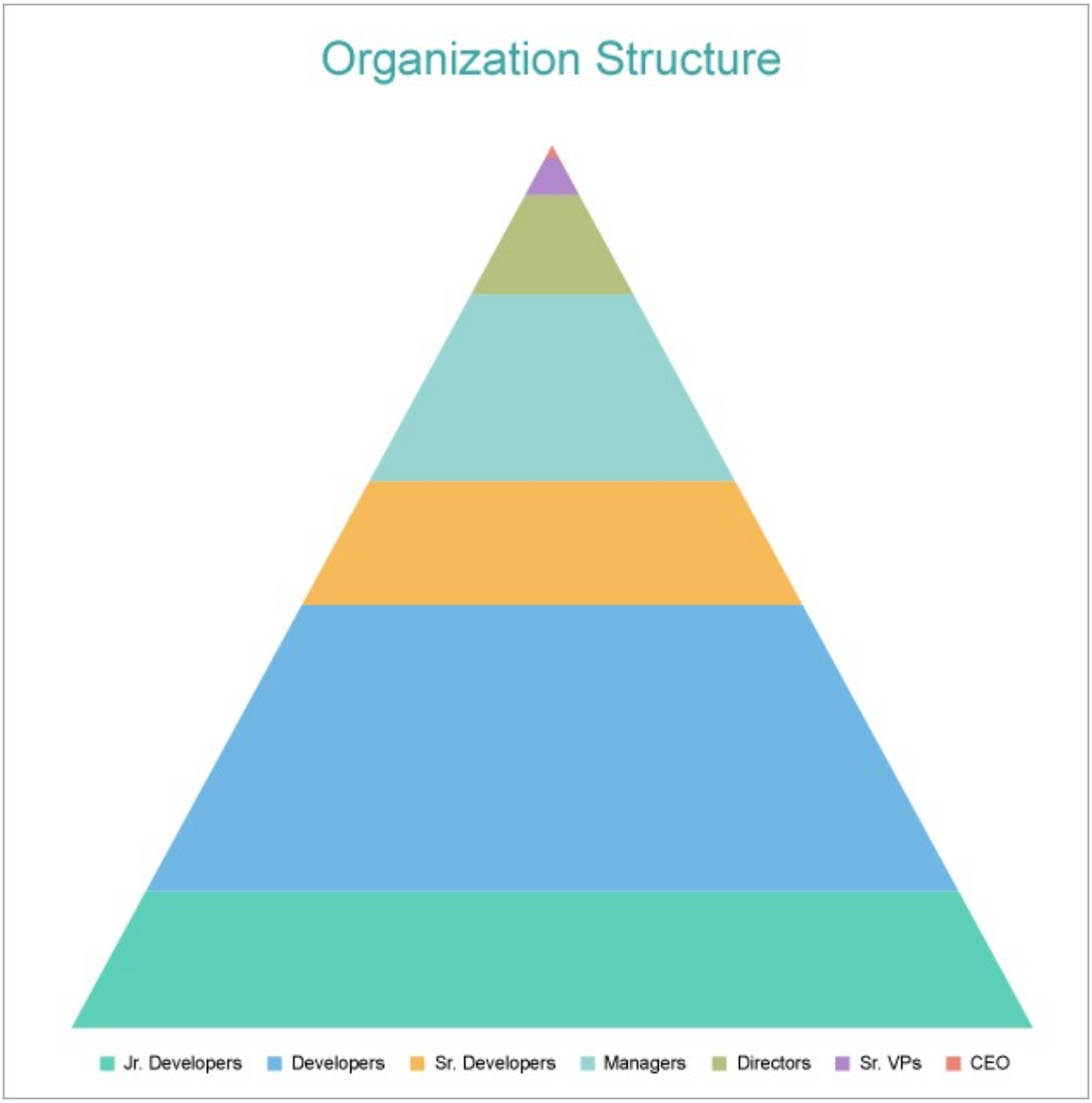


> **Note**: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Pyramid Chart

This walkthrough creates a Clustered Pyramid Chart. The chart shows the hierarchical structure of employees in an organization, which is depicted in the form of discrete vertical pyramids. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data

```
{
   "data":{
      "Counts":[
         {
            "Title":"Managers",
            "Layer":4,
            "Count":15
         },
         {
            "Title":"Sr. VPs",
            "Layer":2,
            "Count":3
         },
         {
            "Title":"Directors",
            "Layer":3,
            "Count":8
         },
         {
            "Title":"Developers",
            "Layer":6,
            "Count":23
         },
         {
            "Title":"CEO",
            "Layer":1,
            "Count":1
         },
         {
            "Title":"Sr. Developers",
            "Layer":5,
            "Count":10
         },
         {
            "Title":"Jr. Developers",
            "Layer":7,
            "Count":11
         }
      ]
   }
}
```


For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'OrgDS'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

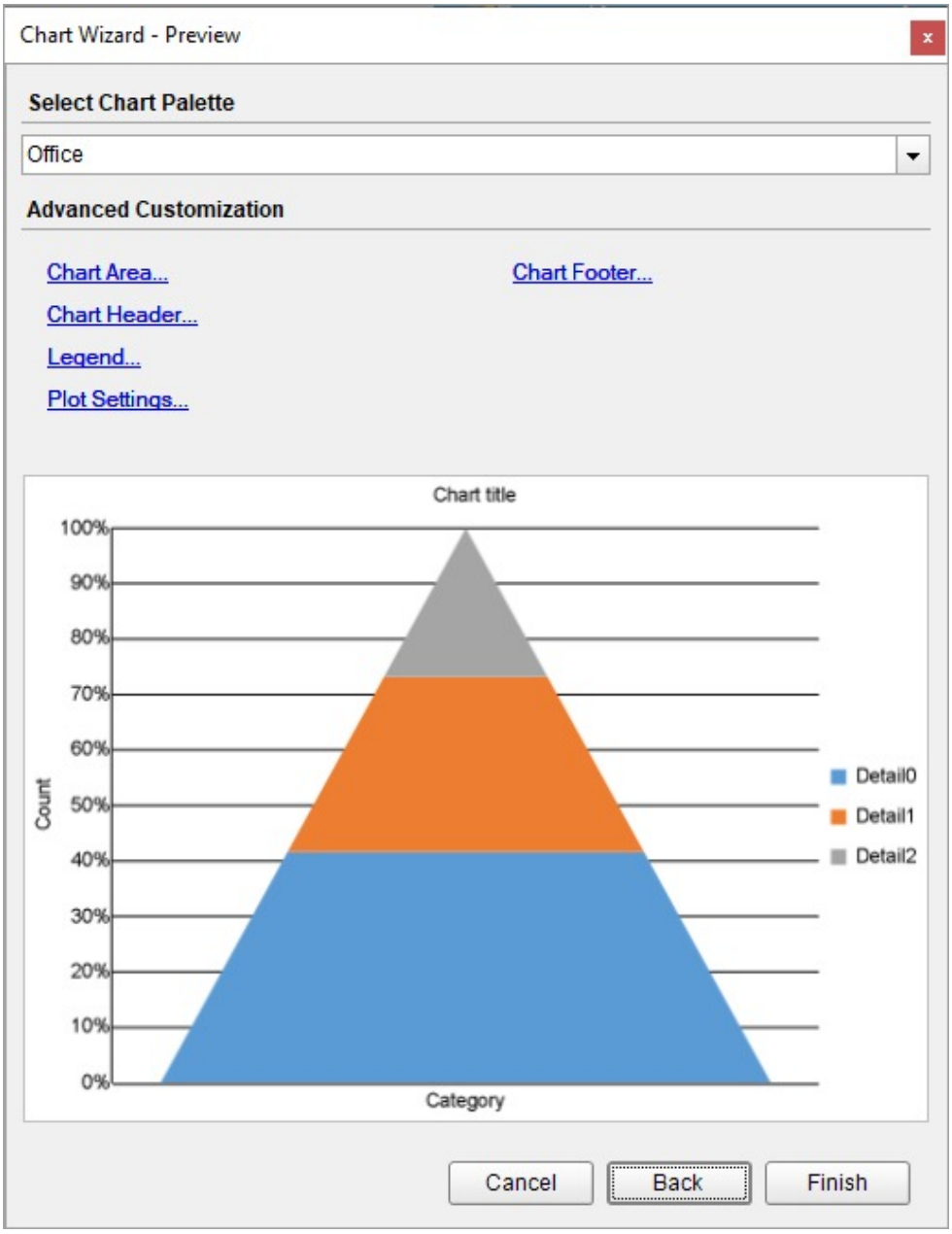
Query
`$.data.Counts[*]`

3. Click **OK** to complete adding the dataset.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'OrgDS' and the **Chart Type** as 'Pyramid'.
3. Click **Next** to proceed. Here, we will define a data value.
4. Under **Choose Data Values**, add a new data value, and set and set the **Field** to `=[Count]`.
5. In **Choose Data Category**, set the **Field** to `=[Title]` and the **Break-down method** to 'Stack'.
6. Click **Next** to preview your chart.



7. Click **Finish** to complete configuring the chart.
   You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Encodings** page > **Detail** tab, ensure that **Expression** is set to `[Title]`, and set following properties as

below.
    - Sorting field: `[Layer]`
    - Sorting direction: Descending
3. Go to the **Appearance** page and set **Opacity** to 70%.
4. Click **OK** to close the dialog.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' from the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** tab and uncheck the **Show Labels** option to hide the data labels.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Go to the **Major Gridline** page and uncheck the **Show Grid** option to hide the gridlines.
6. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page and uncheck the **Show Labels** option to hide the data labels.
4. Go to the **Line** page and uncheck the **Show Line** option.
5. Click **OK** to complete setting up the X-axis.

Chart Palette

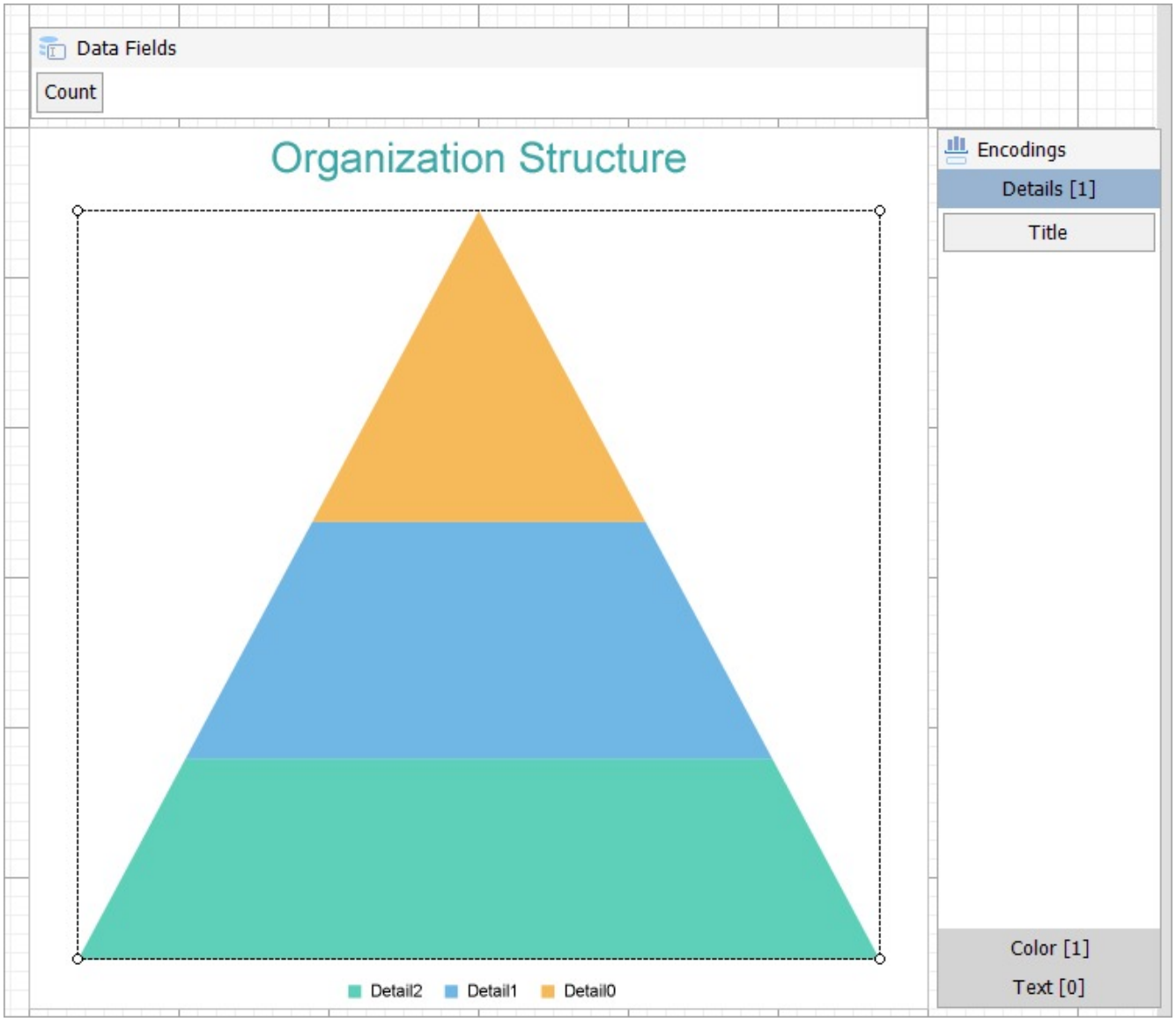
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select the 'Flatly' palette.
3. Click **OK** to complete setting up the chart palette.

Legend - Color

1. To open the smart panel for the legend, right-click 'Legend - Color' on the Report Explorer, and choose **Property Dialog**.
2. Go to the **Layout** page and set the **Orientation** to Horizontal and **Position** to Bottom.
3. Click **OK** to save the settings.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Organization Structure'.
3. Go to the **Font** page and set the properties as below.
    - **Size**: 20pt
    - **Color**: DimGray
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

High Low Close Chart

A High Low Close chart encodes the price points using lines with a marker. The marker stands for the closing value. Also, the high and low values are indicated via the upper and the lower ends of the line. For example, the High-Low-Close plot type can be used to depict the stock price movement for a virtual company.

High Low Close Plot Properties

The plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

LineStyle

The line style for the borders. Includes LineColor, LineStyle, and LineWidth properties for the customization.

- **LineColor**: Specify the color of the border around the sectors.
- **LineStyle:** Specify the line style of the border around the sectors as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the sectors.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to:

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for charts.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

Design

BarSettings

The BarSettings specifies the bar-style settings.

- **NeckHeight**: Determines the bar neck height in percent.
- **BottomWidth**: Determines the Bottom width in percent.
- **Overlap**: Determines the Percentage bar overlap.
- **TopWidth**: Determines the Top width in percent.
- **Width**: Determines the bar width in percent.

Encodings

Category Encoding

The Category Encoding of an plot is a set of properties that determine the period over which the plot generates connected data points representing those above Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Values Encoding

The Values encoding specifies the data values, and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for charts, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the charts take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create High Low Close Chart

This walkthrough creates a High Low Close Chart. The chart shows the movement of stock price for a certain company over time. In the high low close chart, the upper and the lower ends of the line represent the high and low values, while the marker

represents the closing value. The final chart appears like this:

Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.
4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data

```
[
  {
    "Date": "1/4/2021",
    "Close": 129.41,
    "Open": 133.52,
    "High": 133.61,
```

```
    "Low": 126.76
  },
  {
    "Date": "1/5/2021",
    "Close": 131.01,
    "Open": 128.89,
    "High": 131.74,
    "Low": 128.43
  },
  {
    "Date": "1/6/2021",
    "Close": 126.6,
    "Open": 127.72,
    "High": 131.05,
    "Low": 126.38
  },
  {
    "Date": "1/7/2021",
    "Close": 130.92,
    "Open": 128.36,
    "High": 131.63,
    "Low": 127.86
  },
  {
    "Date": "1/8/2021",
    "Close": 132.05,
    "Open": 132.43,
    "High": 132.63,
    "Low": 130.23
  },
  {
    "Date": "1/11/2021",
    "Close": 128.98,
    "Open": 129.19,
    "High": 130.17,
    "Low": 128.5
  },
  {
    "Date": "1/12/2021",
    "Close": 128.8,
    "Open": 128.5,
    "High": 129.69,
    "Low": 126.86
  },
  {
    "Date": "1/13/2021",
    "Close": 130.89,
    "Open": 128.76,
    "High": 131.45,
    "Low": 128.49
  },
```

```
{
  "Date": "1/14/2021",
  "Close": 128.91,
  "Open": 130.8,
  "High": 131,
  "Low": 128.76
},
{
  "Date": "1/15/2021",
  "Close": 127.14,
  "Open": 128.78,
  "High": 130.22,
  "Low": 127
},
{
  "Date": "1/19/2021",
  "Close": 127.83,
  "Open": 127.78,
  "High": 128.71,
  "Low": 126.94
},
{
  "Date": "1/20/2021",
  "Close": 132.03,
  "Open": 128.66,
  "High": 132.49,
  "Low": 128.55
},
{
  "Date": "1/21/2021",
  "Close": 136.87,
  "Open": 133.8,
  "High": 139.67,
  "Low": 133.59
},
{
  "Date": "1/22/2021",
  "Close": 139.07,
  "Open": 136.28,
  "High": 139.85,
  "Low": 135.02
},
{
  "Date": "1/25/2021",
  "Close": 142.92,
  "Open": 143.07,
  "High": 145.09,
  "Low": 136.54
},
{
```

```
  "Date": "1/26/2021",
  "Close": 143.16,
  "Open": 143.6,
  "High": 144.3,
  "Low": 141.37
},
{
  "Date": "1/27/2021",
  "Close": 142.06,
  "Open": 143.43,
  "High": 144.3,
  "Low": 140.41
},
{
  "Date": "1/28/2021",
  "Close": 137.09,
  "Open": 139.52,
  "High": 141.99,
  "Low": 136.7
},
{
  "Date": "1/29/2021",
  "Close": 131.96,
  "Open": 135.83,
  "High": 136.74,
  "Low": 130.21
},
{
  "Date": "2/1/2021",
  "Close": 134.14,
  "Open": 133.75,
  "High": 135.38,
  "Low": 130.93
},
{
  "Date": "2/2/2021",
  "Close": 134.99,
  "Open": 135.73,
  "High": 136.31,
  "Low": 134.61
},
{
  "Date": "2/3/2021",
  "Close": 133.94,
  "Open": 135.76,
  "High": 135.77,
  "Low": 133.61
},
{
  "Date": "2/4/2021",
  "Close": 137.39,
```

```
  "Open": 136.3,
  "High": 137.4,
  "Low": 134.59
},
{
  "Date": "2/5/2021",
  "Close": 136.76,
  "Open": 137.35,
  "High": 137.42,
  "Low": 135.86
},
{
  "Date": "2/8/2021",
  "Close": 136.91,
  "Open": 136.03,
  "High": 136.96,
  "Low": 134.92
},
{
  "Date": "2/9/2021",
  "Close": 136.01,
  "Open": 136.62,
  "High": 137.88,
  "Low": 135.85
},
{
  "Date": "2/10/2021",
  "Close": 135.39,
  "Open": 136.48,
  "High": 136.99,
  "Low": 134.4
},
{
  "Date": "2/11/2021",
  "Close": 135.13,
  "Open": 135.9,
  "High": 136.39,
  "Low": 133.77
},
{
  "Date": "2/12/2021",
  "Close": 135.37,
  "Open": 134.35,
  "High": 135.53,
  "Low": 133.69
},
{
  "Date": "2/16/2021",
  "Close": 133.19,
  "Open": 135.49,
```

```
  "High": 136.01,
  "Low": 132.79
},
{
  "Date": "2/17/2021",
  "Close": 130.84,
  "Open": 131.25,
  "High": 132.22,
  "Low": 129.47
},
{
  "Date": "2/18/2021",
  "Close": 129.71,
  "Open": 129.2,
  "High": 130,
  "Low": 127.41
},
{
  "Date": "2/19/2021",
  "Close": 129.87,
  "Open": 130.24,
  "High": 130.71,
  "Low": 128.8
},
{
  "Date": "2/22/2021",
  "Close": 126,
  "Open": 128.01,
  "High": 129.72,
  "Low": 125.6
},
{
  "Date": "2/23/2021",
  "Close": 125.86,
  "Open": 123.76,
  "High": 126.71,
  "Low": 118.39
},
{
  "Date": "2/24/2021",
  "Close": 125.35,
  "Open": 124.94,
  "High": 125.56,
  "Low": 122.23
},
{
  "Date": "2/25/2021",
  "Close": 120.99,
  "Open": 124.68,
  "High": 126.46,
  "Low": 120.54
```

```
},
{
  "Date": "2/26/2021",
  "Close": 121.26,
  "Open": 122.59,
  "High": 124.85,
  "Low": 121.2
},
{
  "Date": "3/1/2021",
  "Close": 127.79,
  "Open": 123.75,
  "High": 127.93,
  "Low": 122.79
},
{
  "Date": "3/2/2021",
  "Close": 125.12,
  "Open": 128.41,
  "High": 128.72,
  "Low": 125.01
},
{
  "Date": "3/3/2021",
  "Close": 122.06,
  "Open": 124.81,
  "High": 125.71,
  "Low": 121.84
},
{
  "Date": "3/4/2021",
  "Close": 120.13,
  "Open": 121.75,
  "High": 123.6,
  "Low": 118.62
},
{
  "Date": "3/5/2021",
  "Close": 121.42,
  "Open": 120.98,
  "High": 121.94,
  "Low": 117.57
},
{
  "Date": "3/8/2021",
  "Close": 116.36,
  "Open": 120.93,
  "High": 121,
  "Low": 116.21
},
```

```
{
  "Date": "3/9/2021",
  "Close": 121.09,
  "Open": 119.03,
  "High": 122.06,
  "Low": 118.79
},
{
  "Date": "3/10/2021",
  "Close": 119.98,
  "Open": 121.69,
  "High": 122.17,
  "Low": 119.45
},
{
  "Date": "3/11/2021",
  "Close": 121.96,
  "Open": 122.54,
  "High": 123.21,
  "Low": 121.26
},
{
  "Date": "3/12/2021",
  "Close": 121.03,
  "Open": 120.4,
  "High": 121.17,
  "Low": 119.16
},
{
  "Date": "3/15/2021",
  "Close": 123.99,
  "Open": 121.41,
  "High": 124,
  "Low": 120.42
},
{
  "Date": "3/16/2021",
  "Close": 125.57,
  "Open": 125.7,
  "High": 127.22,
  "Low": 124.72
},
{
  "Date": "3/17/2021",
  "Close": 124.76,
  "Open": 124.05,
  "High": 125.86,
  "Low": 122.34
},
{
  "Date": "3/18/2021",
```

```
  "Close": 120.53,
  "Open": 122.88,
  "High": 123.18,
  "Low": 120.32
},
{
  "Date": "3/19/2021",
  "Close": 119.99,
  "Open": 119.9,
  "High": 121.43,
  "Low": 119.68
},
{
  "Date": "3/22/2021",
  "Close": 123.39,
  "Open": 120.33,
  "High": 123.87,
  "Low": 120.26
},
{
  "Date": "3/23/2021",
  "Close": 122.54,
  "Open": 123.33,
  "High": 124.24,
  "Low": 122.14
},
{
  "Date": "3/24/2021",
  "Close": 120.09,
  "Open": 122.82,
  "High": 122.9,
  "Low": 120.07
},
{
  "Date": "3/25/2021",
  "Close": 120.59,
  "Open": 119.54,
  "High": 121.66,
  "Low": 119
},
{
  "Date": "3/26/2021",
  "Close": 121.21,
  "Open": 120.35,
  "High": 121.48,
  "Low": 118.92
},
{
  "Date": "3/29/2021",
  "Close": 121.39,
```

```
  "Open": 121.65,
  "High": 122.58,
  "Low": 120.73
},
{
  "Date": "3/30/2021",
  "Close": 119.9,
  "Open": 120.11,
  "High": 120.4,
  "Low": 118.86
},
{
  "Date": "3/31/2021",
  "Close": 122.15,
  "Open": 121.65,
  "High": 123.52,
  "Low": 121.15
},
{
  "Date": "4/1/2021",
  "Close": 123,
  "Open": 123.66,
  "High": 124.18,
  "Low": 122.49
},
{
  "Date": "4/5/2021",
  "Close": 125.9,
  "Open": 123.87,
  "High": 126.16,
  "Low": 123.07
},
{
  "Date": "4/6/2021",
  "Close": 126.21,
  "Open": 126.5,
  "High": 127.13,
  "Low": 125.65
},
{
  "Date": "4/7/2021",
  "Close": 127.9,
  "Open": 125.83,
  "High": 127.92,
  "Low": 125.14
},
{
  "Date": "4/8/2021",
  "Close": 130.36,
  "Open": 128.95,
  "High": 130.39,
```

```
    "Low": 128.52
  },
  {
    "Date": "4/9/2021",
    "Close": 133,
    "Open": 129.8,
    "High": 133.04,
    "Low": 129.47
  },
  {
    "Date": "4/12/2021",
    "Close": 131.24,
    "Open": 132.52,
    "High": 132.85,
    "Low": 130.63
  },
  {
    "Date": "4/13/2021",
    "Close": 134.43,
    "Open": 132.44,
    "High": 134.66,
    "Low": 131.93
  },
  {
    "Date": "4/14/2021",
    "Close": 132.03,
    "Open": 134.94,
    "High": 135,
    "Low": 131.66
  },
  {
    "Date": "4/15/2021",
    "Close": 134.5,
    "Open": 133.82,
    "High": 135,
    "Low": 133.64
  },
  {
    "Date": "4/16/2021",
    "Close": 134.16,
    "Open": 134.3,
    "High": 134.67,
    "Low": 133.28
  },
  {
    "Date": "4/19/2021",
    "Close": 134.84,
    "Open": 133.51,
    "High": 135.47,
    "Low": 133.34
```

```
},
{
  "Date": "4/20/2021",
  "Close": 133.11,
  "Open": 135.02,
  "High": 135.53,
  "Low": 131.81
},
{
  "Date": "4/21/2021",
  "Close": 133.5,
  "Open": 132.36,
  "High": 133.75,
  "Low": 131.3
},
{
  "Date": "4/22/2021",
  "Close": 131.94,
  "Open": 133.04,
  "High": 134.15,
  "Low": 131.41
},
{
  "Date": "4/23/2021",
  "Close": 134.32,
  "Open": 132.16,
  "High": 135.12,
  "Low": 132.16
},
{
  "Date": "4/26/2021",
  "Close": 134.72,
  "Open": 134.83,
  "High": 135.06,
  "Low": 133.56
},
{
  "Date": "4/27/2021",
  "Close": 134.39,
  "Open": 135.01,
  "High": 135.41,
  "Low": 134.11
},
{
  "Date": "4/28/2021",
  "Close": 133.58,
  "Open": 134.31,
  "High": 135.02,
  "Low": 133.08
},
{
```

```
  "Date": "4/29/2021",
  "Close": 133.48,
  "Open": 136.47,
  "High": 137.07,
  "Low": 132.45
},
{
  "Date": "4/30/2021",
  "Close": 131.46,
  "Open": 131.78,
  "High": 133.56,
  "Low": 131.07
},
{
  "Date": "5/3/2021",
  "Close": 132.54,
  "Open": 132.04,
  "High": 134.07,
  "Low": 131.83
},
{
  "Date": "5/4/2021",
  "Close": 127.85,
  "Open": 131.19,
  "High": 131.49,
  "Low": 126.7
},
{
  "Date": "5/5/2021",
  "Close": 128.1,
  "Open": 129.2,
  "High": 130.45,
  "Low": 127.97
},
{
  "Date": "5/6/2021",
  "Close": 129.74,
  "Open": 127.89,
  "High": 129.75,
  "Low": 127.13
},
{
  "Date": "5/7/2021",
  "Close": 130.21,
  "Open": 130.85,
  "High": 131.26,
  "Low": 129.48
},
{
  "Date": "5/10/2021",
```

```
  "Close": 126.85,
  "Open": 129.41,
  "High": 129.54,
  "Low": 126.81
},
{
  "Date": "5/11/2021",
  "Close": 125.91,
  "Open": 123.5,
  "High": 126.27,
  "Low": 122.77
},
{
  "Date": "5/12/2021",
  "Close": 122.77,
  "Open": 123.4,
  "High": 124.64,
  "Low": 122.25
},
{
  "Date": "5/13/2021",
  "Close": 124.97,
  "Open": 124.58,
  "High": 126.15,
  "Low": 124.26
},
{
  "Date": "5/14/2021",
  "Close": 127.45,
  "Open": 126.25,
  "High": 127.89,
  "Low": 125.85
},
{
  "Date": "5/17/2021",
  "Close": 126.27,
  "Open": 126.82,
  "High": 126.93,
  "Low": 125.17
},
{
  "Date": "5/18/2021",
  "Close": 124.85,
  "Open": 126.56,
  "High": 126.99,
  "Low": 124.78
},
{
  "Date": "5/19/2021",
  "Close": 124.69,
  "Open": 123.16,
```

```
  "High": 124.92,
  "Low": 122.86
},
{
  "Date": "5/20/2021",
  "Close": 127.31,
  "Open": 125.23,
  "High": 127.72,
  "Low": 125.1
},
{
  "Date": "5/21/2021",
  "Close": 125.43,
  "Open": 127.82,
  "High": 128,
  "Low": 125.21
},
{
  "Date": "5/24/2021",
  "Close": 127.1,
  "Open": 126.01,
  "High": 127.94,
  "Low": 125.94
},
{
  "Date": "5/25/2021",
  "Close": 126.9,
  "Open": 127.82,
  "High": 128.32,
  "Low": 126.32
},
{
  "Date": "5/26/2021",
  "Close": 126.85,
  "Open": 126.96,
  "High": 127.39,
  "Low": 126.42
},
{
  "Date": "5/27/2021",
  "Close": 125.28,
  "Open": 126.44,
  "High": 127.64,
  "Low": 125.08
},
{
  "Date": "5/28/2021",
  "Close": 124.61,
  "Open": 125.57,
  "High": 125.8,
```

```
    "Low": 124.55
  },
  {
    "Date": "6/1/2021",
    "Close": 124.28,
    "Open": 125.08,
    "High": 125.35,
    "Low": 123.94
  },
  {
    "Date": "6/2/2021",
    "Close": 125.06,
    "Open": 124.28,
    "High": 125.24,
    "Low": 124.05
  },
  {
    "Date": "6/3/2021",
    "Close": 123.54,
    "Open": 124.68,
    "High": 124.85,
    "Low": 123.13
  },
  {
    "Date": "6/4/2021",
    "Close": 125.89,
    "Open": 124.07,
    "High": 126.16,
    "Low": 123.85
  },
  {
    "Date": "6/7/2021",
    "Close": 125.9,
    "Open": 126.17,
    "High": 126.32,
    "Low": 124.83
  },
  {
    "Date": "6/8/2021",
    "Close": 126.74,
    "Open": 126.6,
    "High": 128.46,
    "Low": 126.21
  }
]
```


For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'StockPrice'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Click **OK** to save the changes.

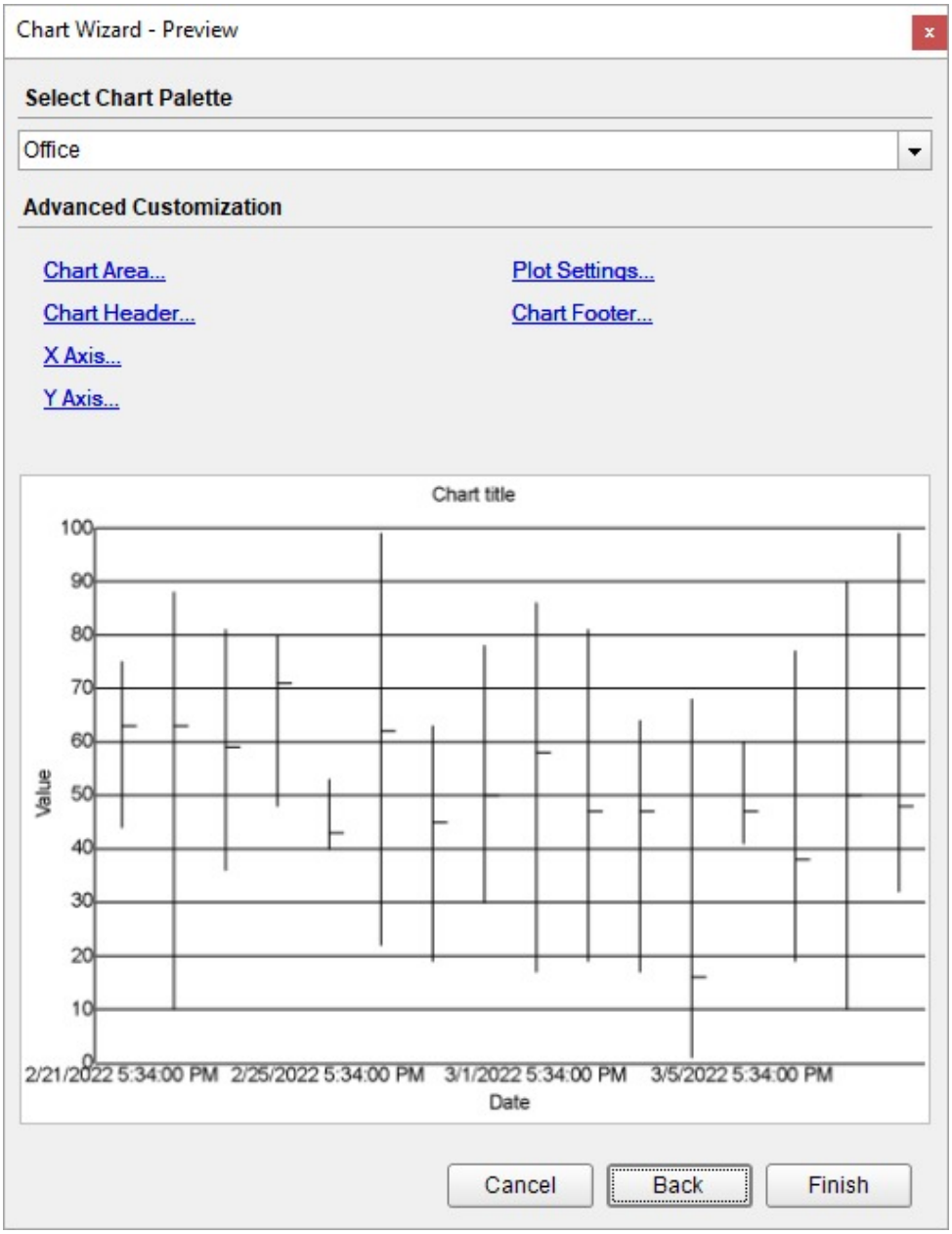
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'StockPrice' and the **Chart Type** as 'High Low Close'.
3. Click **Next** to proceed. Here, we will define the high, low, and close field values in the chart.
4. Under **Choose Data Values**, set the High, Low, and Close fields to the following values.

High Field	Low Field	Close Field
`=[High]`	`=[Low]`	`=[Close]`

5. In **Choose Data Category**, set **Field** to `=[Date]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Line Style > Style**: Solid

- **Line Style > Color**: #8fcd37
- **Line Style > Width**: 1pt

3. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** page and set the **Format** to 'Currency (with 0 decimal points)'.
4. Navigate to the **Appearance** tab and set the following properties.
   - **Font > Size:** 12pt
   - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
   - **Show Line**: Check-on
   - **Color**: Black
   - **Width**: 1pt
   - **Style**: Solid
6. Go to the **Major Gridline** page and set the following properties.
   - **Grid Interval**: 10
   - **Grid appearance > Show Grid**: Check-on
   - **Grid appearance > Width**: 0.25pt
   - **Grid appearance > Color**: #cccccc
   - **Grid appearance > Style**: Dashed
   - **Tick mark appearance > Position**: Outside
   - **Tick mark appearance > Color**: #cccccc
   - **Tick mark appearance > Width**: 0.25pt
   - **Tick mark appearance > Style**: Solid
7. Click **OK** to complete setting up the Y-axis.

X-Axis

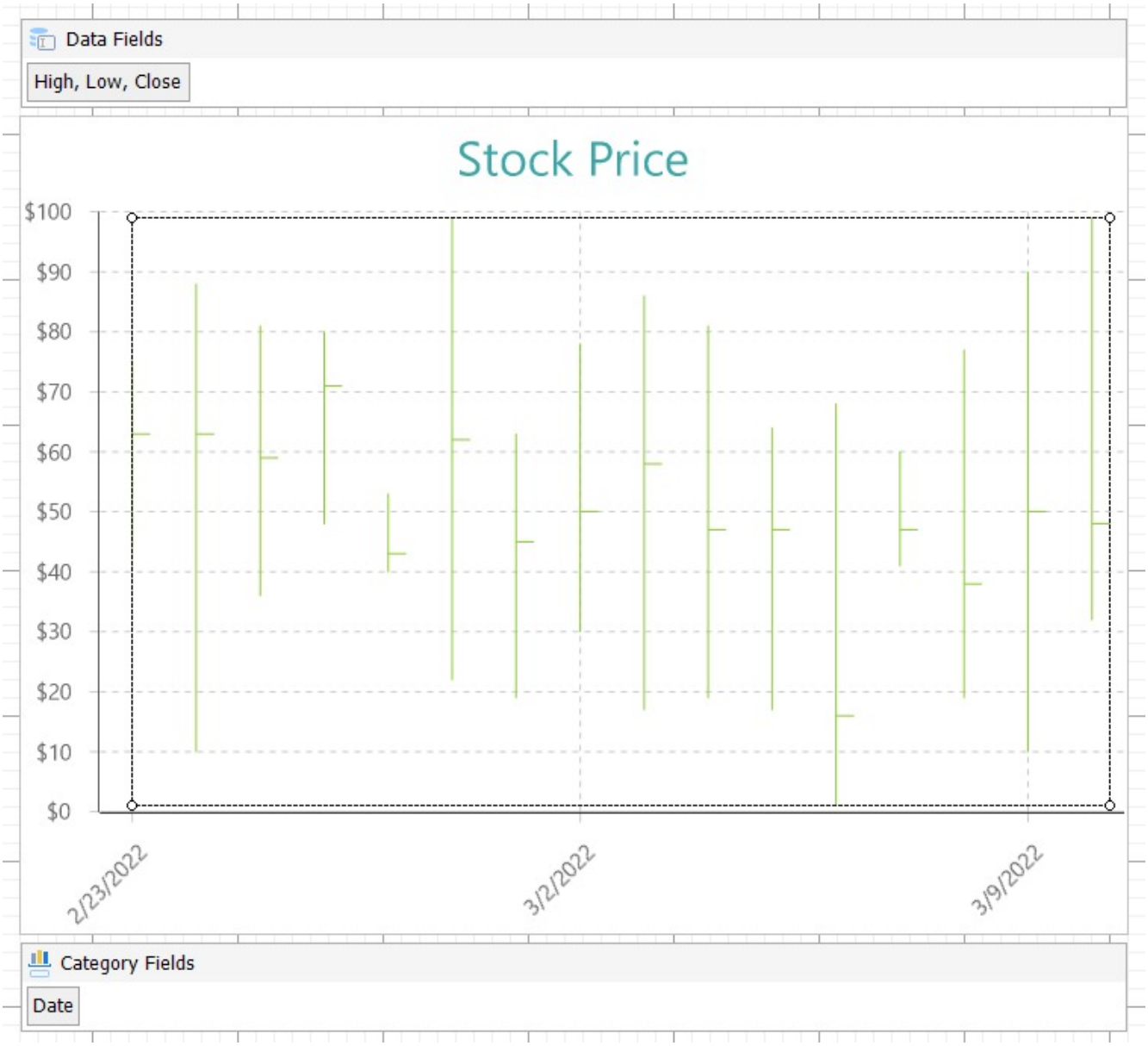
1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the following properties.
   - **Format**: Short Date
   - **Angle**: -45
4. Now, navigate to the **Appearance** page and set the following properties.
   - **Font > Size:** 12pt
   - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
   - **Show Line**: Check-on
   - **Color**: Black
   - **Width**: 1pt
   - **Style**: Solid
6. Go to the **Major Gridline** page and set the following properties.
   - **Grid Interval**: 7 (to view weekly stock prices)
   - **Grid appearance > Show Grid**: Check-on
   - **Grid appearance > Width**: 0.25pt
   - **Grid appearance > Color**: #cccccc
   - **Grid appearance > Style**: Dashed
   - **Tick mark appearance > Position**: Outside
   - **Tick mark appearance > Color**: #cccccc

- **Tick mark appearance > Width**: 0.25pt
- **Tick mark appearance > Style**: Solid

7. Click **OK** to complete setting up the X-axis.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Stock Price'.
3. Go to the **Font** page and set the properties as below.
   - **Size**: 24pt
   - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the

> chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

High Low Open Close Chart

HighLowOpenClose charts help to portray the price movements of a security, currency, or derivative over time with clarity. Though these are quite similar to the candlestick charts, a striking feature of the HighLowOpenClose charts is that they show the opening values using lines to the left, and closing values using lines to the right. The vertical line size is determined by the High and Low values. A suitable example for the HighLowOpenClose plot type is the movement of the stock price of a virtual company.

High Low Open Close Plot Properties

The plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Appearance

BackgroundColor

Indicates the color used to fill the chart area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

LineStyle

The line style for the borders. Includes LineColor, LineStyle, and LineWidth properties for the customization.

- **LineColor**: Specify the color of the border around the sectors.

- **LineStyle:** Specify the line style of the border around the sectors as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the sectors.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to:

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for charts.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

Design

BarSettings

The BarSettings specifies the bar-style settings.

- **NeckHeight**: Determines the bar neck height in percent.
- **BottomWidth**: Determines the Bottom width in percent.
- **Overlap**: Determines the Percentage bar overlap.
- **TopWidth**: Determines the Top width in percent.
- **Width**: Determines the bar width in percent.

Encodings

Category Encoding

The Category Encoding of an plot is a set of properties that determine the period over which the plot generates connected data points representing those above Data Values. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Values Encoding

The Values encoding specifies the data values and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for charts, 'Simple' is acceptable.

Value

The Value property is the collection and usually takes a bound field. However, the charts take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

Count:{Text0}
Sum:{valueField.value}

Template Key

A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create High Low Open Close Chart

This walkthrough creates a High Low Open Close Chart. The chart shows the movement of stock price for a certain company over time. In the high low open close chart, the upper and the lower ends of the line represent the high and low values, while the markers on the left and right of the line represent the opening and closing values. The final chart appears like this:

Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'Embedded'.

4. In the **Select or type the file name or URL or enter the data to be embedded** field, enter the following data:

JSON Data


```
[
  {
    "Date": "1/4/2021",
    "Close": 129.41,
    "Open": 133.52,
    "High": 133.61,
    "Low": 126.76
  },
  {
    "Date": "1/5/2021",
    "Close": 131.01,
    "Open": 128.89,
    "High": 131.74,
    "Low": 128.43
  },
  {
    "Date": "1/6/2021",
    "Close": 126.6,
    "Open": 127.72,
    "High": 131.05,
    "Low": 126.38
  },
  {
    "Date": "1/7/2021",
    "Close": 130.92,
    "Open": 128.36,
    "High": 131.63,
    "Low": 127.86
  },
  {
    "Date": "1/8/2021",
    "Close": 132.05,
    "Open": 132.43,
    "High": 132.63,
    "Low": 130.23
  },
  {
    "Date": "1/11/2021",
    "Close": 128.98,
    "Open": 129.19,
    "High": 130.17,
    "Low": 128.5
  },
  {
    "Date": "1/12/2021",
    "Close": 128.8,
```

```
  "Open": 128.5,
  "High": 129.69,
  "Low": 126.86
},
{
  "Date": "1/13/2021",
  "Close": 130.89,
  "Open": 128.76,
  "High": 131.45,
  "Low": 128.49
},
{
  "Date": "1/14/2021",
  "Close": 128.91,
  "Open": 130.8,
  "High": 131,
  "Low": 128.76
},
{
  "Date": "1/15/2021",
  "Close": 127.14,
  "Open": 128.78,
  "High": 130.22,
  "Low": 127
},
{
  "Date": "1/19/2021",
  "Close": 127.83,
  "Open": 127.78,
  "High": 128.71,
  "Low": 126.94
},
{
  "Date": "1/20/2021",
  "Close": 132.03,
  "Open": 128.66,
  "High": 132.49,
  "Low": 128.55
},
{
  "Date": "1/21/2021",
  "Close": 136.87,
  "Open": 133.8,
  "High": 139.67,
  "Low": 133.59
},
{
  "Date": "1/22/2021",
  "Close": 139.07,
  "Open": 136.28,
  "High": 139.85,
```

```
  "Low": 135.02
},
{
  "Date": "1/25/2021",
  "Close": 142.92,
  "Open": 143.07,
  "High": 145.09,
  "Low": 136.54
},
{
  "Date": "1/26/2021",
  "Close": 143.16,
  "Open": 143.6,
  "High": 144.3,
  "Low": 141.37
},
{
  "Date": "1/27/2021",
  "Close": 142.06,
  "Open": 143.43,
  "High": 144.3,
  "Low": 140.41
},
{
  "Date": "1/28/2021",
  "Close": 137.09,
  "Open": 139.52,
  "High": 141.99,
  "Low": 136.7
},
{
  "Date": "1/29/2021",
  "Close": 131.96,
  "Open": 135.83,
  "High": 136.74,
  "Low": 130.21
},
{
  "Date": "2/1/2021",
  "Close": 134.14,
  "Open": 133.75,
  "High": 135.38,
  "Low": 130.93
},
{
  "Date": "2/2/2021",
  "Close": 134.99,
  "Open": 135.73,
  "High": 136.31,
  "Low": 134.61
```

```
},
{
  "Date": "2/3/2021",
  "Close": 133.94,
  "Open": 135.76,
  "High": 135.77,
  "Low": 133.61
},
{
  "Date": "2/4/2021",
  "Close": 137.39,
  "Open": 136.3,
  "High": 137.4,
  "Low": 134.59
},
{
  "Date": "2/5/2021",
  "Close": 136.76,
  "Open": 137.35,
  "High": 137.42,
  "Low": 135.86
},
{
  "Date": "2/8/2021",
  "Close": 136.91,
  "Open": 136.03,
  "High": 136.96,
  "Low": 134.92
},
{
  "Date": "2/9/2021",
  "Close": 136.01,
  "Open": 136.62,
  "High": 137.88,
  "Low": 135.85
},
{
  "Date": "2/10/2021",
  "Close": 135.39,
  "Open": 136.48,
  "High": 136.99,
  "Low": 134.4
},
{
  "Date": "2/11/2021",
  "Close": 135.13,
  "Open": 135.9,
  "High": 136.39,
  "Low": 133.77
},
{
```

```
  "Date": "2/12/2021",
  "Close": 135.37,
  "Open": 134.35,
  "High": 135.53,
  "Low": 133.69
},
{
  "Date": "2/16/2021",
  "Close": 133.19,
  "Open": 135.49,
  "High": 136.01,
  "Low": 132.79
},
{
  "Date": "2/17/2021",
  "Close": 130.84,
  "Open": 131.25,
  "High": 132.22,
  "Low": 129.47
},
{
  "Date": "2/18/2021",
  "Close": 129.71,
  "Open": 129.2,
  "High": 130,
  "Low": 127.41
},
{
  "Date": "2/19/2021",
  "Close": 129.87,
  "Open": 130.24,
  "High": 130.71,
  "Low": 128.8
},
{
  "Date": "2/22/2021",
  "Close": 126,
  "Open": 128.01,
  "High": 129.72,
  "Low": 125.6
},
{
  "Date": "2/23/2021",
  "Close": 125.86,
  "Open": 123.76,
  "High": 126.71,
  "Low": 118.39
},
{
  "Date": "2/24/2021",
```

```
  "Close": 125.35,
  "Open": 124.94,
  "High": 125.56,
  "Low": 122.23
},
{
  "Date": "2/25/2021",
  "Close": 120.99,
  "Open": 124.68,
  "High": 126.46,
  "Low": 120.54
},
{
  "Date": "2/26/2021",
  "Close": 121.26,
  "Open": 122.59,
  "High": 124.85,
  "Low": 121.2
},
{
  "Date": "3/1/2021",
  "Close": 127.79,
  "Open": 123.75,
  "High": 127.93,
  "Low": 122.79
},
{
  "Date": "3/2/2021",
  "Close": 125.12,
  "Open": 128.41,
  "High": 128.72,
  "Low": 125.01
},
{
  "Date": "3/3/2021",
  "Close": 122.06,
  "Open": 124.81,
  "High": 125.71,
  "Low": 121.84
},
{
  "Date": "3/4/2021",
  "Close": 120.13,
  "Open": 121.75,
  "High": 123.6,
  "Low": 118.62
},
{
  "Date": "3/5/2021",
  "Close": 121.42,
  "Open": 120.98,
```

```
  "High": 121.94,
  "Low": 117.57
},
{
  "Date": "3/8/2021",
  "Close": 116.36,
  "Open": 120.93,
  "High": 121,
  "Low": 116.21
},
{
  "Date": "3/9/2021",
  "Close": 121.09,
  "Open": 119.03,
  "High": 122.06,
  "Low": 118.79
},
{
  "Date": "3/10/2021",
  "Close": 119.98,
  "Open": 121.69,
  "High": 122.17,
  "Low": 119.45
},
{
  "Date": "3/11/2021",
  "Close": 121.96,
  "Open": 122.54,
  "High": 123.21,
  "Low": 121.26
},
{
  "Date": "3/12/2021",
  "Close": 121.03,
  "Open": 120.4,
  "High": 121.17,
  "Low": 119.16
},
{
  "Date": "3/15/2021",
  "Close": 123.99,
  "Open": 121.41,
  "High": 124,
  "Low": 120.42
},
{
  "Date": "3/16/2021",
  "Close": 125.57,
  "Open": 125.7,
  "High": 127.22,
```

```
    "Low": 124.72
  },
  {
    "Date": "3/17/2021",
    "Close": 124.76,
    "Open": 124.05,
    "High": 125.86,
    "Low": 122.34
  },
  {
    "Date": "3/18/2021",
    "Close": 120.53,
    "Open": 122.88,
    "High": 123.18,
    "Low": 120.32
  },
  {
    "Date": "3/19/2021",
    "Close": 119.99,
    "Open": 119.9,
    "High": 121.43,
    "Low": 119.68
  },
  {
    "Date": "3/22/2021",
    "Close": 123.39,
    "Open": 120.33,
    "High": 123.87,
    "Low": 120.26
  },
  {
    "Date": "3/23/2021",
    "Close": 122.54,
    "Open": 123.33,
    "High": 124.24,
    "Low": 122.14
  },
  {
    "Date": "3/24/2021",
    "Close": 120.09,
    "Open": 122.82,
    "High": 122.9,
    "Low": 120.07
  },
  {
    "Date": "3/25/2021",
    "Close": 120.59,
    "Open": 119.54,
    "High": 121.66,
    "Low": 119
  },
```

```
{
  "Date": "3/26/2021",
  "Close": 121.21,
  "Open": 120.35,
  "High": 121.48,
  "Low": 118.92
},
{
  "Date": "3/29/2021",
  "Close": 121.39,
  "Open": 121.65,
  "High": 122.58,
  "Low": 120.73
},
{
  "Date": "3/30/2021",
  "Close": 119.9,
  "Open": 120.11,
  "High": 120.4,
  "Low": 118.86
},
{
  "Date": "3/31/2021",
  "Close": 122.15,
  "Open": 121.65,
  "High": 123.52,
  "Low": 121.15
},
{
  "Date": "4/1/2021",
  "Close": 123,
  "Open": 123.66,
  "High": 124.18,
  "Low": 122.49
},
{
  "Date": "4/5/2021",
  "Close": 125.9,
  "Open": 123.87,
  "High": 126.16,
  "Low": 123.07
},
{
  "Date": "4/6/2021",
  "Close": 126.21,
  "Open": 126.5,
  "High": 127.13,
  "Low": 125.65
},
{
```

```
  "Date": "4/7/2021",
  "Close": 127.9,
  "Open": 125.83,
  "High": 127.92,
  "Low": 125.14
},
{
  "Date": "4/8/2021",
  "Close": 130.36,
  "Open": 128.95,
  "High": 130.39,
  "Low": 128.52
},
{
  "Date": "4/9/2021",
  "Close": 133,
  "Open": 129.8,
  "High": 133.04,
  "Low": 129.47
},
{
  "Date": "4/12/2021",
  "Close": 131.24,
  "Open": 132.52,
  "High": 132.85,
  "Low": 130.63
},
{
  "Date": "4/13/2021",
  "Close": 134.43,
  "Open": 132.44,
  "High": 134.66,
  "Low": 131.93
},
{
  "Date": "4/14/2021",
  "Close": 132.03,
  "Open": 134.94,
  "High": 135,
  "Low": 131.66
},
{
  "Date": "4/15/2021",
  "Close": 134.5,
  "Open": 133.82,
  "High": 135,
  "Low": 133.64
},
{
  "Date": "4/16/2021",
  "Close": 134.16,
```

```
  "Open": 134.3,
  "High": 134.67,
  "Low": 133.28
},
{
  "Date": "4/19/2021",
  "Close": 134.84,
  "Open": 133.51,
  "High": 135.47,
  "Low": 133.34
},
{
  "Date": "4/20/2021",
  "Close": 133.11,
  "Open": 135.02,
  "High": 135.53,
  "Low": 131.81
},
{
  "Date": "4/21/2021",
  "Close": 133.5,
  "Open": 132.36,
  "High": 133.75,
  "Low": 131.3
},
{
  "Date": "4/22/2021",
  "Close": 131.94,
  "Open": 133.04,
  "High": 134.15,
  "Low": 131.41
},
{
  "Date": "4/23/2021",
  "Close": 134.32,
  "Open": 132.16,
  "High": 135.12,
  "Low": 132.16
},
{
  "Date": "4/26/2021",
  "Close": 134.72,
  "Open": 134.83,
  "High": 135.06,
  "Low": 133.56
},
{
  "Date": "4/27/2021",
  "Close": 134.39,
  "Open": 135.01,
```

```
    "High": 135.41,
    "Low": 134.11
  },
  {
    "Date": "4/28/2021",
    "Close": 133.58,
    "Open": 134.31,
    "High": 135.02,
    "Low": 133.08
  },
  {
    "Date": "4/29/2021",
    "Close": 133.48,
    "Open": 136.47,
    "High": 137.07,
    "Low": 132.45
  },
  {
    "Date": "4/30/2021",
    "Close": 131.46,
    "Open": 131.78,
    "High": 133.56,
    "Low": 131.07
  },
  {
    "Date": "5/3/2021",
    "Close": 132.54,
    "Open": 132.04,
    "High": 134.07,
    "Low": 131.83
  },
  {
    "Date": "5/4/2021",
    "Close": 127.85,
    "Open": 131.19,
    "High": 131.49,
    "Low": 126.7
  },
  {
    "Date": "5/5/2021",
    "Close": 128.1,
    "Open": 129.2,
    "High": 130.45,
    "Low": 127.97
  },
  {
    "Date": "5/6/2021",
    "Close": 129.74,
    "Open": 127.89,
    "High": 129.75,
    "Low": 127.13
```

```
},
{
  "Date": "5/7/2021",
  "Close": 130.21,
  "Open": 130.85,
  "High": 131.26,
  "Low": 129.48
},
{
  "Date": "5/10/2021",
  "Close": 126.85,
  "Open": 129.41,
  "High": 129.54,
  "Low": 126.81
},
{
  "Date": "5/11/2021",
  "Close": 125.91,
  "Open": 123.5,
  "High": 126.27,
  "Low": 122.77
},
{
  "Date": "5/12/2021",
  "Close": 122.77,
  "Open": 123.4,
  "High": 124.64,
  "Low": 122.25
},
{
  "Date": "5/13/2021",
  "Close": 124.97,
  "Open": 124.58,
  "High": 126.15,
  "Low": 124.26
},
{
  "Date": "5/14/2021",
  "Close": 127.45,
  "Open": 126.25,
  "High": 127.89,
  "Low": 125.85
},
{
  "Date": "5/17/2021",
  "Close": 126.27,
  "Open": 126.82,
  "High": 126.93,
  "Low": 125.17
},
```

```
{
  "Date": "5/18/2021",
  "Close": 124.85,
  "Open": 126.56,
  "High": 126.99,
  "Low": 124.78
},
{
  "Date": "5/19/2021",
  "Close": 124.69,
  "Open": 123.16,
  "High": 124.92,
  "Low": 122.86
},
{
  "Date": "5/20/2021",
  "Close": 127.31,
  "Open": 125.23,
  "High": 127.72,
  "Low": 125.1
},
{
  "Date": "5/21/2021",
  "Close": 125.43,
  "Open": 127.82,
  "High": 128,
  "Low": 125.21
},
{
  "Date": "5/24/2021",
  "Close": 127.1,
  "Open": 126.01,
  "High": 127.94,
  "Low": 125.94
},
{
  "Date": "5/25/2021",
  "Close": 126.9,
  "Open": 127.82,
  "High": 128.32,
  "Low": 126.32
},
{
  "Date": "5/26/2021",
  "Close": 126.85,
  "Open": 126.96,
  "High": 127.39,
  "Low": 126.42
},
{
  "Date": "5/27/2021",
```

```
  "Close": 125.28,
  "Open": 126.44,
  "High": 127.64,
  "Low": 125.08
},
{
  "Date": "5/28/2021",
  "Close": 124.61,
  "Open": 125.57,
  "High": 125.8,
  "Low": 124.55
},
{
  "Date": "6/1/2021",
  "Close": 124.28,
  "Open": 125.08,
  "High": 125.35,
  "Low": 123.94
},
{
  "Date": "6/2/2021",
  "Close": 125.06,
  "Open": 124.28,
  "High": 125.24,
  "Low": 124.05
},
{
  "Date": "6/3/2021",
  "Close": 123.54,
  "Open": 124.68,
  "High": 124.85,
  "Low": 123.13
},
{
  "Date": "6/4/2021",
  "Close": 125.89,
  "Open": 124.07,
  "High": 126.16,
  "Low": 123.85
},
{
  "Date": "6/7/2021",
  "Close": 125.9,
  "Open": 126.17,
  "High": 126.32,
  "Low": 124.83
},
{
  "Date": "6/8/2021",
  "Close": 126.74,
```

```
    "Open": 126.6,
    "High": 128.46,
    "Low": 126.21
  }
]
```

For more information, see the JSON Provider topic.

5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'StockPrice'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Click **OK** to save the changes.

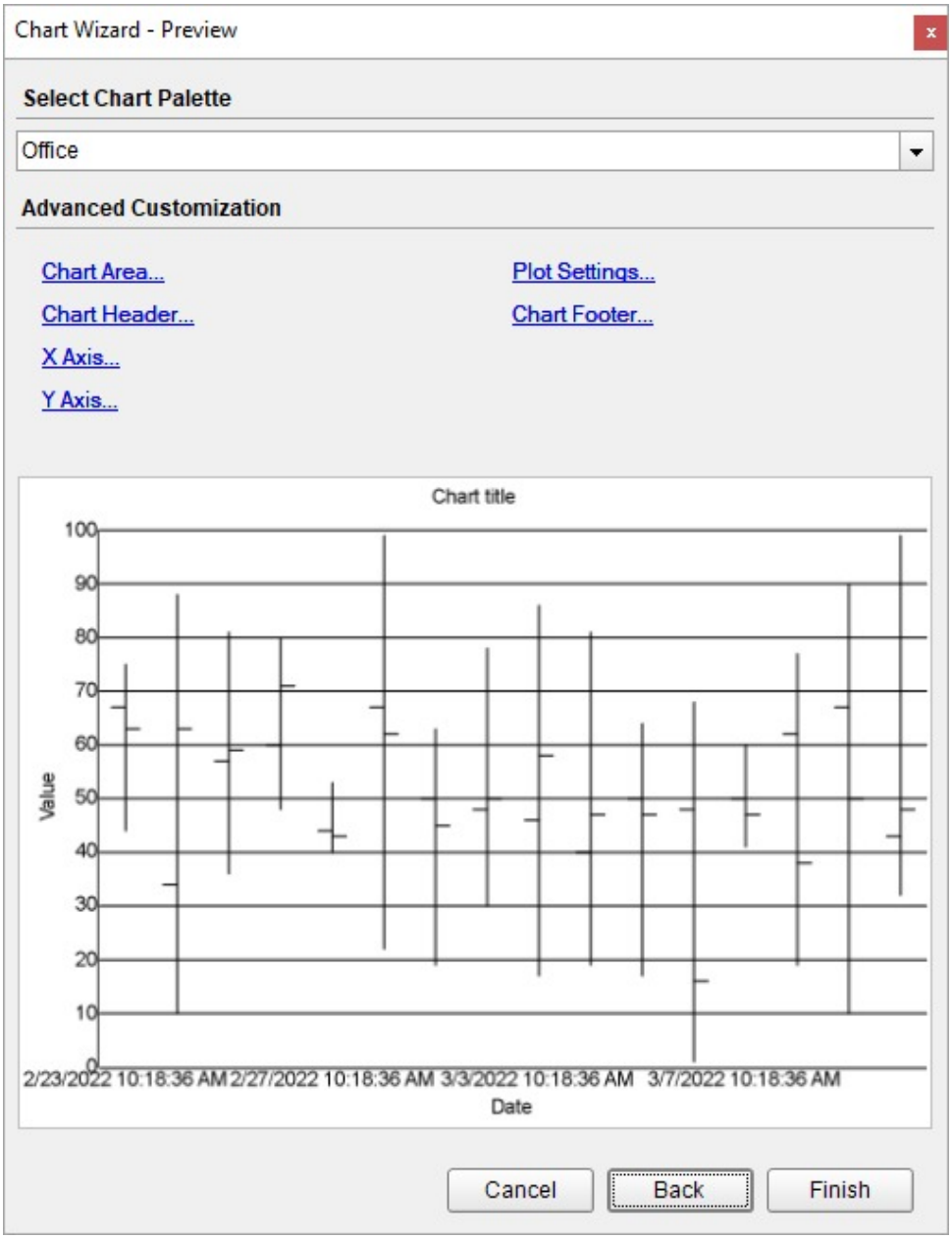
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'StockPrice' and the **Chart Type** as 'High Low Open Close'.
3. Click **Next** to proceed. Here, we will define the high, low, open, and close field values in the chart.
4. Under **Choose Data Values**, set the High, Low, Open, and Close fields to the following values.

High Field	Low Field	Open Field	Close Field
`=[High]`	`=[Low]`	`=[Open]`	`=[Close]`

5. In **Choose Data Category**, set **Field** to `=[Date]`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Appearance** page and set the following properties.
    - **Line Style > Style**: Solid

- **Line Style > Color**: #8fcd37
- **Line Style > Width**: 1pt

3. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the Y-axis title in the chart.
3. Go to the **Labels** page > **General** page and set the **Format** to 'Currency (with 0 decimal points)'.
4. Navigate to the **Appearance** tab and set the following properties.
   - **Font > Size:** 12pt
   - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
   - **Show Line**: Check-on
   - **Color**: Black
   - **Width**: 1pt
   - **Style**: Solid
6. Go to the **Major Gridline** page and set the following properties.
   - **Grid Interval**: 10
   - **Grid appearance > Show Grid**: Check-on
   - **Grid appearance > Width**: 0.25pt
   - **Grid appearance > Color**: #cccccc
   - **Grid appearance > Style**: Dashed
   - **Tick mark appearance > Position**: Outside
   - **Tick mark appearance > Color**: #cccccc
   - **Tick mark appearance > Width**: 0.25pt
   - **Tick mark appearance > Style**: Solid
7. Click **OK** to complete setting up the Y-axis.

X-Axis

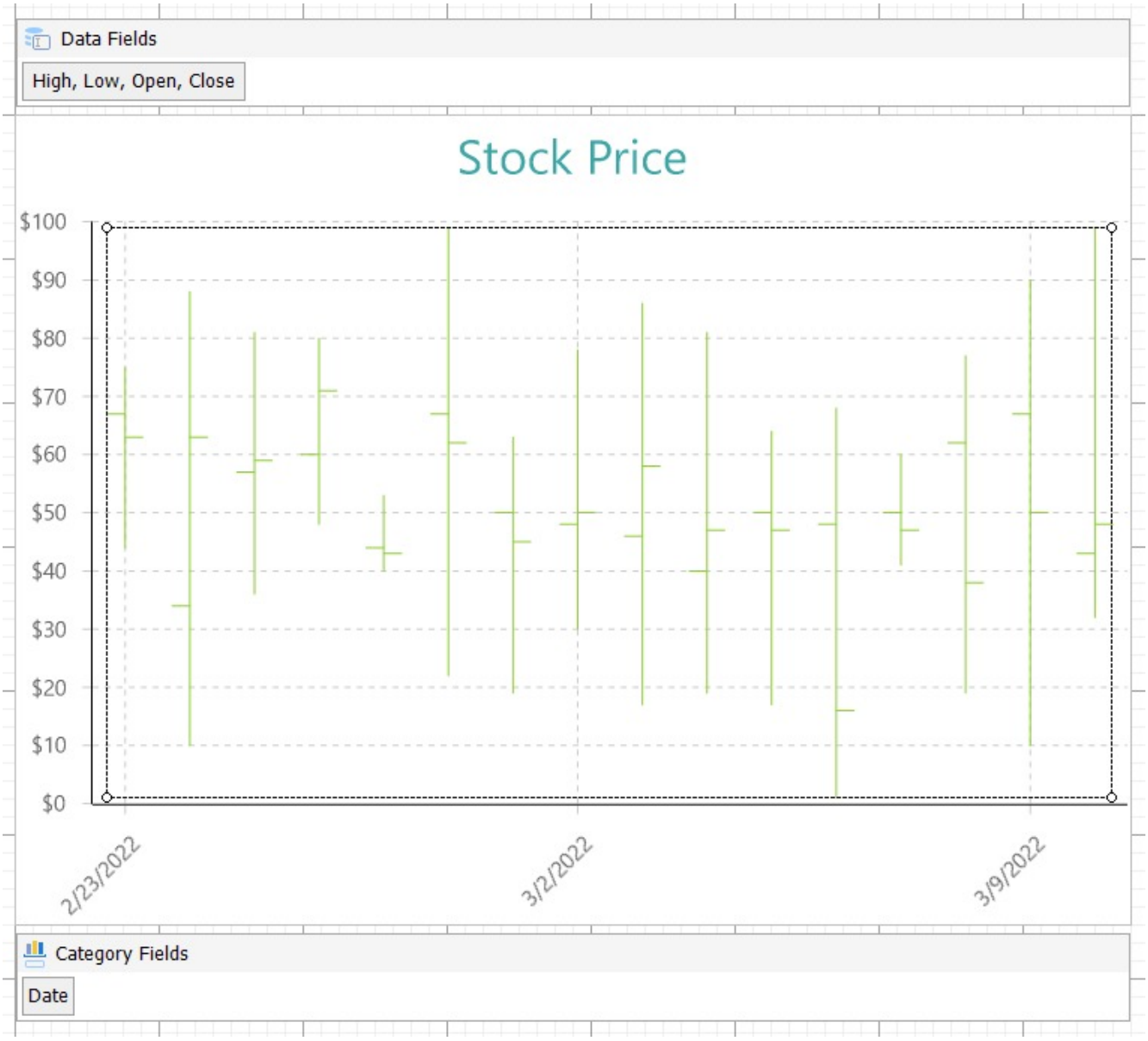
1. To open the smart panel for advanced X-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and remove the text from the **Title** field to hide the X-axis title in the chart.
3. Go to the **Labels** page > **General** tab and set the following properties.
   - **Format**: Short Date
   - **Angle**: -45
4. Now, navigate to the **Appearance** page and set the following properties.
   - **Font > Size:** 12pt
   - **Font > Color**: DimGray
5. Go to the **Line** page and set the following properties.
   - **Show Line**: Check-on
   - **Color**: Black
   - **Width**: 1pt
   - **Style**: Solid
6. Go to the **Major Gridline** page and set the following properties.
   - **Grid Interval**: 7 (to view weekly stock prices)
   - **Grid appearance > Show Grid**: Check-on
   - **Grid appearance > Width**: 0.25pt
   - **Grid appearance > Color**: #cccccc
   - **Grid appearance > Style**: Dashed
   - **Tick mark appearance > Position**: Outside
   - **Tick mark appearance > Color**: #cccccc

- **Tick mark appearance > Width**: 0.25pt
- **Tick mark appearance > Style**: Solid

7. Click **OK** to complete setting up the X-axis.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to 'Stock Price'.
3. Go to the **Font** page and set the properties as below.
   - **Size**: 24pt
   - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

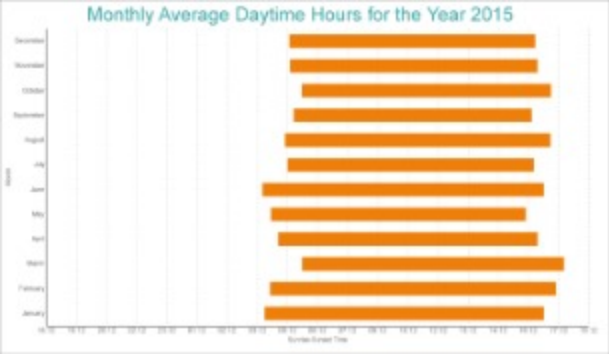
5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Range Charts

Range Charts display a pair of values (low and high) for each data point to visualize a range of values rather than a single value. These charts emphasize the distance (range) between the two values.

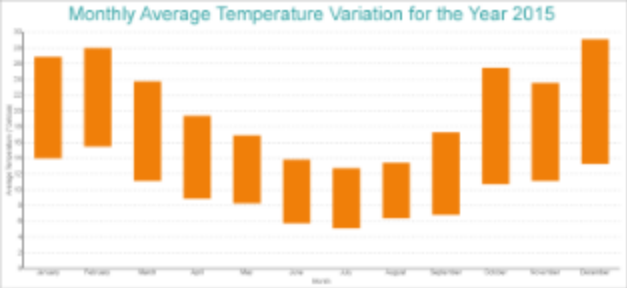
Range Bar

Range Bar Chart is a variation of regular Bar Chart. While the Range Bar Chart visualizes two values along the horizontal axis at a time — a low value and a high value, in Bar chart, the values are plotted on the horizontal axis one by one. The Create Range Bar Chart walkthrough showcases plotting the average daytime hours across one year.



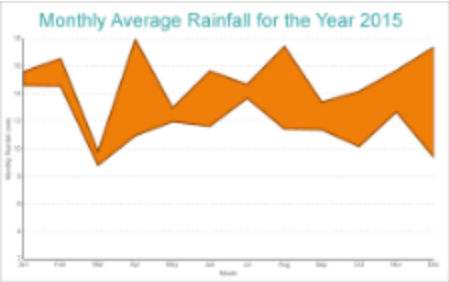
Range Column

Range Column Chart is a variation of regular Column Chart. While the Range Column Chart visualizes two values on the vertical axis at a time — a low value and a high value, in Column chart, the values are plotted on the vertical axis one by one. The Create Range Column Chart walkthrough showcases plotting the average daytime temperature variation across one year.



Range Area

Range Area chart is a variation of regular Area Chart. While the filled area in Range Area chart is between its lowest and highest values, in Area chart, the filled area is between the line segments plotted by data values and the horizontal axis. The Create Range Area Chart walkthrough showcases plotting the average rainfall across one year.



The Range Area Chart fills in the area between the top and the bottom value for each data point.

Range Bar and Column Plot Properties

The Range Bar and Range Column Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when a column or a bar is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for each column or bar.

- **Template**: The template for the data label.
- **Offset**: The pixels by which the data label should move relative to the column or the bar edge.
- **OverlappingLabels:** Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **ConnectingLine**: The line that draws connecting the column or bar edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - Position: The position of the connecting line relative to the data label. 'Auto' (default) draws the line connecting the data label and the edge of the column or the bar, while 'Center' draws the line connecting the data label to the center of the column or bar.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the columns or the bars.
  - Center: Positions the data label text on the center of the column or the bar.
  - Inside: Positions the data label text inside the column or the bar.
  - Outside: Positions the data label text outside the column or the bar.
  - Auto: The default setting, same as Outside for column and bar.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.

LineStyle

The line style for the column and bar borders.

- **LineColor**: Specify the color of the border around columns or bars.
- **LineStyle:** Specify the line style of the border around columns or bars as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around columns or bars.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

BarLineStyle

The BarLineStyle settings are applied when the **ShowBarLines** property is set to True.

- **LineColor:** Specify the color of the bar line.
- **LineStyle:** Specify the bar line style as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the bar line width.

ClippingMode

The Clipping Mode determines how a plot (columns or bars) extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area
- **Clip:** Clips off the excess bar or column lengths toward the right or the bottom
- **None:** Same as 'Fit' for bar and column plots.

Offset

The Offset is the percentage relative to the single-column width or a single bar height, by which the column and bar plots should move to the right or the bottom side, respectively.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means the columns or the bars are opaque while 0% opacity means that they are completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on the column or bar plot data points. For more

information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

ShowBarLines

Determines whether to show the connecting lines between the columns or the bars of the same Details Encoding or the same Category Encoding (if the Details Encoding is empty) across different categories.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

UnpivotData

Determines whether to display multiple data fields as a single data field. In other words, if this property is set to 'True', the columns or the bars appear stacked in case they represent multiple data fields. By default, the property is set to 'False'.

Design

BarSettings

The BarSettings property specifies the percentage relative to the column width or the bar height that should be specified in order to change the shape of the columns or bars.

- **BottomWidth**: Bottom width of the column or bar in percent.
- **NeckHeight**: Creates a neck in the otherwise rectangular columns or bars. The resulting shape of the column or bar can have a different width or height as specified in the BottomWidth or the TopWidth properties.
- **Overlap**: Percentage overlap between the adjacent columns or bars representing different data values, relative to the containing category size.
- **TopWidth**: Top width of the column or bar in percent.
- **Width**: Width of the column or the bar.

Encodings

Values Encoding

The Values encoding specifies the data values. The Values property is the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Range Bar and Range Column plots, 'Complex' is acceptable.

Subfields

The Subfields property is the collection and takes a start field that goes to the lower value and an end field that goes to the upper value. Each subfield can have its own caption.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

If multiple data values are added in the Values encoding, a clustered plot is created.

Category Encoding

The Category Encoding for Range Column and Range Bar plots forms the axis of the plot across which distinct categories are arranged. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field, but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: ExcludeNulls property determines whether to exclude the null details values in the dataset field.
- **Group**: The Group property determines how the columns or bars are arranged when divided into subcategories.
  - Stack: Stacks the subcategories vertically in the case of column plots and side-by-side in the case of bar plots. But in case of Range plots, 'Stack' is not applicable.
  - Cluster: Creates a cluster of groups for the subcategories.
  - None: It is the default value.
- **SortDirection**: The SortDirection defines the ascending or descending order in which the subcategories should be sorted.
- **SortingAggregate**: The SortingAggregate property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: The SortingField defines the order in which the subcategories are displayed.
- **Values**: The Values property is the collection and takes the field as a subcategory.

Color Encoding

The Color Encoding enables the color legend of the Category Encoding or Details Encoding to display a match between the column or bar colors and the data values. It includes the following properties:

Action

The action to take when the color legend is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values
The Values is the collection where the value of the Color expression is specified. However, the Range Column and Range Bar plots take the first item from the collection.

If the Details Encoding is empty, the Range Column and Range Bar plots calculate distinct Color Encoding results for the categories produced by the Category Encoding, convert them to the background color of the corresponding columns or bars, and display the match in the legend.

Otherwise, Range Column and Range Bar plots calculate distinct Color Encoding results for the subcategories produced by the Details Encoding, convert them to the background color of the corresponding columns or bars subsections and display the match in the legend. In both cases, plots pick up colors from the Chart Palette.

In most cases, you will use the same configuration for both Details Encoding and Color Encoding to enable a visual map of data values breakdown.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields, for example:

```
Count:{Text0}
Sum:{valueField.value}
```

Template Key

A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Range Area Plot Properties

The Range Area Plot properties discussed below can be accessed from the Properties Panel by selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when the range area plot is clicked. The action can be a hyperlink, a bookmark, or a drill-through.

Labels

The data labels for range area chart.

- **BackgroundColor**: The background color of the box containing the data label.
- **Border**: Border of the box containing the data label text. Includes LineColor, LineStyle, and LineWidth properties.
- **Color**: The color name or hex value indicating the text color of the label.
- **ConnectingLine**: The line that draws connecting the range area plot edge with the data label. Customize the appearance of the connecting line using the following properties:
  - LineColor: Specify the color of the connecting line.
  - LinePosition: Specify the position of the connecting line relative to the data label. Customize the position of connecting line with Auto and Center properties.
  - LineStyle: Specify the line style as 'Dashed', 'Dotted', 'Double', etc.
  - LineWidth: Specify the width of the connecting line.
- **Font**: The font used to render the text of the label. Customize the text font using the Font Family, Font Size, Font Style, and Font Weight properties.
- **Offset**: It gets or sets the text offset of the data label in pixel.
- **OverlappingLabels**: Indicates the handling of labels in case they overlap. The property takes the following values:
  - Auto: Hides labels that overlap.
  - Show: Shows the labels even if the labels overlap.
  - Hide: Hides the labels that overlap.
- **Template**: The template for the data label.
- **TextDecoration**: Decorate the data label text with an Underline, a DoubleUnderline, an Overline, or a LineThrough.
- **TextPosition**: The position of the data label text relative to the range area plot.
  - Center: Positions the data label text on the center of the range area chart.
  - Inside: Positions the data label text inside the range area chart.
  - Outside: Positions the data label text outside the range area chart.
  - Auto: The default setting, same as Outside for range area chart.

LineStyle

The line style for the range area plot borders.

- **LineColor**: Specify the color of the border around the range area plot.
- **LineStyle:** Specify the line style of the border around the range area plot as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around the range area plot.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Tooltip Template

Contains the tooltip template settings. You can choose from the list of predefined settings or set your own in the Expression Editor.

Configurations

ClippingMode

The Clipping Mode determines how a plot extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area.
- **Clip:** Clips off the excess area lengths toward the right or the bottom.
- **None:** Same as 'Fit' for range area plots.

LineAspect

The Line Aspect determines the line style that connects data points.

- **Default:** Indicates a straight line.
- **Spline:** Indicates a bezier curve.
- **StepCenter, StepLeft, and StepRight:** Indicates a stepped line with different step directions.

Opacity

The Opacity determines the opacity of areas filled with color. 100% opacity means that the plot fill color is opaque while 0% opacity means that the plot fill color is completely transparent.

Overlays

Overlays property is a collection for superimposing the data trend on a plot. For more information, see Trendlines topic.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules.

ShowNulls

Represents how null or empty values should be shown in the plot - Gaps (default), Connected, or Zeros.

UnpivotData

Determines whether to display multiple data fields as a single data field. By default, the property is set to 'False'.

Encodings

Category Encoding

The Category Encoding of an Area plot creates an area between the axis and the line generated from the connected data points representing the periodic change. The Category encoding includes the following properties.

Values

The Values property is the collection and takes the field as a category.

SortingField

The SortingField defines the order in which the categories are displayed. It takes the default same as the Values field,

but you can also specify another field to sort the categories.

SortDirection

The SortDirection defines the ascending or descending order in which the categories should be sorted.

SortingAggregate

The SortingAggregate property specifies the aggregate to use for sorting the categories.

Color Encoding

The Colors Encoding enables the color legend of the Details or Category Encoding. It includes the following properties:

Action

The action to take when the report item is clicked.

Aggregate

Aggregates the value of Color expression.

ShowValuesName

If set to True, the legend is displayed based on the value specified in Details encoding or Color encoding.

Values

The Values is the collection where the value of the Color expression is specified. However, the area plots take the first item from the collection.

Details Encoding

The Details Encoding breaks down the data values into subcategories and produces additional groups. The Details property is the collection of items and each item includes the following properties that define the Details encoding:

- **ExcludeNulls**: This property determines whether to exclude the null details values in the dataset field.
- **Group**: This property determines the area subsection arrangement of the plot. In Range plots, use 'Cluster' instead of 'Stack'.
- **SortDirection**: This property determines the sorting direction, ascending or descending in which the subcategories should be sorted.
- **SortingAggregate**: This property specifies the aggregate to use for sorting the subcategories.
- **SortingField**: This property defines the order in which the subcategories are displayed.
- **Values**: The Values property is the collection and takes the field as a subcategory.

Values Encoding

The Values encoding specifies the data values, and represents the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Range Area plots, 'Complex' is acceptable.

Subfields

The Value property is the collection and usually takes a bound field. However, the Range Area plots take the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Text Encoding

The Text Encoding provides support of any dataset field in chart labels and tooltips and allows displaying additional info on the chart. When two text encodings are added, they are displayed by default with the ";" delimiter. A text encoding includes the following properties:

Aggregate

Aggregates the value of the Text expression.

Target

Specifies whether this text is for a Label or a Tooltip Label. The label format is controlled with the chart's **Labels > Template** property or **Tooltip Template** property.

The Template property can use both the predefined values and the added text encoding fields.

Template Key

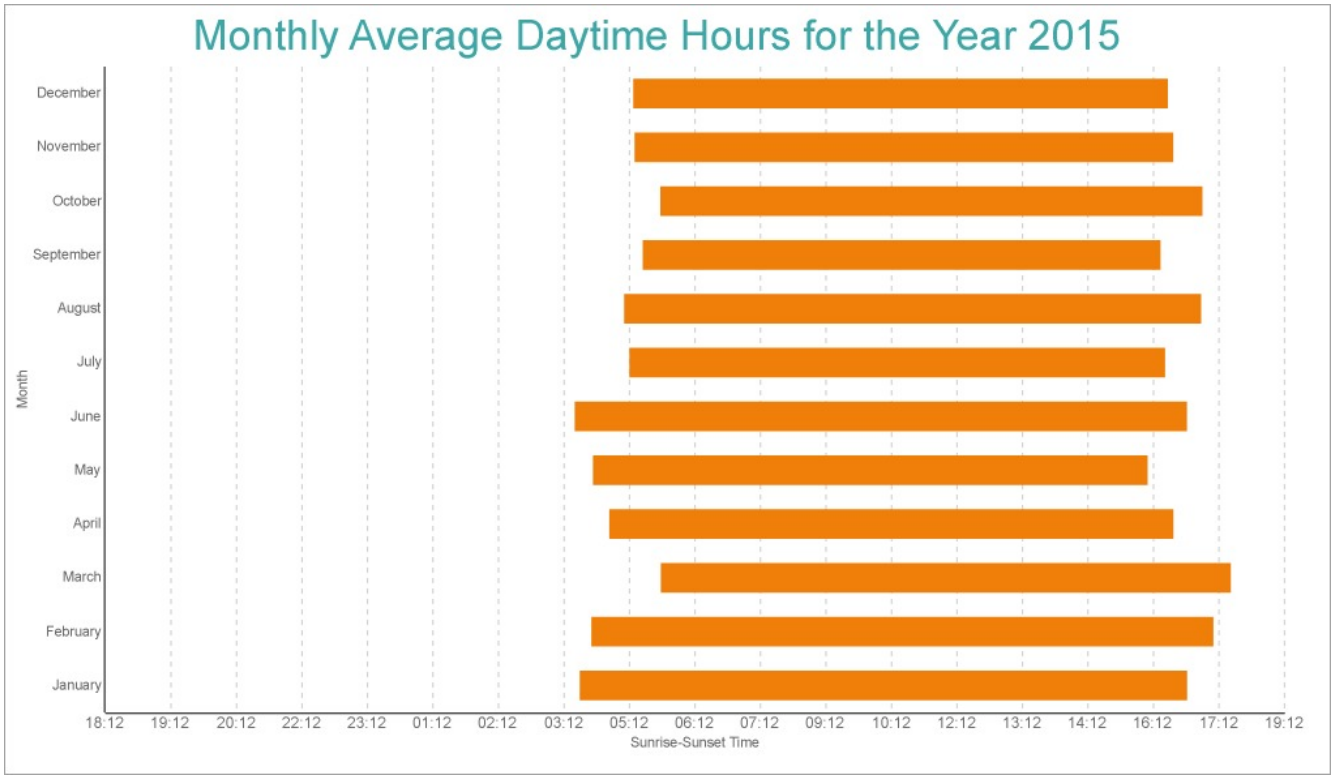
A unique key, used in the chart plot's **Labels > Template** and **Tooltip Template** to access the text encoding value.

Value

A field, constant or expression to be displayed.

Create Range Bar Chart

This walkthrough shows creating a Range Bar chart. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Csv Provider'.
   For more information, see the CSV topic.
3. Go to the **Connection String** tab and click Build
4. In the **Path** field, navigate to the CSV file with content: **weatherdata.csv (on-line documentation)**
5. Click **Get from preview** and edit the **Data Type** column for the following fields as shown.

Name	Data Type
DATE	DateTime
CITY	String
AVG MIN TEMP	Float
AVG MAX TEMP	Float
RAINFALL Min	Float
RAINFALL Max	Float
SUNRISE	DateTime
SUNSET	DateTime

6. Click **OK** to view the generated connection string, which will be similar to the following:

Connection String
`Path=C:\\Data\\weatherdata.csv;Locale=en-US;TextQualifier=";ColumnsSeparator=,;RowsSeparator=\r\n;Columns=DATE(DateTime),CITY,AVG MIN TEMP,AVG MAX TEMP,RAINFALL Min,RAINFALL Max,SUNRISE,SUNSET;HasHeaders=True`

7. Click **OK** to complete connecting to data.

Add Report Parameters

CityParameter

1. In the **Report Explorer**, right-click **Parameters** and select **Add Parameters**.
2. In the **Report - Parameters** dialog's **General** tab, add a name for the parameter, 'CityParameter'.
3. Ensure that the **Data type** matches that of the field (here, String).
4. In the **Available Values** tab, select **Non-queried** and enter:
   - **Label**: Melbourne Airport
   - **Value**: MelbourneAirport
   - **Order By** > **Condition**: Value
   - **Order By** > **Direction**: Ascending
5. In the **Default Values** tab, select **Non-queried** and add 'MelbourneAirport' (same as the Value in Available Value tab).

YearParameter

1. Add another parameter and in the **General** tab, add a name for the parameter, 'YearParameter'.
2. Ensure that the **Data type** matches that of the field (here, Integer).

3. In the **Available Values** tab, select Non-queried and enter:
    - **Label**: 2015
    - **Value**: 2015
    - **Order By** > **Condition**: Value
    - **Order By** > **Direction**: Ascending
4. In the **Default Values** tab, select **Non-queried** and add '2015' (same as the Value in Available Value tab).

Add Filter to Dataset

1. Right-click the 'weatherdata' dataset and select **Edit**.
2. Go to **Filters** page and add following two filters:

S.no.	Expression	Operator	Value
1.	`=Fields!CITY.Value`	Equal	`=Parameters!CityParameter.Value`
2.	`=Year(Fields!DATE.Value)`	Equal	`=Parameters!YearParameter.Value`

3. Now that the filters are added in the dataset, click **OK**.

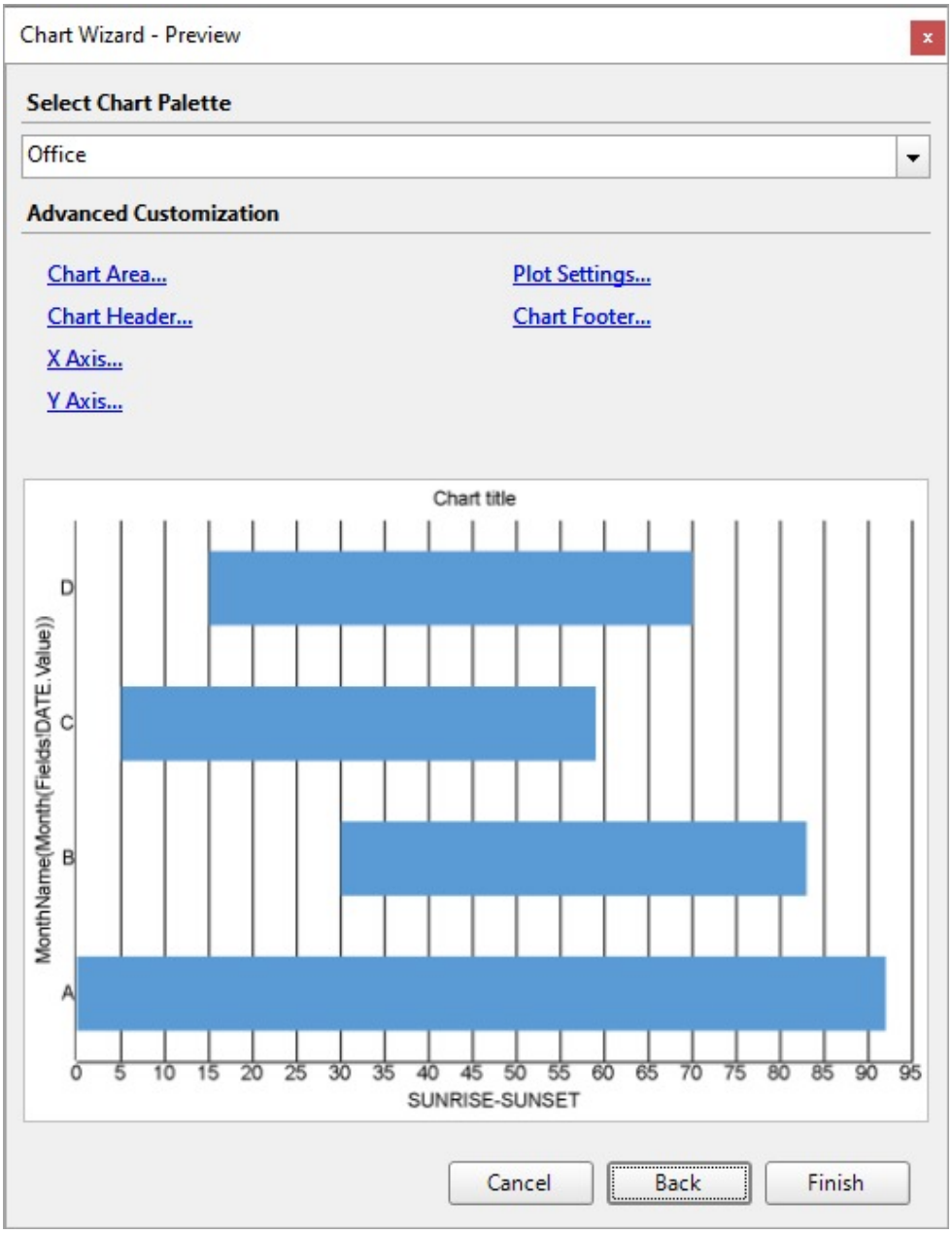
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset name** as 'weatherdata' and the **Chart Type** as 'Range Bar'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Range Bar chart.
4. In **Choose Data Values** section, we will define the start and end fields.

Start Field	End Field
`=Fields!SUNRISE.Value`	`=Fields!SUNSET.Value`

5. In **Choose Data Categories**, enter the field as `=MonthName(Month([DATE]))`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page and in the **Sorting** > **Sorting field**, select `=Fields!DATE.Value` field and set the **Sort direction** to 'Ascending'.

3. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page. Enter the text in the **Title** field as 'Sunrise-Sunset Time' and set **Color** to 'DimGray'.
3. Go to the **Labels** page > **General** tab and set **Format** to 'HH:MM'.
4. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: DimGray
5. Go to the **Major Gridline** page and set the following properties.
    - **Show grid**: Check-on
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Style**: Dashed
6. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and edit the text in the **Title** field to 'Month' and set Color to 'DimGray'.
3. Go to the **Labels** page > **Appearance** tab and set **Font > Color** to DimGray.

Chart Palette

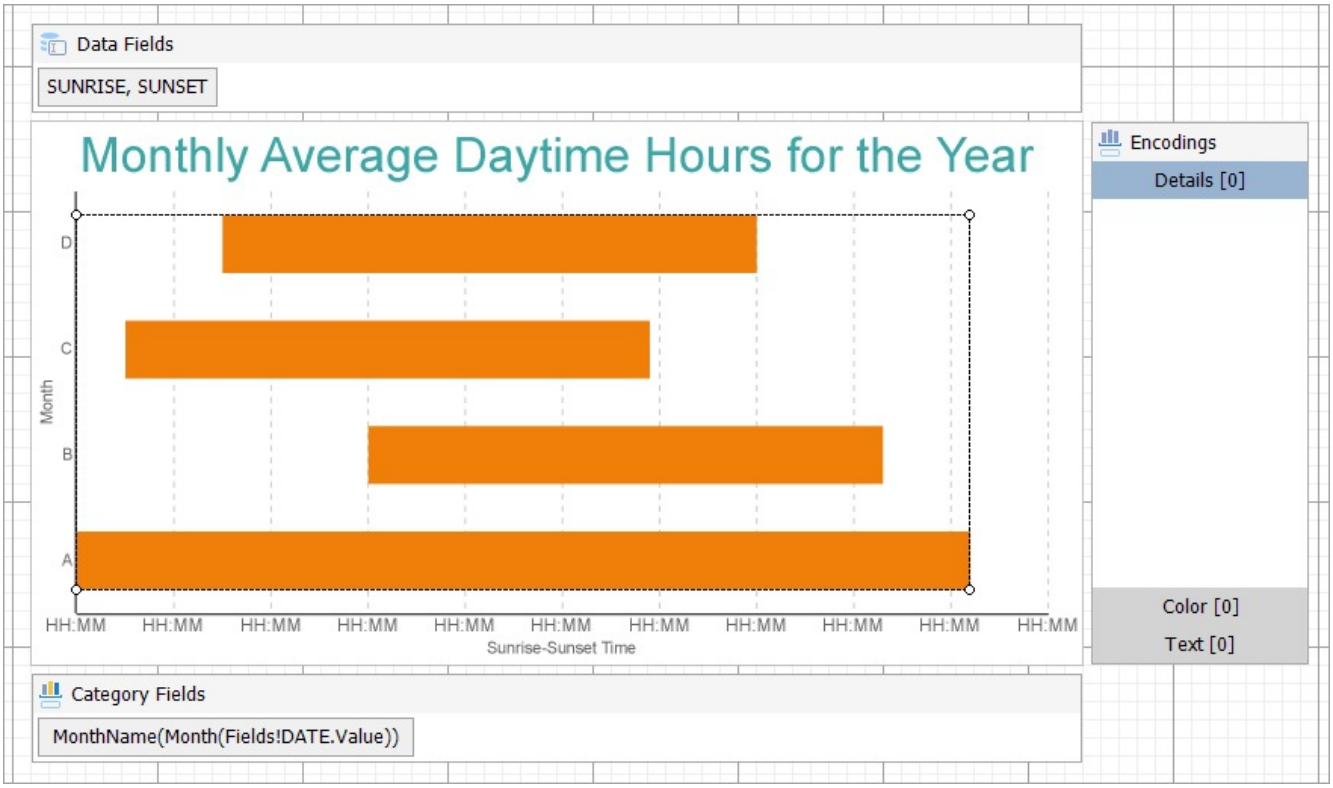
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select 'Aspect'
3. Click **OK** to complete setting up the chart palette.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set the **Title** to the following expression:

```
="Monthly Average Daytime Hours for the Year " & Parameters!YearParameter.Value
```

3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

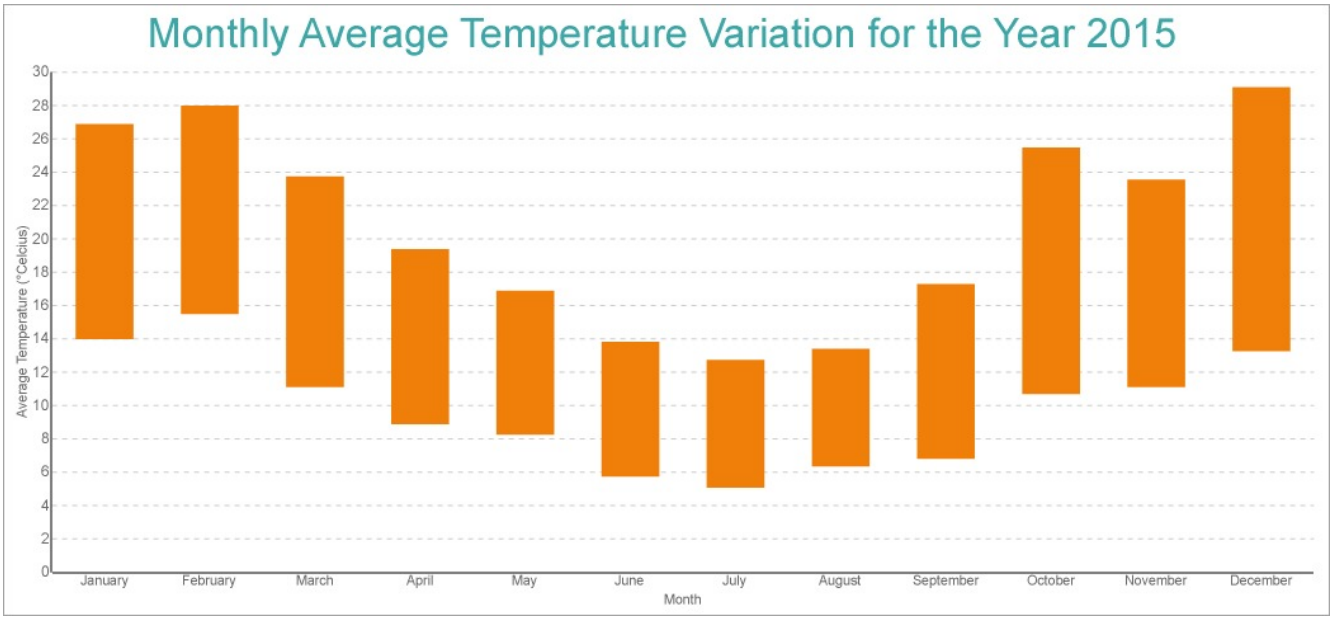


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Range Column Chart

This walkthrough shows creating a Range Column chart. The final chart appears like this:



Create a Report and Bind the Report to Data

See this section on creating the report and binding the report to the data.

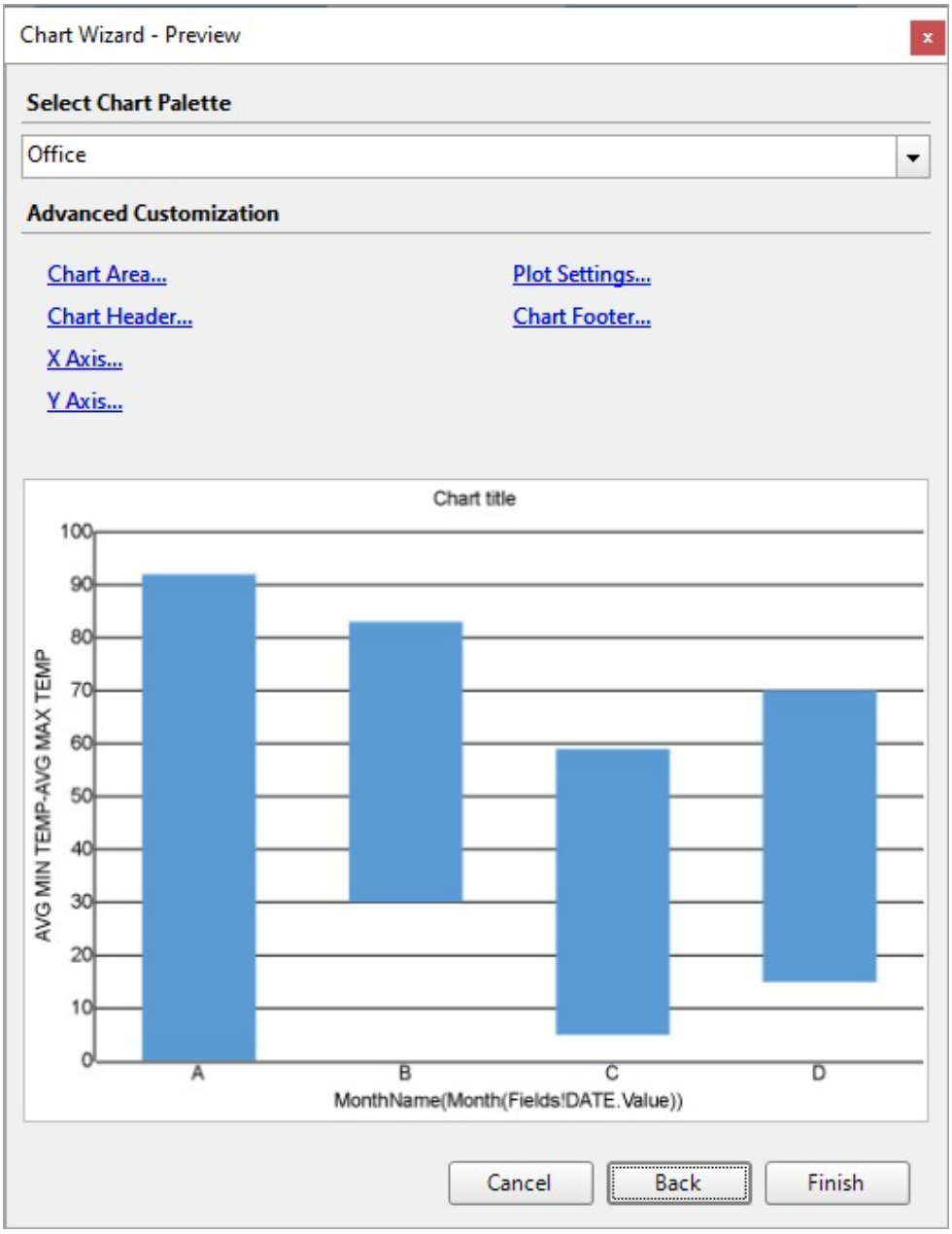
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset name** as 'weatherdata' and the **Chart Type** as 'Range Column'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Range Column chart.
4. In **Choose Data Values** section, we will define the start and end fields.

Start Field	End Field
`=Fields.Item("AVG MIN TEMP").Value`	`=Fields.Item("AVG MAX TEMP").Value`

5. In **Choose Data Categories**, enter the field as `=MonthName(Month(Fields!DATE.Value))`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page and in the **Sorting** > **Sorting field**, select `=Fields!DATE.Value` field and set the **Sort direction** to 'Ascending'.

3. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page. Enter the text in the **Title** field as 'Average Temperature (°Celsius)' and set **Color** to 'DimGray'.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
    - **Font > Size:** 9pt
    - **Font > Color**: DimGray
4. Go to the **Major Gridline** page and set the following properties.
    - **Show grid**: Check-on
    - **Grid appearance > Color**: #cccccc
    - **Grid appearance > Width**: 0.25pt
    - **Grid appearance > Style**: Dashed
5. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and edit the text in the **Title** field to 'Month' and set Color to 'DimGray'.
3. Go to the **Labels** page > **Appearance** tab and set **Font > Color** to DimGray.

Chart Palette

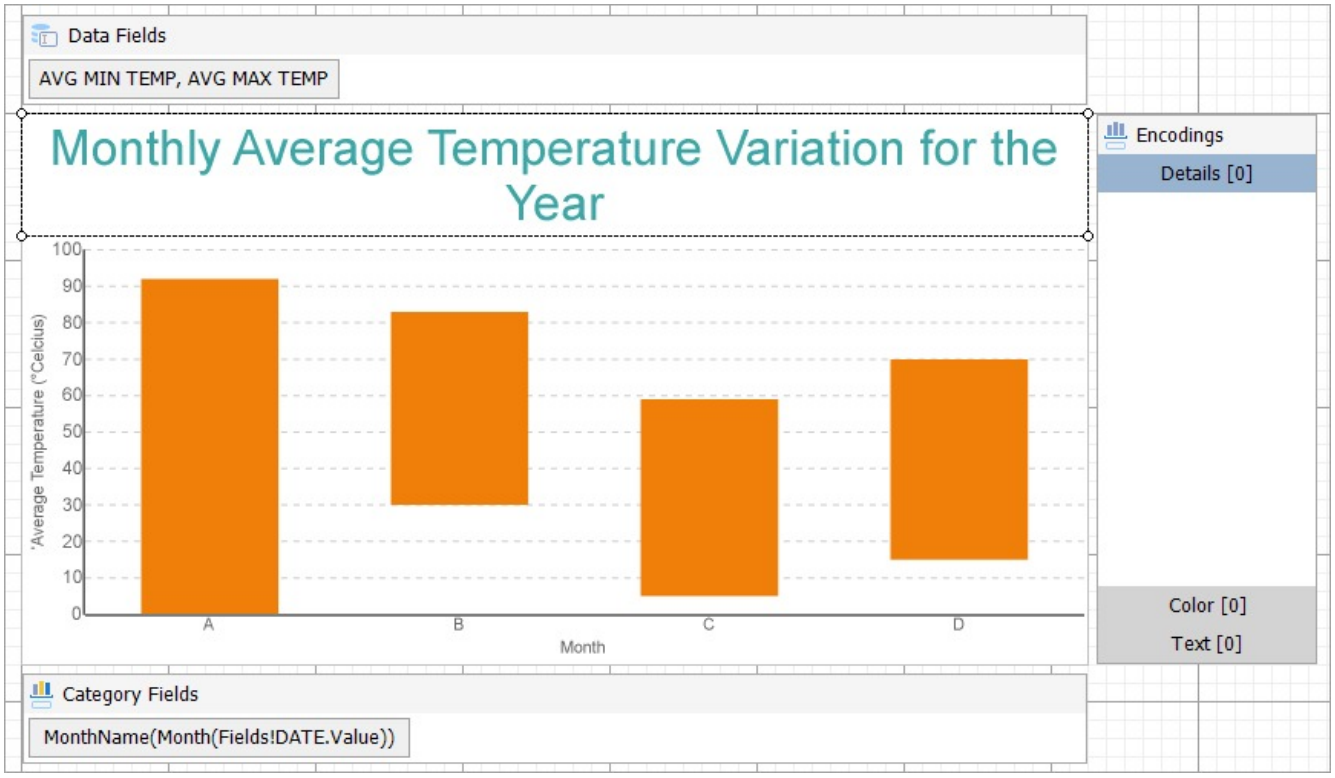
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select 'Aspect'
3. Click **OK** to complete setting up the chart palette.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set the **Title** to the following expression:

```
="Monthly Average Temperature Variation for the Year " & Parameters!YearParameter.Value
```

3. Go to the **Font** page and set the properties as below.
    - **Size**: 24pt
    - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.

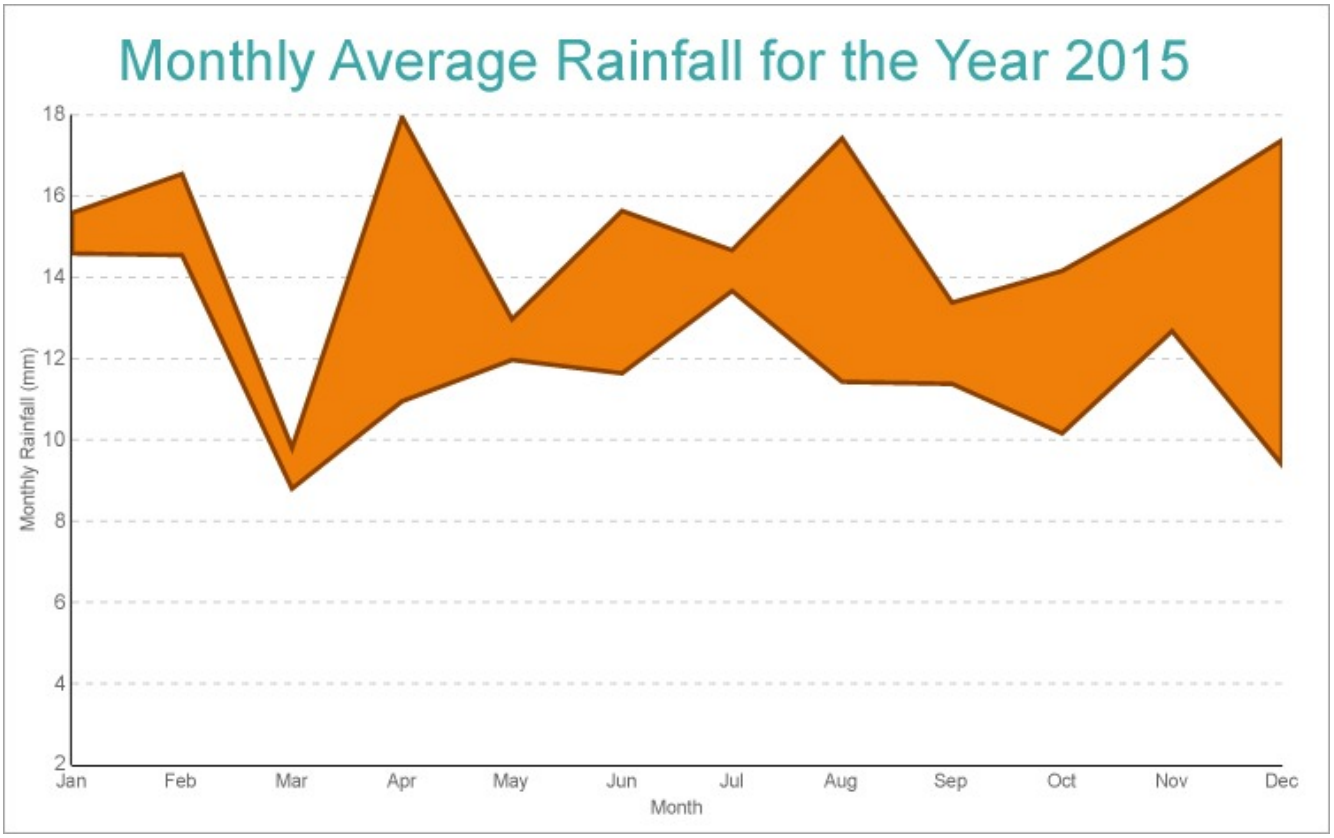


Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Create Range Area Chart

This walkthrough shows creating a Range Area chart. The final chart appears like this:



Create a Report and Bind the Report to Data

See this section on creating the report and binding the report to the data.

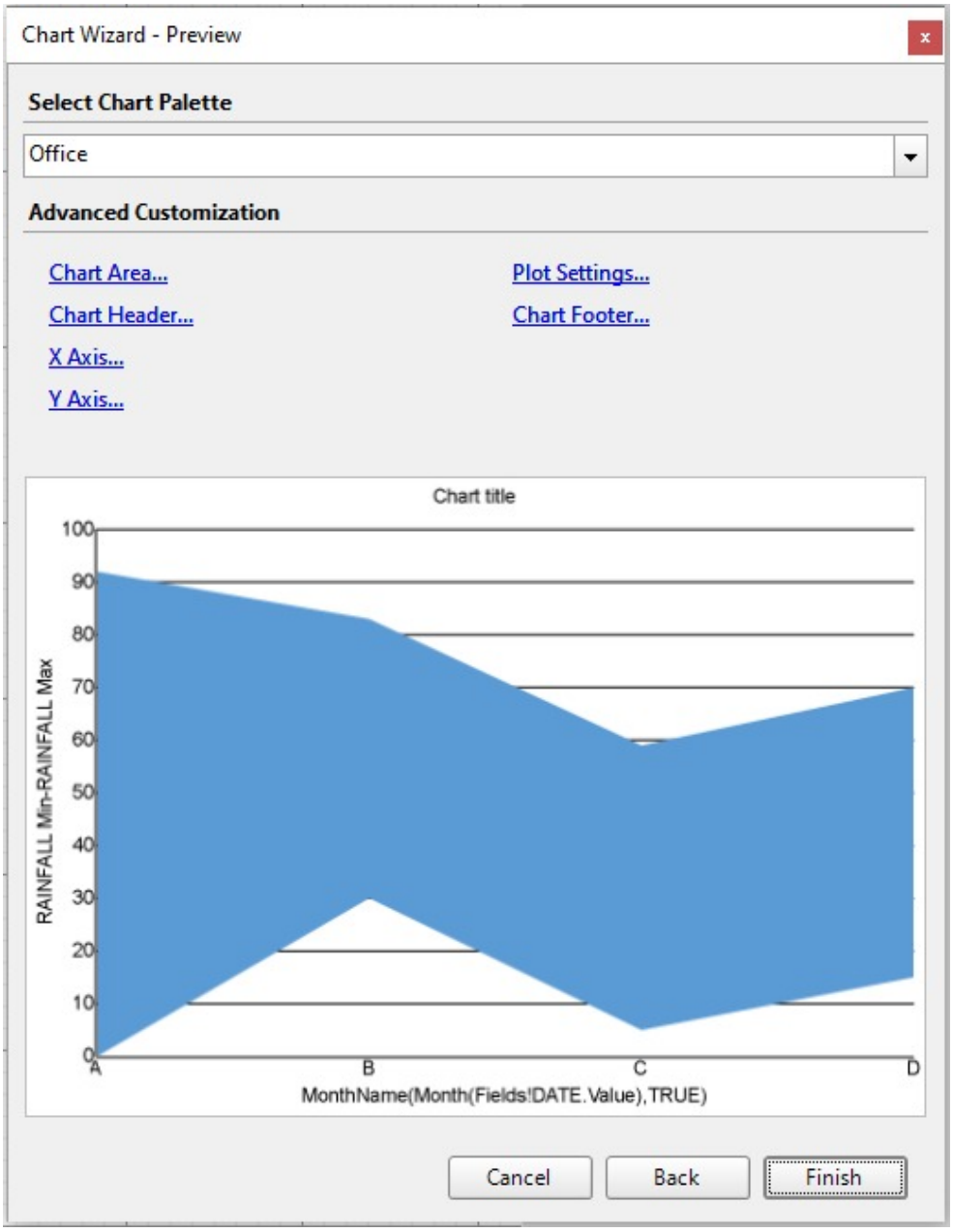
Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset name** as 'weatherdata' and the **Chart Type** as 'Range Area'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Range Area chart.
4. In **Choose Data Values** section, we will define the start and end fields.

Start Field	End Field
`=Fields.Item("RAINFALL Min").Value`	`=Fields.Item("RAINFALL Max").Value`

5. In **Choose Data Categories**, enter the field as `=MonthName(Month(Fields!DATE.Value),TRUE)`. We will add more customizations to the category in later steps.
6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or, you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Categories** page and in the **Sorting** > **Sorting field**, select `=Fields!DATE.Value` field and set the **Sort**

direction to 'Ascending'.
3. Go to Appearance page and set **Line Style** >**:**
   - **Style**: Solid
   - **Color**: #8B400
   - **Width**: 2pt
4. Click **OK** to complete setting up the plot.

Y-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'Y-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page. Enter the text in the **Title** field as 'Monthly Rainfall (mm)' and set **Color** to 'DimGray'.
3. Go to the **Labels** page > **Appearance** tab and set the following properties.
   - **Font > Size:** 9pt
   - **Font > Color**: DimGray
4. Go to the **Major Gridline** page and set the following properties.
   - **Show grid**: Check-on
   - **Grid appearance > Color**: #cccccc
   - **Grid appearance > Width**: 0.25pt
   - **Grid appearance > Style**: Dashed
5. Click **OK** to complete setting up the Y-axis.

X-Axis

1. To open the smart panel for advanced Y-axis settings, right-click 'X-axis' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Title** page and edit the text in the **Title** field to 'Month' and set Color to 'DimGray'.
3. Go to the **Labels** page > **Appearance** tab and set **Font > Color** to DimGray.

Chart Palette

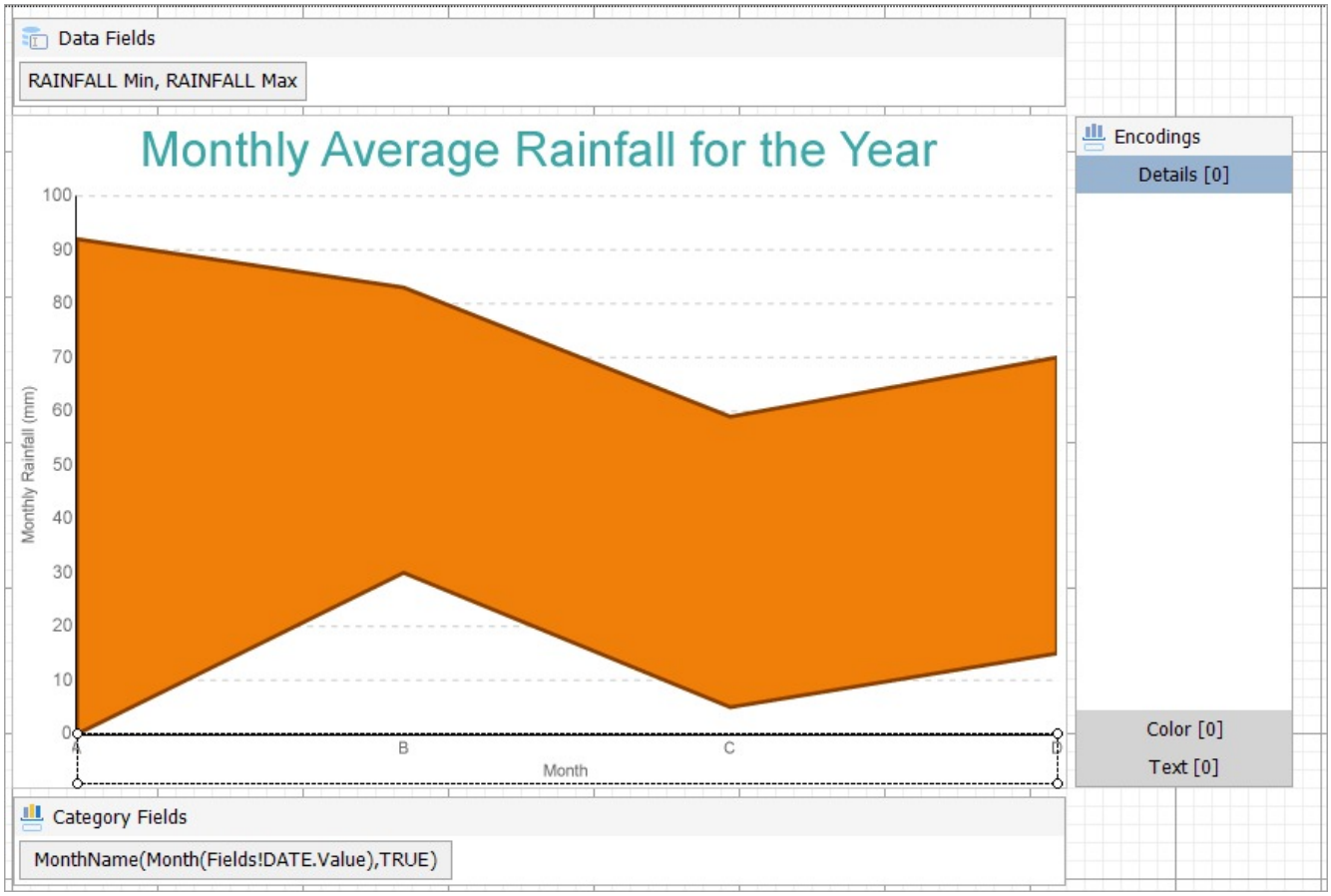
1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Palette** page and select 'Aspect'
3. Click **OK** to complete setting up the chart palette.

Chart Header

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set the **Title** to the following expression:

```
="Monthly Average Rainfall for the Year " & Parameters!YearParameter.Value
```

3. Go to the **Font** page and set the properties as below.
   - **Size**: 24pt
   - **Color**: #3da7a8
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Gauge Chart

A Gauge chart is a Bar chart with a radial axis and overlapping bars to show ranges. It uses the needle and the dial to represent the data. The direction of the needle and the colors in the dial shows the performance at a glance.

Gauge Plot Properties

The Gauge Plot properties discussed below can be accessed from the Properties Panel on selecting the **Chart** > **Plot** from the **Report Explorer**. You can access some of the important properties from the Plot's Smart Panel too. See Chart Smart Panels topic for more information. Both the Properties Panel and the Smart Panel can be accessed by right-clicking the plot in the design area.

Common

Action

The action to perform when the plot is clicked. The action can be a hyperlink, a bookmark, a drill-through, or a slice. For more details on each of these actions, see

LineStyle

The line style for the column and bar borders.

- **LineColor**: Specify the color of the border around columns or bars.
- **LineStyle:** Specify the line style of the border around columns or bars as 'Dashed', 'Dotted', 'Double', etc.
- **LineWidth:** Specify the line width of the border around columns or bars.

Name

The name of the plot. By default, a chart containing a single plot has the plot name as 'Plot1'.

Configurations

ClippingMode

The Clipping Mode determines how a plot (columns or bars) extends within the plot area. The ClippingMode can be set to

- **Fit:** Utilizes the free space to fit the plots within the plot area
- **Clip:** Clips off the excess bar or column lengths toward the right or the bottom

- **None:** Same as 'Fit' for bar and column plots.

Gauge Labels

Gauge labels collection that lets you set the labels, usually to display the exact pointer value.

- **OffsetX** and **OffsetY**: Positions label relative to chart center.
- **Text**: The text for the label. It supports expression format and can use aggregate functions such as `=Sum(Fields!Quantity.value)`.
  By default, when a field from dataset is drag and dropped to Labels region of the Gauge chart, expression includes 'Sum' (`=Sum(Fields!Quantity.value)` functions, since it is likely to be used most commonly.

Gauge Pointers

Gauge pointers collection that lets you customize the needle and the needle pin.

- **End**: Specifies where the needle will point to. The value is same as the Data Field. By default, when a field from dataset is dragged and dropped to Data Fields region of Gauge chart, the expression includes 'Sum' (`=Sum(Fields!Quantity.value)` function, since it is likely to be used most commonly.
- **NeedlePinWidth**: Width of the needle pin.
- **NeedleWidth**: Width of needle.

InnerRadius

Inner radius of the gauge is the percentage relative to the chart's outer radius.

Opacity

The Opacity is the percentage value of the opacity of the plot fill color. 100% opacity means the columns or the bars are opaque while 0% opacity means that they are completely transparent.

Rules

Rules control the appearance of plots based on specified conditions. For more information on conditions, see Rules topic.

StartAngle

Indicates the arc angle that defines the clockwise rotation of the chart. For Gauge charts, the default is '-45' degrees. A full rotation makes 360 degrees.

Sweep

Indicates the arc degrees from 0 to 360 which decides the length of the arc occupied by the plot. For Gauge charts, the default is '270' degrees.

Encodings

Gauge chart has only one encoding type.

Values Encoding

The Values encoding specifies the data values. This encoding plots the ranges on the gauge's dial. It is important to note that the data value fields should be added in the descending order, for example, 800 > 600 > 400 >200 or the actual field names. In design time, the Values encoding is located below the chart, in the 'Ranges' section. See the Create Gauge Chart walkthrough for more information.

The Values property is the collection of items and each item includes the following properties.

Type

The Type property provides 'Simple' and 'Complex' options to choose from. However, for Gauge plots, 'Simple' is acceptable.

Value

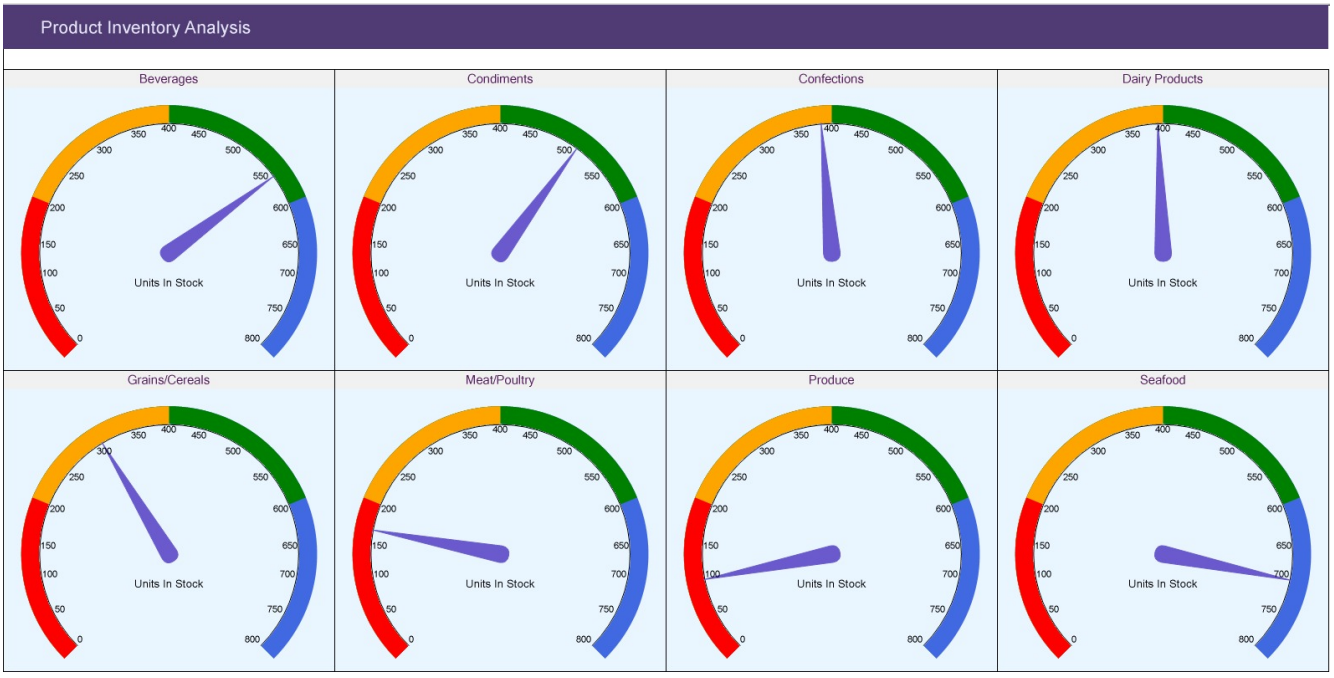
The Value property is the collection and usually takes a bound field. However, the Gauge plot takes the first item from the collection.

Aggregate

To show aggregated values such as Average, Count, and Sum instead of individual values, specify an Aggregate function.

Create Gauge Chart

This walkthrough shows creating a Gauge chart. The Gauge chart is placed in a multi-column list to show the units in stock in each category of products. The final chart appears like this:



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/northwind/api/v1/Products
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'Products'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.[*]`

3. Click **OK** to save the changes.

Similarly, we will connect to another data source from where we will fetch `[categoryName]` corresponding to the `[categoryId]` in 'Products' dataset, in the Chart Header.

The data source URL is https://demodata.mescius.io/northwind/api/v1/Categories and the dataset query is `$.[*]`. Let's name this dataset, 'Categories'.

Design Report Layout

1. Drag-drop the **List** data region.
2. In the Properties panel, set the following properties:

Property	**Value**
DataSetName	Products
GrowDirection	Column
RowsOrColumnsCount	4

3. With **List** data region selected, click **Property dialog**.
4. Go to **Detail Grouping** page, and in the **General** tab, provide a **Name** to the group and in the **Group on: Expression**, select `=[categoryId]`.
5. Go to the **Sorting** tab and add a sorting **Expression** `=[categoryId]` and **Direction** as 'Ascending'.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values and customization. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region inside the first column of the List data region. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
   You may want to increase the size of the List data region to accommodate the chart.
2. Select the **Dataset Name** as 'Products' and the **Chart Type** as 'Gauge'.
3. Click **Next** to proceed. Here, you need to specify the settings for the Gauge chart.
4. In **Gauge Ranges** section, we will define the constant data values to display the ranges on the gauge's dial. Add four data values and enter the **Field** as follows:

- 800
- 600
- 400
- 200

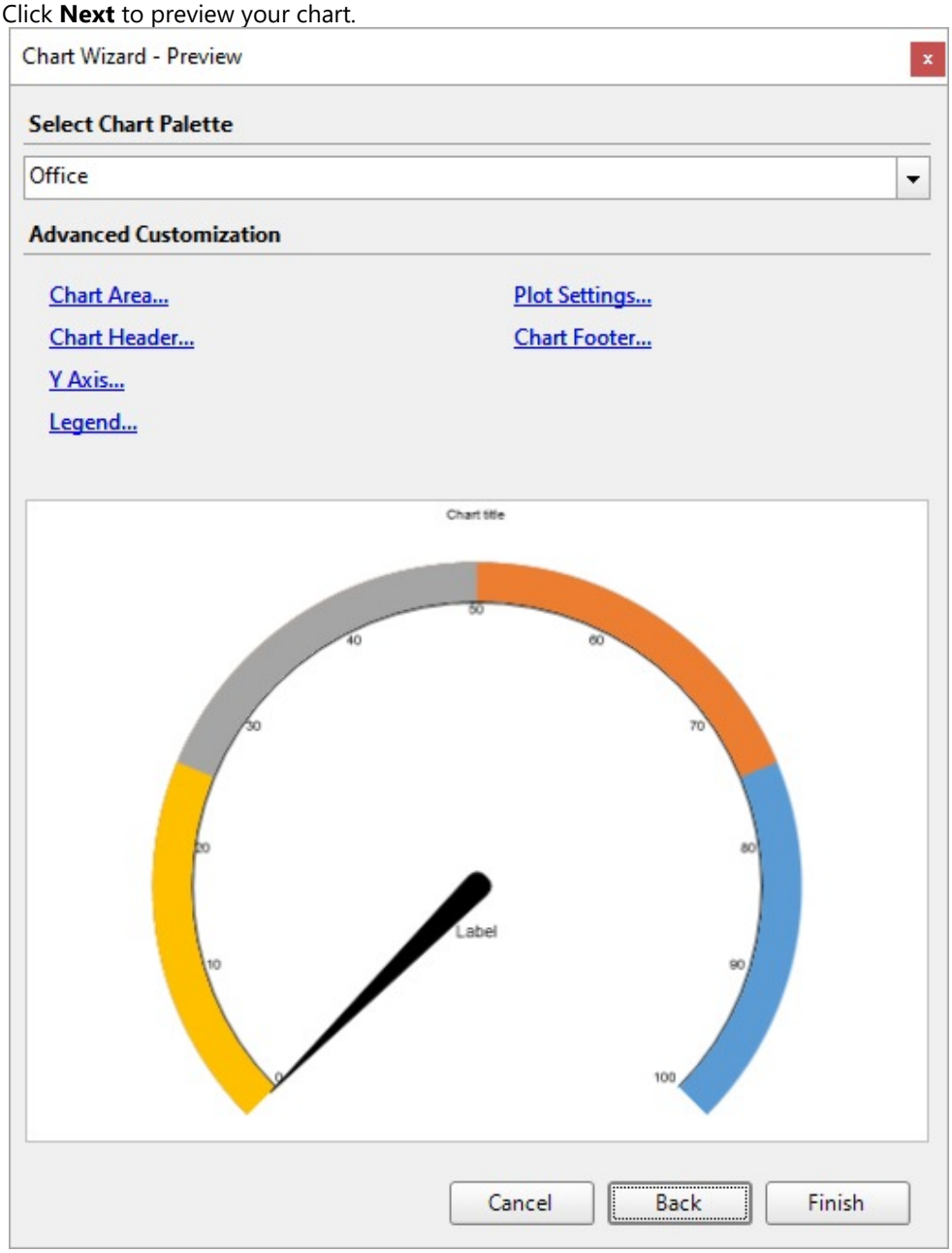
  Set **Aggregate** as 'Max' for all the above fields.

> **Note**: The first field should be the one with greater value.

5. In the **Gauge Pointer** section, enter the Expression: `=Sum(Fields!unitsInStock.Value)`
   This expression defines the gauge pointer's **End** configuration.

   We will add more customizations to the category in later steps

6. Click **Next** to preview your chart.



You can also modify the chart palette and do other customizations as the last step in the process of chart creation. Or,

you can exit the wizard and access these smart panels as explained below.

Set Advanced Customization

Now that the chart is configured with data values, let us do some customizations on the chart elements using the smart panels.

Plot Settings

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. Go to the **Labels** page and update **Expression** to 'Units in Stock'. You can also update the **Offset Y** and **Offset X** settings.
3. Click **OK** to complete setting up the plot.

Chart Palette

1. To open the smart panel for advanced chart settings, right-click 'Chart' on the Report Explorer and choose **Property Dialog**.
2. In **Palette**, select **Custom** from the drop-down and add the following four colors for each value.
    - RoyalBlue
    - Green
    - Orange
    - Red
3. Click **OK** to complete setting up the custom chart palette.

Gauge Pointer

1. To set a needle color, go to **Gauge Pointers** property.
2. In the **PointerDesigner Collection Editor**, go to **BackgroundColor** property and set 'SlateBlue'.

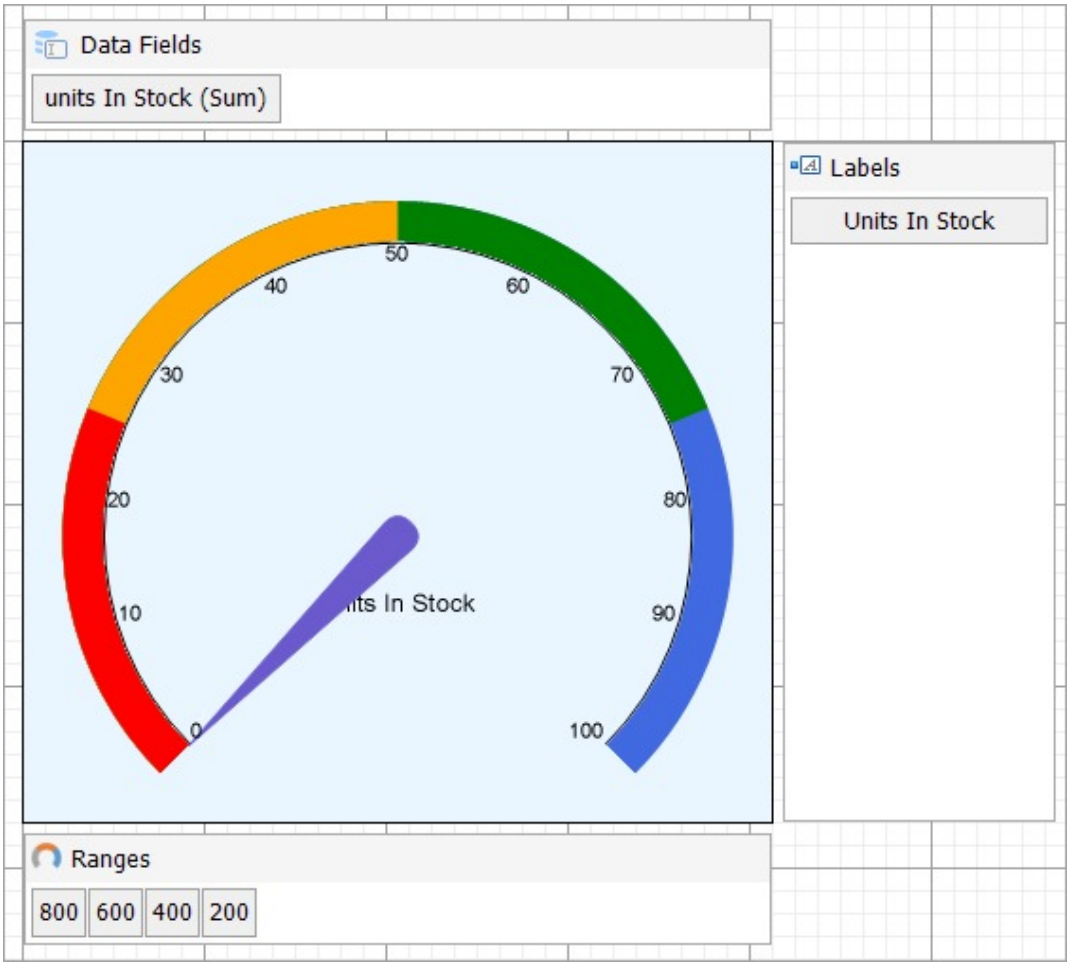
Chart Header

We want chart header to display the category name from another dataset, 'Categories'. For this, we will use Lookup expression.

1. To open the smart panel for the chart header, right-click 'Header' on the Report Explorer and choose **Property Dialog.**
2. Go to the **General** page and set **Title** to the following expression:

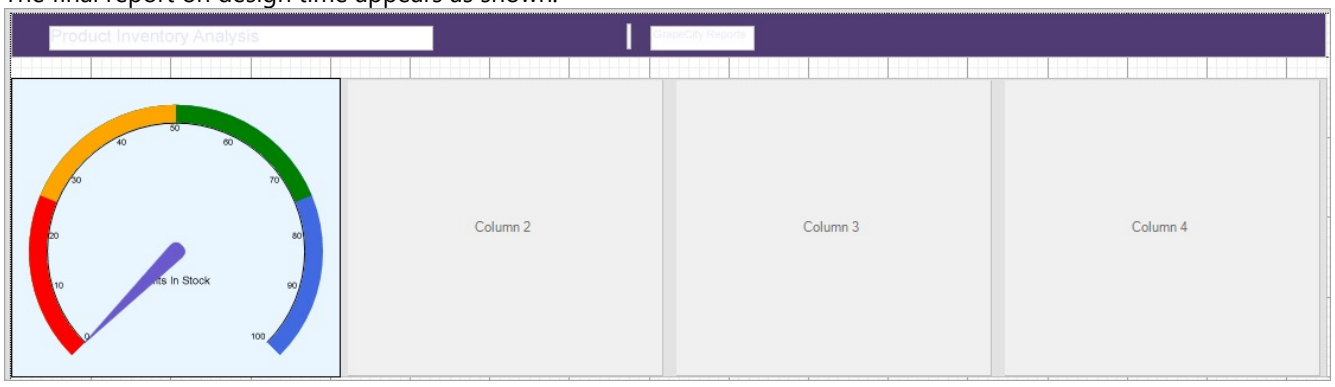
```
=Lookup([categoryId],[categoryId],[categoryName],"Categories")
```

3. Go to the **Font** page and set the properties as below.
    - **Size**: 11pt
    - **Color**: #551e5f
4. Click **OK** to complete setting up the chart header.
   You may want to resize the chart.



Note: We use stub data at design time and not real data. So to view the actual final chart, you need to view the chart on the preview.

The final report on design time appears as shown.

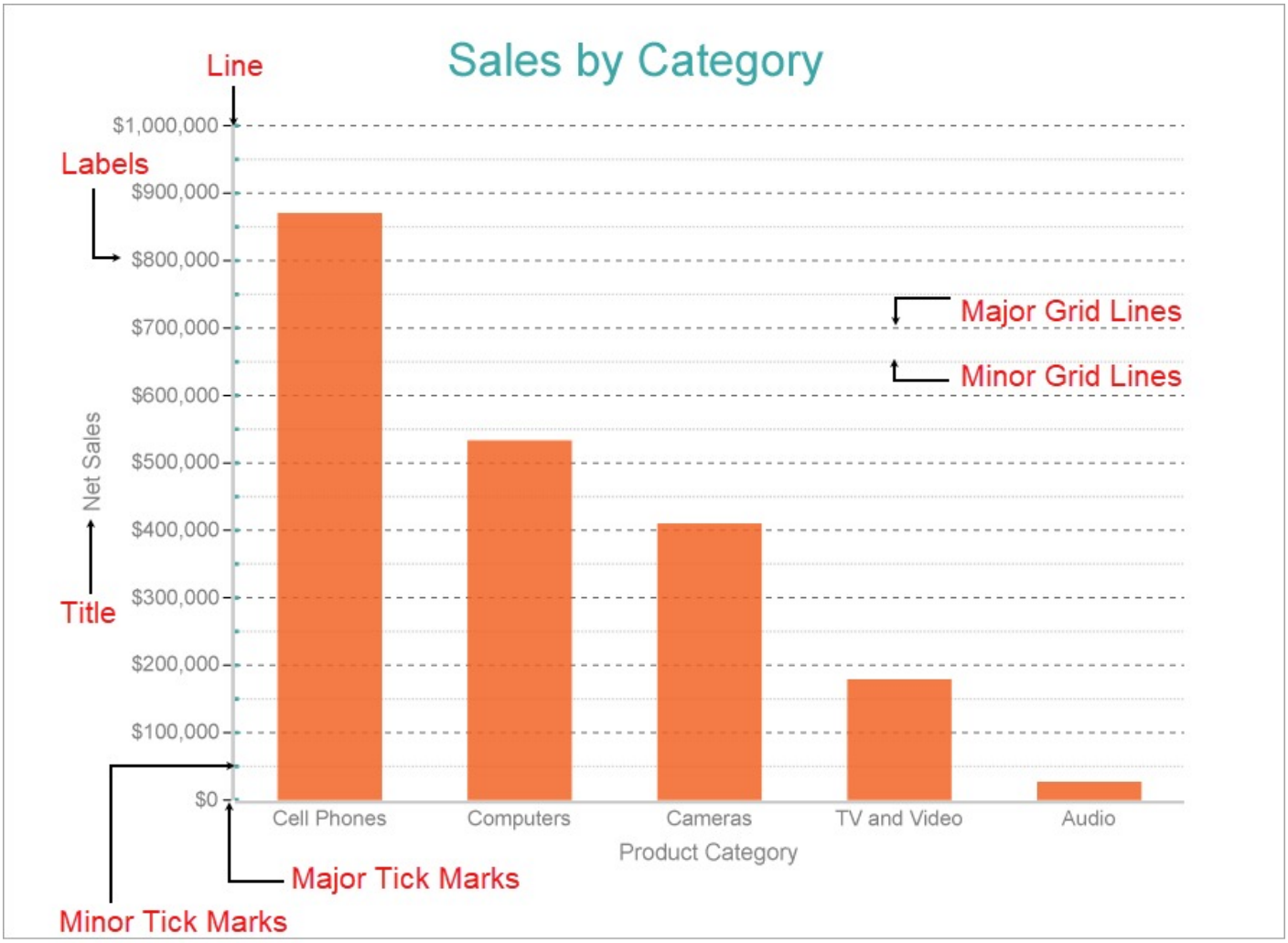


5. Once you are done with configuring and customizing the chart, press **F5** to preview the report.

Axes

Being an essential part of most chart types, the Axis can perform the following functions:

- Provide a coordinate system for one or multiple plots.
- Display measuring units for the data to be visualized.
- Produce the grid lines.



General Axis Properties

Appearance

Height: Represents the height of the Axis in percentage relative to the overall chart height. This property is applicable to the horizontal axis.

MaxHeight: Represents the maximum height of the Axis in percentage relative to the overall chart height. This property applies to the horizontal axis and its default value is 100%.

MaxWidth: Represents the maximum width of the Axis in percentage relative to the overall chart width. This property applies to the vertical axis and its default value is 100%.

Width: Represents width of the Axis in percentage relative to the overall chart width. This property applies to the

vertical axis.

See the section **Set Position of Axis** to know about customizing the position of Y-axis.

Common

AxisType: Designates the type of Axis, whether be it X or Y.

Plots: Designates the plot in which the X or Y axis exists.

Labels

Format: Represents the formatting string used for rendering dates and numbers.

LabelField: Represents customized labels for the X-axis. See the section **Set Custom Labels for X-Axis** to know about displaying the custom labels for category fields .

LabelsAngle: Represents the rotation angle of axis labels in degrees.

LabelsStyle: Represents the style of the axis labels

- Color: The label text color.
- Font: The font properties for the label.
- Padding: The padding to place between the label text and the axis.
- TextDecoration: The decorative lines on the label text.
- WritingMode: The horizontal (lr-tb) or vertical direction (tb-rl) of the label text. The default value is lr-tb (left to right top to bottom). If set to tb-lr (top to bottom left to right), the chart axis labels appear vertically.

ShowLabels: Indicate whether to show or hide axis labels.

Layout

Origin: Represents the value or ordinal number at which both the axes cross each other.

Overlapping: Indicates whether the labels can be overlapped.

Position: Indicates the axis position.

- Far
- Near
- None

Reversed: Indicates whether the axis is reversed from top to bottom, or left to right.

Line

LineStyle: Select the style of the line.

- LineColor: The line color.
- LineStyle: The style of the line.
- LineWidth: The width of the line.

ShowLine: Indicates whether or not to show the line.

Major Grid

MajorGridInterval: Number of units between the major axis ticks.

MajorGridStyle: Style of the major gridline.

MajorTickMark: Location of the major grid tick marks.

MajorTickSize: Length of the major grid tick marks.

MajorTickStyle: Style of the major grid ticks.

ShowMajorGrid: Indicates whether or not to show the major gridlines.

Minor Grid

MinorGridInterval: Number of units between minor axis ticks.

MinorGridStyle: Style of the minor grid line.

MinorTickMark: Location of the minor grid line tick marks.

MinorTickSize: Length of the minor gridline tick marks.

MinorTickStyle: Style of the minor grid ticks.

ShowMinorGrid: Indicates whether or not to show the minor gridlines.

Scale

LogarithmicBase: Represents the axis logarithmic base for logarithmic scale.

Max: Shows the maximum axis value.

Min: Shows the minimum axis value.

Scale: Represents the scale of the axis.

- Linear: In the Linear scale type, the values are evenly distributed along the axis. It is the default scale type. The Create Clustered Bar Chart walkthrough sets the Linear scale on Y-axis.
- Logarithmic: In the Logarithmic scale type, the axis uses the logarithm of values rather than the values themselves. This scale type is best suitable for displaying a wide range of numerical values in a compact manner. When using a logarithmic scale, the axis can use a logarithmic base. The Create Simple Polar Chart walkthrough sets the Logarithmic scale on Y-axis.
- Ordinal: In the Ordinal scale type, the values are evenly distributed along the axis according to their order. It applies to non-numerical values like product categories.
- Percentage: In the Percentage scale, the values are represented as a percentage. The Create Stacked Percentage Area Chart walkthrough sets the Percentage scale on Y-axis.

Title

Title: Represents the title of the axis.

TitleStyle: Represents the styling of the Axis Title.

- Color: The title color.
- Font: The font properties for the title.
- Padding: The padding to place between the title and the axis.
- TextDecoration: The decorative lines on the title.

User Scenarios

Set Custom Labels for X-Axis

Using **LabelField** property, you can display a custom label for the X-axis, that is, the Category Field. For example, in the following chart, the default labels for X-Axis show Ship Names since the Category Field is bound to `[ShipName]`. The chart binds to the 'Invoices' table of the Nwind.db. See Custom Data Provider for more information.

You can customize the X-Axis labels of the chart as follows.

1. From the **Report Explorer**, go to Chart > Plot Area and select **X Axis - [Plot1].**
2. In the **Properties** window, go to **Labels** > **LabelField** property.
3. Enter the following custom label in the **LabelField** property to view 'City' along with 'Ship Name'.

```
=Fields!ShipName.Value + " , " + Fields!City.Value
```

4. Preview the report.

The below image shows how the X-Axis labels show customized labels.

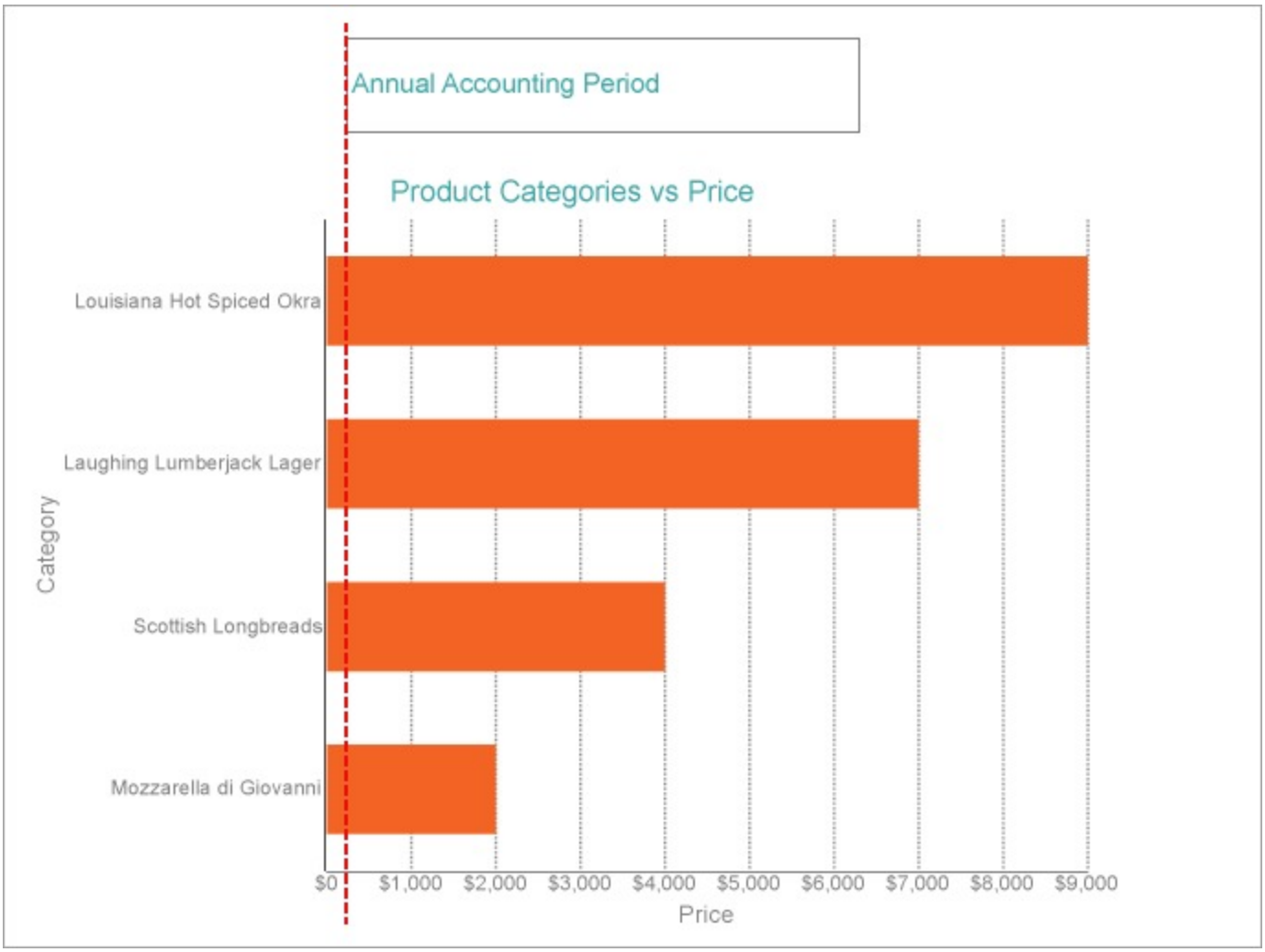
Set Position of Axis

A set of four properties for the X and Y axes - Width, MaxWidth, Height, and MaxHeight help users to control the position of the axes by specifying the height and width values for the X and Y axes, and their corresponding maximum values up to which the axes can grow. If the length of the data labels and titles in a chart exceeds the specified maximum height and width, labels are clipped-off. These properties are used to align a chart with the other controls or data regions.

Note that when a user does not specify these properties, the height and width for the chart axes are calculated automatically depending on the bound data. If the user wants to set the exact Y-axis position, the 'Width' and 'MaxWidth' properties should be set to one value. This value should be found by the user at design time.

Let us see how to align the vertical axis of a Chart data region relative to a Textbox control on the same page. The below image shows how the chart appears with the following properties set for the vertical axis.

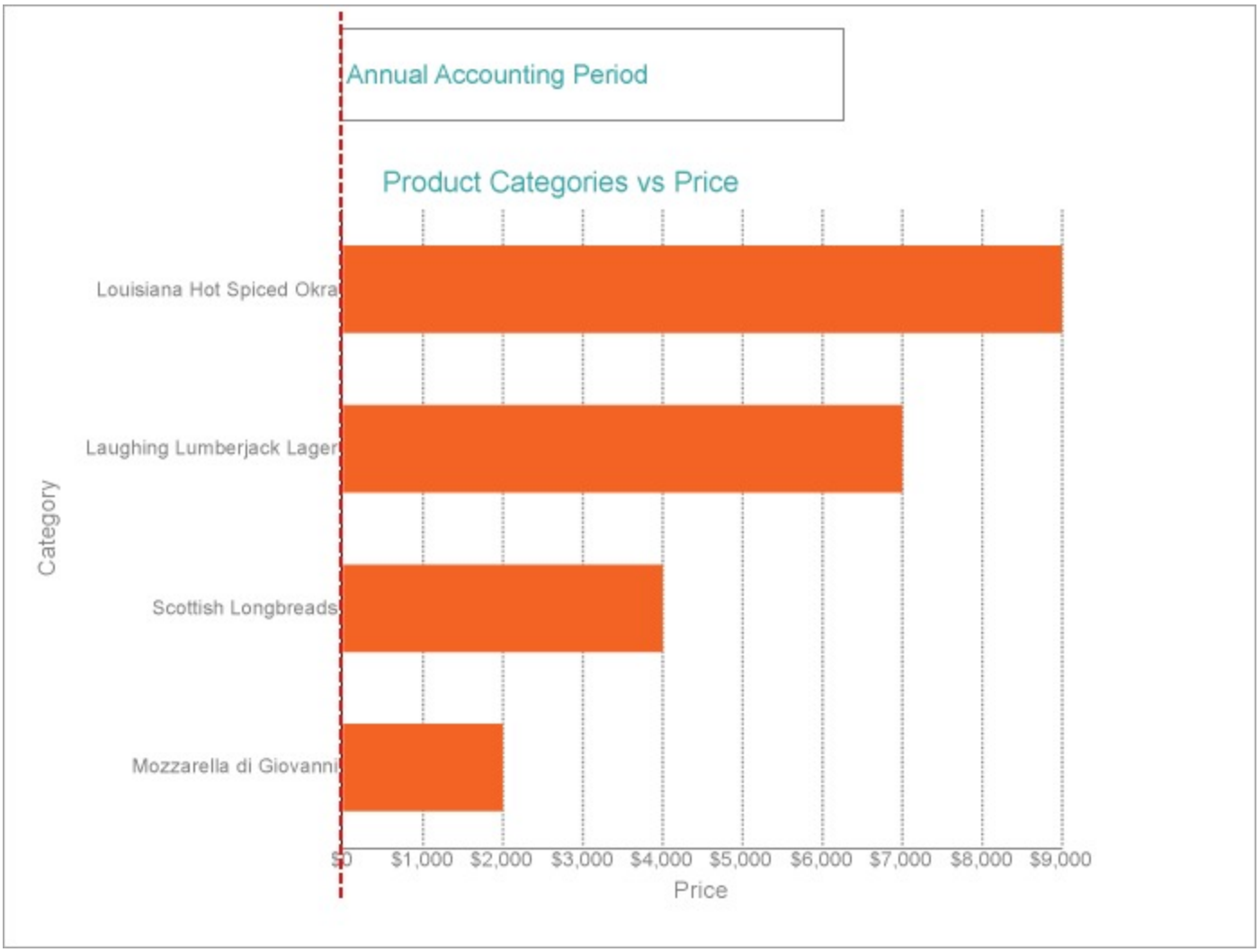
- **MaxWidth**: 100%
- **Width**: Default



The steps to set the Y-Axis position of the chart are as follows.

1. From the **Report Explorer**, go to Chart > Plot Area and select **Y Axis - [Plot1].**
2. In the **Properties** window, go to **Appearance** > **Width** property and enter a suitable value.
   In our case, 30% aligns the Y-Axis with the TextBox control.
3. Preview the report.

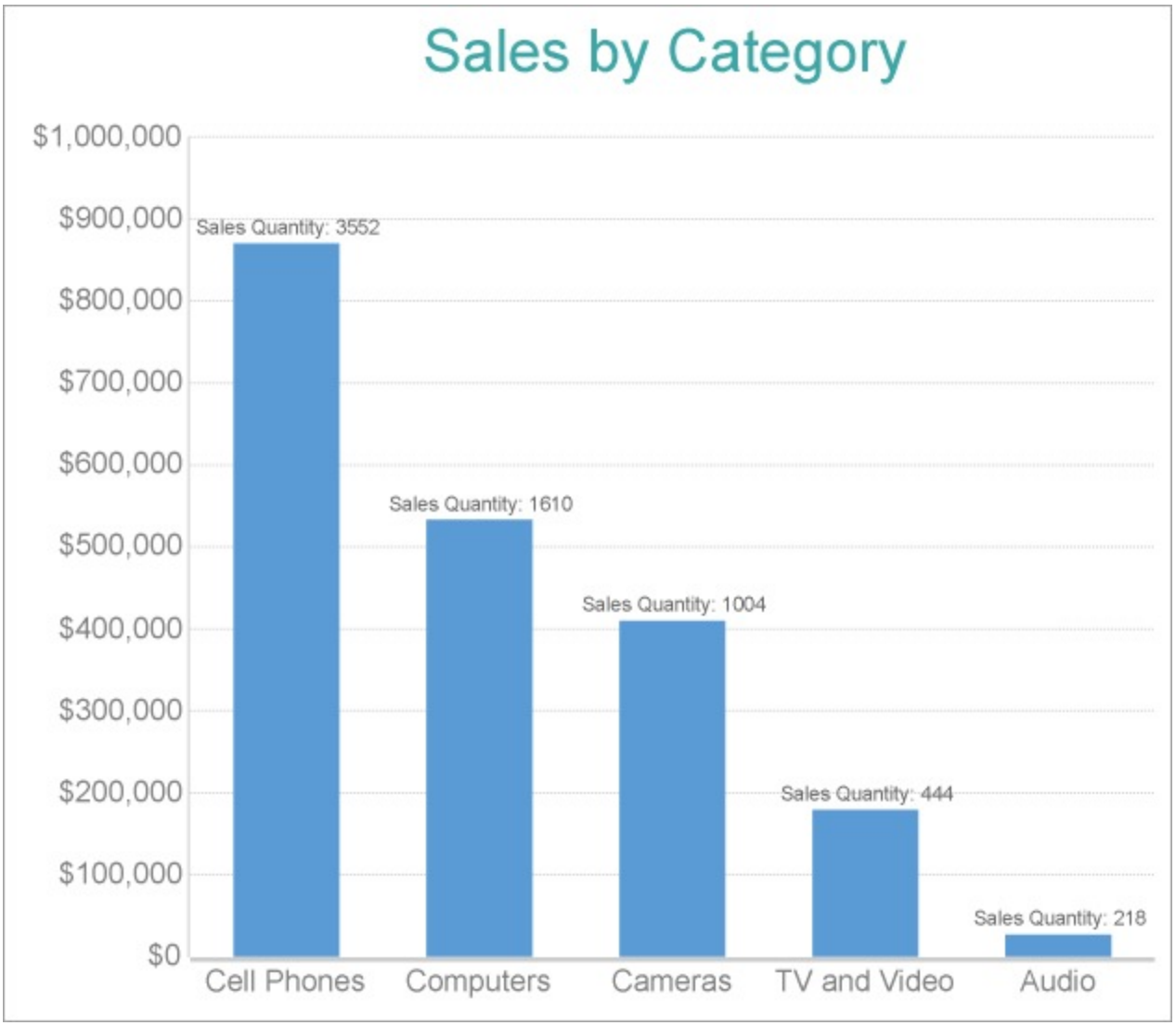
The below image shows how the vertical axis of the chart is aligned with the textbox after the 'Width' property is set to a suitable value observed at the design time:



Custom Labels

With Text encoding that supports any dataset field in chart labels and tooltips, you can create custom labels in a chart to show additional information.

This example uses the Column Chart with the FactSales dataset and displays Net Sales and Product Category data. The custom labels display the additional information on the total sales quantity across each product category.



Create a Report and Bind Report to Data

In the ActiveReports Designer, create a new RDLX report and follow the **New Report** wizard to bind the report to data. You can also perform data binding later using the **Report Data Source** dialog accessed from the **Report Explorer**.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under **Type**, select 'Json Provider'.
3. Go to the **Content** tab under **Connection** and set the type of JSON data to 'External file or URL'.
4. In the **Select or type the file name or URL** field, enter the following URL:
   https://demodata.mescius.io/contoso/odata/v1/FactSales
   For more information, see the JSON Provider topic.
5. Go to the **Connection String** tab and verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and open the **DataSet** dialog.

Add a Dataset

1. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'FactSales'.
2. Go to the **Query** page and enter the following query to fetch the required fields:

Query
`$.value[*]`

3. Go to the **Fields** page to view the available fields and modify the **Name** of the `[SalesAmount]` field to `[Net Sales]`.
4. On the same page, add following calculated field:

Name	Value
Product Category	`=Switch([ProductKey] < 116, "Audio", [ProductKey] >= 116 And [ProductKey] < 338, "TV and Video", [ProductKey] >= 338 And [ProductKey] < 944, "Computers", [ProductKey] >= 944 And [ProductKey] < 1316, "Cameras", [ProductKey] >= 1316, "Cell Phones")`

5. Click **OK** to save the changes.

Create a Chart

We will use the Chart Wizard dialog to configure chart data values. The wizard appears by default if you have a dataset added to your report. See the topic on Chart Wizard for more information.

1. Drag-drop **Chart** data region onto the design area. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
2. Select the **Dataset Name** as 'FactSales' and the **Chart Type** as 'Column'.
3. Click **Next** to proceed. Here, you need to specify the column settings. We will define a data series value to display the net sales values across the horizontal axis. We will later sub-categorize the data series value to form a cluster.
4. Under **Choose Data Values**, add a new data value `[Net Sales]`.
5. In **Choose Data Categories**, select `[Product Category]` as the **Field**.
6. Click **Next** and then **Finish.**

Add a Text Encoding (Plot Settings)

1. To open the smart panel for advanced plot settings, right-click 'Plot-Plot1' on the Report Explorer and choose **Property Dialog**.
2. In the Chart Pot dialog that opens, go to the **Encodings** page.
3. On the **Text** tab, add a new text encoding and set its properties as below.
    - Expression: `=Fields!SalesQuantity.Value` (to display total sales quantity for each product category)
    - Aggregate: Sum
    - Target: Label
    - Template Key: SalesQuantity
4. In the same Chart Plot dialog, go to the **Labels** page and set the label's properties as below.
    - Template: `Sales Quantity: {SalesQuantity}`
    - Text Position: Outside

5. Click **OK** to complete setting up the plot.

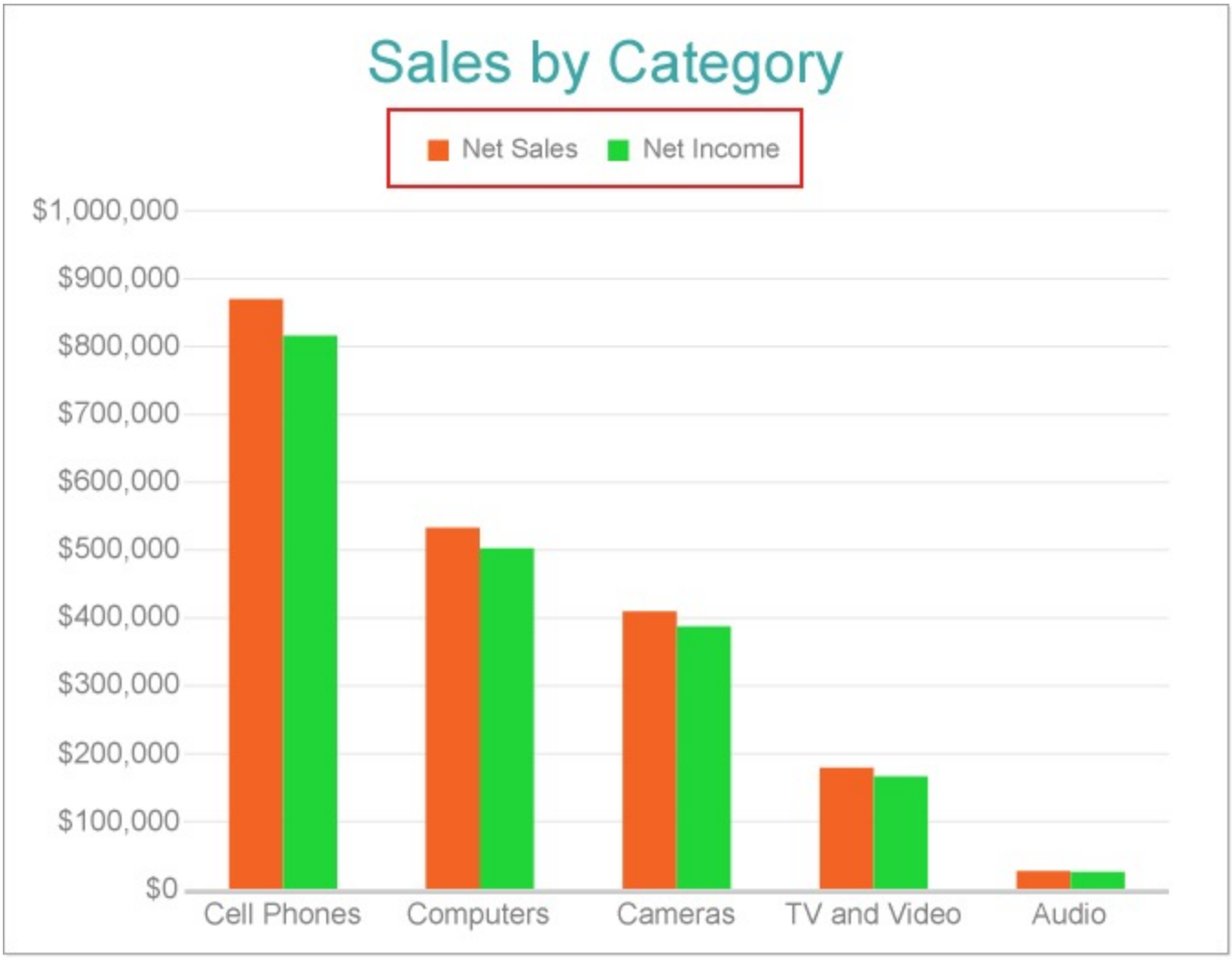
Chart Customization

See topic Create Clustered Column Chart for other chart customization options.

Legends

A Legend in a Chart data region helps a user understand the data plots easily. A legend is connected to the data which is being graphically displayed in the plot area of the chart and is an especially useful tool to analyze data in the case of multiple series.
Several legend types depending on the Chart type are available.

Global Legend



You can use this type of legend if the plot has several Data Values to display their names. In order to display the global legend, set the the **ShowValuesNames** property to True from the plot properties (or **Show values name** property in the smart panel for Plot properties). Refer Create Clustered Column Chart walkthrough.

Global legend has lower priority than Color, Size, or Shape legends. So if you have any settings changed in Color, Shape, or Size legends, they will override global settings.

You can configure the appearance of Global Legend using the properties given below.

BackgroundColor: Indicates the color of the background.

Border: Represents the border settings.

Hidden: Indicates whether the legend is hidden.

MaxHeight: Represents the maximum height of the element in percentage.

MaxWidth: Represents the maximum width of the element in percentage.

Orientation: The Orientation property determines the direction of legend item's appearance:

- Horizontal: from left to right
- Vertical: from top to bottom

Padding: Represents the amount of padding to place between the text or graphics and the edge of the report item.

Position: Represents the legend position.

TextStyle and **TitleStyle**: The TextStyle and TitleStyle properties determine how to add styling to the Text and Title respectively. For instance, the TextStyle or TitleStyle has the following settings:

- Color: Adds the color for the Text or Title.
- Font: Adds the FontFamily, FontSize, FontStyle and FontWeight.
- TextDecoration: Adds a line under, over or through the text.

Wrapping: Represents whether the title text is wrapped.

Color Legend

The color legend displays the match between the plot subsections's fill or stroke color and the corresponding data value subcategories. You can configure the Color Encoding of a plot to show the Color Legend. Almost all types of plot support this legend. See Create Clustered Polar Chart walkthrough.

You can configure the appearance of Color Legend using the properties given below.

Title: The string Expression determines the legend heading.

BackgroundColor: Indicates the color of the background.

Border: Represents the border settings

MaxHeight: Represents the maximum height of the element in percentage.

MaxWidth: Represents the maximum width of the element in percentage.

Hidden: Indicates whether the legend is hidden.

Orientation: The Orientation property determines the direction of legend item's appearance:

- Horizontal: from left to right
- Vertical: from top to bottom

Position: The Position value sets the location of the legend relative to the plot area.

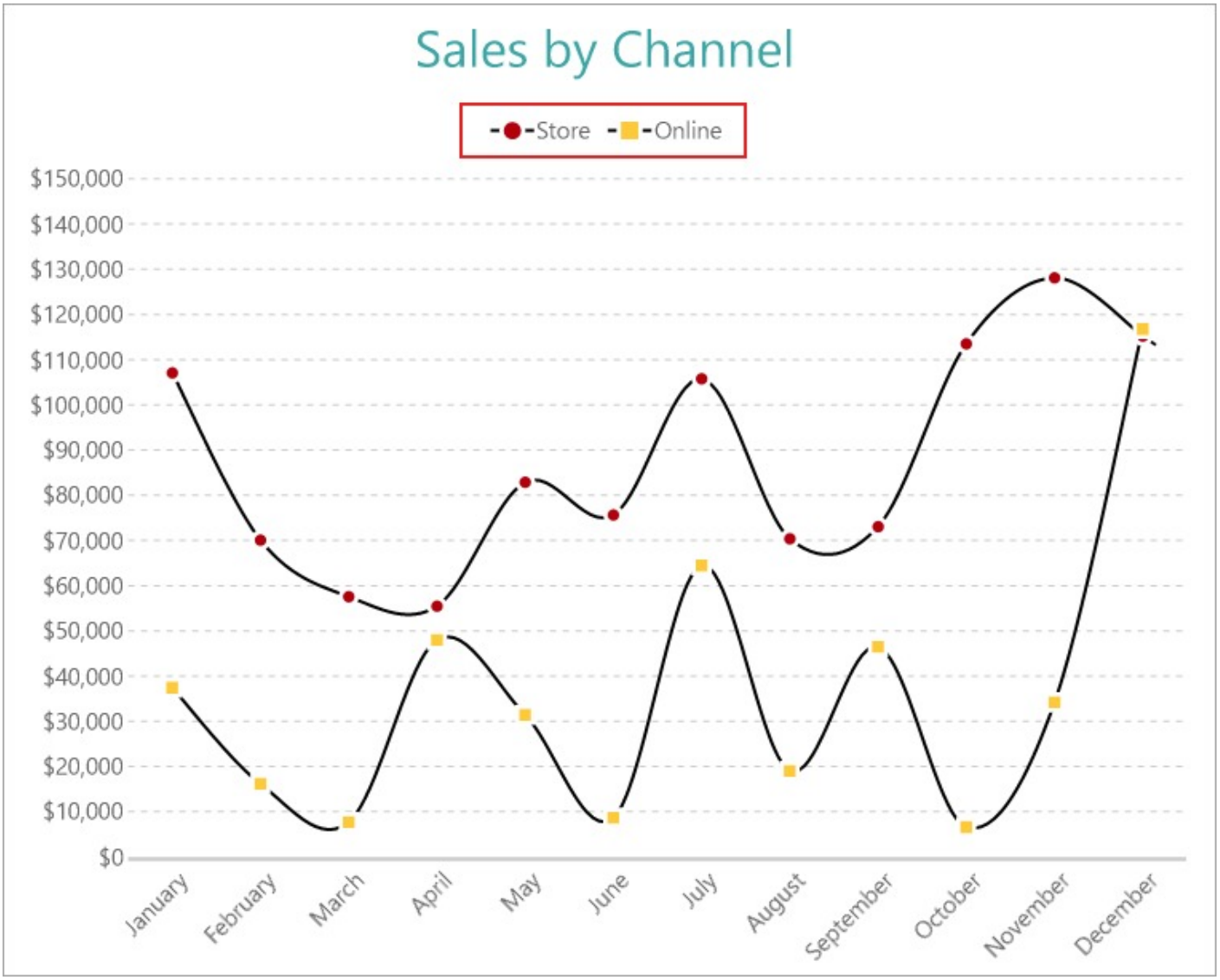
Padding: You could also set the Padding between the legend box and its content. This will automatically increase the

legend size.

TextStyle and **TitleStyle**: The TextStyle and TitleStyle properties determine how to add styling to the Text and Title respectively. For instance, the TextStyle or TitleStyle has the following settings:

- Color: Adds the color for the Text or Title.
- Font: Adds the FontFamily, FontSize, FontStyle and FontWeight.
- TextDecoration: Adds a line under, over or through the text.

Shape Legend



The Shape Legend shows the relationship between the shape of data point symbols and corresponding subcategories of a value in the plot data. You can configure the Shape Encoding in a Line plot to depict the Shape Legend. Consider the Create Multiple Line Chart walkthrough. With the chart selected, set the Shape Encoding to 'Channel Name' (same as the Details encoding) and the Symbols > Shape property to 'Auto'.

You can configure the appearance of Shape Legend using the properties given below.

BackgroundColor: Indicates the color of the background.

Border: Represents the border settings.

Hidden: Indicates whether the legend is hidden.

MaxHeight: Represents the maximum height of the element in percentage.

MaxWidth: Represents the maximum width of the element in percentage.

Orientation: The Orientation property determines the direction of legend item's appearance:

- Horizontal: from left to right
- Vertical: from top to bottom

Padding: Represents the amount of padding to place between the text or graphics and the edge of the report item.

Position: Represents the legend position.

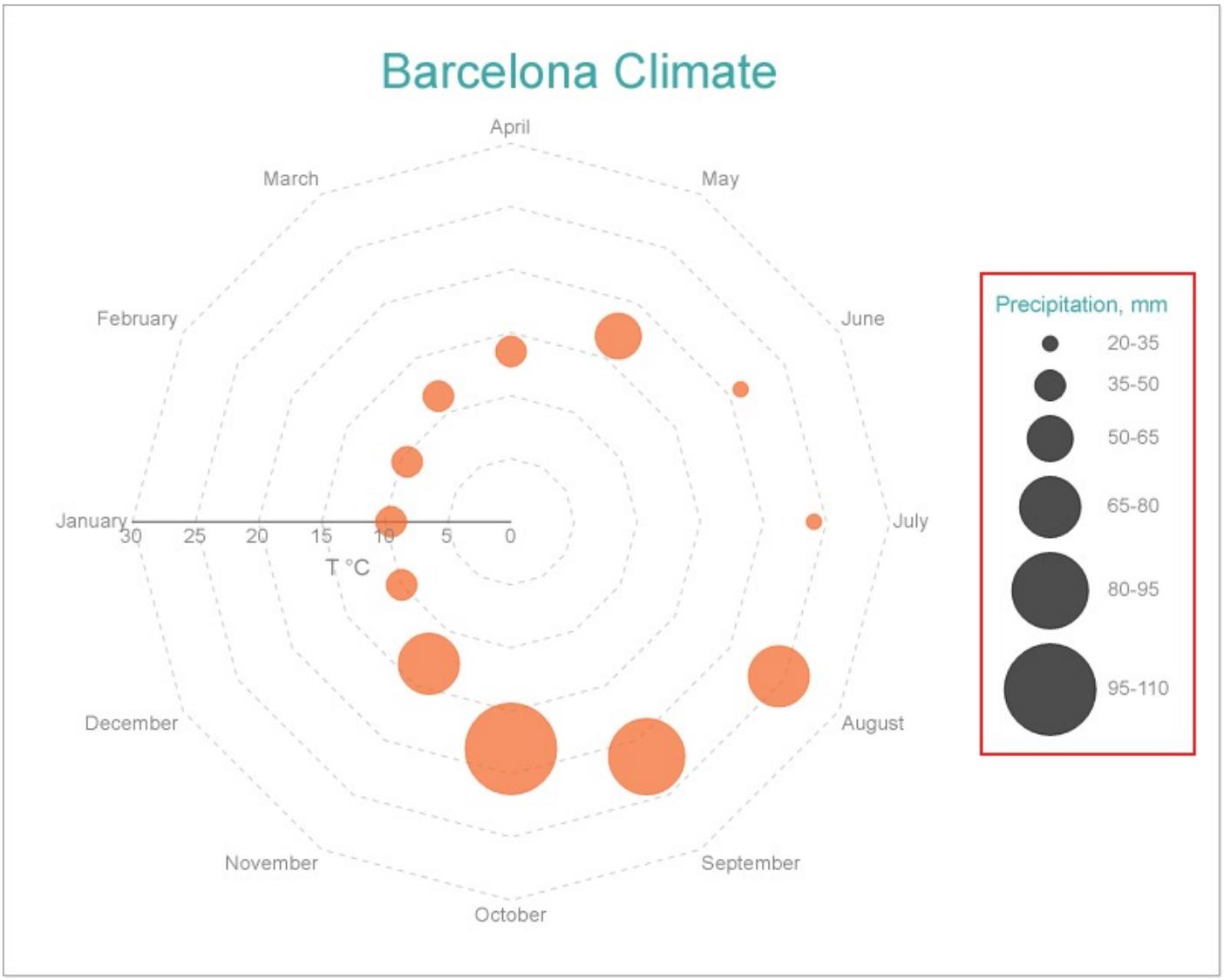
TextStyle and **TitleStyle**: The TextStyle and TitleStyle properties determine how to add styling to the Text and Title respectively. For instance, the TextStyle or TitleStyle has the following settings:

- Color: Adds the color for the Text or Title.
- Font: Adds the FontFamily, FontSize, FontStyle and FontWeight.
- TextDecoration: Adds a line under, over or through the text.

Wrapping: Represents whether the title text is wrapped.

IconColor: Represents the color of the icon.

Size Legend



The Size Legend shows the match between the size of data point symbols and corresponding subcategories of a data value. You need to configure the Size Encoding of a plot to depict the Size Legend. The Create Radar Bubble Chart walkthrough illustrates setting up a Size Legend.

You can configure the appearance of Size Legend using the properties given below.

BackgroundColor: Indicates the color of the background.

Border: Represents the border settings.

Hidden: Indicates whether the legend is hidden.

MaxHeight: Represents the maximum height of the element in percentage.

MaxWidth: Represents the maximum width of the element in percentage.

Orientation: The Orientation property determines the direction of legend item's appearance:

- Horizontal: from left to right
- Vertical: from top to bottom

Padding: Represents the amount of padding to place between the text or graphics and the edge of the report item.

Position: Represents the legend position.

TextStyle and **TitleStyle**: The TextStyle and TitleStyle properties determine how to add styling to the Text and Title respectively. For instance, the TextStyle or TitleStyle has the following settings:

- Color: Adds the color for the Text or Title.
- Font: Adds the FontFamily, FontSize, FontStyle and FontWeight.
- TextDecoration: Adds a line under, over or through the text.

Wrapping: Represents whether the title text is wrapped.

IconColor: Represents the color of the icon.

Ranges: Represents the collection of ranges for size legend.

User Scenario

Set a Custom Legend

Though the data fields in the legend display the actual dataset field names, you can also define a custom name for a data field using the **Caption** property as elaborated in below. This is useful in scenarios when the data field names in a legend are difficult to understand.

> **Note**: The **Caption** property is unavailable in two scenarios:
>
> 1. For the chart types that do not support color encoding. For instance, Candlestick, High Low Close, and High Low Open Close charts.
> 2. If you set the **Type** property of the data field to 'Complex' (an exception is the Gantt Chart because it has Color encoding).

Consider the Single Line Chart walkthrough. The following steps set a custom legend for the data field name in the chart.

1. On the design area, select the **Chart** data region and then select the data field (for which you want a custom name) from 'Data Fields'.
2. In the **Properties** panel, go to the **Misc** > **Value** property and open the collection editor.
3. In the **FieldExpression Collection Editor** dialog, set a new field name in **Caption**, 'Monthly Return Quantity'.
4. Verify that **ShowValuesNames** of **Color Encodings** is set to 'True'.
5. Preview the report

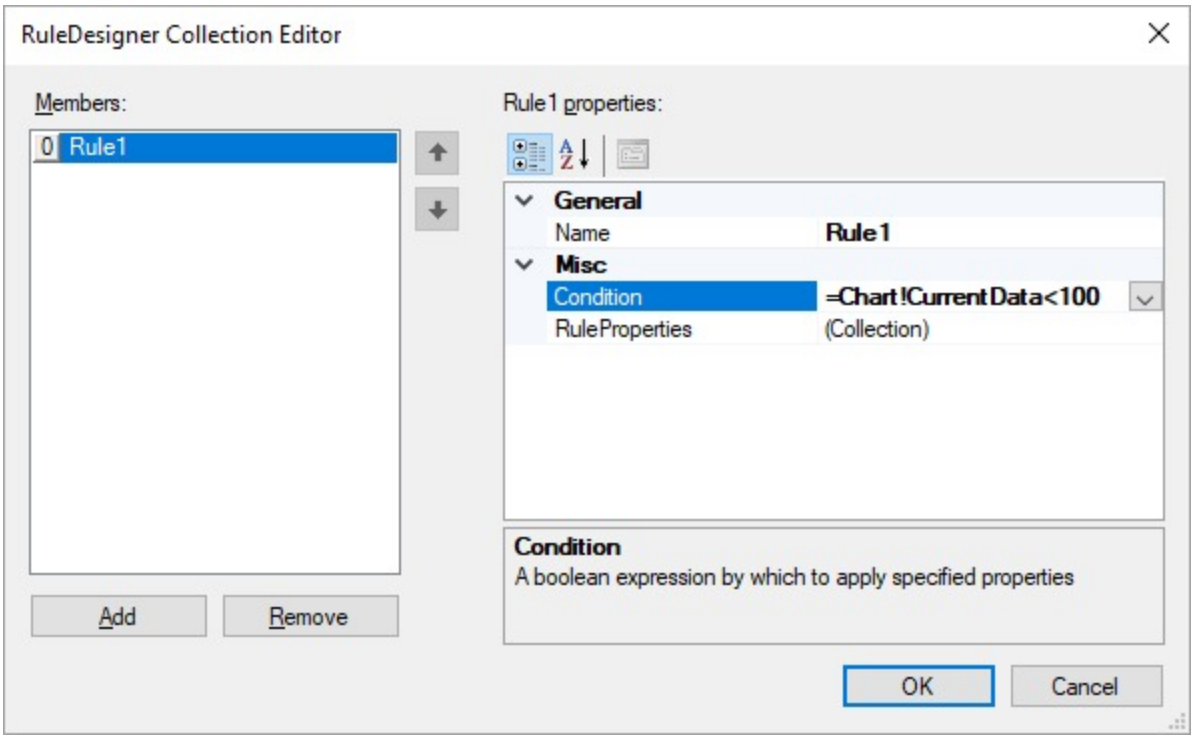
Rules

You can dynamically control the chart appearance by creating a rule or several rules for each plot and specifying the background color, line color, line style, line width, and other elements of the chart visual presentation at rendering.

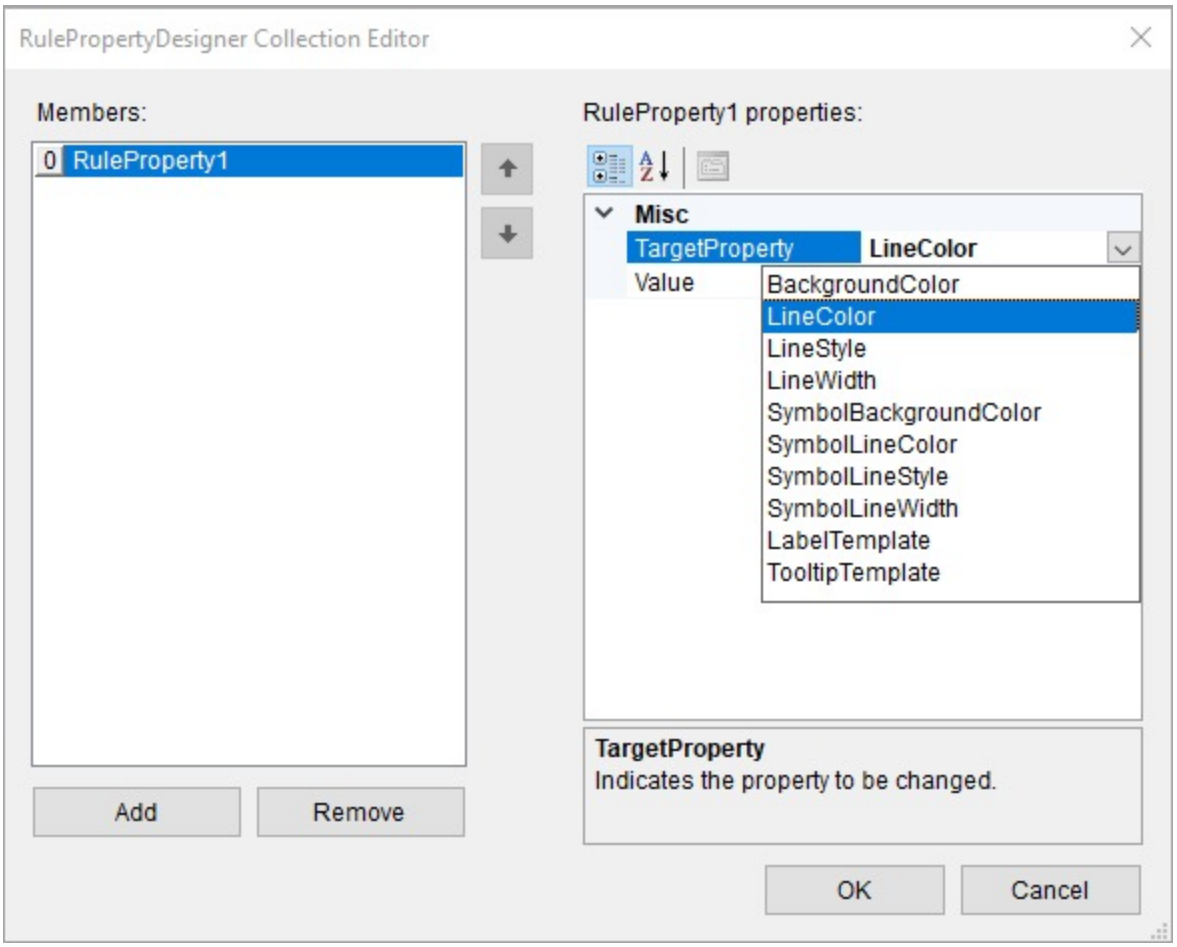
A rule consists of a condition - an expression that specifies how to apply a target property, and a target property - a chart element that you can change.

Add a rule

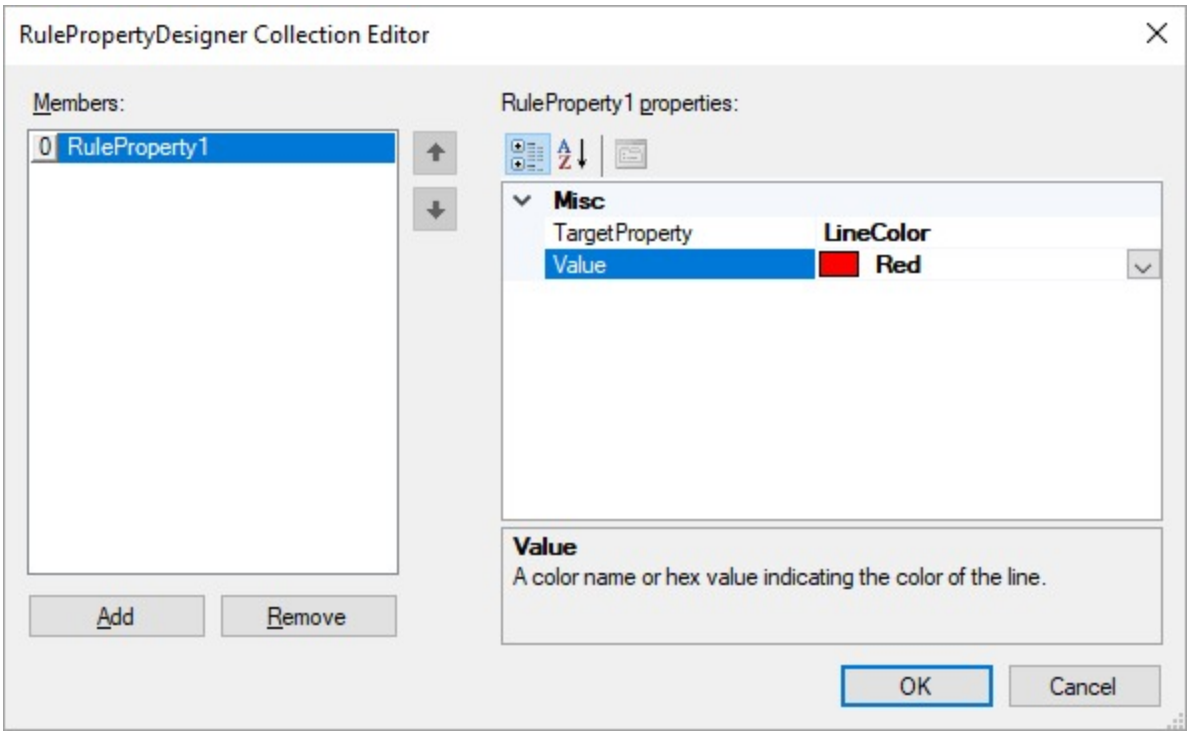
1. Select a Chart plot.
2. In the Properties window, go to the **Rules** property and click (Collection).
3. In the RuleDesigner Collection Editor that opens, click **Add** to add a new rule.



4. In the **Condition** field, specify a condition expression for a target property.
5. In the **RuleProperties** field, click (Collection).
6. In the RulePropertyDesigner that opens, click **Add** to add a target property.



7. In the **TargetProperty** field, select a property from the pre-defined list of properties - e.g. LineColor.



8. In the **Value** field, select a value of the target property.
9. Click **OK**.

Use Rules to Customize Chart Appearance

You can customize the chart appearance by changing these target properties of the chart plot.

- DataPoint Styles
- DataPoint Symbol Styles
- LabelTemplate
- ToolTipTemplate

> **Note**: If the chart legend is set to display and a rule condition changes the color of a part of the chart, the color of the legend will not change. The color of the legend will match the chart color only if the whole chart changes color.

DataPoint Styles

DataPoint Styles include such target properties as **BackgroundColor**, **LineColor**, **LineStyle**, and **LineWidth**. These properties are applicable to the chart types that are composed of data points, for example, Bar, Column, Pie, Bubble, etc.

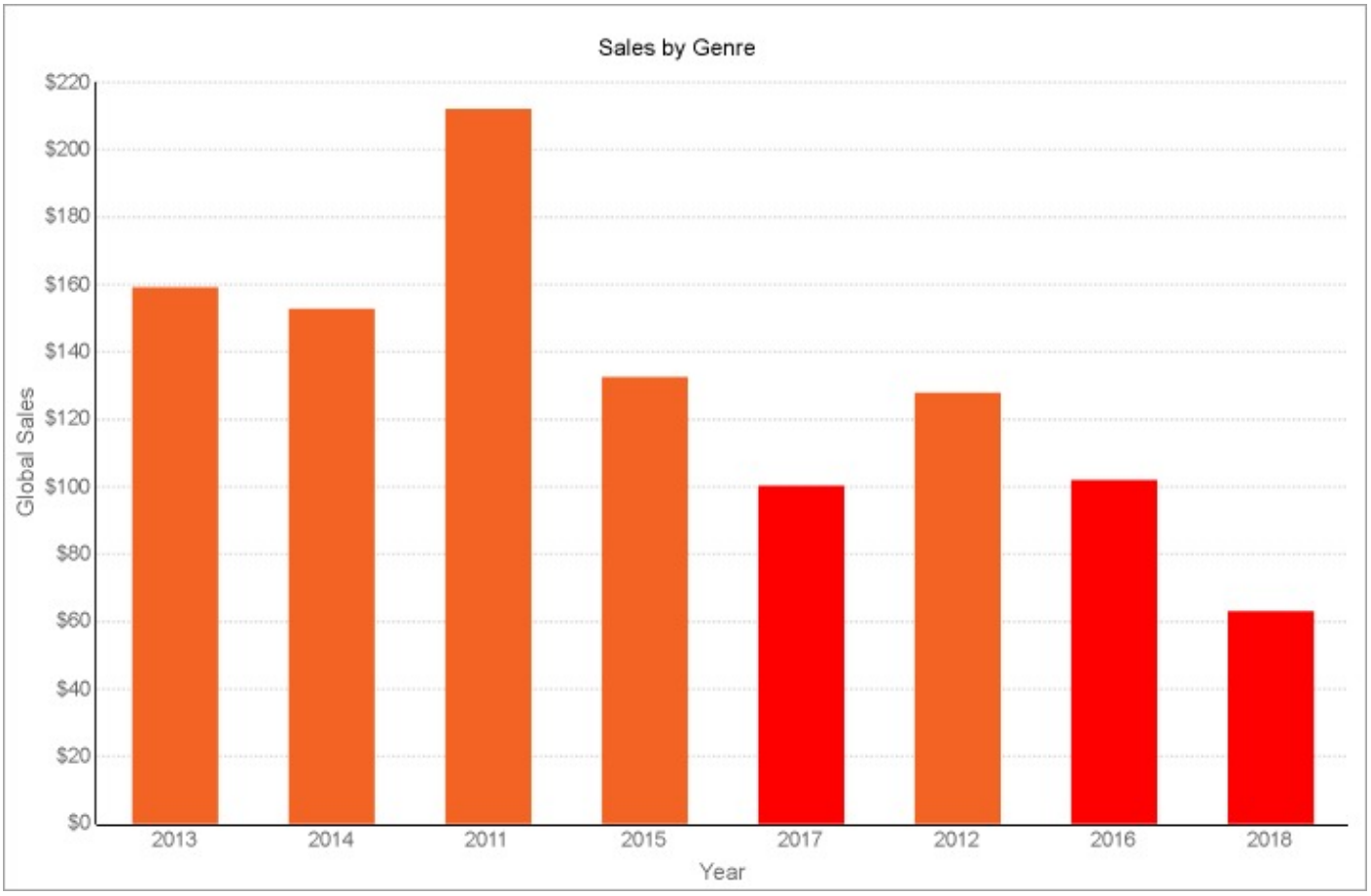
This topic uses a report with a Column chart that displays global sales by year.

BackgroundColor

To change the BackgroundColor target property of the chart, you can add a rule with the following settings.

Property	Value
RuleDesigner Collection Editor > Condition	=Chart!CurrentData<110
RulePropertyDesigner Collection Editor > TargetProperty	BackgroundColor
RulePropertyDesigner Collection Editor > Value	Red

At preview, you will see Sales with the data below 110 get the green background color.



LineColor

Add the LineColor target property to the same condition (see previous image) for the chart with the following settings:

Property	Value
RulePropertyDesigner Collection Editor > TargetProperty	LineColor
RulePropertyDesigner Collection Editor > Value	Black

LineStyle

Add the LineStyle target property to the same condition (see previous image) for the chart with the following settings:

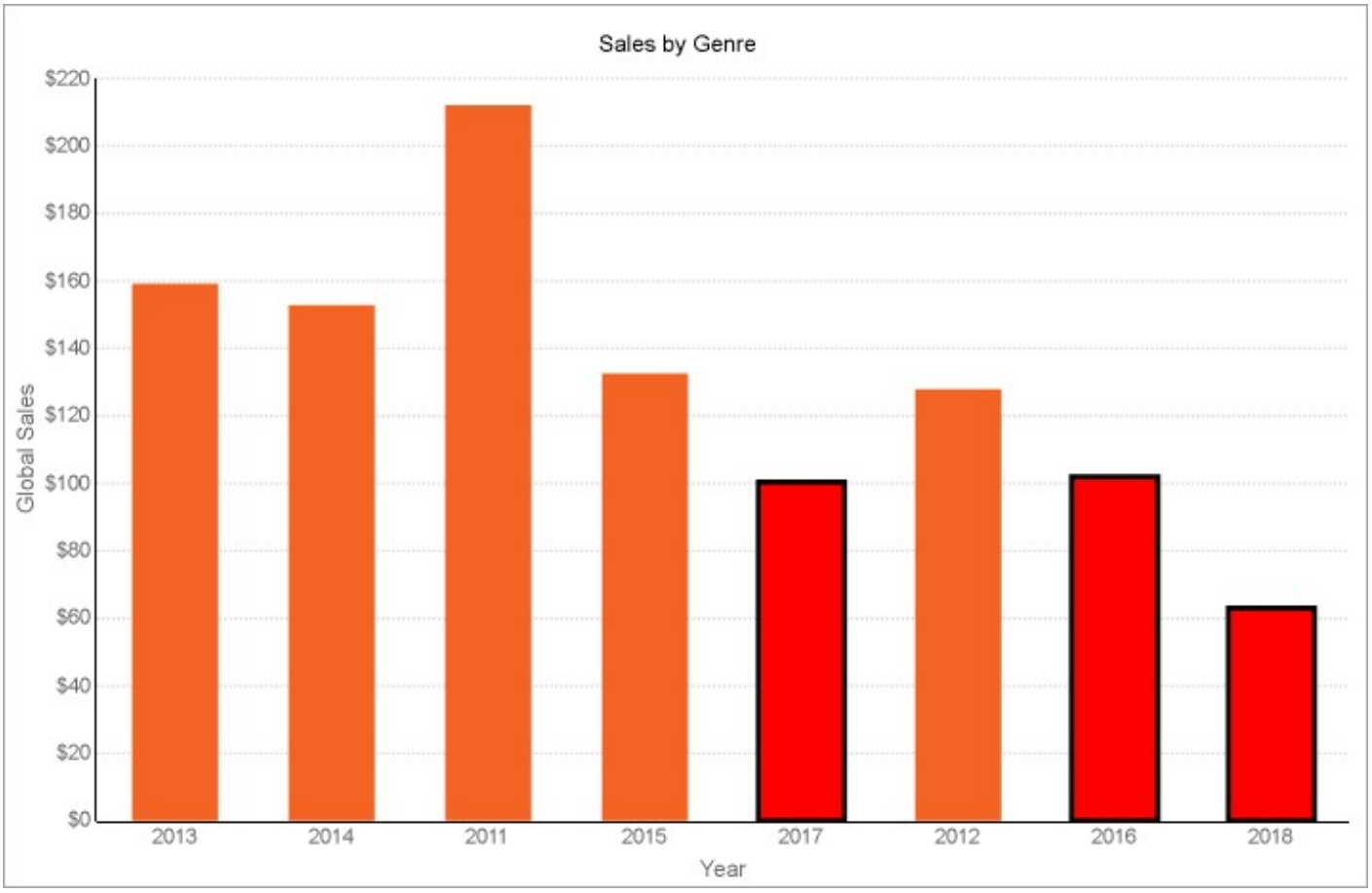
Property	Value
RulePropertyDesigner Collection Editor > TargetProperty	LineStyle
RulePropertyDesigner Collection Editor > Value	Double

LineWidth

Add the LineWidth target property to the same condition (see previous image) for the chart with the following settings:

Property	Value
RulePropertyDesigner Collection Editor > TargetProperty	LineWidth
RulePropertyDesigner Collection Editor > Value	2

The final chart looks as follows.



DataPoint Symbol Styles

DataPoint Symbol Styles include such target properties as **SymbolBackgroundColor**, **SymbolLineColor**, **SymbolLineStyle**, and **SymbolLineWidth**. These properties are applicable to the chart types that are composed of data point symbols, for example, Line, Radar Line, etc.

This topic uses a report with a Line chart that displays global sales by year.

SymbolBackgroundColor

To change the SymbolBackgroundColor target property of the chart, you can add a rule with the following settings.

Property	Value
RuleDesigner Collection Editor > Condition	=Chart!CurrentData<110
RulePropertyDesigner Collection Editor > TargetProperty	SymbolBackgroundColor
RulePropertyDesigner Collection Editor > Value	Green

SymbolLineColor

To change the SymbolLineColor target property of the chart, you can add a rule with the following settings.

Property	Value
RulePropertyDesigner Collection Editor > TargetProperty	SymbolLineColor
RulePropertyDesigner Collection Editor > Value	Red

SymbolLineStyle

To change the SymbolLineStyle target property of the chart, you can add a rule with the following settings:

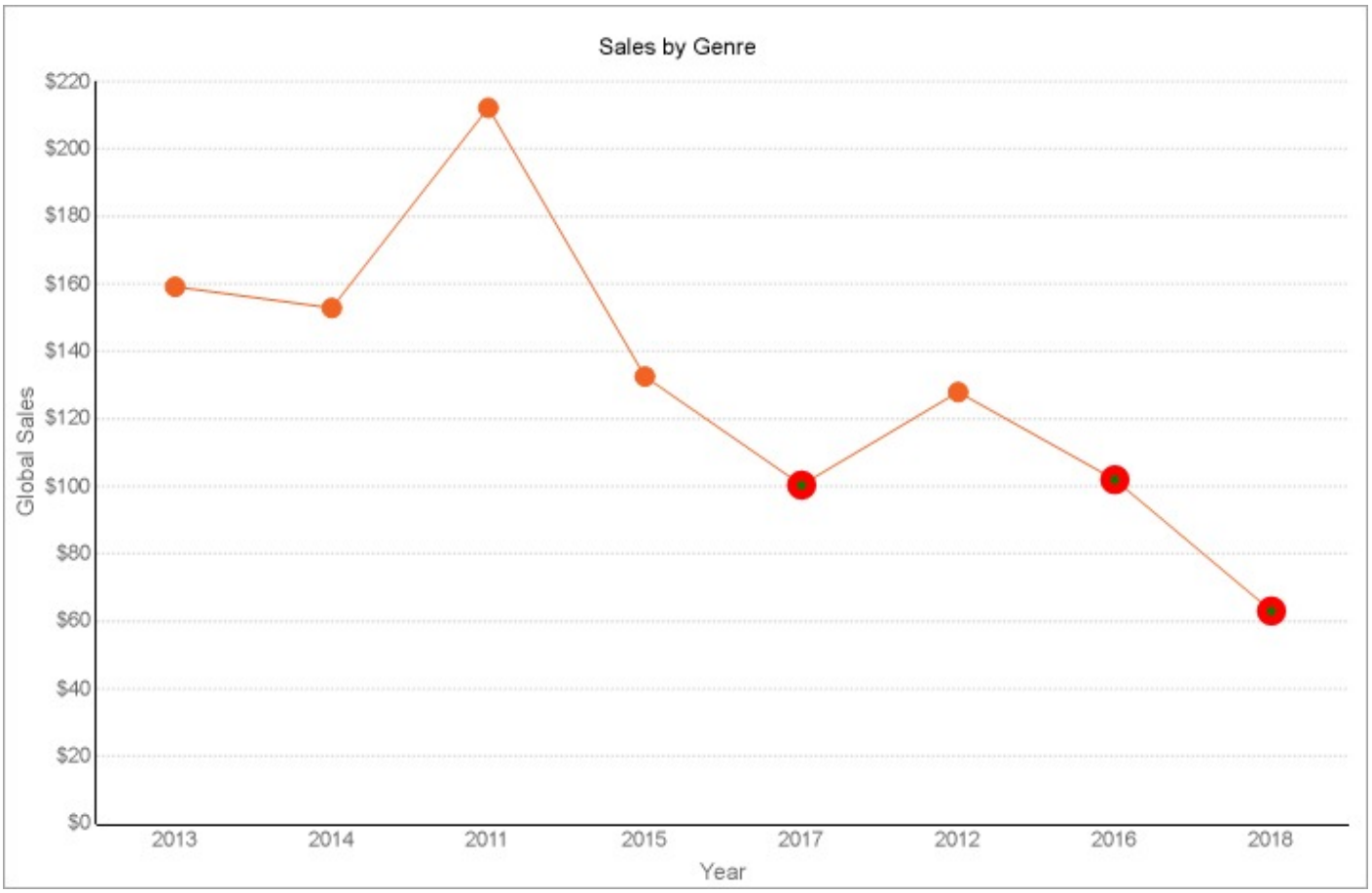
Property	Value
RulePropertyDesigner Collection Editor > TargetProperty	SymbolLineStyle
RulePropertyDesigner Collection Editor > Value	Double

SymbolLineWidth

To change the SymbolLineWidth target property of the chart, you can add a rule with the following settings.

Property	Value
RulePropertyDesigner Collection Editor > TargetProperty	SymbolLineWidth
RulePropertyDesigner Collection Editor > Value	4

The final chart looks as follows.



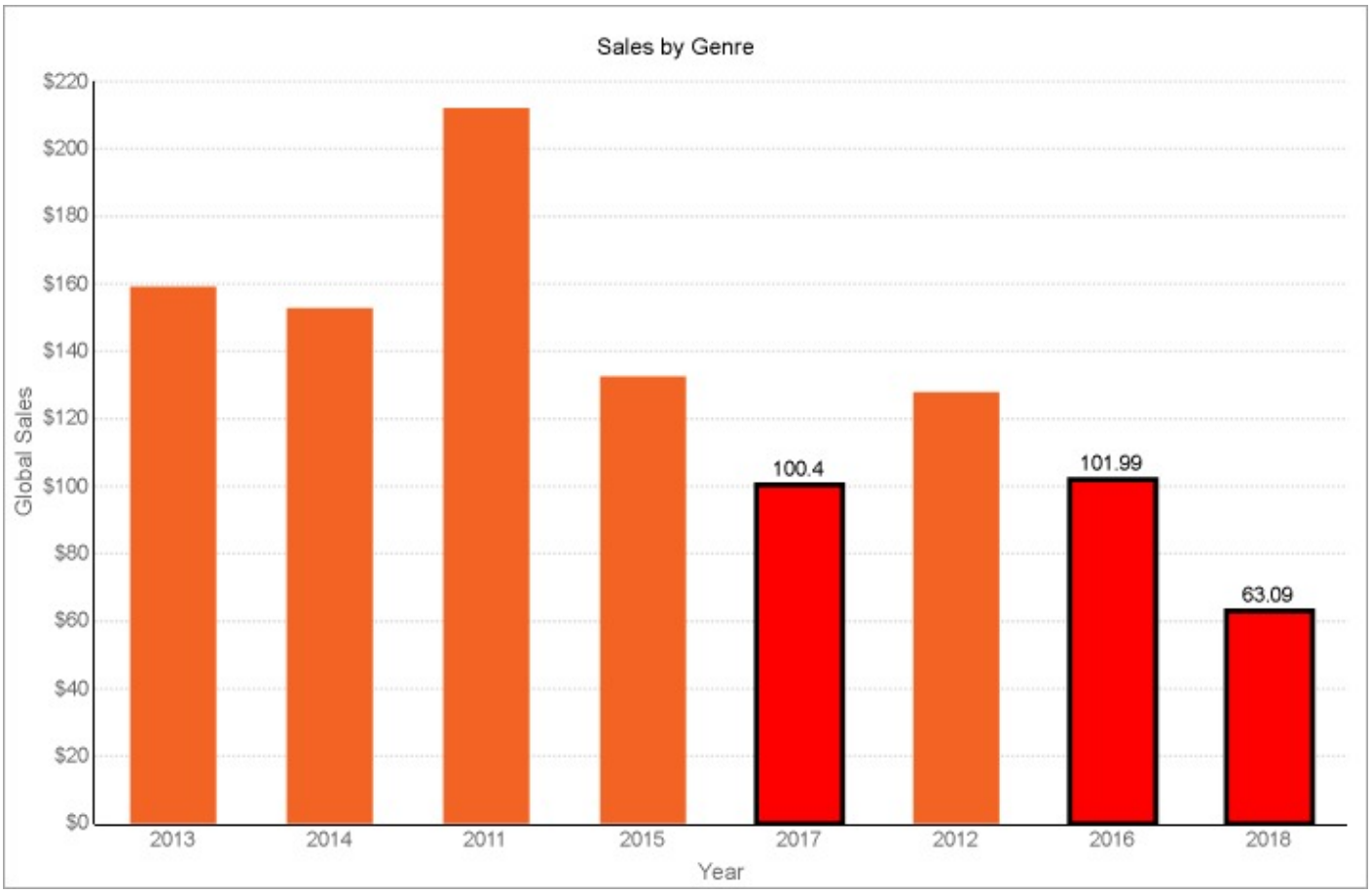
LabelTemplate

The **LabelTemplate** target property allows you to display labels and specify the label format for the chart data that meets the condition of a rule.

Let's say that you have a chart with a rule to mark data below 110 and you want the chart to display a label for this marked data. To do this, you can add a rule with the following settings.

Property	**Value**
RuleDesigner Collection Editor > Condition	=Chart!CurrentData<110
RulePropertyDesigner Collection Editor > TargetProperty	LabelTemplate
RulePropertyDesigner Collection Editor > Value	{valueField.value}

At preview, you will see Sales with the data below 110 marked with the labels. These labels display data values according to the rule you have just set.



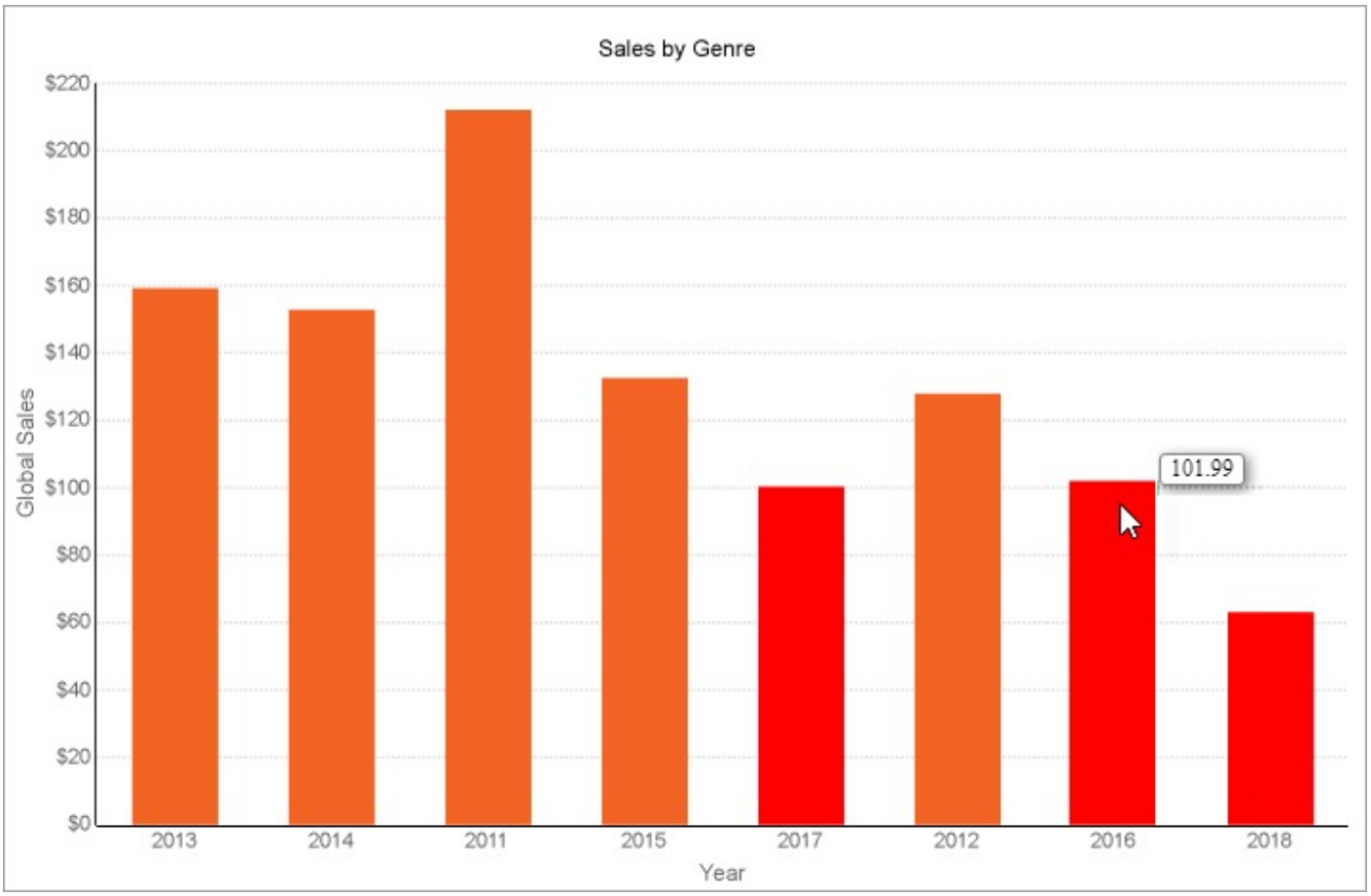
TooltipTemplate

The **TooltipTemplate** target property allows you to display tooltips in the Web Viewer and specify the tooltip format for the chart data that meets the condition of a rule.

Let's say that you have a chart with a rule to mark data below 110 and you want the chart to display a tooltip for this marked data. To do this, you can add a rule with the following settings.

Property	**Value**
RuleDesigner Collection Editor > Condition	=Chart!CurrentData<110
RulePropertyDesigner Collection Editor > TargetProperty	TooltipTemplate
RulePropertyDesigner Collection Editor > Value	{valueField.value}

At preview, you will see Sales with the data below 110 display tooltips. These tooltips display data values according to the rule you have just set.



Trendlines

You can add a trendline element to the chart to display additional information like a trendline for the chart data, reference information, and more. A trendline is a helpful element for the analysis of chart data.

The following trendline types are available.

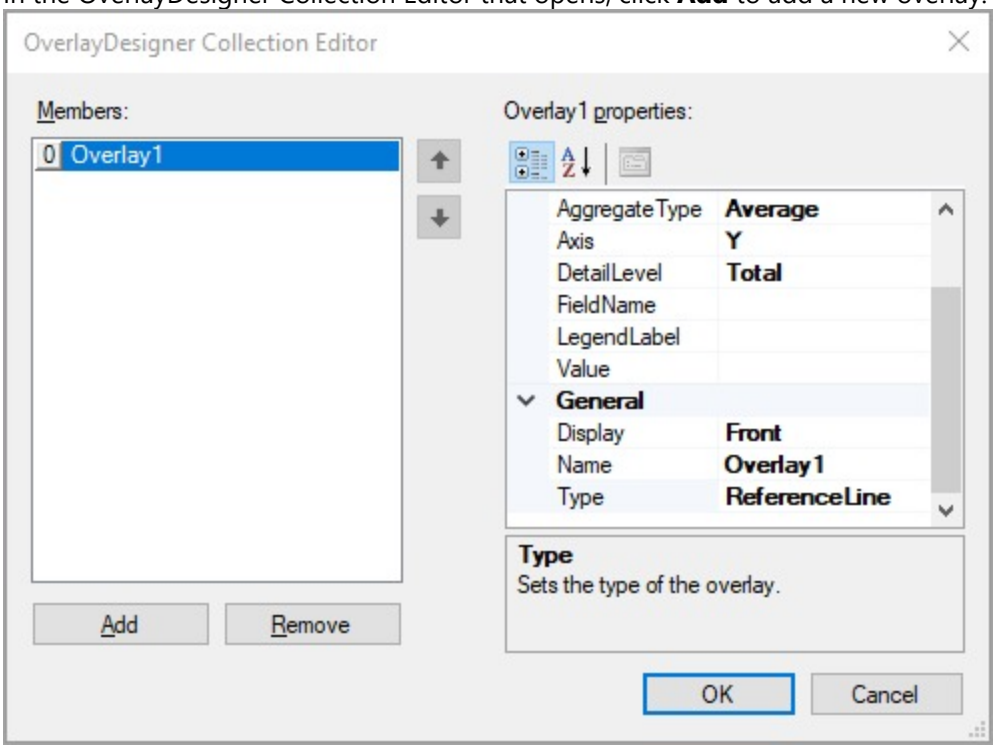
- Reference Line
- Reference Band
- Linear Trendline
- Exponential Trendline
- Power Trendline
- Logarithmic Trendline
- Polynomial Trendline
- Fourier Trendline
- MovingAverage Trendline
- CumulativeMovingAverage Trendline
- ExponentialMovingAverage Trendline
- WeightedMovingAverage Trendline
- MovingAnnualTotal Trendline

Property	Description
Aggregate Type (Reference Line)	Sets the reference line aggregation fuction. Select from Sum, Count, Average, Max, Min, Median, and Percentile.
Axis	Specifies the axis to which the reference line belongs. Select from X or Y.
BackwardForecastPeriod (Linear Trendline, Exponential Trendline, Power Trendline, Logarithmic Trendline, Polynomial Trendline, Fourier Trendline)	Sets a number of periods that the forecast extends backward.
DetailLevel	Specifies if the overlay calculation should include the entire data set or each detail group. Select from Total or Group.
End (Reference Band)	Specifies the end position on the value axis or the end index on the category axis.
Start (Reference Band)	Specifies the start position on the value axis or the start index on the category axis.
FieldName	Sets the field name for the reference line to use.
ForwardForecastPeriod (Linear Trendline, Exponential Trendline, Power Trendline, Logarithmic Trendline, Polynomial Trendline, Fourier Trendline)	Sets a number of periods that the forecast extends forward.
Intercept (Linear Trendline, Exponential Trendline, Power Trendline, Logarithmic Trendline, Polynomial Trendline, Fourier Trendline)	Sets the intercept value of Y-axis.
Order (Polynomial Trendline, Fourier Trendline)	Sets a number of terms in the Polynomial equation. The default value is 2, the value less than 2 is treated as 2, values greater than 6 are treated as 6.
Period (MovingAverage Trendline, CumulativeMovingAverage Trendline, ExponentialMovingAverage Trendline, WeightedMovingAverage Trendline, MovingAnnualTotal Trendline)	Specifies the number of data values used to create a line. The default value is 2.
LegendLabel	Sets the legend label of the reference line.
Value	Sets the position of the specified axis.
Display	Sets the display position. Select from Front or Back.
Name	Enter the name of the trendline.
Type	Select the type of the trendline from the list of available types.
LineStyle	Sets the style of the line. Select the LineColor, LineStyle and LineWidth properties of the line.
BackgroundColor (Reference Band)	Select the color to use for the background, or select

	the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

To add a trendline

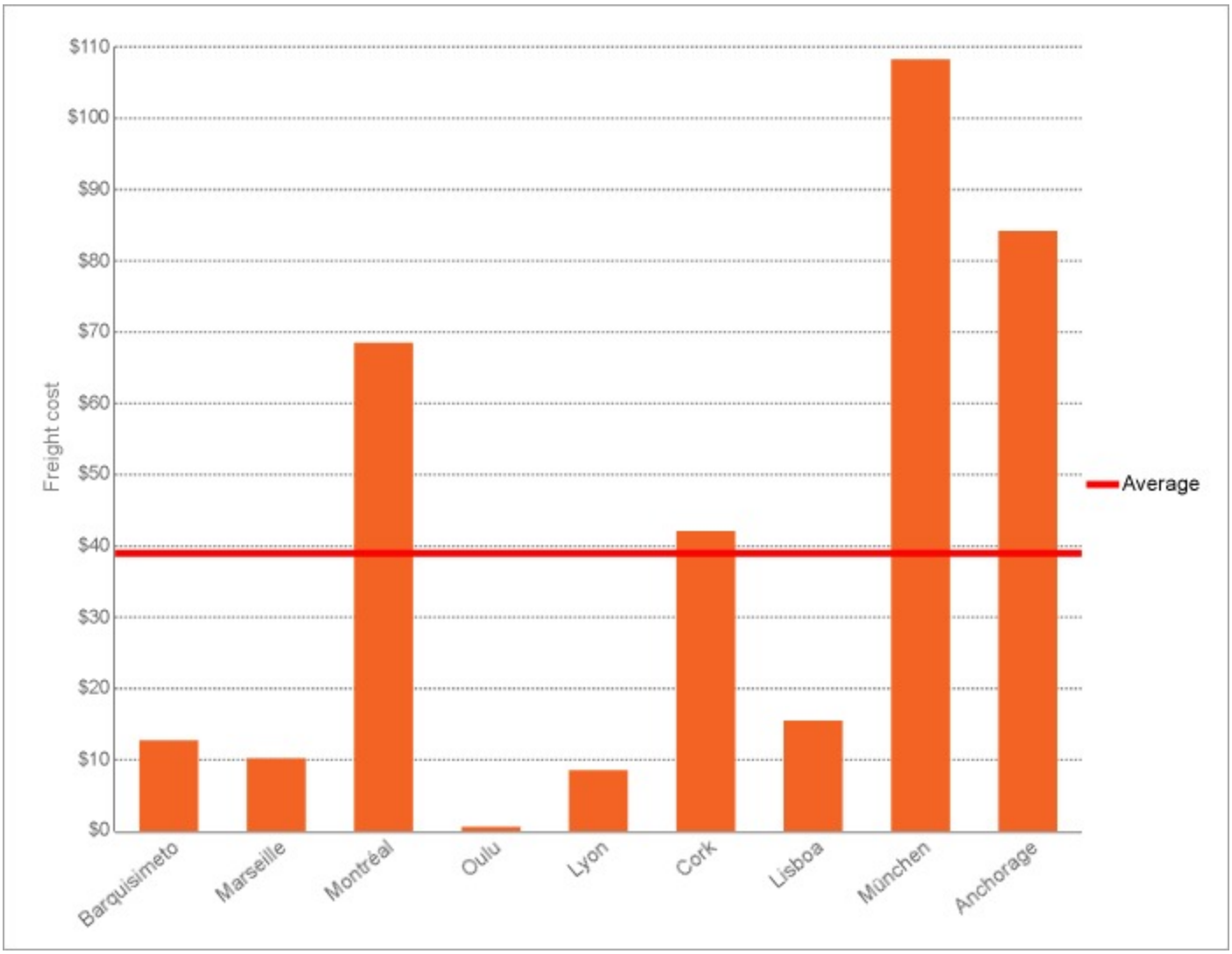
1. Select a Chart plot.
2. In the Properties window, go to the **Overlays** property and click (Collection).
3. In the OverlayDesigner Collection Editor that opens, click **Add** to add a new overlay.



4. Specify the **Type** property and other important properties.
5. Click **OK**.

Reference Line

A Reference Line is a static line, drawn at a single point along a single axis that stretches indefinitely along the opposite axis. You can set the line's position or have it calculated by the data set. For example, you may want to add a reference line at the average value to see how each value is compared against the average.



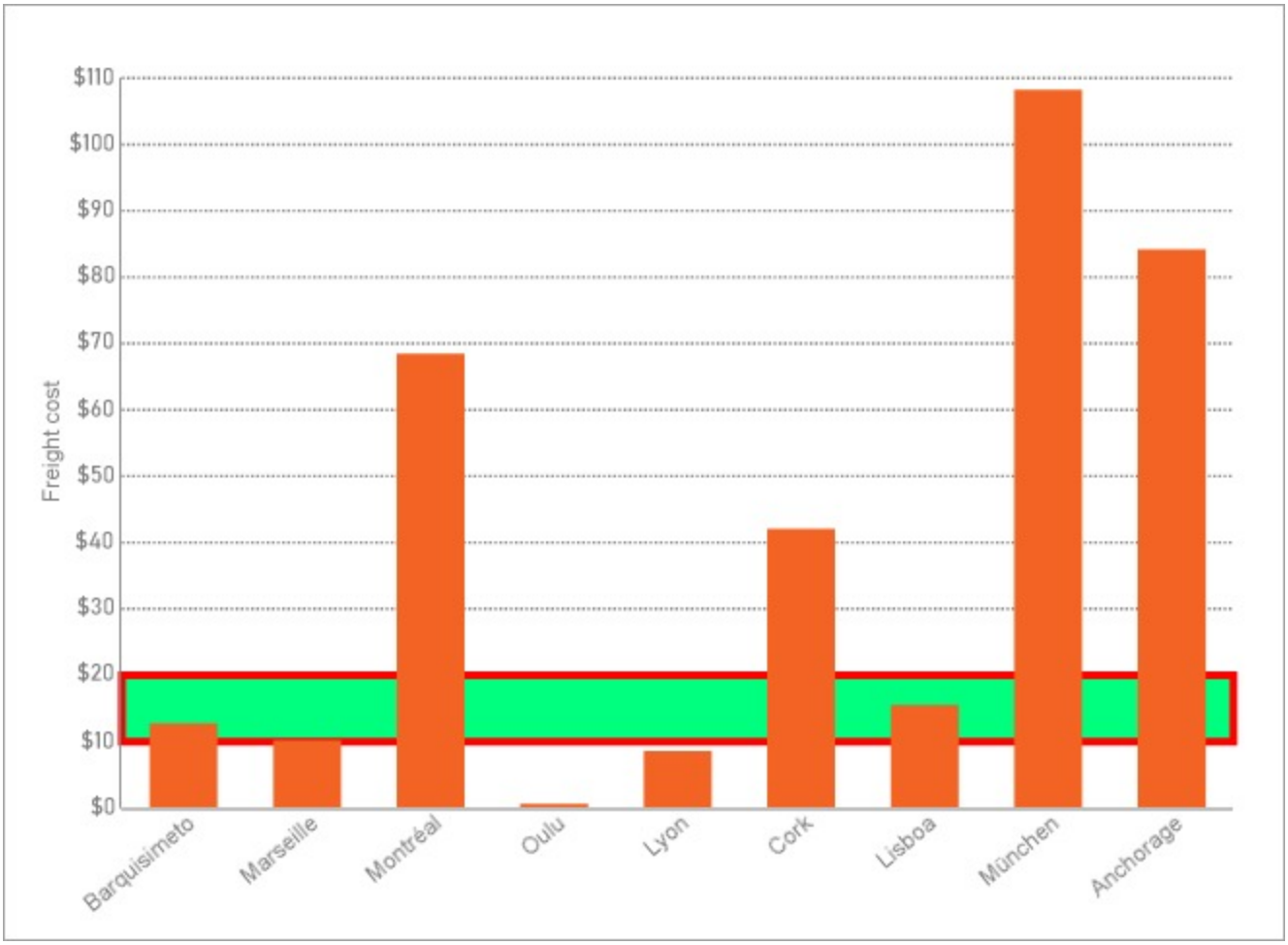
In the **OverlayDesigner Collection Editor**, you can set a Reference Line with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > AggregateType	Average
Configurations > Axis	Y
Configurations > DetailLevel	Total
Configurations > LegendLabel	Average
General > Display	Front
General > Type	ReferenceLine

Reference Band

A Reference Band is a shaded area, which is positioned along a single axis between two calculated values and extend indefinitely along the opposite axis. Each band has a start and end position that you set in the Properties.

This overlay type is not available for Candlestick, High Low Close, and High Low Open Close type charts.



In the **OverlayDesigner Collection Editor**, you can set a Reference Band with the properties as follows.

Property	Value
BackgroundColor	Green
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > Axis	Y
Configurations > End	20
Configurations > Start	10

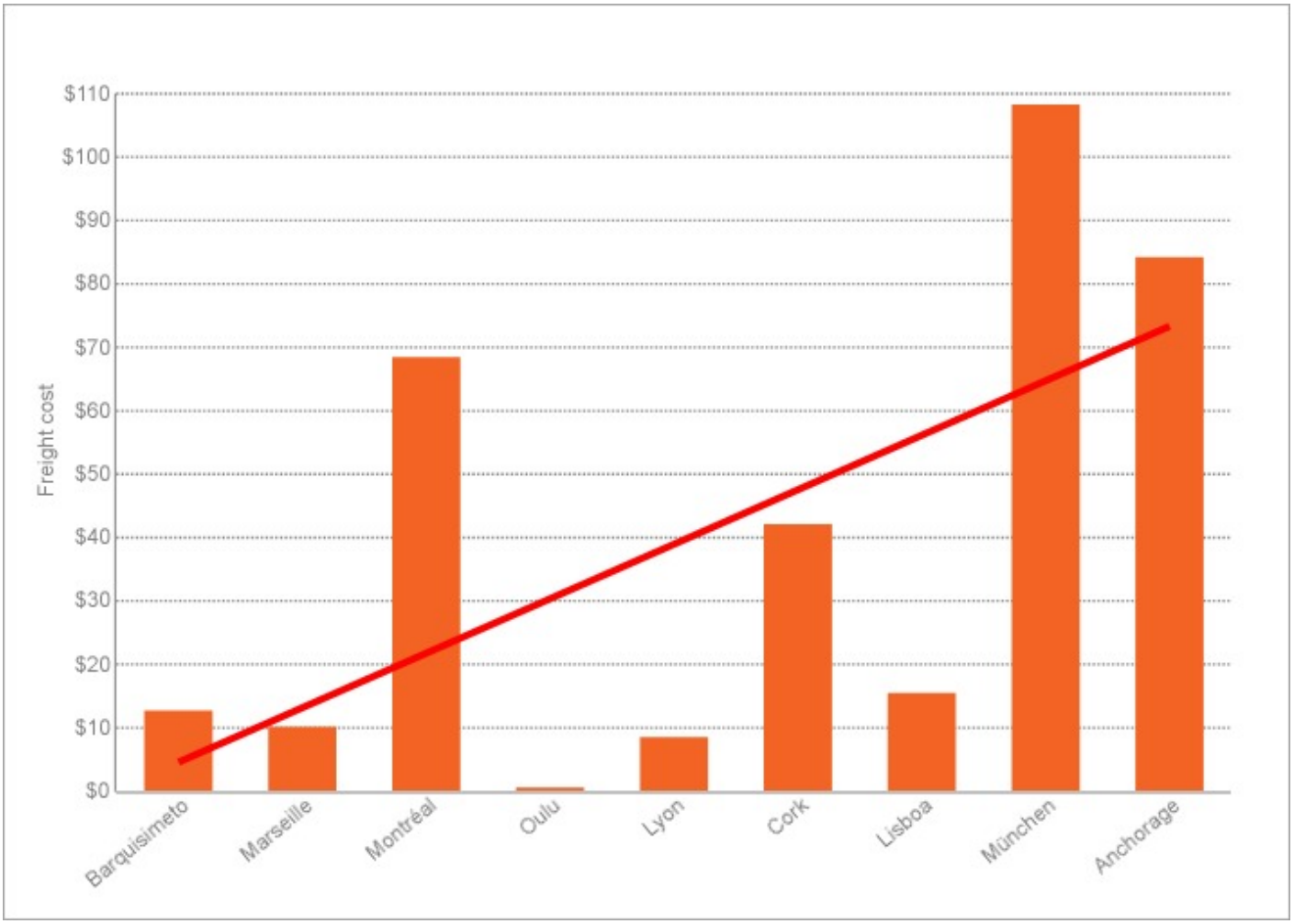
General > Display	Back
General > Type	ReferenceBand

Linear Trendline

A Linear trendline creates a best fit straight line that shows how values in a data series increase or decrease at a steady rate. This trendline uses the following equation:

```
y=mx + b
```

where "m" is the slope and "b" is the intercept.



In the **OverlayDesigner Collection Editor**, you can set a Linear trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > BackwardForecastPeriod	0

Configurations > DetailLevel	Total
Configurations > ForwardForecastPeriod	0
General > Display	Front
General > Type	LinearTrendline

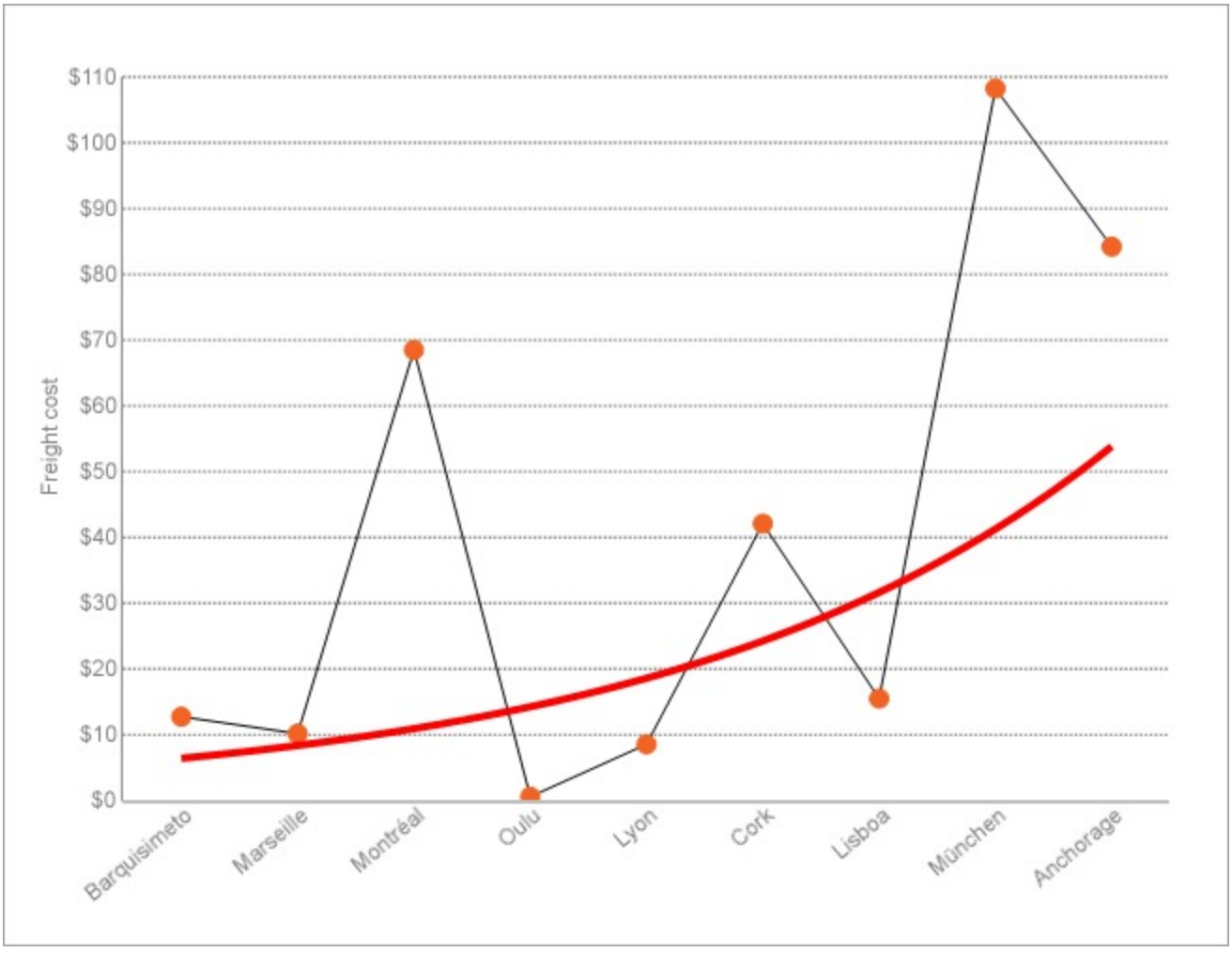
Exponential Trendline

An Exponential trendline creates a best-fit curved line that illustrates how data values increase or decrease and then level out. You can use this trendline only with positive numbers. If the data series contains any zero or negative numbers, then this trendline is not available.

This trendline uses the following equation:

y=cebx

where "c" and "b" are constants, and "e" is the base of the natural logarithm.



In the **OverlayDesigner Collection Editor**, you can set an Exponential trendline with the properties as follows.

Property	**Value**

LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > BackwardForecastPeriod	0
Configurations > DetailLevel	Total
Configurations > ForwardForecastPeriod	0
General > Display	Front
General > Type	ExponentialTrendline

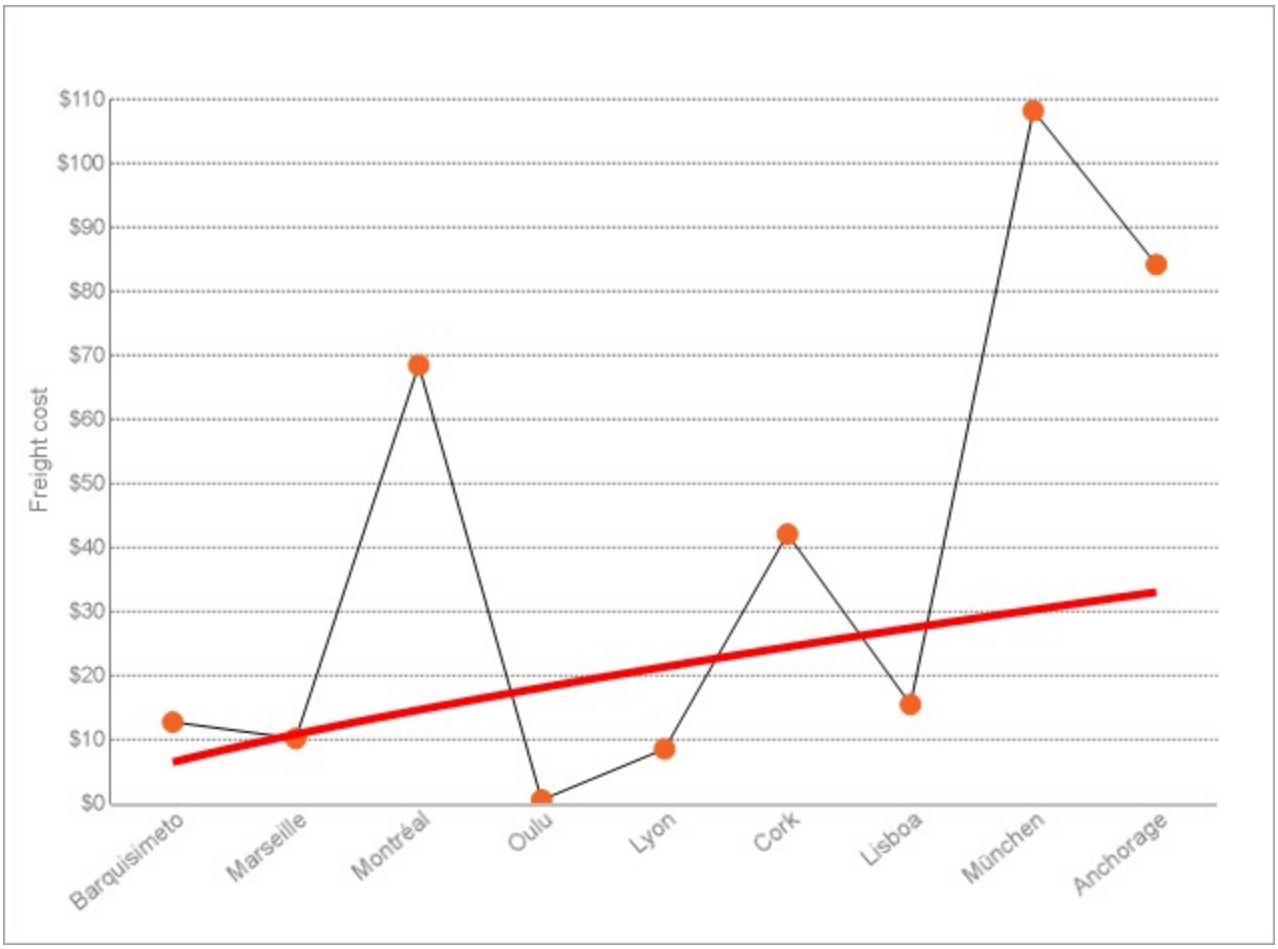
Power Trendline

A Power trendline creates a curved line to compare measurements that increase at a specific rate. You can use this trendline only with positive numbers. If the data series contains any zero or negative numbers, then this trendline is not available.

This trendline uses the following equation:

```
y = ax^b
```

where "a" and "b" are constants.



In the **OverlayDesigner Collection Editor**, you can set a Power trendline with the properties as follows.

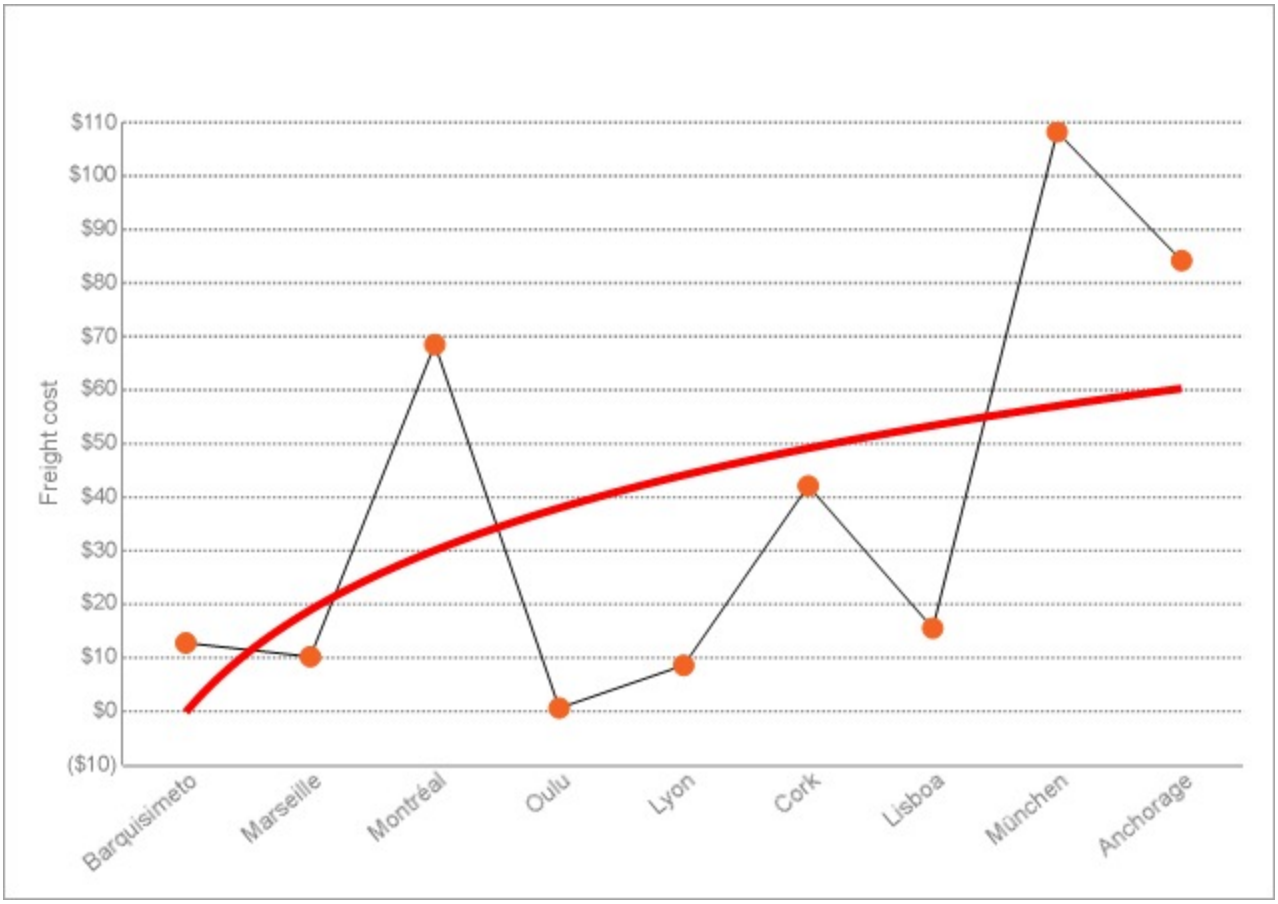
Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > BackwardForecastPeriod	0
Configurations > DetailLevel	Total
Configurations > ForwardForecastPeriod	0
General > Display	Front
General > Type	PowerTrendline

Logarithmic Trendline

A Logarithmic trendline creates a best-fit curved line that illustrates how data values increase or decrease and then level out. This trendline uses the following equation:

```
y=c ln x + b
```

where "c" and "b" are constants, and "ln" is the natural logarithm function.



In the **OverlayDesigner Collection Editor**, you can set a Logarithmic trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > BackwardForecastPeriod	0
Configurations > DetailLevel	Total
Configurations > ForwardForecastPeriod	0
General > Display	Front
General > Type	LogarithmicTrendline

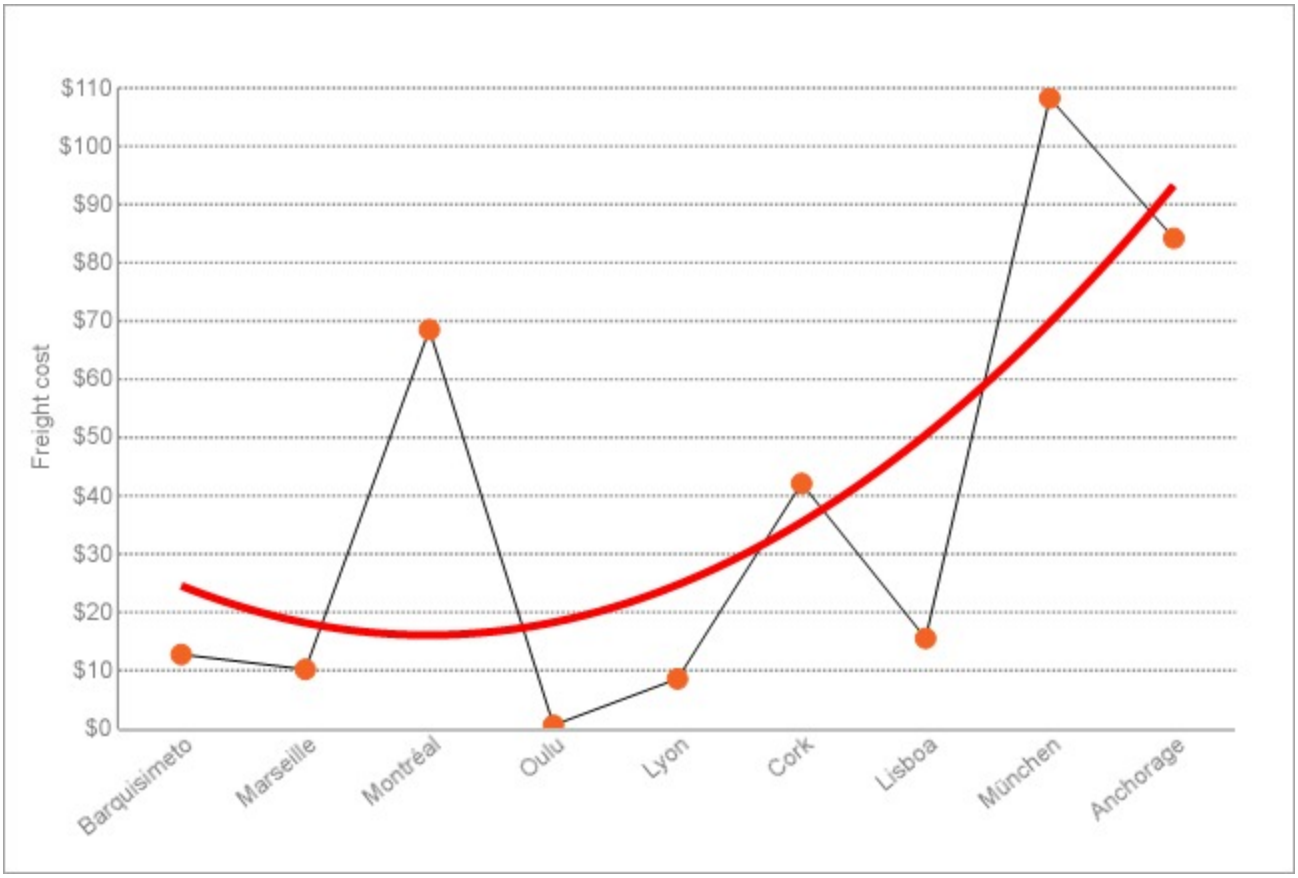
Polynomial Trendline

A Polynomial trendline creates a curved line, illustrating fluctuations in the data values. The trendline uses the following

equation:

y=b+c1x+c2x2+c3x3+

where c1, c2, c3 are constants.

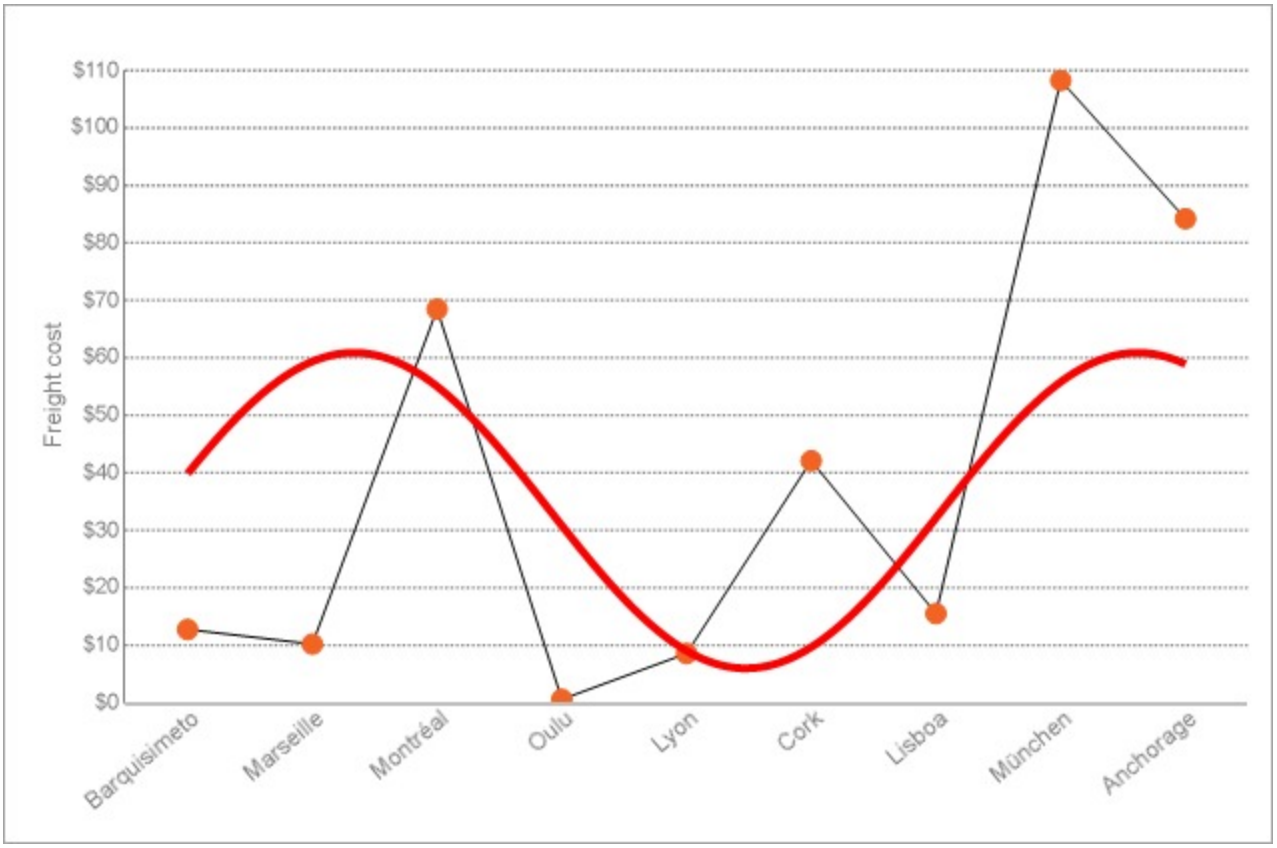


In the **OverlayDesigner Collection Editor**, you can set a Polynomial trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > BackwardForecastPeriod	0
Configurations > DetailLevel	Total
Configurations > ForwardForecastPeriod	0
Configurations > Order	2
General > Display	Front
General > Type	PolynomialTrendline

Fourier Trendline

A Fourier trendline creates a trendline, based on Fourier Series.



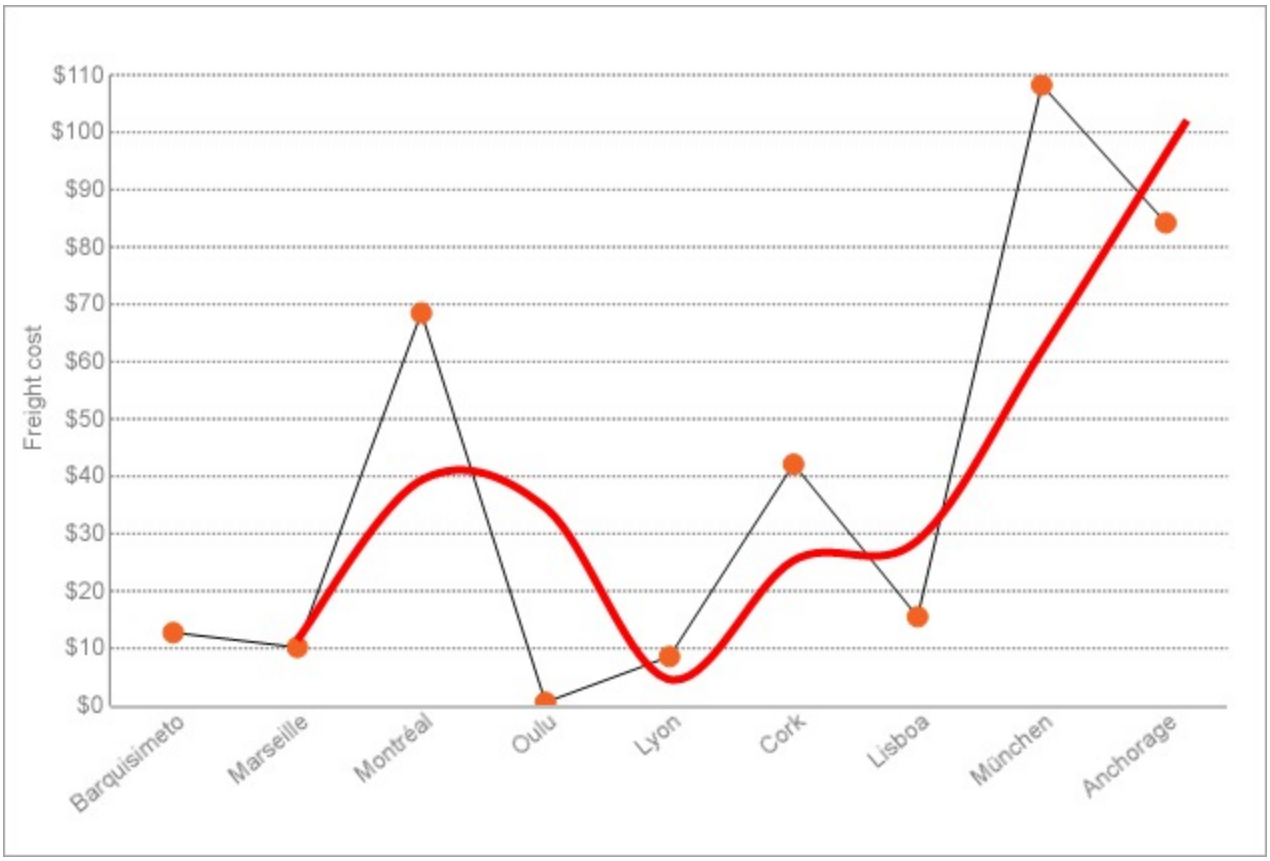
In the **OverlayDesigner Collection Editor**, you can set a Fourier trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > BackwardForecastPeriod	0
Configurations > DetailLevel	Total
Configurations > ForwardForecastPeriod	0
Configurations > Order	1
General > Display	Front
General > Type	FourierTrendline

MovingAverage Trendline

A MovingAverage trendline reduces the fluctuations in the trendline to show a smoother pattern. This trendline uses the following equation:

```
Ft = (At + At-1 + At-2 ) / n
```

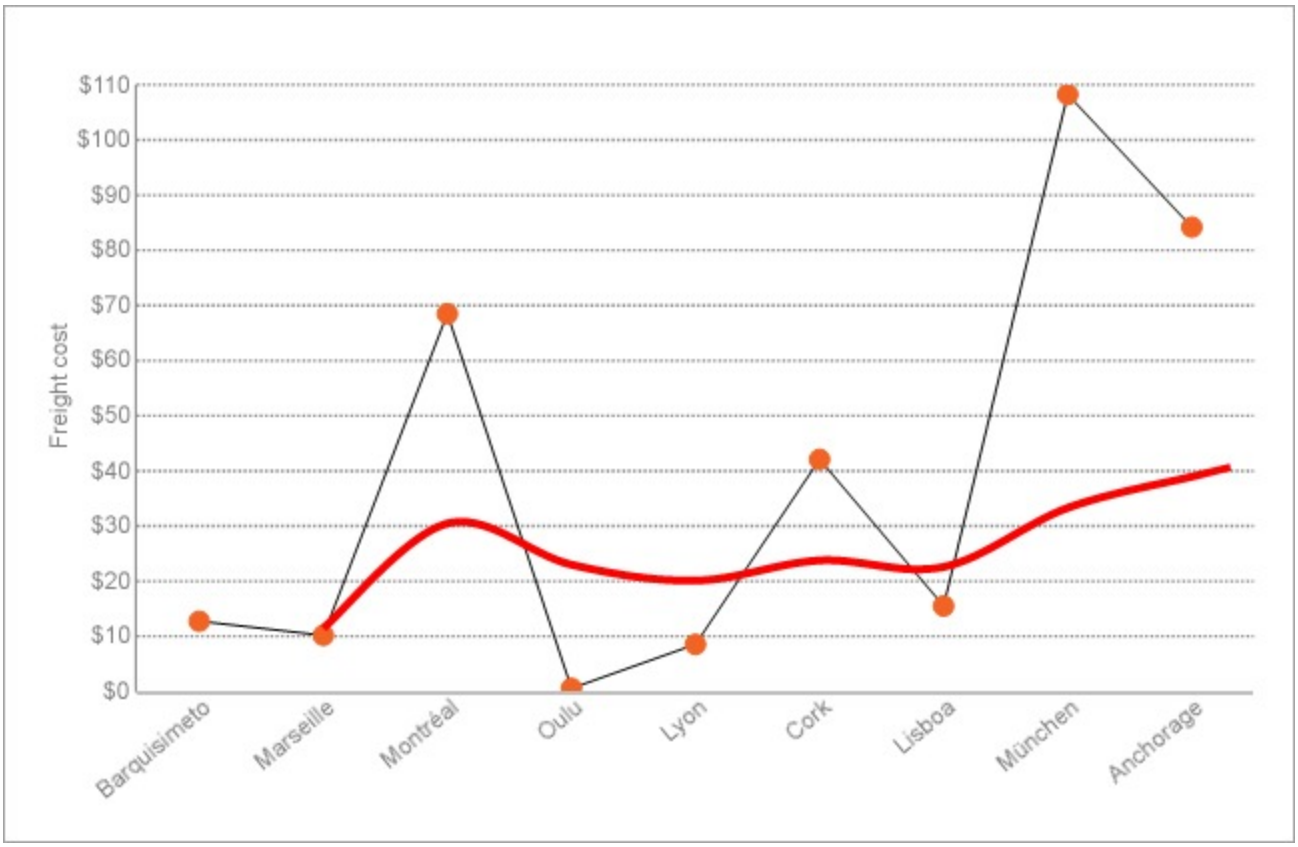


In the **OverlayDesigner Collection Editor**, you can set a MovingAverage trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > DetailLevel	Total
Configurations > Period	2
General > Display	Front
General > Type	MovingAverageTrendline

CumulativeMovingAverage Trendline

In a CumulativeMovingAverage trendline, the data arrive in an ordered datum stream, and you get the average of all data up until the current datum point.

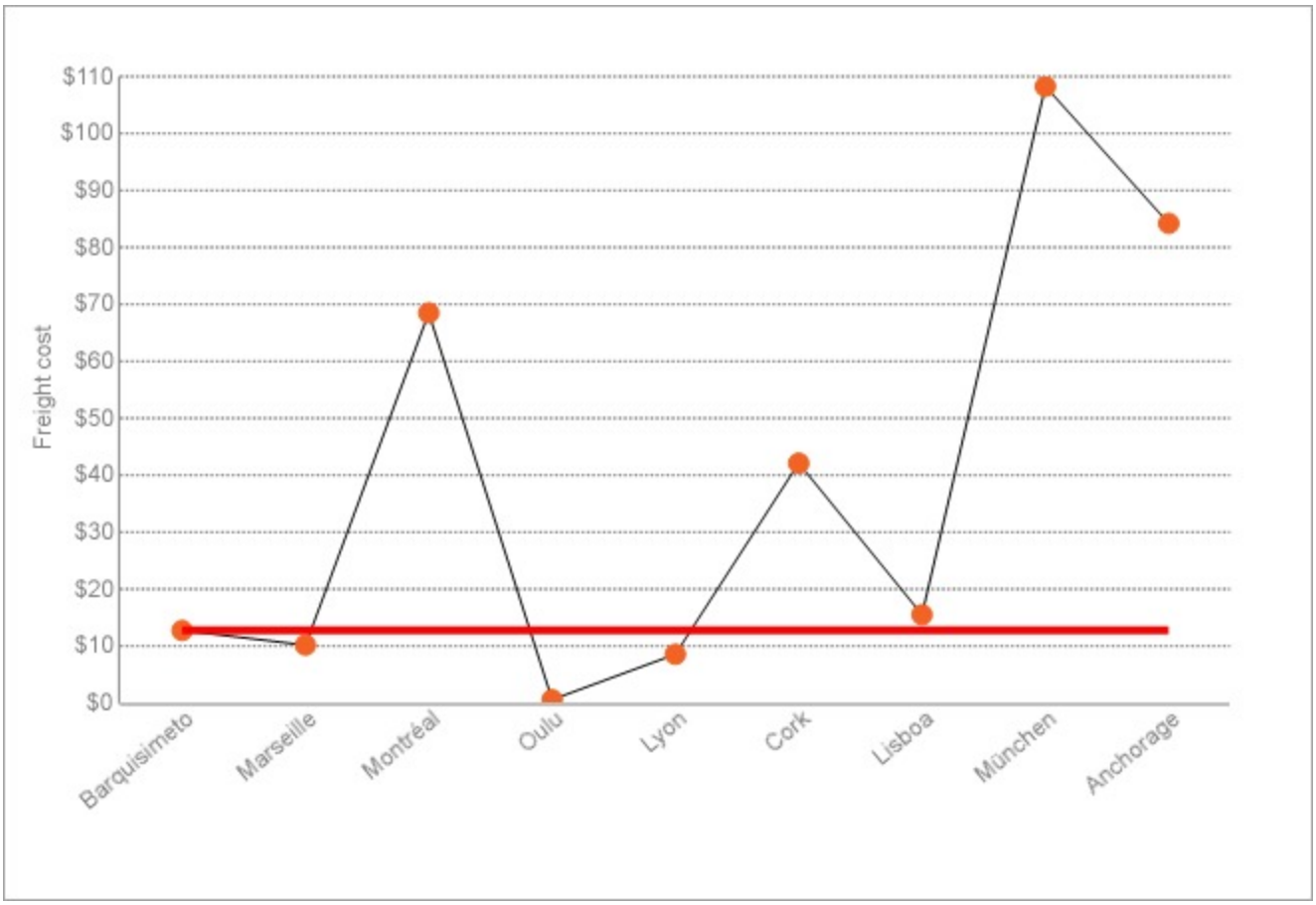


In the **OverlayDesigner Collection Editor**, you can set a CumulativeMovingAverage trendline with the properties as follows.

Property	Value
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > DetailLevel	Total
Configurations > Period	2
General > Display	Front
General > Type	CumulativeMovingAverageTrendline

ExponentialMovingAverage Trendline

An exponential moving average (EMA), also known as an exponentially weighted moving average (EWMA), is a first-order infinite impulse response filter that applies weighting factors, which decrease exponentially. The weighting for each older datum decreases exponentially, never reaching zero.

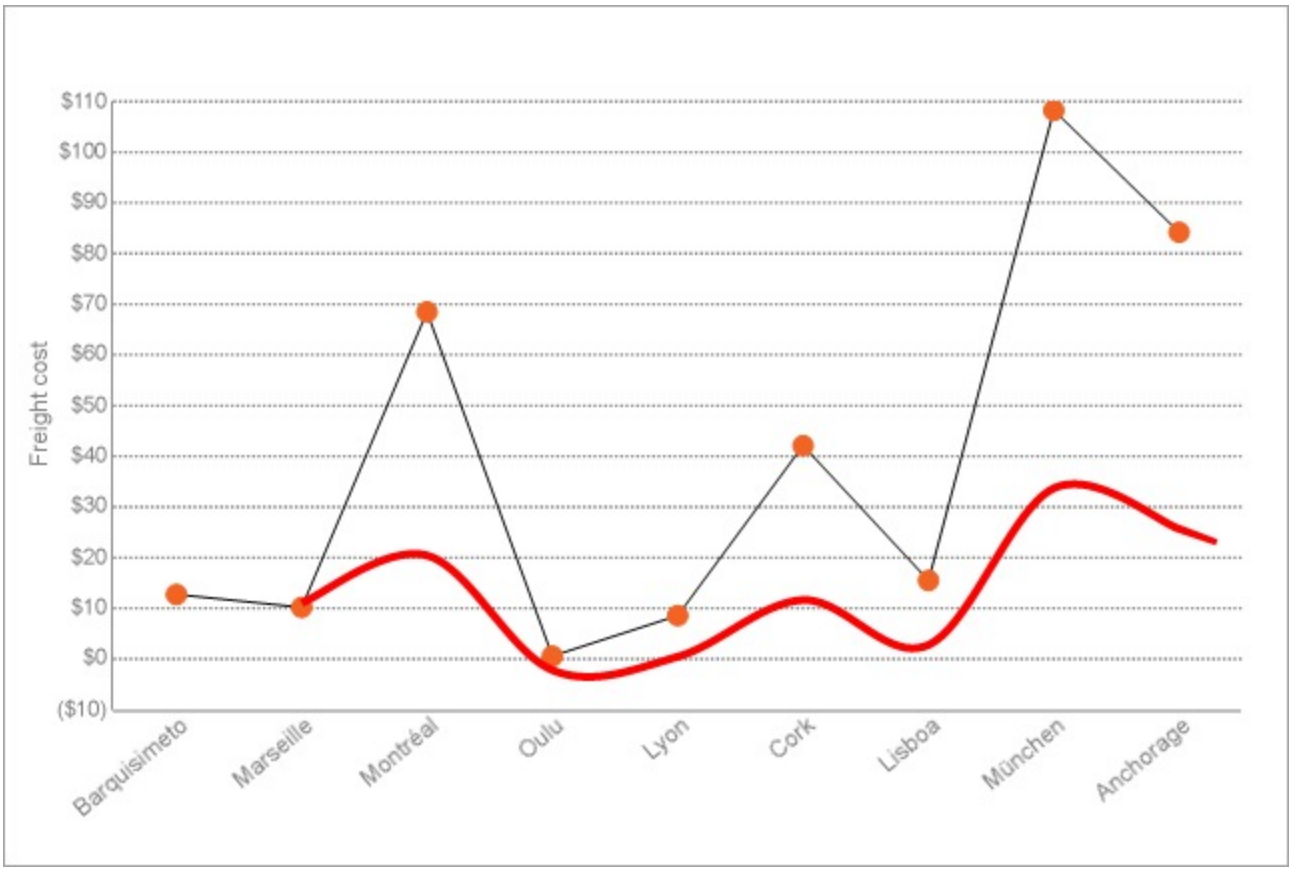


In the **OverlayDesigner Collection Editor**, you can set an ExponentialMovingAverage trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > DetailLevel	Total
Configurations > Period	2
General > Display	Front
General > Type	ExponentialMovingAverageTrendline

WeightedMovingAverage Trendline

A WeightedMovingAverage trendline is an average that has multiplying factors to give different weights to data at different positions. Mathematically, the moving average is the convolution of the datum points with a fixed weighting function.

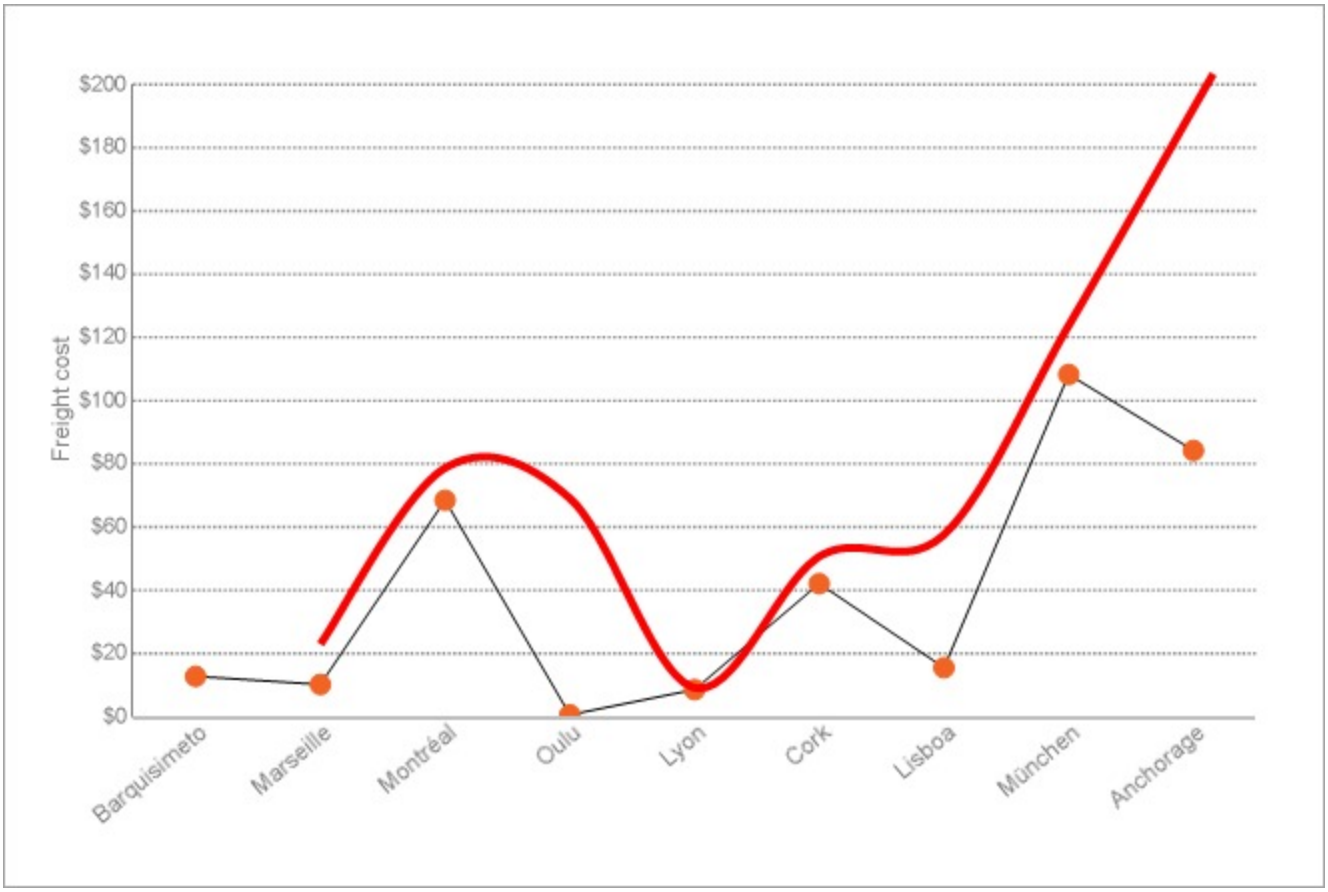


In the **OverlayDesigner Collection Editor**, you can set a WeightedMovingAverage trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > DetailLevel	Total
Configurations > Period	2
General > Display	Front
General > Type	WeightedMovingAverageTrendline

MovingAnnualTotal Trendline

A MovingAnnualTotal trendline is a moving total of the prior 'n' points at an abstract level.



In the **OverlayDesigner Collection Editor**, you can set a MovingAnnualTotal trendline with the properties as follows.

Property	**Value**
LineStyle > LineColor	Red
LineStyle > LineStyle	Solid
LineStyle > LineWidth	3pt
Configurations > DetailLevel	Total
Configurations > Period	2
General > Display	Front
General > Type	MovingAnnualTotalTrendline

Classic Chart

Note: The Classic Chart is by default hidden from the toolbox. If you want to use Classic Chart instead of new Chart, you can enable it from ActiveReports.config file located at C:\Program Files (x86)\MESCIUS\ActiveReports 18. See Configure ActiveReports for more information

The **Chart** data region shows your data in a graphical representation that often makes it easier for users to comprehend large amounts of data quickly. Different types of charts are more efficient for different types of

information, so we offer a wide variety of chart types. This makes it easy and cost effective to add charting to your reports, as there is no need to purchase and integrate a separate charting tool.

Chart Types

Bar Charts

Bar charts present each series as a horizontal bar, and group the bars by category. The x-axis values determine the lengths of the bars, while the y-axis displays the category labels. With a bar chart, you can select from the following subtypes.

- **Plain or Simple Bar:** A bar chart used to compare values of items across categories.
- **Stacked Bar:** A bar chart with two or more data series stacked one on top of the other that shows how each value contributes to the total.
- **Percent Stacked Bar:** A bar chart with two or more data series stacked one on top of the other to sum up to 100% that shows how each value contributes to the total with the relative size of each series representing its contribution to the total.

Column Charts

Column charts present each series as a vertical column, and group the columns by category. The y-axis values determine the heights of the columns, while the x-axis displays the category labels. With a column chart, you can select from the following subtypes.

- **Plain or Simple Bar:** Compares values of items across categories.
- **Stacked Bar:** A column chart with two or more data series stacked one on top of the other that shows how each value contributes to the total.
- **Percent Stacked Bar:** A column chart with two or more data series stacked one on top of the other to sum up to 100% that shows how each value contributes to a total with the relative size of each series representing its contribution to the total.

Scatter Charts

Scatter charts present each series as a point or bubble. The y-axis values determine the heights of the points, while the x-axis displays the category labels. With a scatter chart, you can select from the following subtypes.

- **Plain or Simple Scatter:** Shows the relationships between numeric values in two or more series sets of XY values.
- **Scatter or Lines:** Plots points on the X and Y axes as one series and uses a line to connect points to each other.
- **Scatter or Smooth Lines:** Plots points on the X and Y axes as one series and uses a line with the angles smoothed out to connect points to each other.

Line Charts

Line charts present each series as a point, and connect the points with a line. The y-axis values determine the heights of the points, while the x-axis displays the category labels. With a line chart, you can select from the following subtypes.

- **Plain or Simple Line:** Compares trends over a period of time or in certain categories.
- **Smooth Line:** Plots curves rather than angled lines through the data points in a series to compare trends over a period of time or in certain categories. Also known as a Bezier chart.

Pie Charts

Pie charts are circular plots that display the proportionate contribution of each category which is represented by a slice or pie. The magnitude of the dependent variable is proportional to the pie or slice angle. These charts can be used for plotting a single series with non-zero and positive values.

- **Simple Pie:** Shows how the percentage of each data item contributes to the total.
- **Exploded Pie**: Shows how the percentage of each data item contributes to the total, with the pie slices pulled out from the center to show detail.

Bubble Charts

Bubble charts are used to plot three dimensional data. These charts are often used to depict financial data.

Doughnut Charts

A doughnut chart is a pie chart with a hole in the center. The chart is divided into different portions that show the percentage each value contributes to the total. Like pie charts, the doughnut chart is used with small sets of data to compare categories.

- **Simple Doughnut:** Shows how the percentage of each data item contributes to a total percentage.
- **Exploded Doughnut:** Shows how the percentage of each data item contributes to the total, with the pie slices pulled out from the center to show detail.

Stock Charts

Stock charts present each series as a line with markers showing some combination of high, low, open, and close values. The y-axis values determine the heights of the lines, while the x-axis displays the category labels.

- **High Low Close**: Displays stock information using High, Low, and Close values. High and low values are displayed using vertical lines, while tick marks on the right indicate closing values.
- **Open High Low Close**: Displays stock information using Open, High, Low, and Close values. Opening values are displayed using lines to the left, while lines to the right indicate closing values. The high and low values determine the top and bottom points of the vertical lines.
- **Candlestick**: Displays stock information using High, Low, Open and Close values. The height of the wick line is determined by the High and Low values, while the height 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.

Funnel Charts

Funnel charts show how the percentage of each data item contributes to the whole, with the largest value at the top and the smallest at the bottom. This chart type works best with relatively few data items.

Pyramid Charts

Pyramid charts show how the percentage of each data item contributes to the whole, with the smallest value at the top and the largest at the bottom. This chart type works best with relatively few data items.

Three Line Break Charts

In Three Line Break Charts, vertical boxes or lines illustrate price changes of an asset or market. The price in a three line

break graph must break the prior high or low set in the NewLineBreak property in order to reverse the direction of the graph.

Kagi Charts

Kagi charts display 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).

Renko Charts

Renko charts depict bricks of uniform size 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. A change in box color and direction signifies a trend reversal.

Point and Figure Charts

In Point and Figure charts, stacked columns of Xs indicate that demand exceeds supply and columns of Os 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 you set. A new column is added when the price reverses to the level of the BoxSize value multiplied by the ReversalAmount you set. This calculation of pricing trends is best suited for long-term financial analysis.

Gantt Charts

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

Dot Plot Chart

A Dot Plot chart is a statistical chart containing group of data points plotted on a simple scale. Dot Plot chart are used for continuous, quantitative, and univariate data. The dot plot chart has one subtype. The plain Dot Plot chart displays simple statistical plots. It is ideal for small to moderate sized data sets. You can also highlight clusters and gaps, as well as outliers, while conserving numerical information.

Smart Panels

Smart panels are the dialogs that group commonly-used properties of the chart elements using tabs and groups. ActiveReports .NET provides you with the smart panels for the following chart features:

- Chart appearance
- Chart data
- Chart legend
- Chart X axis
- Chart Y axis

These smart panels can be accessed from the Property dialog link on the **Properties** pane:

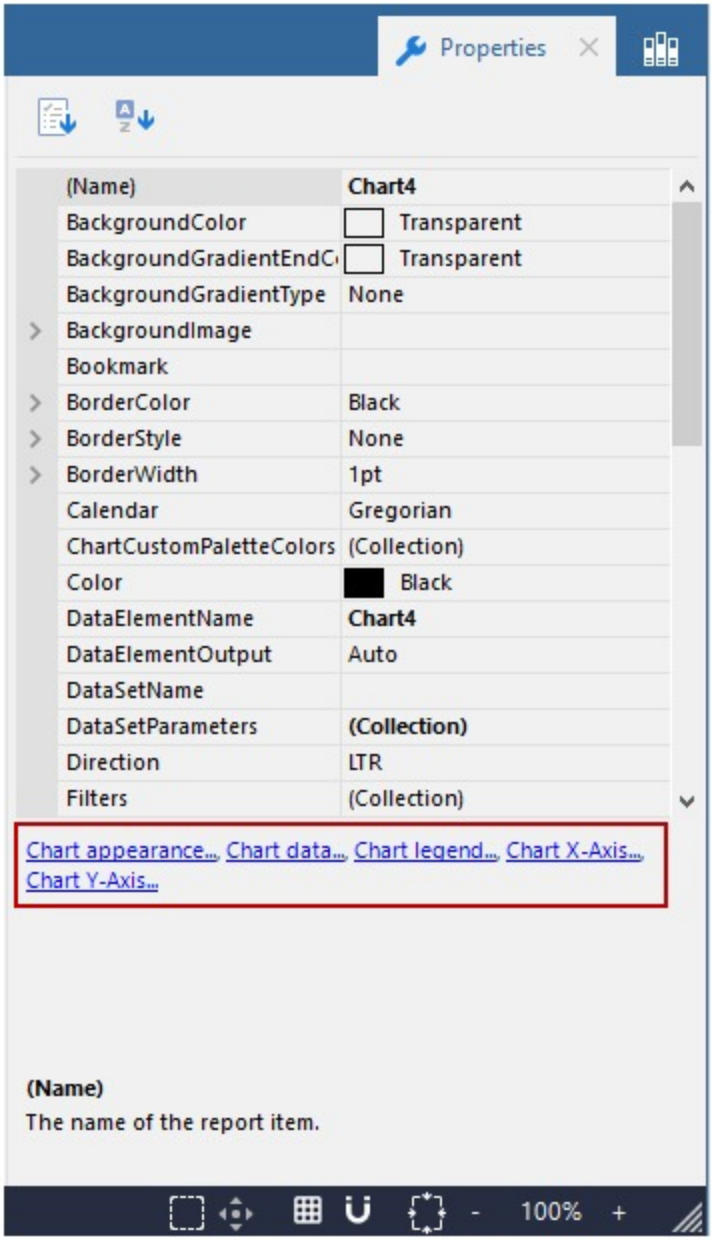
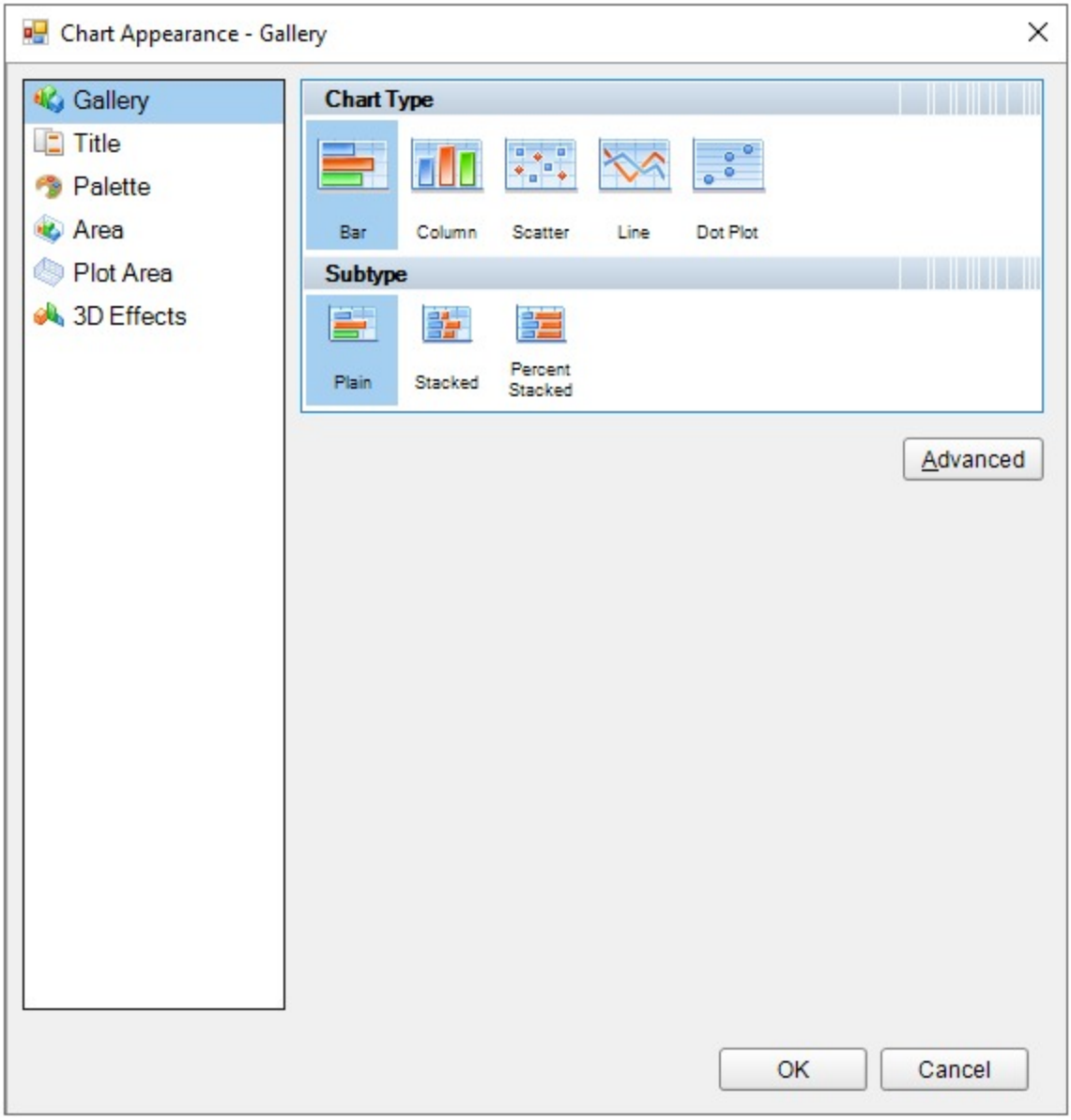


Chart Appearance Smart Panel



Gallery

- Chart Types: Select the type of classic chart.
- Subtype: Select a sub-category of the selected chart types.

Title

Chart title: Enter an expression or text to use for the title.

Font

Family: Choose the font family name.

Size: Choose the size in points for the font.

Style: Choose Normal or Italic.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, and LineThrough.

Palette

- Palette: Select a suitable palette from the available color palettes.

Area and Plot Area

Border

Style: Choose an enumerated style for the border.

Width: Choose a width value between **0.25pt** and **20pt**.

Color: Select a Web or Custom color.

Background Fill Color

Fill Color: Select a Web or Custom color.

Gradient: Choose from one of the following gradient styles.

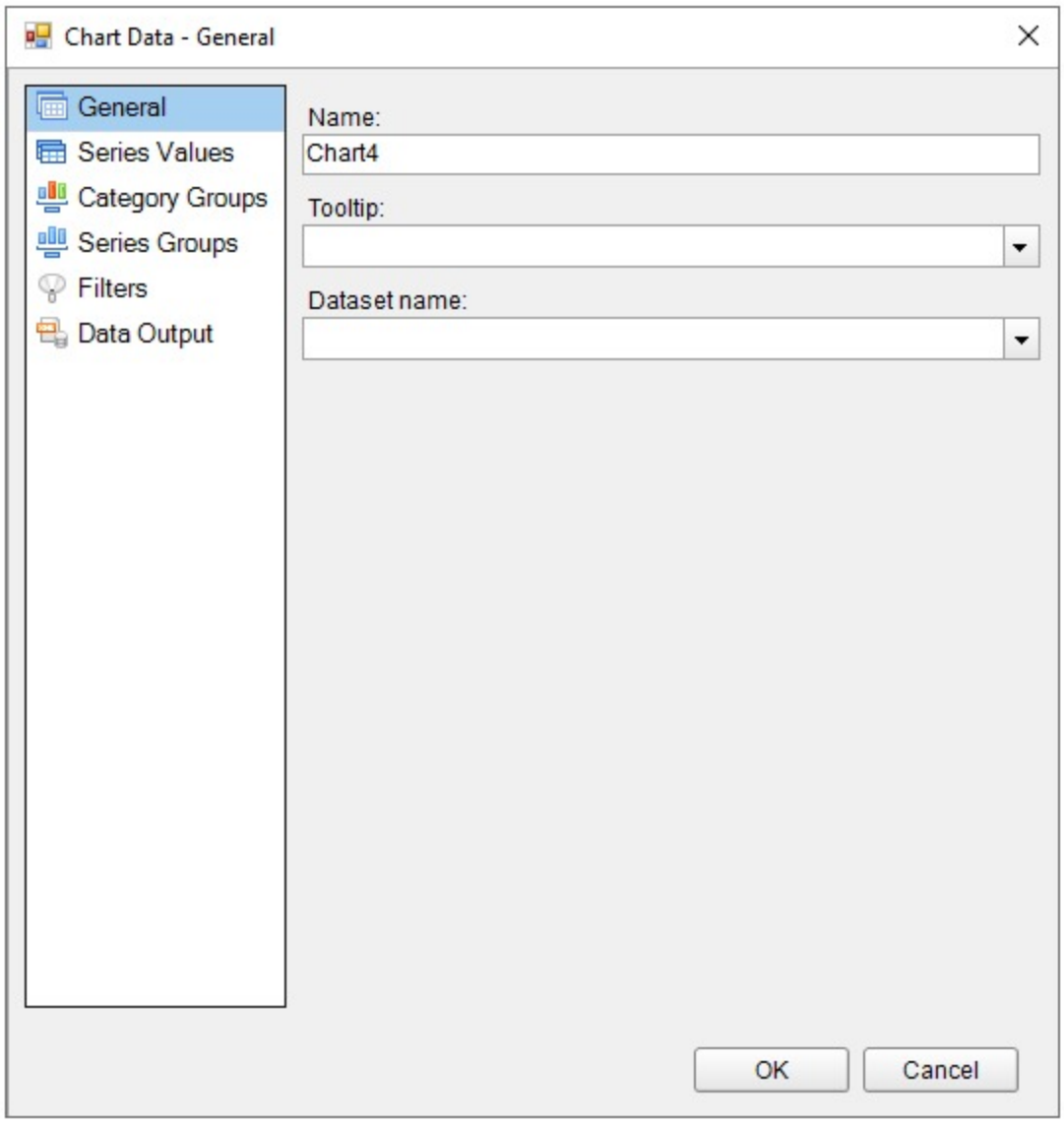
- **None**: No gradient is used. The Fill Color is used to fill the area and the Gradient End Color property is ignored.
- **LeftRight**: A gradient is used. The Fill Color property defines the color at the left, and the Gradient End Color property defines the color at the right. The two colors are gradually blended in between these areas.
- **TopBottom**: A gradient is used. The Fill Color property defines the color at the top, and the Gradient End Color property defines the color at the bottom. The two colors are gradually blended in between these areas.
- **Center**: A gradient is used. The Fill Color property defines the color at the center, and the Gradient End Color property defines the color at the edges. The two colors are gradually blended in between these areas.
- **DiagonalLeft**: A gradient is used. The Fill Color property defines the color at the top left, and the Gradient End Color property defines the color at the bottom right. The two colors are gradually blended in between these areas.
- **DiagonalRight**: A gradient is used. The Fill Color property defines the color at the top right, and the Gradient End Color property defines the color at the bottom left. The two colors are gradually blended in between these areas.
- **HorizontalCenter**: A gradient is used. The Gradient End Color property defines the horizontal band of color across the center, and the Fill Color property defines the color at the top and bottom. The two colors are gradually blended in between these areas.
- **VerticalCenter**: A gradient is used. The Gradient End Color property defines the vertical band of color across the center, and the Fill Color property defines the color at the left and right. The two colors are gradually blended in between these areas.

Gradient End Color: When you choose any gradient style other than None, this property becomes available. Choose a Web or Custom color.

3D Effect

Display the chart with 3D visual effects.

Chart Data Smart Panel



When you first open the Chart Data dialog, you can select a **Dataset name** to associate with the chart. The list is populated with all of the datasets in the report's dataset collection.

This dialog also gives you access to the following related pages.

General

Name: Enter a name for the chart that is unique within the report. This name is displayed in the Document Outline and in XML exports.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Dataset name: Assign a dataset name to associate with the chart. The combo box is populated with all of the datasets

in the report's dataset collection.

Series Values

Add at least one Value series to determine the size of the chart element. Click the plus sign button to enable the General tab. Once you have one or more value series in place, you can use the arrow buttons to change the order or the X button to delete them.

Another way to add Chart Series Values is to drag fields from the Report Explorer onto the tray along the top edge of the chart that reads **Drop data fields here**.

If you have already added values, you can right-click any value displayed in the UI along the top of the chart and choose **Edit** to open this dialog.

The Series Values page has the following tabs.

General

The General tab of the Series Values page allows you to control different items depending on the Chart Type you have chosen.

For all Chart types

Series label: Enter an expression to use as a series label to display in the legend.

For Scatter or Bubble Chart types

X: Enter an expression to use as an X value.

Y: Enter an expression to use as a Y value.

Size: If the chart type is bubble, enter an expression to use as the bubble size value.

For Stock Chart

High: Enter an expression to use as the high value.

Low: Enter an expression to use as the low value.

Open: Enter an expression to use as the open value.

Close: Enter an expression to use as the close value.

For Column, Bar, Line, Pie, Area, Doughnut, Funnel, Pyramid, ThreeLineBreak, Kagi, Renko, PointAndFigure, or DotPlot Chart types

Value: Enter an expression to use as a series value.

For Column, Line, or Area Chart types

Chart Type: For Composite charts, select the chart type to use in combination with other chart types within the same plot area. The available chart types are:

- Column Plain
- Column Stacked
- Column Percent Stacked
- Area Plain
- Area Stacked

- Area Percent Stacked
- Line Plain
- Smooth Line

Y-Axis: Select the Y-axis from the list of available Y-axes.

Styles

Line/Border

These properties control the appearance of the border of bars or columns, or the lines, depending on the type of chart.

Style: Choose one of the enumerated styles for the lines.

Width: Choose a width value between 0.25pt and 20pt for the thickness of the lines.

Color: Choose a Web or Custom color to use for the lines.

Background Fill Color

These properties control the appearance of the background of the series values.

Fill Color: Choose a Web or Custom color to fill the background.

Gradient: Choose from one of the following gradient styles.

- **None**: No gradient is used. A single color (defined by the **Fill Color** property above) is used to fill the area and the **Gradient End Color** property remains disabled.
- **LeftRight**: A gradient is used. The **Fill Color** property defines the color at the left, and the **Gradient End Color** property defines the color at the right. The two colors are gradually blended in between these areas.
- **TopBottom**: A gradient is used. The **Fill Color** property defines the color at the top, and the **Gradient End Color** property defines the color at the bottom. The two colors are gradually blended in between these areas.
- **Center**: A gradient is used. The **Fill Color** property defines the color at the center, and the **Gradient End Color** property defines the color at the edges. The two colors are gradually blended in between these areas.
- **DiagonalLeft**: A gradient is used. The **Fill Color** property defines the color at the top left, and the **Gradient End Color** property defines the color at the bottom right. The two colors are gradually blended in between these areas.
- **DiagonalRight**: A gradient is used. The **Fill Color** property defines the color at the top right, and the **Gradient End Color** property defines the color at the bottom left. The two colors are gradually blended in between these areas.
- **HorizontalCenter**: A gradient is used. The **Gradient End Color** property defines the horizontal band of color across the center, and the **Fill Color** property defines the color at the top and bottom. The two colors are gradually blended in between these areas.
- **VerticalCenter**: A gradient is used. The **Gradient End Color** property defines the vertical band of color across the center, and the **Fill Color** property defines the color at the left and right. The two colors are gradually blended in between these areas.

Gradient End Color: When you choose any gradient style other than **None**, this property becomes available. Choose a Web or Custom color to blend with the **Fill Color** in the background of the series.

Markers

Marker type: Choose one of the following values to determine the shape of the marker or whether one is displayed.

- **None** - Markers are not used. (Default)
- **Square** - Markers are square.
- **Circle** - Markers are circular.
- **Diamond** - Markers are diamond shaped.
- **Triangle** - Markers are triangular.
- **Cross** - Markers are cross shaped.
- **Auto** - A shape is chosen automatically.

Marker size: Enter a value between **2pt** and **10pt** to determine the size of the plotting area of the markers.

Plot data as secondary: If the chart type is Column, Bar, or DotPlot, you can select this check box and select whether to use a Line or Points to show the data.

Labels

Show point labels: Select this check box to display a label for each chart value. Selecting this box enables the disabled properties on this page.

Data label: Enter a value to use as the label, or select **<Expression...>** to open the Expression Editor.

Format code: Select one of the provided format codes or use a custom .NET formatting code to format dates or numbers. For more information, see MSDN's Formatting Types topic.

Position: Leave **Auto** selected to use the default point label position for the chart type, or select an enumerated value to position the labels.

Angle: Enter the value in tenths of degrees to use for the angle of the point label text. The default (0°) position denotes no angle and renders regular horizontal text.

Font

Family: Choose the font family name.

Size: Choose the size in points for the font.

Style: Choose **Normal** or **Italic**.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, and LineThrough.

Action

Choose from the following actions to perform when the user clicks on the chart element.

None: The default behavior is to do nothing when a user clicks the chart element at run time.

Jump to report: For drill-through reporting, select this option and provide the name of a local report, the relative path of a report in another folder, or the full path of a report on another server.

Parameters

- **Name**: Supply the exact names of any parameters required for the targeted report. Note that parameter names you supply in this must match parameters in the target report.

> **Important:** The Parameter Name must exactly match the name of the parameter in the detail report. If any parameter is spelled differently, capitalized differently, or if an expected parameter is not supplied, the drill-through report will fail.

- **Value**: Enter a Parameter Value to pass to the detail report. This value must evaluate to a valid value for the parameter.
- **Omit**: Select this check box to omit this parameter from the report.

Jump to bookmark: Select this option and provide a valid Bookmark ID to allow the user to jump to the report control with that Bookmark ID.

Jump to URL: Select this option and provide a valid URL to create a hyperlink to a Web page.

Data Output

Element name: Enter a name to be used in the XML output for this chart element.

Output: Choose Yes or No to decide whether to include this chart element in the XML output.

Category Groups

Add Category Groups to group data and provide labels for the chart elements. Click the **Add** button to enable the General tab. Once you have one or more category groups in place, you can use the arrow buttons to change the order or the X button to delete them.

Another way to add Category Groups is to drag fields from the Report Explorer onto the tray along the bottom edge of the chart that reads **Drop category fields here**.

If you have already added values, you can right-click the value displayed in the UI along the bottom of the chart and choose **Edit** to open this dialog.

The Category Groups page has the following tabs.

General

Name: Enter a name for the group that is unique within the report. This name can be called in code.

Group on: Enter an expression to use for grouping the data.

Label: Enter an expression to use as a label for the group. You can select **<Expression...>** to open the Expression Editor.

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right:

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on

the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

The Sorting tab of Category Groups page allows you to enter new sort expressions and remove or change the order of them using the X or arrow buttons. For each sort expression in this list, you can also choose the direction.

Expression: Enter an expression by which to sort the data in the group.

Direction: Select whether you want to sort the data in an Ascending or Descending direction.

Data Output

Element name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose Yes or No to decide whether to include this group in the XML output.

Series Groups

Optionally add Series Groups for extra levels of data (for example, Orders by Country can be broken down by year as well). Labels for the series are displayed in the chart legend. Click the **Add** button to open the General page. Once you have one or more series groups in place, you can use the arrow buttons to change the order or the X button to delete them.

Another way to add Series Groups is to drag fields from the Report Explorer onto the tray along the right edge of the chart that reads **Optionally drop series fields here**.

If you have already added values, you can right-click the value displayed in the UI along the right edge of the chart and choose **Edit** to open this dialog.

The Series Groups page has the following tabs.

General

Name: Enter a name for the group that is unique within the report. This name can be called in code.

Group on: Enter an expression to use for grouping the data.

Label: Enter an expression to use as a label for the group. You can select **<Expression...>** to open the Expression Editor.

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right:

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

The Sorting tab of Series Groups page allows you to enter new sort expressions and remove or change the order of them using the X or arrow buttons. For each sort expression in this list, you can also choose the direction.

Expression: Enter an expression by which to sort the data in the group.

Direction: Select whether you want to sort the data in an Ascending or Descending direction.

Data Output

Element name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose Yes or No to decide whether to include this group in the XML output.

Filters

Chart Data Filters Page

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the chart.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right:

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right.
  Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values (used with the **In** and **Between** operators) separate values using commas.

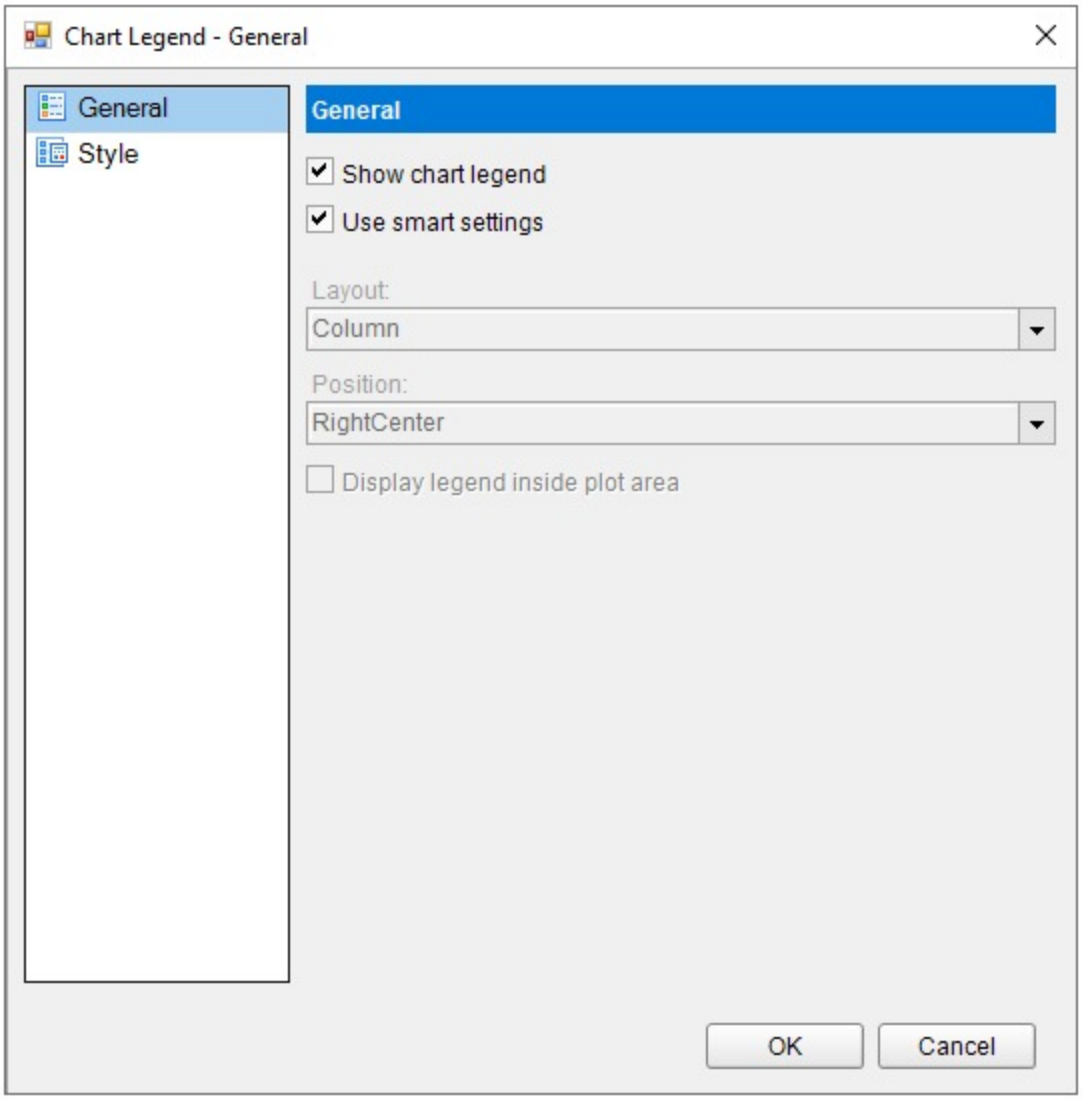
Data Output

Chart Data Output Page

Element name: Enter a name to be used in the XML output for the chart.

Output: Choose one between Auto, Yes, No or Contents Only to decide whether to include this group in the XML output.

Chart Legend Smart Panel



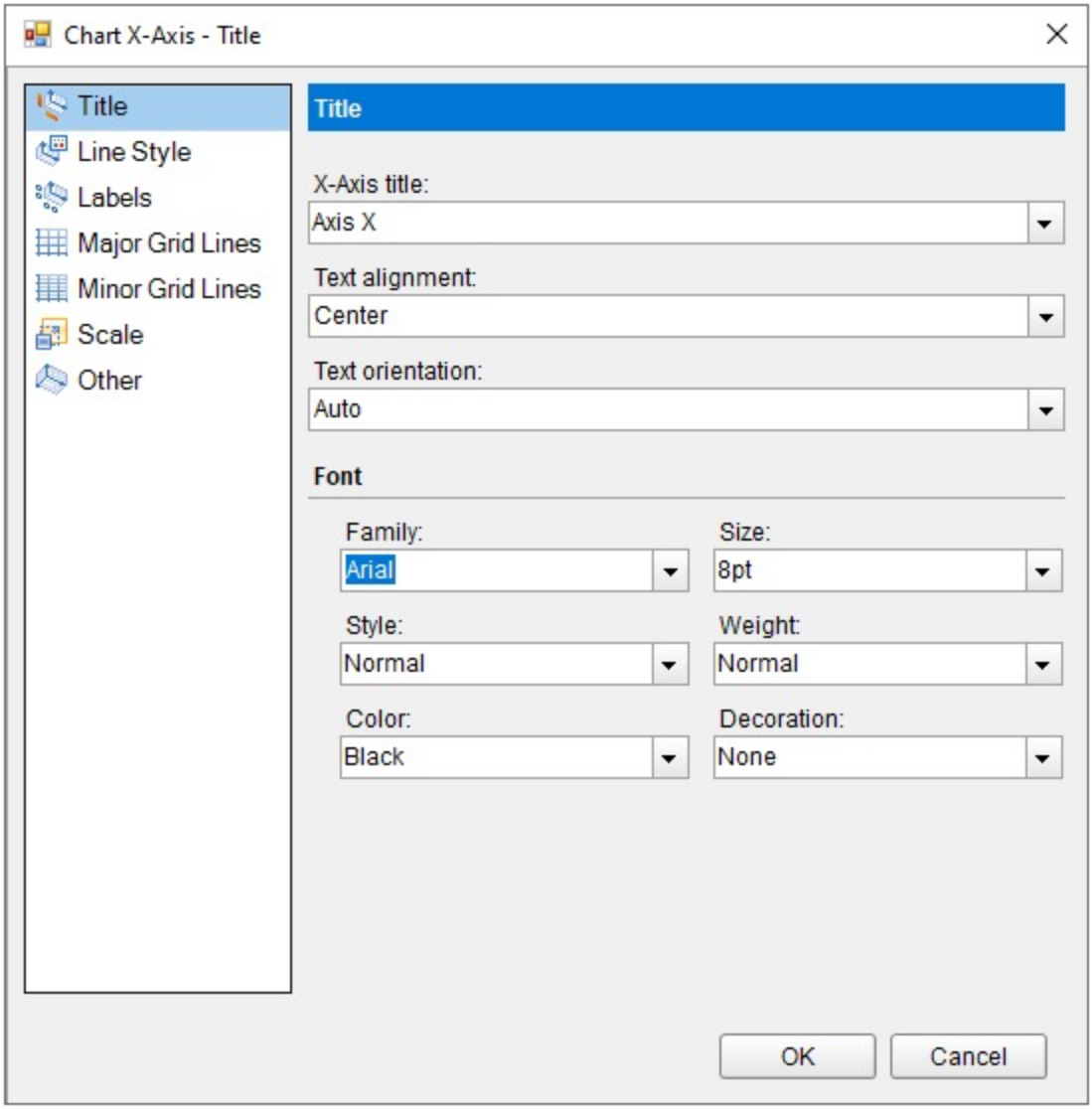
General

- Show chart legend and use smart settings.

Style

- Select style settings related to font, border and background color.

Chart X Axis Smart Panel



Title

Add the x-axis title, change text alignment, orientation and font.

Line Style

Change the styling properties related to the appearance of line.

Labels

Show x-axis labels.

Major Grid Lines

Show major gridlines.

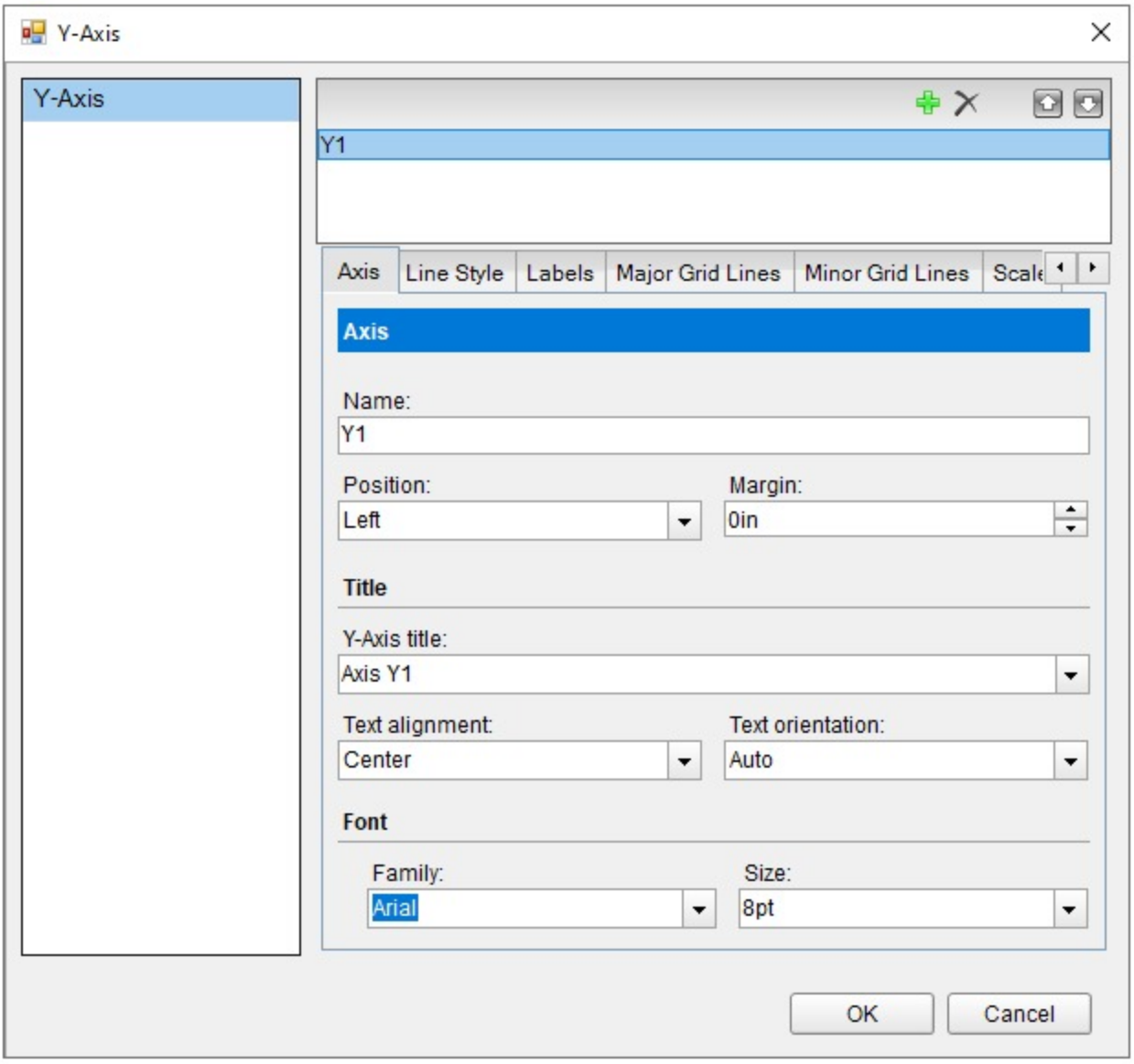
Minor Grid Lines

Show minor grid lines.

Scale

Specify the minimum and maximum scale.

Chart Y Axis Smart Panel



Axis

Add the y-axis title, change text alignment, orientation and font.

Line Style

Change the styling properties related to the appearance of line.

Labels

Show y-axis labels.

Major Grid Lines

Show major gridlines.

Minor Grid Lines

Show minor grid lines.

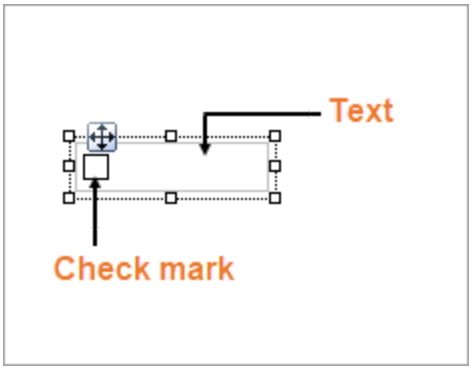
Scale

Specify the minimum and maximum scale.

CheckBox

You can use the CheckBox control to represent a Boolean value in a report. By default, the CheckBox control appears as a small box inside an empty textbox. If the **Checked** value is set to True, the small box appears with a check mark; if False, the box is empty. By default, the checkbox is empty.

Structure



Important Properties

Clicking the four-way arrow selects the control and reveals its properties in the Properties window.

Property	Description
Checked	Gets or sets a value indicating whether the check box is in the checked state. You can also set the **Checked** property of the check box in code or bind it to a Boolean database value.
Text	Gets or sets the printed caption of the check box.
CheckAlignment	Gets or sets the alignment of the check box text within the control drawing area.

You can double-click the CheckBox control to enter edit mode and enter text directly in the control, or you can enter text in the Properties window or you can assign data to display in code through the **Text** property.

In edit mode, using the toolbar you can format text in the CheckBox control using the toolbar or you modify properties in the Properties window. Formats apply to all of the text in the control. Text formatting changes in the Properties window immediately appear in the control, and changes made in the toolbar are immediately reflected in the Properties window.

CheckBox Dialog Properties

You can set the CheckBox properties in the CheckBox dialog. To open it, with the CheckBox selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the checkbox that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Text : Enter an expression or a static label, or choose a field expression from the drop-down list. You can access the expression editor by selecting **<Expression...>** in the list. The value of this expression or text is displayed in the report to the right of the checkbox.

Visibility

Initial visibility

Visible: The checkbox is visible when the report runs.

Hidden: The checkbox is hidden when the report runs.

Expression: Use an expression with a Boolean result to decide whether the checkbox is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this checkbox to specify a report control to use as a toggle to show or hide the checkbox. Then specify the TextBox control to display with a toggle image button. When the user clicks the TextBox control, the checkbox changes between visible and hidden.

Navigation

Choose one of the following actions to carry out when the checkbox is selected:

None: The default behavior is to do nothing when you select the checkbox at runtime.

Jump to Report: It allows you to jump to another report by selecting the checkbox at runtime. For a drill-through report, select the **Jump to Report** option and choose the name of a local report from the dropdown. To display a report from another folder, enter the relative path of the report. For the report from another server, provide the full path of the report. You can also use expressions to create drill-through links.

Parameters: Enter the **Name** and **Value** of each parameter you want to pass to the targeted report. You can also **Omit** the parameter by setting the value to **True**. The parameter names you supply must be the same in both reports.

Jump to bookmark: Select this option and provide a valid Bookmark ID to jump to the report control with that Bookmark ID.

Jump to URL: Select this option and provide a valid URL to create a hyperlink to a Web page, and then at runtime, select the checkbox to jump to that Web page.

Apply Parameters: Select the Name, Type, and Value of the parameter to set a parameter value through user action. See Actionable Parameters for more information.

Appearance

Border

Style: Select a style for the border.

Width: Enter a value in points to set the width of the border.

Color: Select a color to use for the border, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Background

Color: Select a color to use for the background of the checkbox.

Image: Enter an image to use for the background of the CheckBox.

Image Source: Select the location of the image.

MIME Type: Select the MIME type of the image.

Background repeat: Specify how the background image fills the space of the CheckBox.

Font

Family: Select a font family name or a theme font.

Size: Choose the size in points for the font or use a theme.

Style: Choose **Normal** or **Italic** or select a theme.

Weight: Choose an enumerated weight value or select a theme.

Color: Choose a color to use for the text.

Decoration: Choose from **None**, **Underline**, **Overline**, or **LineThrough**.

Alignment

Amount of space to leave around report control

Top padding: Set the top padding in points.

Left padding: Set the left padding in points.

Right padding: Set the right padding in points.

Bottom padding: Set the bottom padding in points.

Data Output

Element Name: Enter a name to be used in the XML output for this checkbox.

Output: Choose **Auto**, **Yes**, or **No** to decide whether to include this checkbox in the XML output. Auto exports the contents of the checkbox only when the value is not a constant.

Render as: Choose **Auto**, **Element**, or **Attribute** to decide whether to render checkboxes as Attributes or Elements in the exported XML file. Auto uses the report's setting for this property

Attribute example: <table1 checkbox3="Report created on: 7/26/2005 1:13:00 PM">

Element example: <table1> <checkbox3>Report created on: 7/26/2005 1:13:28 PM</checkbox3>

Keyboard Shortcuts

In the edit mode, you can use the following keyboard shortcuts.

Key Combination	Action
Enter	New line.
Alt + Enter	Saves modifications and exits edit mode.
Esc	Cancels modifications and exits edit mode.

In Visual Studio Integrated Designer, you can disable this feature in the **EditModeEntering Event (on-line documentation)** and **EditModeExit Event (on-line documentation)**.

Container

Using the Container report control in your Page/RDLX report gives you more possibilities for visual presentation of your report. The Container is a visual element that is used as a container for other report controls. It highlights a part of a report and how the report controls appear inside it. The Container control has no data associated with it.

For Container control to work, you must drag the items into the Container instead of drawing it around the existing items.

The Container control can be used in several ways to enhance your reports:

- Group the report controls visually by placing them within the Container control.
- Anchor report controls, which may otherwise be pushed down by a vertically expanding data region.
- Create visual effects by adding borders to the Container control.
- Display an image behind a group of report controls by adding a background image.

Important Properties

Clicking the four-way arrow selects the control and reveals its properties.

Property	Description
KeepTogether (RDLX)	Change to True to have ActiveReports attempt to keep all of the repeated data together on one page.

Container Dialog Properties

General

Name: Enter a name for the container that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Consume all white space during report rendering (RDLX): Select this checkbox to have all white space consumed at report rendering.

Page breaks (RDLX):

- **Insert a page break before this container (RDLX):** Insert a page break before the container.
- **Insert a page break after this container (RDLX):** Insert a page break after the container.

Appearance

Background

Color: Select a color to use for the background of the container or select the **<Expression...>** option to open the Expression Editor.

Image: Specify the background image of container using Expression or Data Visualizer, or directly open the image file on your system.

> If the Hatch and Gradient background styles are set using Data Visualizers, these are not displayed at design time.

Image Source: Select the location of the image for the background from External, Embedded, Database or select the **<Expression...>** option to open the Expression Editor.

MIME Type: Select the MIME type of the image.

Background repeat: Specify how the background image fills the space of the container.

Border

Style: Select a style for the border.

Width: Enter a value in points to set the width of the border or select the **<Expression...>** option to open the Expression Editor.

Color: Select a color to use for the border, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Rounded Rectangle: Specify the radius for each corner of the shape independently. Drag the handlers ◇ available at each corner of the shape to set the value of the radius at each corner.

> **Note:** To enable specific corners, select the checkbox available near each corner of the Container control.

Visibility

Initial visibility

- **Visible**: The container is visible when the report runs.
- **Hidden**: The container is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the container is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the container. The user can click the toggle item to show or hide this container.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this container. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Data Output

The Data Output page of the Container dialog allows you to control the following properties when you export to XML:

- **Element Name:** Enter a name to be used in the XML output for this container.
- **Output:** Choose **Auto**, **Yes**, **No**, or **Contents only** to decide whether to include the contents of this container in the XML output. Choosing **Auto** exports the contents of the container only when the value is not a constant.

Container Control in RDLX Dashboard Report

The Container is a visual element that is used as a container for other report controls. The properties and workings are the same as Page or RDLX Reports. The Container control in the RDLX Dashboard Report has the following additional property to manage the scroll behavior:

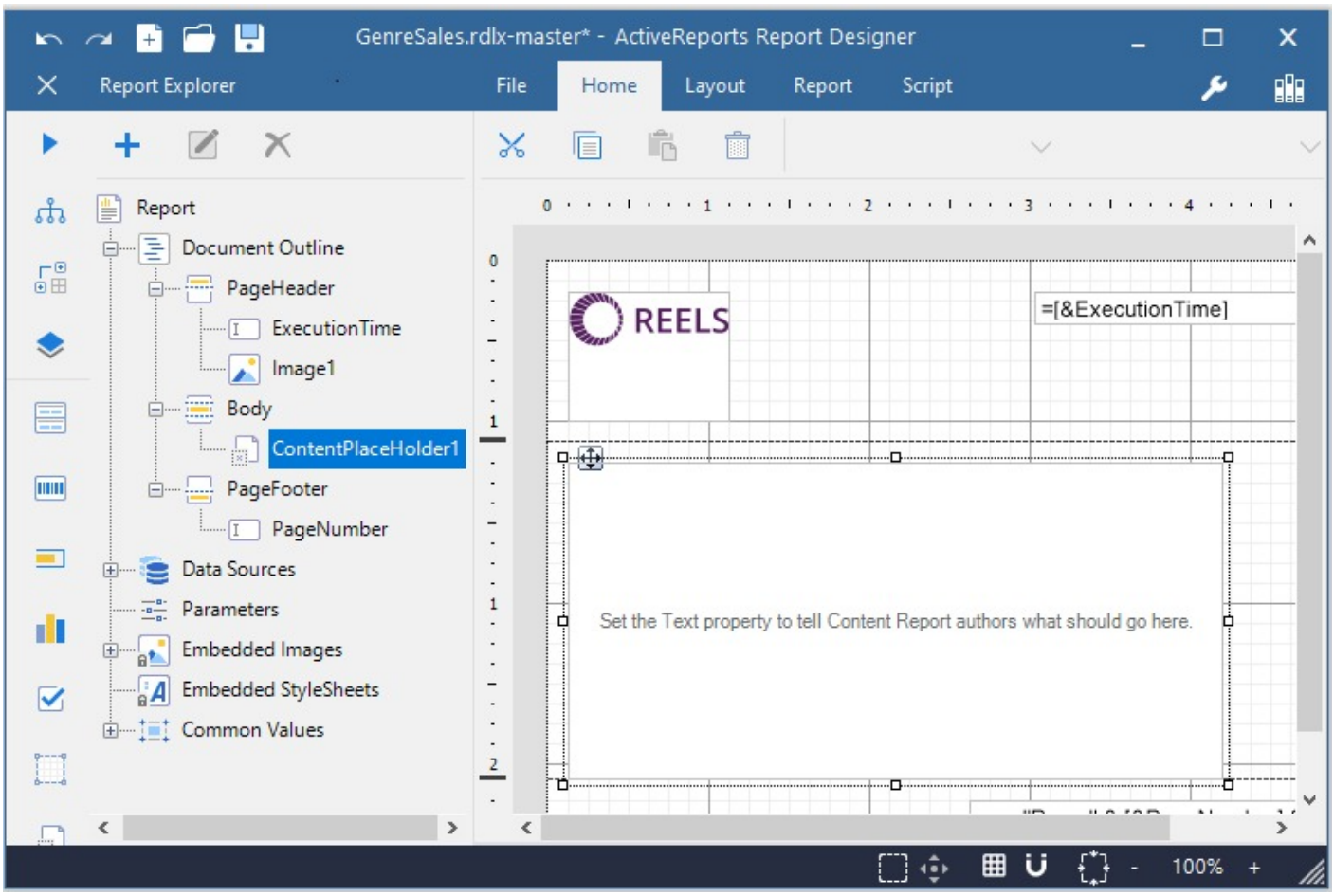
Overflow: By Setting the overflow property to Auto, Clip, Grow, or Scroll, you can specify the scroll behavior of the Container control in the RDLX Dashboard Report.

Choose one of the following options to specify the scroll behavior of the Container:

- **Auto:** The Container size remains the same, but you can scroll the data. By default, the value is set to 'Auto'.
- **Clip:** The Container size remains the same, and only the content that fits in the control is displayed.
- **Grow:** The Container size grows automatically to fit the data completely.
- **Scroll:** The Container size remains the same, and the scroll appears only if necessary.

ContentPlaceHolder (RDLX Master Report)

The **ControlPlaceHolder** control is used to show the edit area when a Master Report is available. See Master Report (RDLX Report) page for complete information on usage and implementation.



FormattedText

The FormattedText report control can perform mail merge operations, plus it displays richly formatted text in HTML. To format text in the FormattedText report control, enter HTML code into the **Html** property.

> **Note:** All text written in the **Html** property must be enclosed in the **<body></body>** tags.

The design-time editor displays the HTML text with the applied formatting, so you can view the text just as it will be displayed on the preview.

Important Properties

By clicking on the FormattedText control, you can set its properties in the Properties window.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
EncodeMailMergeFields	Select whether you want to encode mail merge fields or not.

Html	Enter HTML code to format text.
MailMergeFields	Add new mail merge fields to the FormattedText.
Size	Set the Width and Height of the FormattedText control.

FormattedText Dialog Properties

You can set the FormattedText properties in the FormattedText dialog. To open it, with the FormattedText selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the FormattedText that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Visibility

Initial visibility

- **Visible:** The FormattedText is visible when the report runs.
- **Hidden:** The FormattedText is hidden when the report runs.
- **Expression:** Use an expression with a Boolean result to decide whether the FormattedText is visible. True for hidden, false for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report control. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the FormattedText. The user can click the toggle item to show or hide this FormattedText.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this FormattedText. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Appearance

Background

Color: Select a color to use for the background of the FormattedText.

Image: Add an image to use for the background of the FormattedText.

Border

Style: Select a style for the border.

Width: Enter a value in points to set the width of the border.

Color: Select a color to use for the border, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Data Output

Element Name: Enter a name to be used in the XML output for this FormattedText report control.

Output: Choose **Auto**, **Yes**, **No**, or **Contents Only** to decide whether to include this FormattedText in the XML output. Choosing **Auto** exports the contents of the FormattedText report control.

Mail Merge

Check on the **Encode Mail Merge Fields** option to encode mail merge fields.

Click the plus sign button to add a new mail merge field to the FormattedText, and delete them using the X button.

Field: Enter a name for the field that is unique within the report. This is used in the Html property inside <%FieldName%> tags to display the field in the formatted text.

Value: Enter an expression to pull data into the control for mail merge operations.

Here is a very simple example of HTML code that you can use to add mail merge fields to formatted text. This example assumes that you have added two mail merge fields named **Field1** and **Field2**.

Paste this code in the Html property of the FormattedText control.

```
<body><p>This is <%Field1/%> and this is <%Field2/%>.</p></body>
```

Supported HTML Tags and Attributes

The FormattedText control supports almost all HTML tags and attributes from the XHTML 1.1 specification (see https://www.w3.org/TR/xhtml11/ for details) with some extensions to partially support the HTML5 specification (see https://www.w3.org/TR/2011/WD-html5-20110405/ for details). Additionally, the FormattedText control supports many (but not all) CSS styles.

For example, the FormattedText control supports these HTML tags and attributes. If you use valid HTML tags that are not in the list below, ActiveReports ignores them.

⚠ **Caution**: To enter **&** in the HTML property, you need to use **&**.

Supported HTML Tags and Attributes

HTML Tags	Attributes	Description
<%MergeFieldName%>		**Inserts a mail merge field.**
<!-- -- >		Defines a comment
<!DOCTYPE>		Defines the document type
<a>		Defines an anchor
<abbr>		Defines an abbreviation
<acronym>		Defines an acronym
<address>		Defines an address element
<b>		Defines bold text
<base />		Defines a base URL for all the links in a page
<bdo>		Defines the direction of text display
<big>		Defines big text
<blockquote>		Defines a long quotation
<body>		**Defines the body element (Required)**
 		Inserts a single line break
<caption>		Defines a table caption
		Defines centered text
<cite>		Defines a citation
<code>		Defines computer code text
<col>		Defines attributes for table columns
<dd>		Defines a definition description
<del>		Defines deleted text
<dir>		Defines a directory list
	style	Defines a section in a document
<dfn>		Defines a definition term
<dl>		Defines a definition list
<dt>		Defines a definition term
		Defines emphasized text
<h1> to <h6>	align	Defines header 1 to header 6

<head>		Defines information about the document
<hr />		Defines a horizontal rule
<html>		Defines an html document
<i>		Defines italic text

<img />	align
	src
	height
	width
		Defines an image. Use **src** attribute to define path of the image, for example:
		`<img src="https://.../images/image.png"/>`
		To refer an image embedded within the report, specify the image name:
		`<img src="image_name"/>`

<ins>		Defines inserted text
<kbd>		Defines keyboard text
		Defines a list item
<link>	href	Defines a link
<map>		Defines an image map
<menu>		Defines a menu list
	type align	Defines an ordered list
<p>		Defines a paragraph
<pre>		Defines preformatted text
<q>		Defines a short quotation
<s>		Defines strikethrough text
<samp>		Defines sample computer code
<small>		Defines small text
		Defines a section in a document
<strike>		Defines strikethrough text
		Defines strong text
<style>		Defines a style definition
<sub>		Defines subscripted text
<sup>		Defines superscripted text
<table>	align bgcolor border bordercolor cellpadding cellspacing	Defines a table

	width	
<tbody>		Defines a table body.
<td>	align bgcolor colspan height nowrap rowspan valign width	Defines a table cell
<tfoot>		Defines a table footer
<th>	abbr align axis bgcolor colspan headers height nowrap rowspan scope sorted valign width	Defines a table header
<thead>		Defines a table header. It is used along with <tbody> and <tfoot>.
<tr>	align bgcolor bordercolor height valign	Defines a table row
<tt>		Defines teletype text
<u>		Defines underlined text
		Defines an unordered list

Image

The **Image** report control displays an image that you embed in the report, add to the project, store in a database, or access through a URL. You can choose the **Image Source** in the Properties window after you place the Image report control on the report.

The supported image formats are Base64 string, Byte[], BMP, JPG, JPEG, JPE, GIF, PNG, EMF, WMF, and SVG.

Important Properties

By clicking on the Image control, you can set its properties in the Properties window.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
Action	From Navigation Dialog choose one of the actions to carry out at runtime.
MIMEType	Choose MIME Type from the dropdown.
Size	Enter the Width and Height of the image.
Sizing	Specify the way an image should be sized within the image report control.
Source	Select the source of the image from External, Embedded, or Database.
Value	Depending on the Source chosen specify the path of the image.

Image Dialog Properties

You can set the Image properties in the Image dialog. To open it, with the Image selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the image that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Image Value: Enter the name of the image to display. Depending on the Image Source chosen below, you can give a path to the image, select an image to embed, or pull images from a database. This property also allows you to choose the **<Data Visualizer...>** option to launch a dialog that will let you build a data visualization expression.

Image Source: Select whether the image comes from a source that is **External**, **Embedded**, or **Database**.

MIME Type: Select the MIME type of the image chosen.

Visibility

Initial visibility

- **Visible**: The image is visible when the report runs.
- **Hidden**: The image is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the image is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report control. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the image. The user can click the toggle item to show or hide this image.

Navigation

Action

Select one of the following actions to perform when a user clicks on this image.

None: The default behavior is to do nothing when a user clicks the image at run time.

Jump to report: For drill-through reporting, select this option and provide the name of a local report, the relative path of a report in another folder, or the full path of a report on another server.

Parameters: Supply parameters to the targeted report by entering the Name of each parameter, the Value to send to the targeted report, or whether to Omit the parameter. Note that the parameter names you supply must exactly match the parameters in the target report. You can remove or change the order of parameters using the X and arrow buttons.

Jump to bookmark: Select this option and provide a valid Bookmark ID to allow the user to jump to the report control with that Bookmark ID.

Jump to URL: Select this option and provide a valid URL to create a hyperlink to a Web page.

Apply Parameters: Select the Name, the Type, and the Value of the parameter to set a parameter value through user action. See Actionable Parameters for more information.

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this image. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

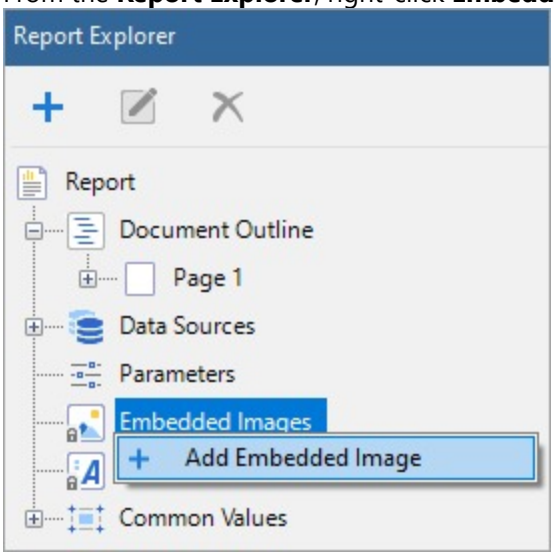
Image Features

Embedded Images

The benefit of using an embedded image is that there is no separate image file to locate or to keep track of when you move the report between projects. The drawback is that the larger the file size of the image you embed, the more inflated your report file size becomes.

To embed an image in your report:

1. From the **Report Explorer**, right-click **Embedded Images**.



2. Select **Add Embedded Image** and navigate to an image file from your local files.
3. Now that the image is embedded in the report and appears under **Embedded Images**, drag-drop the image to the design area.
   The **MIMEType**, **Source**, and **Value properties** are filled in automatically and the image file's data is stored in the report definition.

Data Visualizer Images

You can use a data visualizer to display data in small graphs that are easy to comprehend. To add a data visualizer image to your report:

1. Select the Image control and in the Properties window, drop down the **Value** property and select **<Data Visualizer...>**.
2. In the Data Visualizers dialog that appears, select the **Visualizer Type** that you want to use - Icon Set, Range Bar, Data Bar, Gradient, or Hatch.
3. Use expressions related to your data to set other values in the dialog.

SVG Images

There are several cases where you can use an SVG image:

- as a background image for Chart, Container, CheckBox, FormattedText, List, Shape, Table, Tablix, TextBox.
- as a report's embedded image.
- as a report's theme, where you can add an image in the Images tab of the Theme Editor.

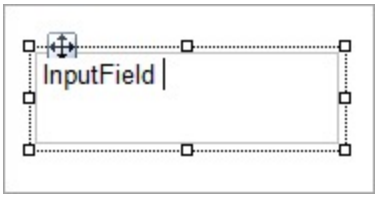
Image files with the mime-type **image/svg+xml** are allowed, which you can set in the **MimeType** property of the Image control.

InputField

The **InputField** report control in Page and RDLX reports provides support for editable fields in an exported PDF report where the InputField's value can be modified. An exported editable PDF file is an AcroForm. See Creating AcroForm fields for details.

The **InputField** report control can be one of the two types – **Text** and **CheckBox**, which you can choose in the **InputType** property. Each selected type has its own set of properties. In case of the Text type, the InputField control gets the set of properties of the TextBox control. If the CheckBox type is selected, then the control inherits the set of properties of the CheckBox control.

Note: The InputField control is part of the **Professional Edition**. With the Standard license, the InputField control is not displayed in an exported file.



Important Properties

Property	Description
InputType	Specifies the form field type - Text or CheckBox, in the resulted PDF file.
ReadOnly	Prevents the user from changing the entered text content in the resulted PDF file.
FieldName	Specifies a unique name of the field in the resulted PDF file.
Required	Forces the user to fill in the selected field of the resulted PDF file. If a user attempts to submit the form where the required field is blank, the error message appears and the empty required field is highlighted.
Checked (for CheckBox)	Gets or sets a value indicating whether the check box is in the checked state.
CheckStyle (for CheckBox)	Gets or sets the style of the check symbol inside the CheckBox. The available options are Check, Circle, Cross, Diamond, Square, and Star.
MaxLength (for Text)	Specifies the maximum length of the entered text in the resulted PDF file. When set to null, the text is not restricted to any specified length.
MultiLine (for Text)	Gets or sets a value indicating whether to allow text to break to multiple lines within the control in the resulted PDF file.
Password (for Text)	Displays the user-entered text as a series of asterisks (*).
SpellCheck (for Text)	Indicates whether to spell check the text during its input or not in the resulted PDF file.
TabIndex	Sets the tab order of editable fields in the resulted PDF file. A field with the lowest TabIndex value is selected first.

Limitations

- (Preview limitation) Border drawing is displayed inside the control area.
- (Preview limitation) A checkmark inside the InputField CheckBox is shown only if the control's size is not smaller than the checkmark size in the **CheckSize** property. A checkmark cannot be displayed partially.
- (Preview limitation) Some properties are not supported at preview, like Multiline (False), Password, Required.
- With the PDF export filter, the InputField control is exported as a static, non-editable control.
- (PDF limitation) Limited set of enum values for the BorderStyle property are available - None, Solid, Dashed, Inset.
- (PDF limitation) Padding, VerticalAlign, Justify settings are not available.
- (PDF limitation) Properties related to the Text/Font settings are not available for the InputType control as Checkbox.
- (PDF limitation) Border properties can only be set for all sides at once (the option for each side is not available).

Line

The Line control allows you to draw vertical, horizontal or diagonal lines that visually separate or highlight areas within a section on a report. You can also use the Line report control to visually separate data regions in a report layout.

Important Properties

Clicking the four-way arrow selects the control and reveals its properties.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
LineColor	Gets or sets the color of the line.
LineStyle	Gets or sets the pen style used to draw the line. The line styles include Transparent, Solid, Dash, Dot, DashDot, DashDotDot, or Double.
LineWidth	Gets or sets the pen width of the line in points.
Visibility	Indicates whether the line is visible or hidden.
LayerName	Gets or sets the name of the containing layer.
EndPoint	Set the vertical and horizontal end points of the line.
Location	Gets or sets the position of the top-left corner of the report item in relation to its container.

Line Dialog Properties

You can set the Line properties in the Line dialog. To open it, with the Line selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the line that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Visibility

The Visibility page of the Line dialog allows you to control the following items:

Initial visibility

- **Visible**: The line is visible when the report runs.
- **Hidden**: The line is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the line is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the line. The user can click the toggle item to show or hide this line.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

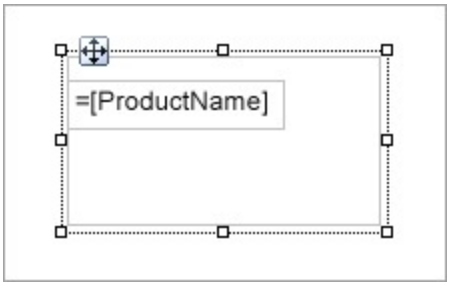
Bookmark ID: Enter an expression to use as a locator for this line. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

List

The **List** data region repeats any report controls it contains for every record in the dataset. Unlike other data regions, the **List** is free-form, so you can arrange report controls in any configuration you like.

Grouping in the list is done on the details group.

The List data region behavior is the same as the one of Table or Tablix that has one Detail cell with the Container control added to it.



List Dialog Properties

You can set the List properties in the List dialog. To open it, with the List selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the list that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), back slash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Dataset name: Select a dataset to associate with the list. The drop-down list is populated with all of the datasets in the report's dataset collection.

Has own page numbering: Select to indicate whether this List is in its own section with regards to pagination.

Consume all white space during report rendering (RDLX): Select to have all white space consumed at report rendering.

Page Breaks (RDLX): Select any of the following options to apply to each instance of the list.

- Insert a page break before this list
- Insert a page break after this list
- Fit list on a single page if possible

NewPage (RDLX): Indicates on which page the content to start after the page break.

- **Next**: A default value that makes a new group start from the immediate next page of the report.
- **Odd**: A new group starts from the next odd page of the report.
- **Even**: A new group starts from the next even page of the report.

Detail Grouping

Detail grouping is useful when you do not want to repeat values within the details. When a detail grouping is set, the value repeats for each distinct result of the grouping expression instead of for each row of data. For example, if you use the Customers table to create a list of countries without setting the details grouping, each country is listed as many times as there are customers in that country. If you set the details grouping to `=Fields!Country.Value` each country is listed only once.

> **Note**: If the detail grouping expression you use results in a value that is distinct for every row of data, a customer number, for example, you will see no difference in the results.

The Detail Grouping page of the List dialog has the following tabs.

General

Name: Enter a name for the group that is unique within the report. This property cannot be set until after a **Group on** expression is supplied. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Group on: Enter an expression to use for grouping the data.

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Filters data for which the value on the left is equal to the value on the right.
- **Like** Filters data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Filters data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Filters data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Filters data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Filters data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Filters data for which the value on the left is less than or equal to the value on the right.
- **TopN** Filters data items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Filters items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Filters items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Filters items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Filters items from the value on the left which are in the array of values specified on the right.
  Selecting this operator enables the Values list at the bottom.
- **Between** Filters items from the value on the left which fall between the pair of values you specify on the

right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

In the **Expression** box, enter an expression by which to sort the data in the group, and under **Direction**, select **Ascending** or **Descending** for the selected sort expression.

Data Output

Element Name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose **Yes** or **No** to decide whether to include this group in the XML output.

Layout

BreakLocation: Select from these options to decide where to insert a page break in relation to the group.

- **None**: Inserts no page break.
- **Start**: Inserts a page break before the group.
- **End**: Inserts a page break after the group.
- **StartAndEnd**: Inserts a page break before and after the group.
- **Between**: Inserts a page break between groups (at the end of a current group and the beginning of the next group).

NewPage:

- **Next:** A default value that makes a new group start from the immediate next page of the report.
- **Odd:** A new group starts from the next odd page of the report.
- **Even:** A new group starts from the next even page of the report.

See Page Breaks in Data Regions topic for more detail.

Has own page numbering: Used in conjunction with the "Page Number in Section" and "Total Pages in Section" properties, tells the report that the group constitutes a new page numbering section.

Visibility

Initial visibility

- **Visible**: The list is visible when the report runs.
- **Hidden**: The list is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the list is visible. True for hidden, false for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the list. The user can click the toggle item to show or hide this list.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this list. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Filters data for which the value on the left is equal to the value on the right.
- **Like** Filters data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Filters data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Filters data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Filters data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Filters data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Filters data for which the value on the left is less than or equal to the value on the right.
- **TopN** Filters data items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Filters items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Filters items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Filters items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Filters items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Filters items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

Expression: Enter an expression by which to sort the data in the group.

Direction: Select **Ascending** or **Descending** for the selected sort expression.

Data Output

Element Name: Enter a name to be used in the XML output for this list.

Output: Choose **Auto**, **Yes**, **No**, or **Contents only** to decide whether to include this List in the XML output. Choosing **Auto** exports the contents of the list.

Instance element name: Enter a name to be used in the XML output for the data element for instances of the list. This name is ignored if you have specified a detail grouping.

Instance element output: Choose Yes or No to decide whether to include the instances of the list in the XML output. This is ignored if you have specified a detail grouping.

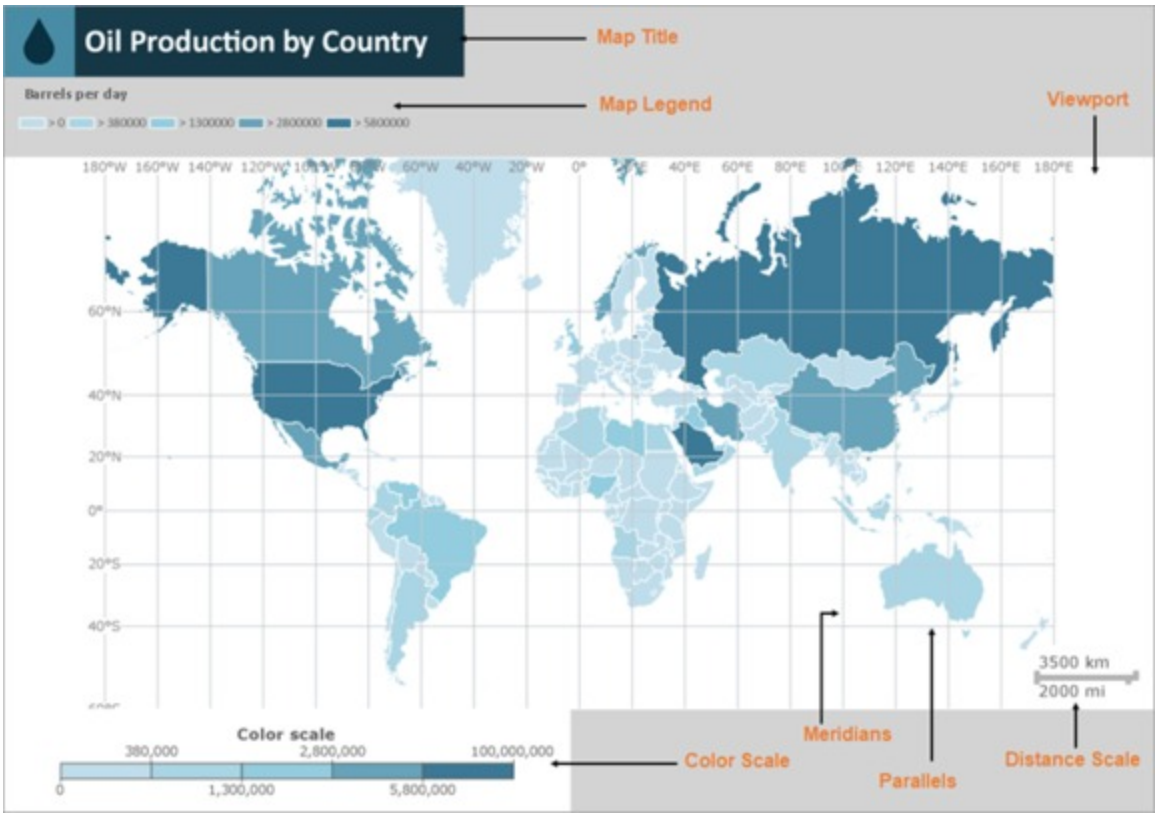
Map

The Map data region is a professional edition feature that shows your business data against a geographical background. You can create different types of map, depending on the type of information you want to communicate in your report.

Maps consist of a title, a viewport for specifying the center point and scale, legend for interpreting data, distance scale to represent real world distance, and color scale for representing the meaning of colors used in maps. Lets us learn these in detail in the next section.

> **Note**:
>
> - Map control is not supported by WebDesigner for now.
> - MESCIUS inc. does not provide access to Tile Providers; please ensure that you have a valid license before using it.

Structure



Title

Map Title describes the theme or subject of the map. The purpose of map title is to tell the viewer of what he is looking at. You can add multiple titles to the Map using the **MapTitleDesigner Collection Editor**. For more information, see Add a Title to the Map tutorial.

Viewport

Viewport refers to the area on the map where data is displayed against a geographical background. It specifies the coordinates, projection system, parallels and meridians, center point, and scale of the map. In other words, it is a map element that actually displays geographical data and occupies most area of the map control depending on the location and dock position of other map elements. By default, color scale and distance scale appear inside the viewport, and the map legend appears outside the viewport. For information on customizing the viewport, see Modify the Appearance of the Viewport tutorial.

Map Viewport Dialog Properties

The **Map Viewport** dialog lets you set properties with the following pages.

General

Coordinate system: Specify the coordinate system of the viewport. Select from Planar, Geographic, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Projection: Specify the projection of the map. Tile layers must use the Mercator projection.

Minimum X: Specify the minimum X coordinate of the map in degrees.

Maximum X: Specify the maximum X coordinate of the map in degrees.

Minimum Y: Specify the minimum Y coordinate of the map in degrees.

Maximum Y: Specify the maximum Y coordinate of the map in degrees.

Projection Center X: Specify the X coordinate of the projection center in degrees.

Projection Center Y: Specify the Y coordinate of the projection center in degrees.

Minimum zoom: Specify the minimum zoom value.

Maximum zoom: Specify the maximum zoom value.

Map resolution: Enables the viewport to simplify vector data for polygon and line layers.

Show grid lines below the map: Specify whether to show the grid lines above or below the content of the map.

Meridians

Hide meridians: Specify whether to hide meridians.

Interval: Specify the spacing between the grid lines in degrees.

Line

- **Style:** Choose from None, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, WindowInset, Outset, or select the <**Expression...**> option to open the Expression Editor and create an expression.
- **Width:** Specify the width of the line.
- **Color:** Select a Web or custom color for the line.

Show labels: Specify whether to show labels for meridians on the map.

Format: Specify the format string to display numeric labels.

Position: Specify the position of the meridians on the map.

Font

- **Family:** Choose the font family name.
- **Size:** Choose the size in points for the font.
- **Style:** Choose Normal or Italic.
- **Weight:** Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.
- **Color:** Select a Web or custom color for the font.
- **Decoration:** Choose from None, Underline, Overline, LineThrough, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Parallels

Hide parallels: Specify whether to hide parallels.

Interval: Specify the spacing between the grid lines in degrees.

Line

- **Style:** Choose from None, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, WindowInset, Outset, or select the <**Expression...**> option to open the Expression Editor and create an expression.
- **Width:** Specify the width of the line.
- **Color:** Select a Web or custom color for the line.

Show labels: Specify whether to show labels for parallels on the map.

Format: Specify the format string to display numeric labels.

Position: Specify the position of the parallels on the map.

Font

- **Family:** Choose the font family name.
- **Size:** Choose the size in points for the font.
- **Style:** Choose Normal or Italic.
- **Weight:** Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.
- **Color:** Select a Web or custom color for the font.
- **Decoration:** Choose from None, Underline, Overline, LineThrough, or select the <**Expression...**> option to open the Expression Editor and create an expression.

View

Center and zoom: Specify how the map viewport zooms and centers during the report processing.

- Custom
- Center map to show a map element
- Center map to show a map layer
- Center map to show all map elements

View Center X: Specify the X coordinate of the current view center.

View Center Y: Specify the Y coordinate of the current view center.

Zoom level: Specify the zoom level of the map view.

Appearance

Border

- **Style:** Choose an enumerated style for the border.
- **Width:** Set a width value in points between 0.25pt and 20pt.
- **Color:** Select a Web or custom color for the border.

Background

- **Color:** Select a color to use for the background of the Viewport.
- **Gradient:** Select a color to use for the border, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
- **Gradient End Color:** Select a color to use for the end color of the background gradient.
- **Pattern:** Select the hatching pattern of a report control.

Shadow offset: Specify the size of the shadow. Shadow offsets are drawn to the right and below an element.

Legend

A legend on a map provides valuable information to users for interpreting the map data visualization rules such as color, size, and marker type differences for map elements on a layer. By default, a single Legend item already exists in the legends collection which can be used by all layers to display items. You can also create additional legends to use them individually with layers that have associated rules to display items in the legend. Legends are added in the **LegendDesigner Collection Editor**. For more information, see Add a Legend to the Map tutorial.

Distance Scale

A distance scale helps a user to understand the scale of the map. Distance on a map is not the same as the actual real-world distance, so a distance scale shows that a certain distance on the map equals a certain distance in a real-world. In distance scale, the distance is displayed in both miles and kilometers. The scale range and values are automatically calculated using the viewport boundaries, projection type, and zoom level. For more information, see Set the Distance Scale on a Map tutorial.

Map Distance Scale Dialog Properties

The Map Distance Scale dialog lets you set properties with the following pages.

General

Location

- **Position:** Specify the docking position of the distance scale panel. Choose from TopCenter, TopLeft, TopRight, LeftTop, LeftCenter, LeftBottom, RightTop, RightCenter, RightBottom, BottomRight, or select the <**Expression...**> option to open the Expression Editor and create an expression.
- **Show distance scale outside the viewport:** Specify whether the panel is docked inside or outside of the map viewport.

Scale

- **Color:** Select the fill color for the distance scale bar.
- **Border color:** Select the border color for the distance scale bar.

Appearance

Border

- **Style:** Choose from None, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, WindowInset, or Outset.
- **Width:** Choose the width of the border line.
- **Color:** Select a color for the border.

Background

- **Color:** Select a color to use for the background of the distance scale.
- **Gradient:** Select a color to use for the border, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
- **Gradient End Color:** Select a color to use for the end color of the background gradient.
- **Pattern:** Select the hatching pattern of the distance scale panel from the list of patterns, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Shadow offset: Specify the size of the shadow of the distance scale panel. Shadow offsets are drawn to the right and below an element.

Font

Family: Choose the font family name.

Size: Choose the size in points for the font.

Style: Choose Normal or Italic, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline and LineThrough, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Visibility

Initial visibility

- **Visible**: The distance scale is visible when the report runs.
- **Hidden**: The distance scale is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the distance scale is visible. True for hidden, false for visible.

Navigation

Action

Select one of the following actions to perform when a user clicks on the distance scale element.

- **None:** The default behavior is to do nothing when a user clicks the distance scale element at run time.
- **Jump to URL:** Select this option and provide a valid URL to create a hyperlink to a Web page.
- **Jump to bookmark:** Select this option and provide a valid Bookmark ID to allow the user to jump to the report control with that Bookmark ID.
- **Jump to report:** For drill-through reporting, select this option and provide the name of a local report, the relative path of a report in another folder, or the full path of a report on another server.
- **Apply Parameters**: Select the Name, the Type, and the Value of the parameter to set a parameter value through user action. See Actionable Parameters for more information.

Color Scale

A color scale helps a user to understand the range of colors that are used for data visualization on a layer. A map has just one color scale and multiple layers can provide data for it. For information, see Set the Color Scale on a Map tutorial.

Map Color Scale Dialog Properties

The Map Color Scale dialog lets you set properties with the following pages.

General

Location

- **Position:** Specify the docking position of the color scale panel. Choose from TopCenter, TopLeft, TopRight, LeftTop, LeftCenter, LeftBottom, RightTop, RightCenter, RightBottom, BottomRight, or select the <**Expression...**> option to open the Expression Editor and create an expression.
- **Show color scale outside the viewport:** Specify whether the panel is docked inside or outside of the map viewport.

Color bar

- **Border color:** Specify the outline color for color scale divisions.
- **Range gap color:** Specify color to fill color divisions for undefined range values.

Labels

Display: Specify whether to display color scale labels on the color scale panel. Select from Auto, ShowMiddleValue, ShowBorderValue, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Hide end labels: Specify whether to display first and last labels on the color scale panel.

Format: Specify the format string to display numeric labels.

Placement: Speciy the position of the color scale labels on the color scale panel. Select from Alternate, Top, Bottom, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Interval: Specify the frequency of the labels on the color scale panel. A value of 0 means no labels are displayed.

Tick mark length: Specify the length of the tick marks on the color scale panel.

Title

Text: Specify the text of the color scale panel.

Font

- **Family:** Choose the font family name.
- **Size:** Choose the size in points for the font.
- **Style:** Choose Normal, Italic or select the <**Expression...**> option to open the Expression Editor and create an expression.
- **Weight:** Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.
- **Color:** Select a Web or custom color for the font.
- **Decoration:** Choose from None, Underline, Overline, LineThrough, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Appearance

Border

- **Style:** Choose from None, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, WindowInset, or Outset.
- **Width:** Specify the width of the border.
- **Color:** Specify a color for the border.

Background

- **Color:** Select a color to use for the background of the distance scale.
- **Gradient:** Specify whether and how to use color gradients in the color scale background. Select from None, LeftRight, TopBottom, Center, DiagonalLeft, DiagonalRight, HorizontalCenter, VerticalCenter, or select the <**Expression...**> option to open the Expression Editor and create an expression.
- **Gradient End Color:** Select a color to use for the end color of the background gradient.
- **Pattern:** Select the hatching pattern of the color scale panel from the list of patterns, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Shadow offset: Specify the size of the shadow of the color scale panel. Shadow offsets are drawn to the right and below an element.

Font

Family: Choose the font family name.

Size: Choose the size in points for the font.

Style: Choose Normal, Italic or select the <**Expression...**> option to open the Expression Editor and create an expression.

Weight: Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder.

Color: Select a Web or custom color for the font.

Decoration: Choose from None, Underline, Overline, LineThrough, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Visibility

Initial visibility

- **Visible**: The color scale is visible when the report runs.
- **Hidden**: The color scale is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the color scale is visible. True for hidden, false for visible.

Navigation

Action

Select one of the following actions to perform when a user clicks on the color scale element.

- **None:** The default behavior is to do nothing when a user clicks the color scale element at run time.
- **Jump to URL:** Select this option and provide a valid URL to create a hyperlink to a Web page.
- **Jump to bookmark:** Select this option and provide a valid Bookmark ID to allow the user to jump to the report control with that Bookmark ID.
- **Jump to report:** For drill-through reporting, select this option and provide the name of a local report, the relative path of a report in another folder, or the full path of a report on another server.
- **Apply Parameters**: Select the Name, the Type, and the Value of the parameter to set a parameter value through user action. See Actionable Parameters for more information.

Important Properties

By clicking on the Map control, you can set its properties in the Properties window.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
AntiAliasing	Select the anti-aliasing type from All, None, Text, or Graphic.
BorderColor	Choose a color for the border from the Color Picker.
BorderStyle	Select a style for the border.
BorderWidth	Enter a value or expression to set the width of the border.
Calander	Select the type of calendar you want to use.
ColorScale	Specify the range of colors that are used for data visualization on a layer. See the **Color Scale** section for details.
DistanceScale	Specify a certain distance on the map that equals a certain distance in the real-world. See the **Distance Scale** section for details.
Language	Choose the language of the map.
Layers	Specify the layers (Polygon layer, Point layer, Line layer, Tile layer) used to display data on the map. See the **Map Layers** section for details.
Legends	Specify the map data visualization rules, such as color, size, and marker type differences, for map elements on a layer. See the **Legends** section for details.
MaximumSpatialElementCount	Specify the maximum number of spatial elements that are allowed in the Map. By default, the value is set to 20000.
MiximumTotalPointCount	Specify the maximum total number of map points in all spatial elements

	that are allowed in the Map.
NumeralLanguage	Choose the numeral language.
NumeralVariant	Choose numeral variant from 1-7.
PageBreak	Set the Page Break.
ShadowIntensity	Entre a value or expression to specify the intensity of the shadow throughout the map. By default, the value is set to 25.
Size	Set the Width and Height of the map.
TextAntiAliasingQuality	Select the text anti- aliasing quality to High, Normal, or SystemDeafult.
TileLanguage	Choose the primary language of the map tiles.
Titles	Specify the theme or subject of the map. See the **Title** section for details.
ViewPort	Specify the coordinates, projection system, parallels and meridians, center point, and scale of the map. See the **Viewport** section for details.

Map Dialog Properties

You can set the Map properties in the Map dialog. To open it, with the Map selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the map that is unique within the report. This name is called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), back slash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Antialiasing: Select the smoothing mode to apply to all map control elements. Choose All, None, Text, Graphic, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Antialiasing quality: Select the quality for antialiasing. Choose High, Normal, SystemDefault, or select the <**Expression...**> option to open the Expression Editor and create an expression.

Visibility

Initial visibility

- **Visible**: The map is visible when the report runs.
- **Hidden**: The map is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the map is visible. True for hidden, false for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the map. The user can click the toggle item to show or hide this map.

Appearance

Border

- **Style:** Choose from None, Dotted, Dashed, Solid, Double, Groove, Ridge, Inset, WindowInset, or Outset.
- **Width:** Specify the width of the border.
- **Color:** Select a Web or custom color for the font.

Background

- **Color:** Select a color to use for the background of the map.
- **Gradient:** Specify whether and how to use color gradients in the color scale background. Select from None, LeftRight, TopBottom, Center, DiagonalLeft, DiagonalRight, HorizontalCenter, VerticalCenter, or select the <**Expression...**> option to open the Expression Editor and create an expression.
- **Gradient End Color:** Select a color to use for the end color of the background gradient.
- **Pattern:** Select the hatching pattern of a report control.

Data Output

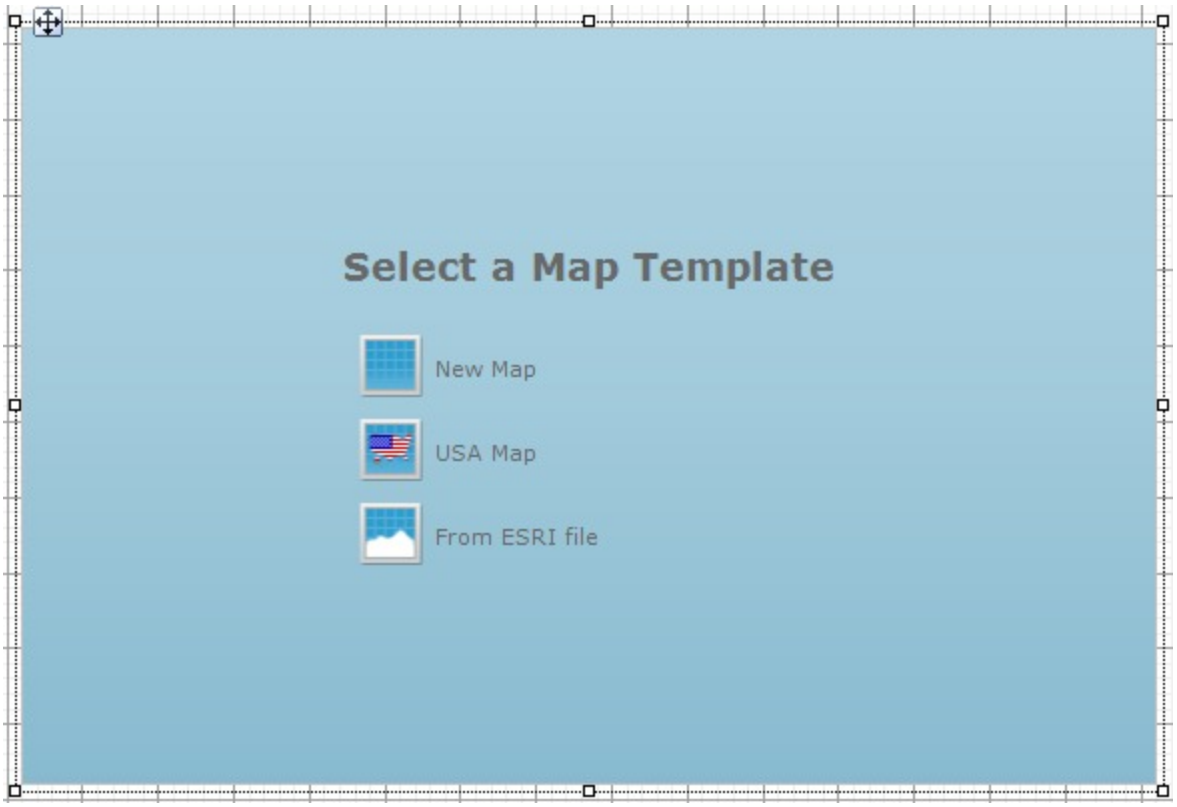
Element name: Enter a name to be used in the XML output for this map.

Output: Choose **Auto**, **Yes**, or **No** to decide whether to include this map in the XML output. Choosing **Auto** exports the contents of the map.

Add a Map to a Report

1. From the Visual Studio toolbox, drag a **Map** control onto the design surface.

2. In the **Select a Map Template** wizard that appears, select a map template from the following options:



- **Empty map:** An empty map without any predefined data.
- **USA map:** A map with the predefined polygon layer that contains the embedded spatial data with the USA map.
- **From ESRI file:** Select from your local .shp file, the shapefile spatial data format that complies with the Environmental Systems Research Institute, Inc. (ESRI). An ESRI Shapefile is a collection of files, where a .shp file defines the geographical or geometrical shapes and the .dbf file provides attributes for the shapes in the .shp file. To successfully add spatial data using this option, both files (.shp and .dbf) must be copied to the same folder. ESRI files are available on public domain data sources on the Web, including government and university sites. For more information, checkout the TIGER/Line Geodatabases article.

For more information on adding a map, see tutorial on Map Data Region in Reports.

Add Data to a Map

The Map data region uses two types of data - **spatial data** and **analytical data**.

Spatial Data

Spatial data is a set of coordinates that defines a map element. Each map layer must have spatial data of one of the following types - a polygon, a line, or a point.

Spatial data can be either embedded in a map or can be linked to a map layer. The only difference between the two is that while having the spatial data embedded in a map, there is no separate file to locate or to keep track of when you move the report between projects or machines.

- **Embedded Spatial Data**:

  Embedded spatial data can refer to the following:

    - **Embedded ESRI shapefile or dataset**: When you use the **Embedded** option in Map Layer Data Properties dialog or the **From ESRI file** option in the Map Wizard to import spatial data from an ESRI shapefile, the ESRI shapefile gets embedded in the Map. Similarly, if a report dataset is been used to provide spatial data to a map layer, you always have an option of embedding that spatial data in the map using the **Embed Spatial Data** option that appears in the layers pane. See, Layers to learn embedding spatial data to a map.
    - **Custom data**: When you add an empty layer to a map and enter spatial data manually in the LayerDesigner Collection Editor, the data entered gets embedded to the map automatically.

- **External Spatial Data**:

  External spatial data can refer to the following:

    - **Remote or Local ESRI shapefile**: When you use the **Linked** option in Map Layer Data Properties dialog or use the **File** property in the Properties Panel to specify the local or remote path/location of an ESRI shapefile, the ESRI shapefile gets linked to the Map layer. However, it remains an external source as the dependent ESRI shapefile needs to moved along with the report between projects or machines.
    - **Report Dataset**: When you use a report dataset to provide spatial data to a map layer and you don't embed the spatial data, it remains an external source. This requires the dependent database file to moved along with the report between projects or machines.

Analytical Data

Analytical data is the data that you want to visualize on the map, for example, tourist attractions in a city or product sales by region. For analytical data, you can associate it with map elements by indicating match fields in the **Match** box of the **Map Layer Data Properties** dialog. You can use one or more fields in the **Match** box of the **Map Layer Data Properties** dialog; for each spatial data field you must indicate a unique analytical data field. This data is optional.

You can get analytical data from the following types of data sources.

- **Dataset field:** A field from a dataset.
- **Spatial data source field:** A field from the spatial data source. For example, you can often find that an ESRI Shapefile contains both spatial and analytical data. Field names from the spatial data source are marked with the # sign in the drop-down list of fields.
- **Embedded data for a map element:** After you embed polygons, lines, or points in a report, you can select the data fields for map elements and define custom values.

See Add Data to a Map tutorial.

Map Layers

A map is a collection of layers that display data on the map control.

- **Polygon layer:** Display outlines of areas or markers for the polygon center point.
- **Point layer:** Display markers for point locations.
- **Line layer:** Display lines for paths or routes.
- **Tile layer:** Adds a Bing map tiles background.

A map can have one or more layers. You can load these layers on top of each other to create a more detailed map. For example, a polygon layer can represent the borders of a country, a line can represent transportation routes, a point can represent the locations and a tile can add a virtual earth background on the map. See Use Map Layers for more information.

Map layer element appearance:

- Properties that you set on a polygon layer, line layer and a point layer apply to all map elements on that layer, whether or not the map elements are embedded in the report definition.
- Properties that you set for rules apply to all map elements on a layer. All data visualization options apply only to map elements that are associated with spatial data.

Map Layer Data Dialog

The Map layer Data dialog is used to set up spatial and analytical data for the map control. For more information on spatial and analytical data, see Add Data to a Map.

General

Spatial data source: Select one of the spatial data source connection types:

- **Embedded:** The map layer data is loaded from the .shp data source that you embed into the map layer by indicating the .shp file in the **Import spatial data from file:** field. This field appears below when you select this option.

  Spatial fields: Use the plus sign button to add a field, and the X button to delete a field. For each newly added spatial field, you must specify the name and type in the corresponding fields below.

  Field name: Enter a name of a spatial field.

  Field type: Select the type of a spatial field from the list.

- **Linked:** The map layer data is linked to the .shp file and is uploaded at report rendering. You select this type of data source by indicating the .shp file in the **File Name** field that appears.
- **Dataset:** The map layer data is loaded from the data source of the report. In the **Dataset** field and in the **Field name** field that appear below, select a dataset from the bound data source and a dataset field.

  > ⚠ **Caution**: In **Field name**, simply type the name of the dataset field that contains spatial data. For example, enter the dataset field name as ***StateName***, not as ***=[StateName]***.

- **Analytical data:** The map layer data is loaded from the the analytical dataset of the bound report data source. In the **Field name** field that appears below, you must set the name of the data field that contains spatial data in the Analytical dataset.

> ⚠ **Caution**: In **Field name**, enter the data field name as ***=[StateName]***, not as ***StateName***.

Analytical Data

Dataset: Select the dataset for the analytical data to be displayed on the map layer.

Match: Use the plus sign button to add a relationship between a spatial data field and an analytical data field.

Spatial field: A field with spatial data that specifies an element on the map surface, for example, boundaries of a country.

Analytical field: A field with analytical data that displays information on the related map element, for example, the country population.

Filters

The **Filters** page of the **Map Layer Data Properties** dialog allows you to filter the data that is included in the map. Use the plus sign button to add a filter, and the arrow and X buttons to move or delete filters. You need to provide three values to add a new filter to the collection.

Expression: Enter the expression to use for evaluating whether data should be included in the map.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right:

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values (used with the **In** and **Between** operators) separate values using commas.

Color, Marker, and Size Rules

Rules apply properties to a layer when the layer has map elements that have a relationship to analytical data. For example, the color rule varies map element color based on color palette, color range, custom colors, and further specify the distribution option to control display values.

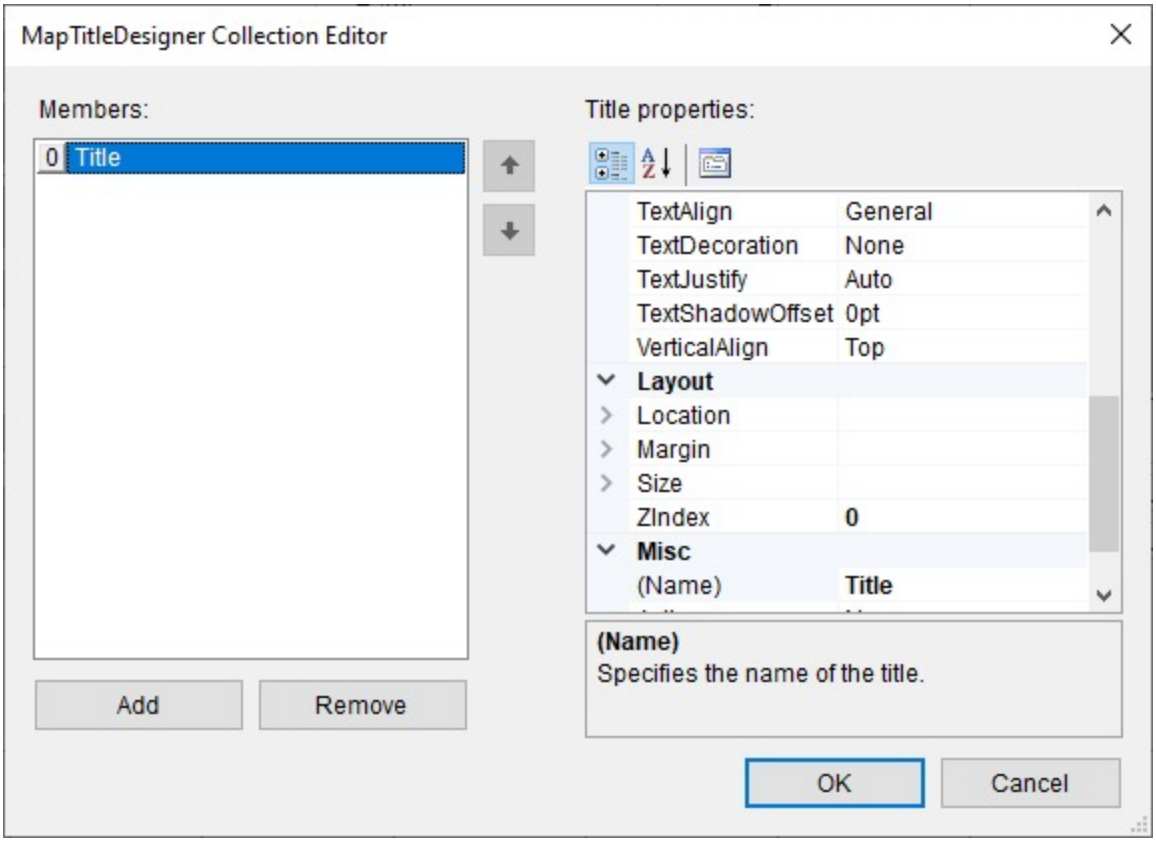
The type of rule depends on the layer type. For example, use point size rules to vary bubble size based on population.

See Use Color Rule, Marker Rule, and Size Rule for more information.

Add a Title to the Map

Map Title describes the theme or subject of the map. Use these steps to learn adding a title on a map control.

1. On the design surface, select the Map control.
2. In the Properties window, click the **Titles (Collection)** property and then click the ellipsis (...) button that appears.
3. In the **MapTitleDesigner Collection Editor** that appears, in the Members list of titles, **Title** with the default property settings already exist.



4. In the **Properties Panel**, you can modify the text, font, border and the background color settings of the map title.

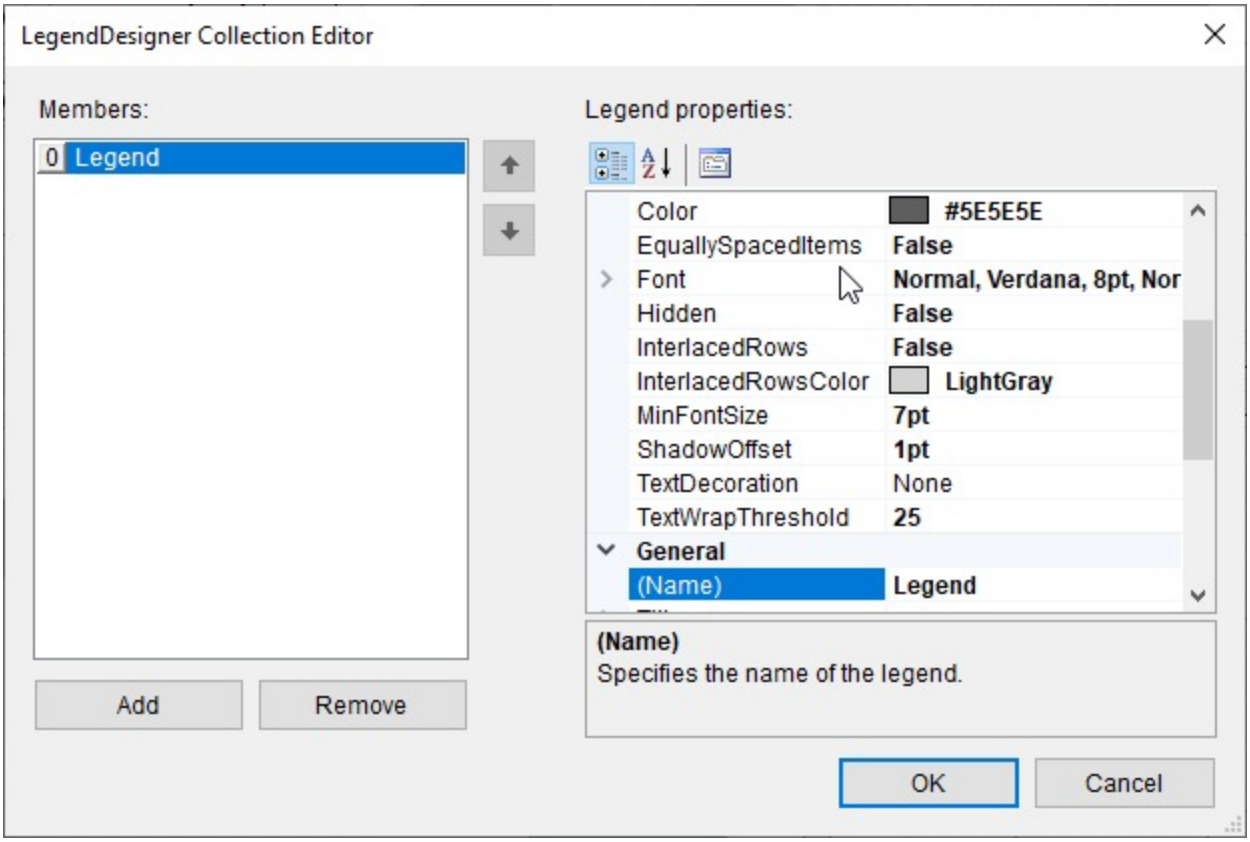
5. Click **OK** to close the dialog.

Add a Legend to the Map

A legend on a map provides valuable information to users for interpreting the map data visualization rules such as color, size, and marker type differences for map elements on a layer. By default, a single Legend item already exists in the legends collection which can be used by all layers to display items. You can also create additional legends to use them individually with layers that have associated rules to display items in the legend. Use these steps to learn adding and setting a legend on a map:

1. On the design surface, select the Map control.
2. In the Properties window, click the **Legends (Collection)** property and then click the ellipsis (...) button that appears.

3. In the **LegendDesigner Collection Editor** that appears, click **Add** under the Members list of legends. **Legend1** with the default legend settings appears in the Members list.



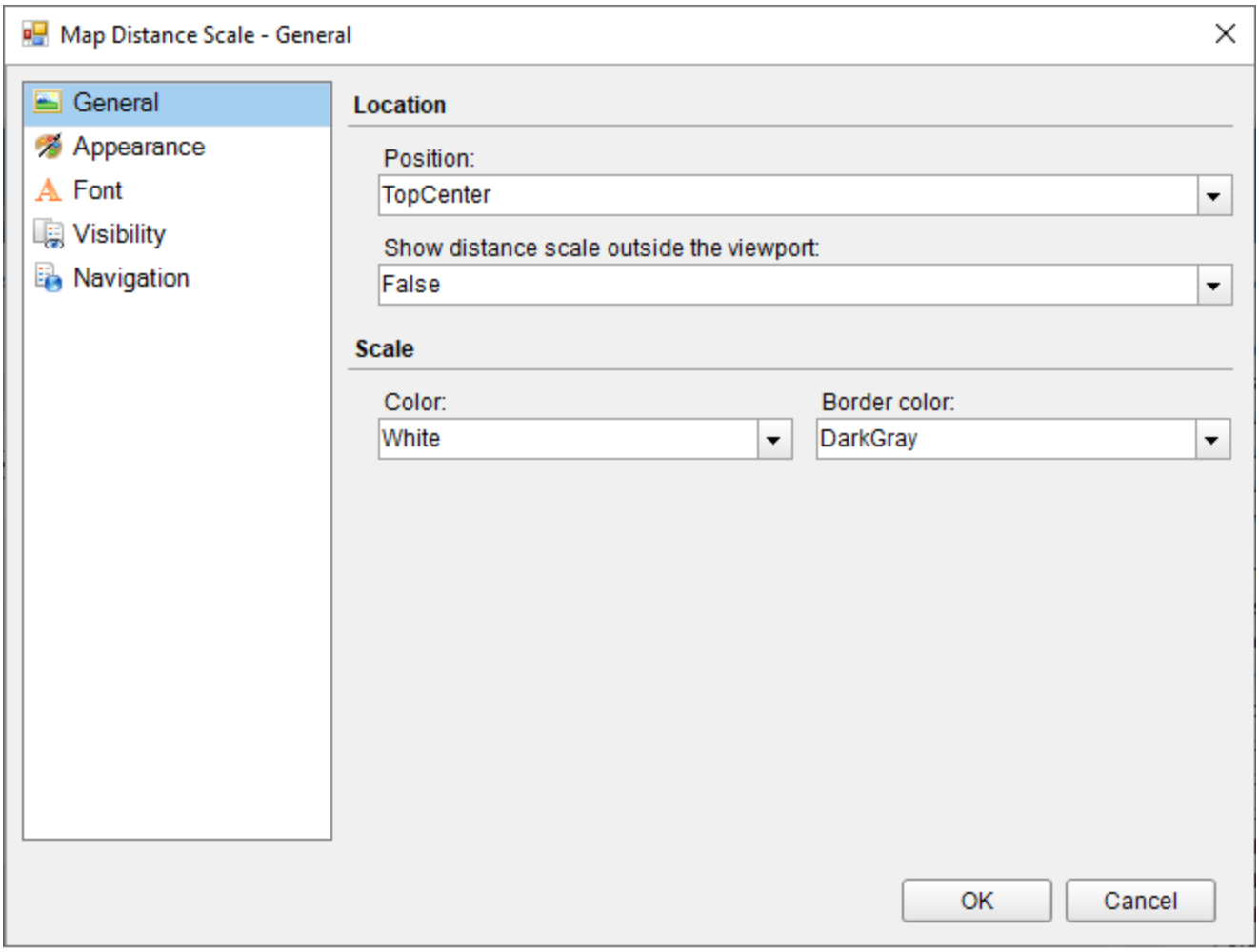
4. With **Legend1** selected in the Members list of legends, you can make modifications to its font, border and background color settings.
5. Click **OK** to close the dialog.

> **Note**: You can associate the newly added legend to a layer by specifying its name in the **Legend Name** field that appears in the Legend tab on a specific rule page of a Layer dialog. See Use Color Rule, Marker Rule, and Size Rule for more information.

Set the Distance Scale on a Map

A distance scale helps a user to understand the scale of the map. Distance on a map is not the same as the actual real-world distance, so a distance scale shows that a certain distance on the map equals a certain distance in a real-world. In distance scale, the distance is displayed in both miles and kilometers. The scale range and values are automatically calculated using the viewport boundaries, projection type, and zoom level. Use these steps to learn setting a distance scale on a map:

1. On the design surface, select the Map control.
2. In the Properties window, click the **DistanceScale** property and then click the ellipsis (...) button that appears.
3. In the **Map Distance Scale** dialog that appears, on the **General** page, you can set the location and the color of the distance scale.

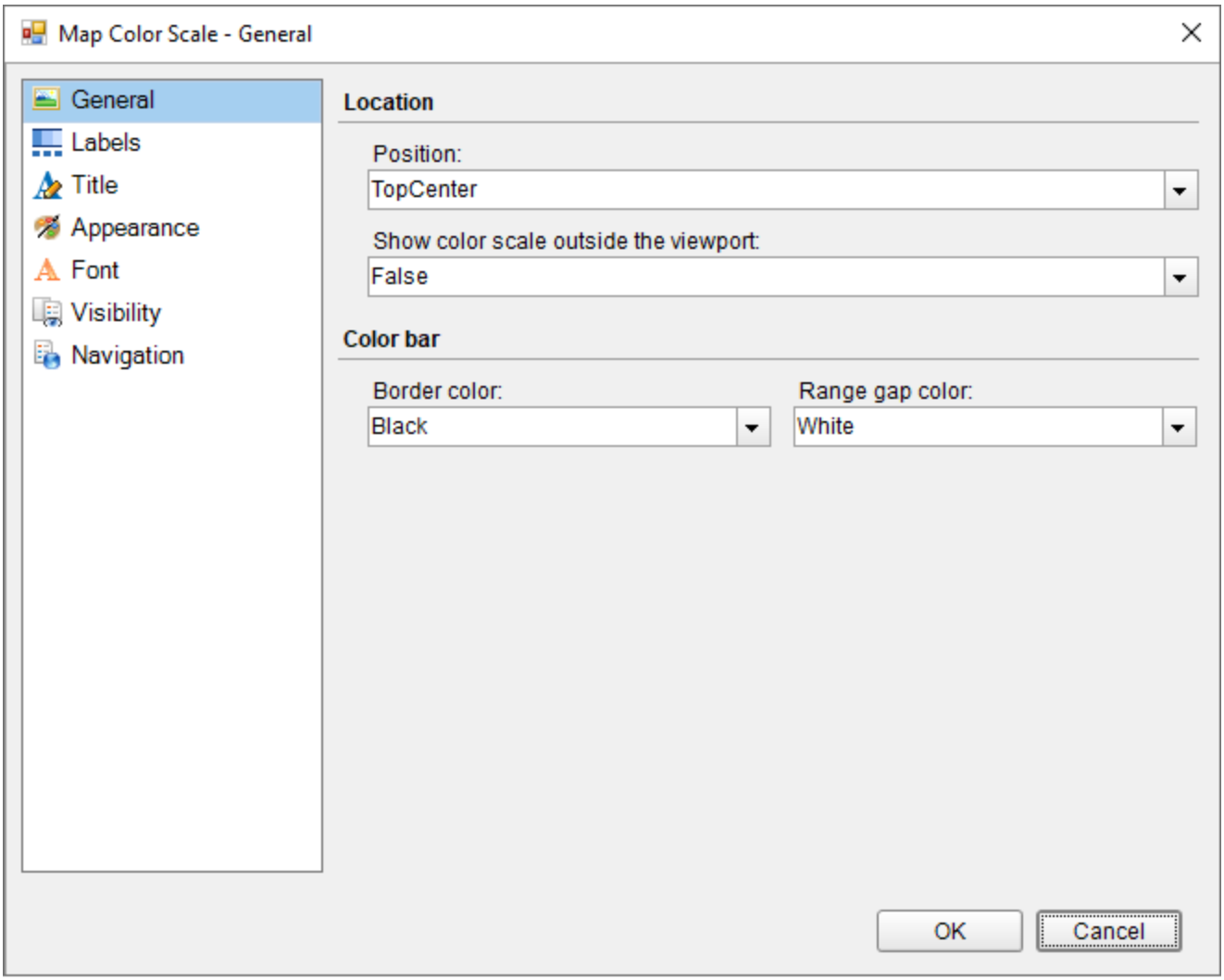


4. On the **Appearance** page of the dialog, you can make modifications to the border width, style, color and background color.
5. On the **Font** page of the dialog, you can make modifications to the font properties of the distance scale.
6. On the **Visibility** page of the dialog, you can setup the visibility mode of the distance scale.
7. On the **Navigation** page of the dialog, you can setup the interactivity features for the distance scale.
8. Click **OK** to close the dialog.

Set the Color Scale on a Map

A color scale helps a user to understand the range of colors that are used for data visualization on a layer. A map has just one color scale and multiple layers can provide data for it. Use these steps to learn setting a color scale on a map.

1. On the design surface, select the Map control.
2. In the Properties window, click the **ColorScale** property and then click the ellipsis (...) button that appears.
3. In the **Map Color Scale** dialog that appears, on the **General** page, you can set the location and the color of the color scale.



4. On the **Labels** page of the dialog, you can make modifications to the properties of the color scale labels.
5. On the **Title** page of the dialog, you can make modifications to title text and font properties of the color scale.
6. On to the **Appearance** page, you can make modifications to the border width, style, color and background color.
7. On the **Font** page, you can make modifications to the font properties of the color scale.
8. On the **Visibility** page, you can setup the visibility mode of the color scale.
9. On the **Navigation** page, you can setup the interactivity features for the color scale.
10. Click **OK** to close the dialog.

Add Data to a Map

ActiveReports provide numerous ways to add spatial data to the map. You can either use the **Map Wizard** and add data from an ESRI shapefile or use the **Map Layer Data Properties** dialog for using advance options. You can also add spatial data from the Properties Panel.

Add Spatial Data

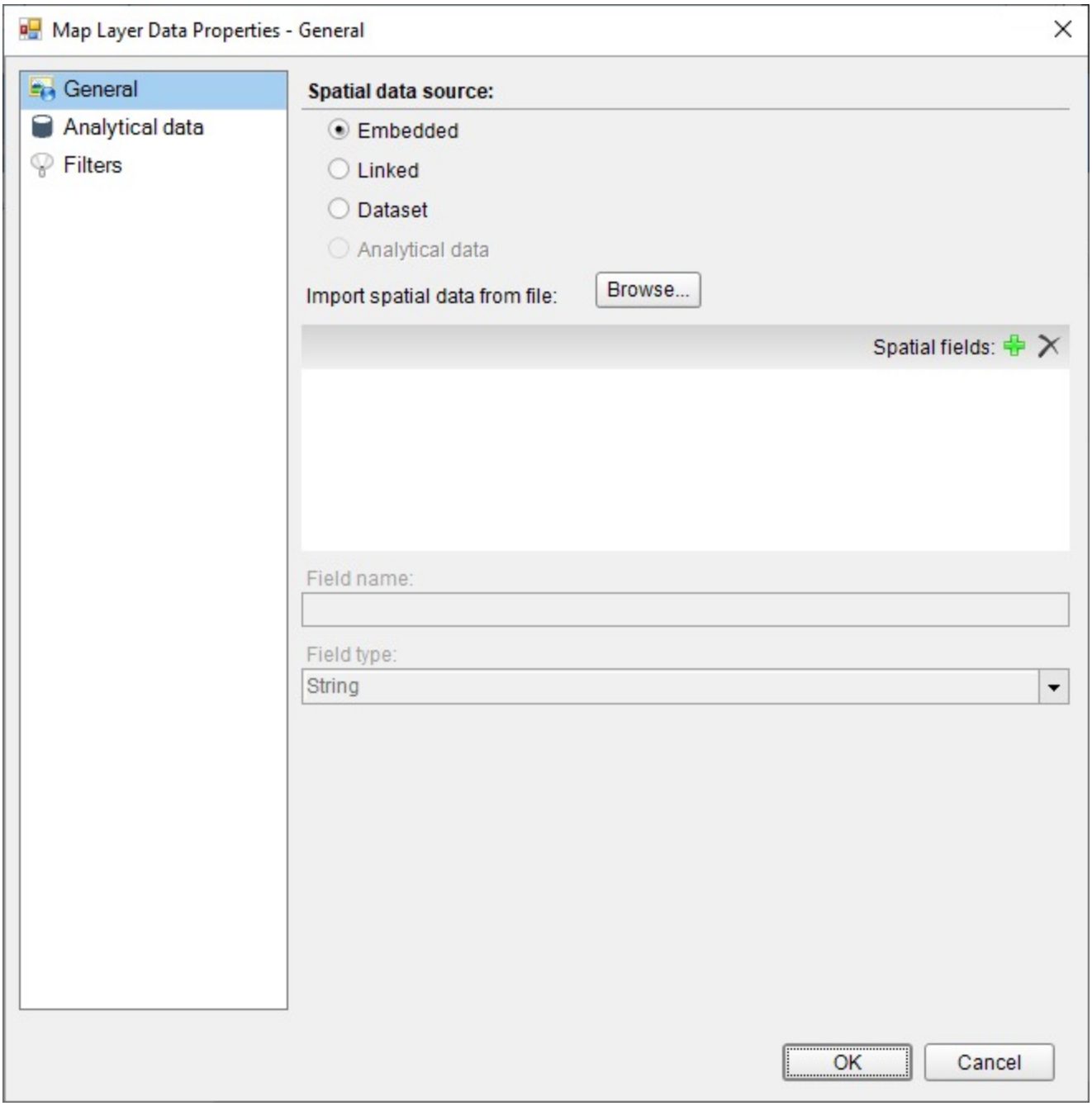
Using Map Wizard

1. From the Visual Studio toolbox, drag and drop a Map control onto the design surface.
2. In the **Select a Map Template** wizard that appears, select **From ESRI file**.
3. In the **Open** dialog that appears, navigate to the folder that contains the .shp and .dbf files, select the .shp file, use the Map Resolution slider to simplify vector data and click **Open**.

This option automatically imports the spatial data stored in the shapefile and adds a related layer to the map control.

Using the Map Layer Data Properties dialog

The **Map Layer Data Properties** dialog provide the following advance options to add spatial data. For more information on Map Layer Data Properties dialog, see Map.



Embedded: Use this option where you want to add spatial data from an ESRI file along with an additional option of adding custom spatial data fields if required.

Add Spatial data from an ESRI file using the Embedded option

1. In the **Map Layer Data Properties** dialog, on the **General** page, select the **Embedded** option and click the **Browse** button.
2. In the **Open** dialog that appears, navigate to the folder that contains the .shp and .dbf files.
3. Select the .shp file, use the Map Resolution slider to simplify vector data and click **Open**. The fields list gets populated with spatial data fields that are added from the shapefile.
4. In **Dataset**, set the name of the dataset.

5. In the Fields list, click the Add (+) button to add a new empty spatial data field.
6. With the newly added spatial data field selected in the list, in **Field name**, enter the name of the field.
7. In **Field type**, set the field type of newly added spatial data field.
8. Click **OK** to close the dialog.

> **Note**: In order to apply the new spatial data field on a layer, you must provide its value for each map layer element like polygons, points or lines in the added map layer Designer Collection Editor dialog.

Linked: Use this option when you just want to add spatial data from an ESRI shapefile without any additional modifications or additions.

Add Spatial data from an ESRI file using the Linked option

1. In the **Map Layer Data Properties** dialog, on the **General** page, select the **Linked** option and click the **Browse** button.
2. In the **Open** dialog that appears, navigate to the folder that contains the .shp and .dbf files.
3. Select the .shp file and click **Open**.
4. Click **OK** to close the dialog.

Dataset: Use this option when you have a dataset that stores a spatial data field to provide spatial data to the Map. You can directly use the data fields from the dataset to display data on a map layer without setting the match fields for analytical data. Therefore, it also provides you an additional option of having different dataset for analytical data and spatial data respectively.

Add Spatial data from a Dataset

1. In the **Map Layer Data Properties** dialog, on the **General** page, select the **Dataset** option.
2. In **Dataset**, set the name of the dataset.
3. In **Field name**, enter the data field name that contains the spatial data.

   > **Caution**:
   >
   > - Simply type the name of the dataset field that contains spatial data. For example, enter value as ***StateName***, not as ***=[StateName]***.
   > - You cannot use the MapPoint() function in parameters.

4. Click **OK** to close the dialog.

> **Note:**
>
> - **MapPoint()** function is supported for Point layer only.
> - Simple (non spatial) data can also be added as Spatial field using MapPoint() expression as `=MapPoint(<Latitude>, <Longitude>)` from Expression Editor.

Analytical : Use this option when you want to use the spatial data field from the same dataset that you may use for adding Analytical data to the layer. For using this option you need to first set the dataset for analytical data and then use the spatial data field from the dataset to provide spatial data to the map layer.

Add spatial data from Analytical data

1. Configure Analytical data for the Map control. See the dropdown section below to learn adding Analytical data.

> **Note**: Once the Analytical data has been set, the Analytical option on the General page of the Map Layer Data Properties dialog becomes active.

2. In the **Map Layer Data Properties** dialog, on the **General** page, select the **Analytical** option.
3. In **Field name**, enter the data field name that contains the spatial data.

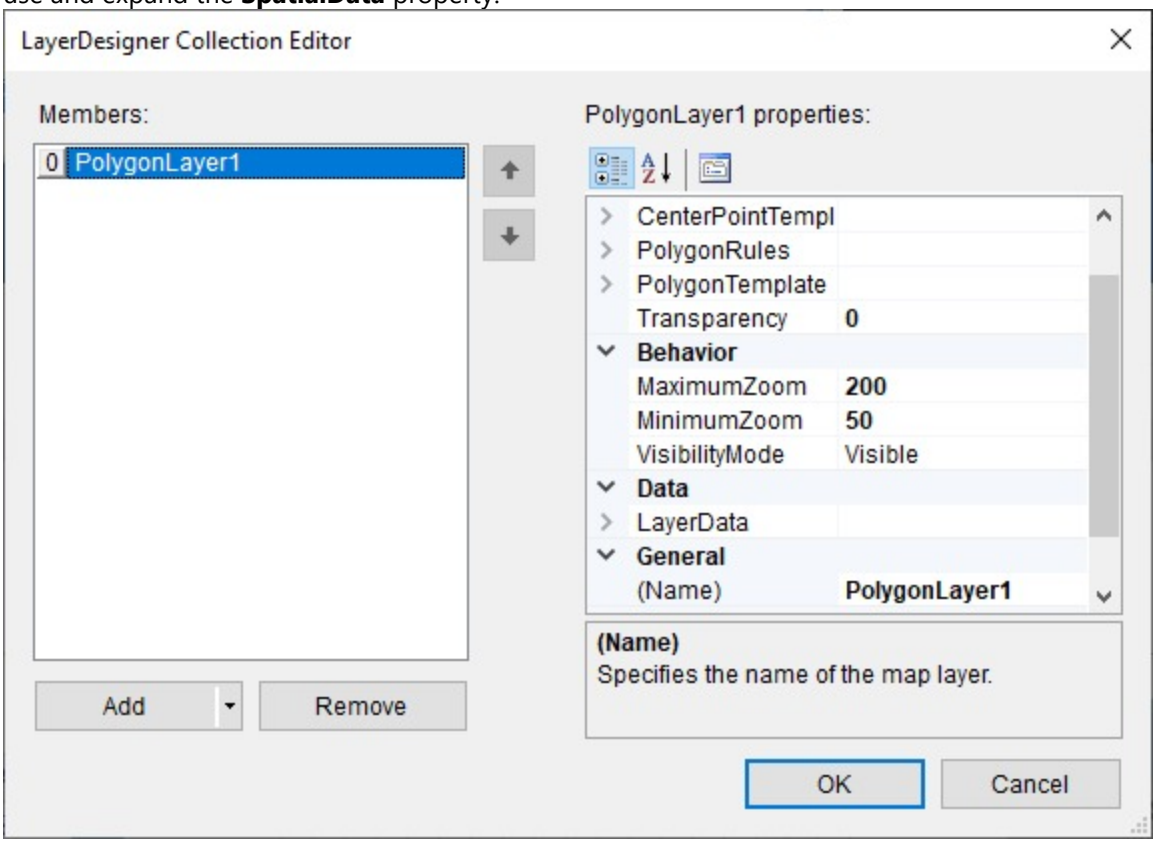
> **Caution**: In **Field name**, enter the data field name as ***=[StateName]***, not as ***StateName***.

4. Click **OK** to close the dialog.

Using Properties Panel

These steps assume that you have a Map control containing at least one map data layer placed on the design surface. To learn how to add a layer to the Map control, see Layers.

1. With the Map control selected, go to the Properties window, click the **Layers (Collection)** property and then click the ellipsis button that appears.
2. In the **LayerDesigner Collection Editor** that appears, under Members, select the map data layer you want to use and expand the **SpatialData** property.



3. In **Type**, choose the spatial data source of the layer from the following supported options:
   - **Embedded**: Use this option when you want to add custom spatial data to the map data layer. This can be done by first adding the spatial data fields for the embedded spatial data using the MapFieldDefinitionDesigner Collection Editor which can be accessed through the **FieldDefinitions** property. And then later adding spatial data (points, polygons or lines) using the PolygonDesigner Collection Editor, PointsDesigner Collection Editor or LineDesigner Collection Editor depending on the layer in use. These Editor dialogs can be accessed using the **SpatialData** > **Polygons**, **SpatialData** >

Points or **SpatialData** > **Lines** property.

- **File**: Use this option when you want to add spatial data from an ESRI file. Use the **Source** property to specify the location of the shapefile.

  > **Note**: The specified location must contain the shape format (.shp) and attribute format (.dbf) files.

- **DataSet**: Use this option when you have a dataset that stores a spatial data field to provide spatial data to the Map. This option is equivalent to the DataSet option in the **Map Layer Data Properties** dialog. Use the **SpatialData** > **DataSetName** property to specify the name of the dataset and the **SpatialData** > **SpatialField** property to specify the data field that contains spatial data.

  > **Note:**
  >
  > - **MapPoint()** function is supported for Point layer only.
  > - Simple (non spatial) data can also be added as Spatial field using MapPoint() expression as `=MapPoint(<Latitude>, <Longitude>)` from Expression Editor.

  > **Caution:** You cannot use the **MapPoint()** function in parameters.

- **DataRegion**: Use this option when you have a dataset that stores a spatial data field to provide spatial data to the Map. This option is equivalent to the **Analytical** option in the **Map Layer Data Properties** dialog. Use the **LayerData** > **DataSetName** property to specify the name of the dataset and the **SpatialData** > **VectorData** property to specify the data field or expression that contains spatial data.

4. Click **OK** to close the dialog.

Add Analytical Data

Analytical data is the data that you want to visualize on the map, for example, tourist attractions in a city or product sales by region. For analytical data, you can associate it with map elements by indicating match fields in the **Match** box of the **Map Layer Data Properties** dialog. You can use one or more fields in the **Match** box of the **Map Layer Data Properties** dialog; for each spatial data field you must indicate a unique analytical data field. This data is optional.

You can get analytical data from the following types of data sources.

- **Dataset field:** A field from a dataset.
- **Spatial data source field:** A field from the spatial data source. For example, you can often find that an ESRI Shapefile contains both spatial and analytical data. Field names from the spatial data source are marked with the # sign in the drop-down list of fields.
- **Embedded data for a map element:** After you embed polygons, lines, or points in a report, you can select the data fields for map elements and define custom values.

Add Analytical Data to the Map control

Use the following steps to add analytical data to the map. These steps assume that you have added a page layout template to your report and have a data connection in place.

1. Add spatial data to the map control. See the dropdown sections above to learn adding spatial data to the map control.
2. In the **Map Layer Data Properties** dialog, go to the **Analytical data** page.
3. Select the dataset that you want to use from the **Dataset** dropdown list and click the **Add** (+) button located next to the **Match** field. This creates an empty match field expression and enables the **Spatial** and **Analytical** field properties.

4. In the **Spatial field** and **Analytical field** options set data fields that contain same data in both Spatial and Analytical databases. This builds the match field expression and relates analytical data to map elements on a map layer.

   > **Note**: It is necessary to set match fields if you want to use a spatial data field from analytical data, or if you want to visualize analytical data on the map layer. Match fields enable the report processor to build a relationship between the analytical data and the spatial data.

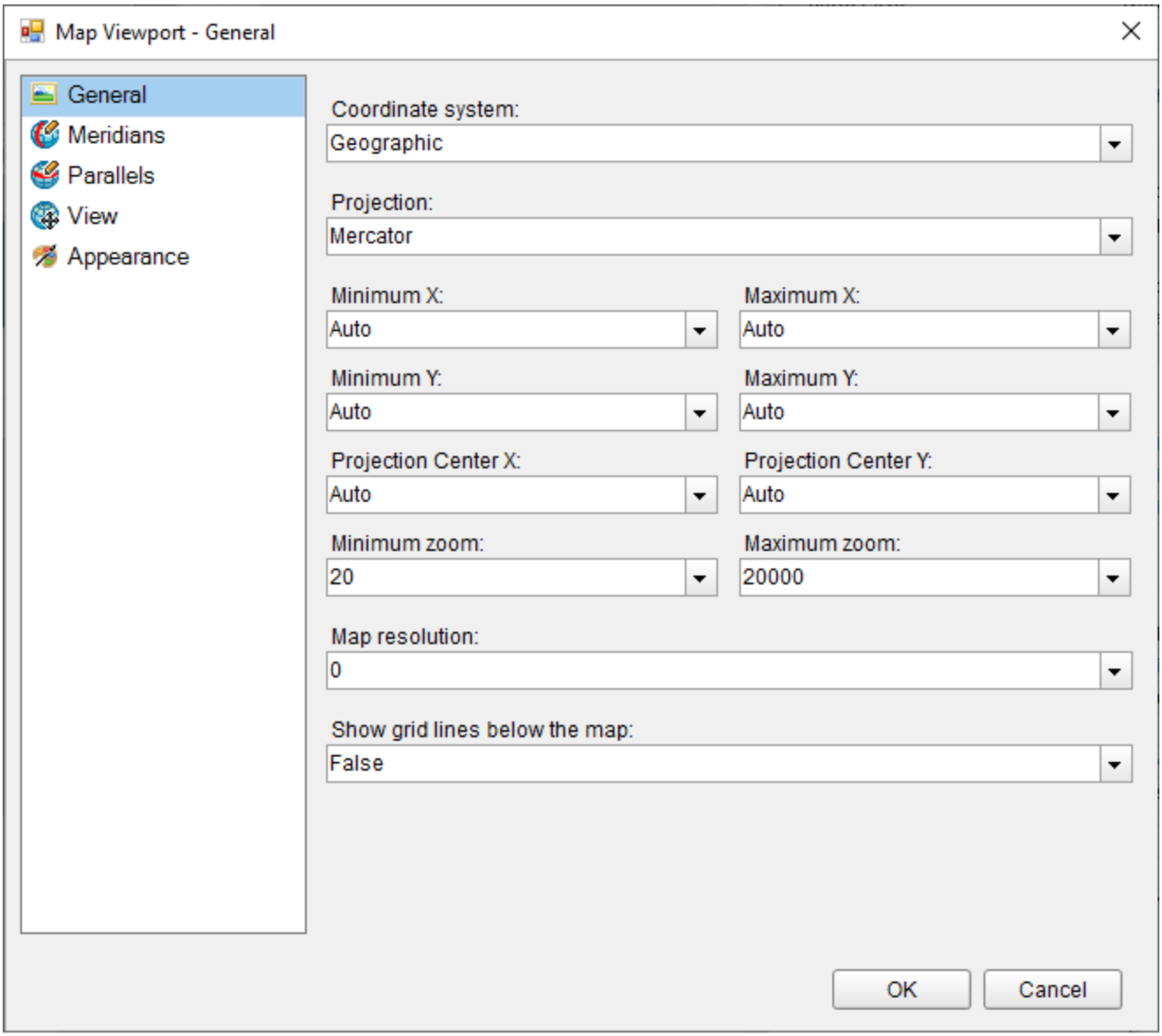
5. Click **OK** to close the dialog.

Modify the Appearance of the Viewport

Viewport refers to the area on the map where map data is displayed against a geographical background. It specifies the coordinates, projection system, parallels and meridians, center point, and scale of the map. In other words, it is a map element that actually display geographical data and occupies most area of the map control depending on the location and dock position of other map elements. See Map for more information.

You can modify Viewport properties to make the map look more attractive.

1. On the design surface, select the Map control.
2. Go to the Properties window, click the **Viewport** property and then click the ellipsis (...) button that appears.
3. In the **Map Viewport** dialog that appears, on the **General** page, choose the **Coordinate system**. The map viewport supports the following two coordinate system:



- **Geographic**: Specifies Earth coordinates by defining longitude and latitude values. If you set the CoordinateSystem property to Geographic, then you must specify the Projection property. A projection is a set of rules on how to locate three-dimensional objects onto a planar surface.
- **Planar**: Specifies geometric coordinates on a two-dimensional surface by using X and Y values. and in case you set it to Geographic then set the **Projection**.

4. Go to the **Meridians** page, set its visibility and its line and font style and color.
5. Similarly, go to the **Parallels** page, set its visibility and its line and font style and color.
6. On the **View** page of the dialog, choose a Center and Zoom mode. The map viewport supports the following four center and zoom modes:
   - **Custom**: Choose this option to specify custom values for the view center and the zoom level.
   - **Center map to show a map layer**: Choose this option to specify a layer and automatically, center the view on its map data. For example, center the view on LineLayer1.
   - **Center map to show a map element**: Choose this option to center the view on a specific data bound map element. For example, center the view on the map element where the name of the match field is [StateName] and the match value is "Washington".

- **Center map to show all map elements**: Choose this option to center the view on all map elements in the layer.

> **Note**: Zoom and View Center level can also be set from the design surface using the zoom slider and arrow keys that appears in the View Pane of the Map control.

7. On the **Appearance** page, set the Border and Background style and color of the viewport.
8. Click **OK** to close the dialog.

Use Map Layers

A map is a collection of layers that display data on the map control. See, Map for more information.

This topic illustrates how to add, remove and change order of layers. It also shows how to add interactive navigational feature to map layer elements.

Add a map layer

From the design surface

1. From the Visual Studio toolbox, drag and drop a Map control onto the design surface.
2. In the **Select a Map Template** wizard that appears, select a map template.
3. Click the Map until the map panes appear.
4. Right-click inside the area labeled "**right-click to add the new layer**." and select the map layer you want to use.

Using the LayerDesigner Collection Editor

1. From the Visual Studio toolbox, drag and drop a Map control onto the design surface.
2. In the **Select a Map Template** wizard that appears, select a map template.
3. With the Map control selected, go to the Properties window, click the **Layers (Collection)** property and then click the ellipsis button that appears.
4. In the **LayerDesigner Collection Editor** that appears, use the **Add** combo-box to view the list of available layers and select the map layer you want to use.

Delete a map layer

From the design surface

1. On the design surface, click the map until the map panes appear.
2. In the layers pane, right-click the layer you want to remove and select **Delete**.

Using the LayerDesigner Collection Editor

1. On the design surface, with the Map control selected, go to the Properties window, click the **Layers (Collection)** property and then click the ellipsis button that appears.
2. In the **LayerDesigner Collection Editor** that appears, under the members list, select the map layer you want to delete and click the **Remove** button.

Change order of layers

Map layers are rendered from left to right in the order that they appear in the map panes. In the image below, the polygon layer is drawn first and the line layer is rendered last. Layers that are rendered later might hide map elements on layers that are rendered earlier. You can change rendering order of layers added to the map control using the **LayerDesigner Collection Editor**. Follow these to steps learn re-ordering the layers on a map.

1. On the design surface, with the Map control selected, go to the Properties window.
2. In the Properties Panel, click the **Layers (Collection)** property and then click the ellipsis button that appears.
3. In the **LayerDesigner Collection Editor** that appears, under the members list, select the map layer you want to reorder and use the up or down arrow to change the rendering order of each layer.
4. Click **OK** to close the Collection Editor.

Embed layer spatial data or tiles in a map

When you embed map elements or map tiles in a report, the spatial data is stored in the report definition.

1. Click the map until the map panes appear.
2. In the layers pane, right-click the added layer that contains spatial data, select **Embed Spatial Data** and then select **All Spatial Data** or **Currently Visible Data**. In case of Tile layer select **Embed Tiles**.

> **Note**: **All Spatial Data** refers to all the spatial data fields, while **Currently Visible Data** refers to the spatial data field that is set in the **Field** property.

Add Hyperlinks, Bookmarks, and Drill-through links

Map layer elements like point, polygon and line provides you a functionality to set interactive navigational features like a bookmark link to jump to other areas in the same report, a hyperlink to jump to a Web address, or a drill-through link to jump to another report. Follow these steps to learn adding hyperlinks, bookmarks and drill-through links to a layer element:

1. On the design surface, click the map until the map panes appear.
2. In the layers pane, right-click the layer in use and select **Edit**.
3. In the selected layer's dialog that appears, go to the **Navigation** page.
4. On the Navigation page, select from the following actions to perform when a user clicks a data layer element :
    - **None**: The default behavior to indicate that the item has no action.
    - **Jump to report**: For drill-through reporting, select this option and provide the name of a local report, the relative path of a report in another folder, or the full path of a report on another server.
        - **Parameters**: Supply parameters to the targeted report by entering the **Name** of each parameter, the **Value** to send to the targeted report, or whether to **Omit** the parameter. Note that parameter names you supply must exactly match parameters in the target report. You can remove or change the order of parameters using the X and arrow buttons.

      For detailed steps on adding a drill-through link, see Drill-Through Links.

    - **Jump to bookmark**: Select this option and provide a valid **Bookmark ID** to allow the user to jump to another report control with the same Bookmark ID.
      For more information on adding bookmarks, see Bookmarks.

- **Jump to URL**: Select this option and provide a valid URL to create a hyperlink to a Web page. For more information on adding hyperlinks, see Hyperlinks.

5. Click **OK** to close the dialog.

Polygon Layer

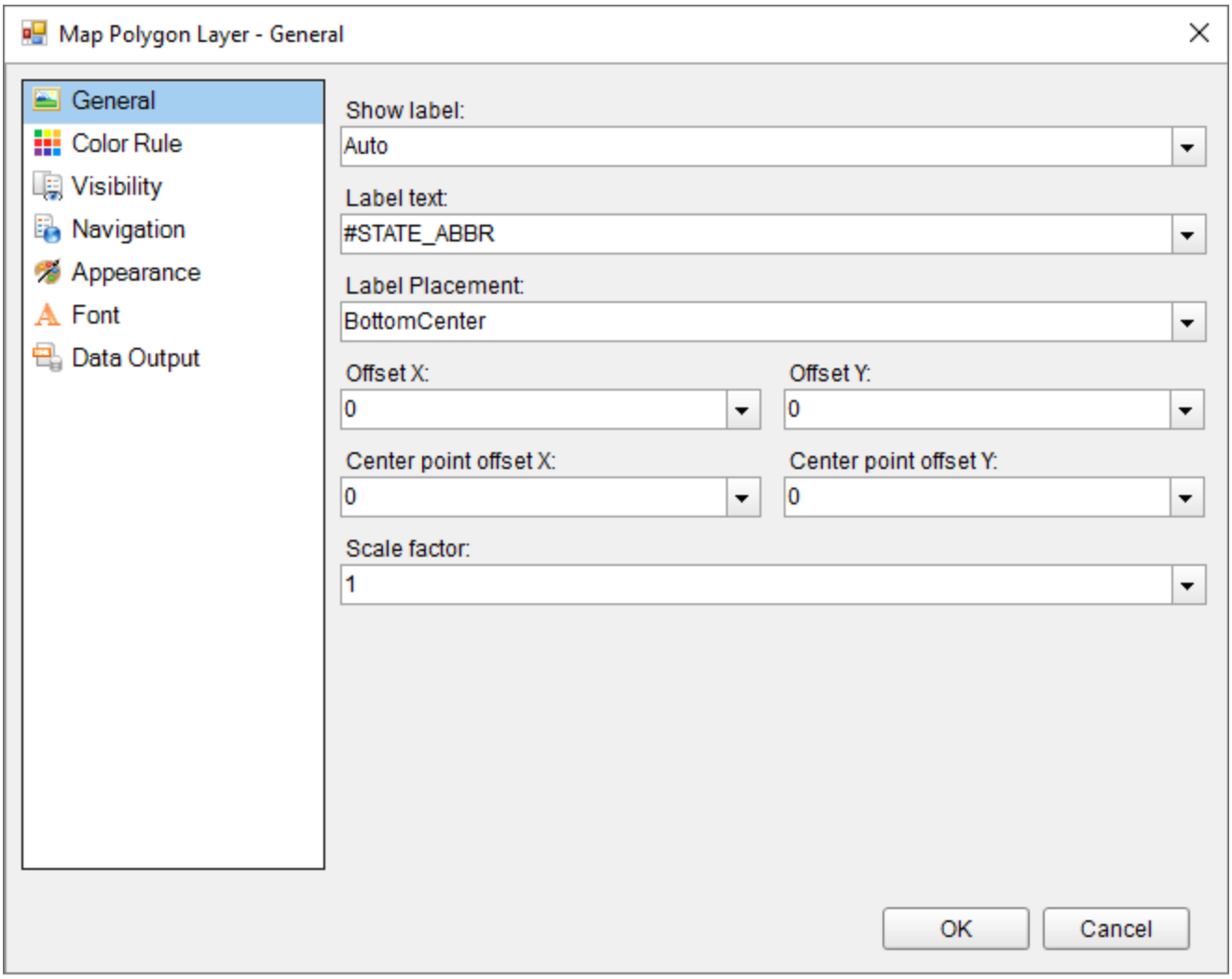
A Polygon layer display outlines of areas or site boundaries that can be linked to Locations and can be used to select records that fall within the boundary for reporting usage.

Use the following steps for creating a basic map using the Polygon layer. These steps assume that you have added a page layout template to your project and have a data connection in place. See Data Binding in Page/RDLX Reports for more information.

1. From the toolbox, drag and drop a Map control onto the design surface.
2. In the **Select a Map Template** wizard that appears, select the **New Map** template.
3. Click the map until the map panes appear.
4. Right-click inside the area labeled "Right click to add the new layer." and select **Add Polygon Layer**. This adds a polygon layer to the map and opens the **Map Layer Data Properties** dialog.
5. In the **Map Layer Data Properties** dialog that appears, on the **General** page, import the spatial data (polygons) from a shape file or use a data field from the analytical data to specify the spatial data source.
6. In case you want to visualize analytical data on the map, go to the **Analytical data** page of the dialog, select your dataset from the **Dataset** dropdown list and click the **Add** (+) button located next to the **Match** field. This creates an empty match field expression and enables the **Spatial** and **Analytical** field properties. See, Add Data to a Map for more information.
7. In the **Spatial field** and **Analytical field** options set data fields that contain similar data in both Spatial and Analytical databases. This builds the match field expression and relates analytical data to map elements on a polygon layer.

   > **Note**: It is necessary to set match fields if you want to use a spatial data field from analytical data, or if you want to visualize analytical data on the map layer. Match fields enable the report processor to build a relationship between the analytical data and the spatial data.

8. Go to the **Filters** page and set filters if any.
9. Click **OK** to close the dialog.
10. In the layers pane, right-click on **PolygonLayer1** and select **Edit** to open **Map Polygon Layer** dialog.



11. In the **General** page of the dialog, select any data field from the **Label Text** combo box to display as labels inside polygons at run time.
12. Go to the **Color Rule** page of the dialog, to set rules to visualize data using color pallets, color ranges or custom colors for polygons or polygon center points that gets displayed on the map or keep it set to the default "Use Appearance settings". See, Use Color Rule, Marker Rule, and Size Rule for more information.
13. Go to the **Visibility** page of the dialog and make sure the layer visibility is set to **Show**. You can also select options to show or hide layer based on any expression or zoom value.
14. In the **Navigation** page of the dialog, you can optionally link the polygon to a URL, bookmark or a report.
15. The **Appearance** page of the dialog either reflects the default appearance of the polygons or the color rule settings if any.
16. In the **Font** page of the dialog, you can optionally set the font family, size, weight, style, set and color for the label text that you had set in the **General** page of the dialog.
17. Go to the **Data Output** page and specify the Data Element Name of the layer to be used while rendering to XML and also specify whether the layer should be included in output while rendering or not.
18. Click **OK** to close the dialog and go to the preview tab to view the map.

Point Layer

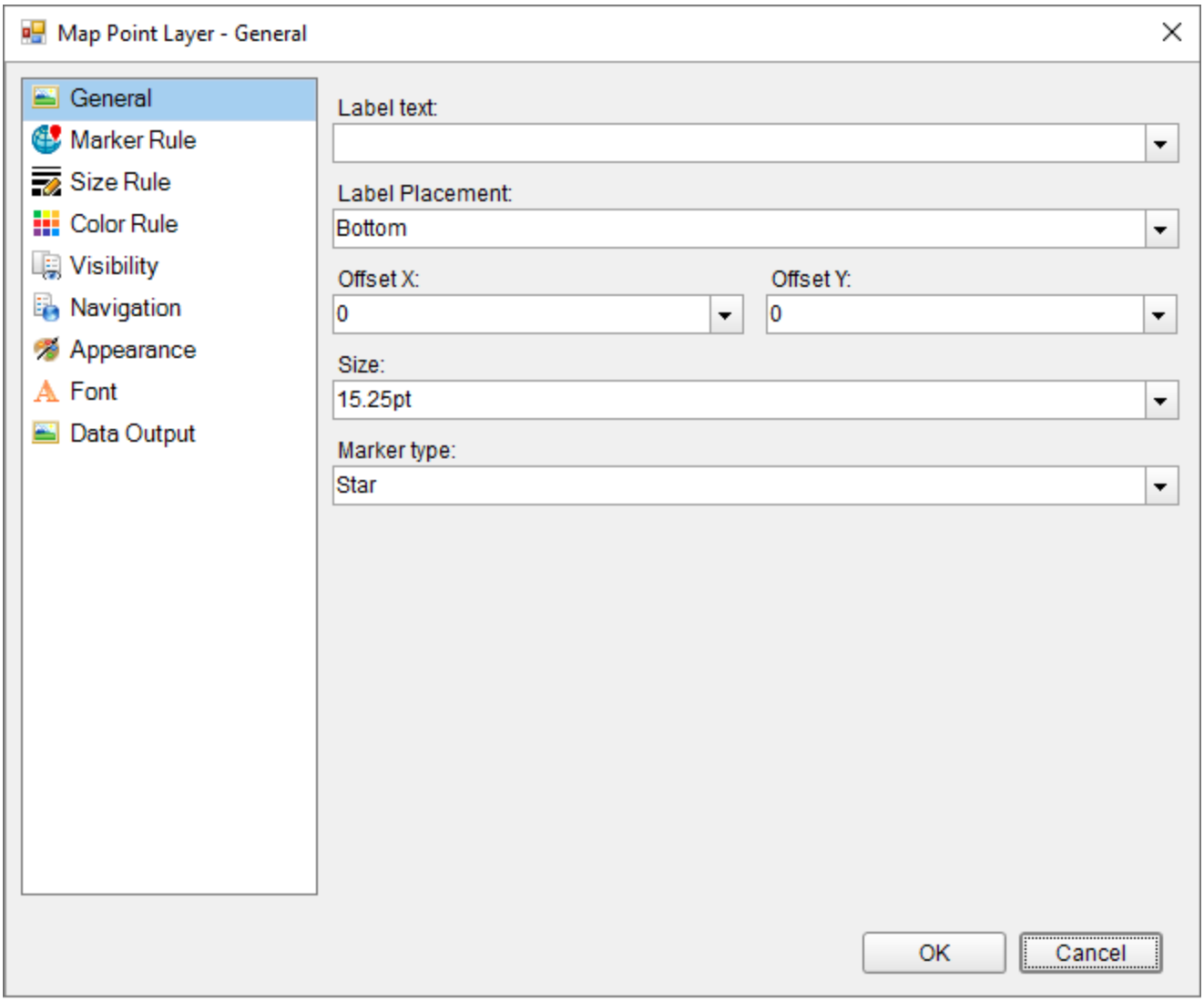
A Point layer displays markers for point locations such as a city or an address for a store, restaurant, or school.

Use the following steps for creating a basic map using the Point layer. These steps assume that you have added a page layout template to your project and have a data connection in place. See Data Binding in Page/RDLX Reports for more information.

1. From the toolbox, drag and drop a Map control onto the design surface.
2. In the **Select a Map Template** wizard that appears, select the **New Map** template.
3. Click the map until the map panes appear.
4. Right-click inside the area labeled "Right click to add the new layer." and add either a Tile layer or a Polygon layer. One of these layers that you add serves as the geographic base to plot points on. See, Polygon Layer or Tile Layer for steps to add these layers on a Map control.
5. Right-click again inside the area labeled "Right click to add the new layer." and select **Add Point Layer**. This adds a point layer to the map and opens the **Map Layer Data Properties** dialog.
6. In the **Map Layer Data Properties** dialog that appears, on the **General** page, import the spatial data (points) from a shape file or use a data field from the analytical data to specify the spatial data source.
7. In case you want to visualize analytical data on the map, go to the **Analytical data** page of the dialog, select your dataset from the **Dataset** dropdown list and click the **Add** (+) button located next to the **Match** field. This makes the **Spatial field** and **Analytical field** options active. See, Add Data to a Map for more information.
8. In the **Spatial field** and **Analytical field** options set data fields that contain similar data in both Spatial and Analytical databases. Match fields are used to relate the spatial data with the analytical data.

   > **Note**: It is necessary to set match fields if you want to use a data field from the analytical data to set spatial data for the layer, or if you want to visualize analytical data on the map.

9. Go to the **Filters** page and set filters if any.
10. Click **OK** to close the dialog and return to the design surface.
11. In the layers pane, right-click on **PointLayer1** and select **Edit** to open **Map Point Layer** dialog.



12. In the **General** page of the dialog, select any data field from the **Label Text** dropdown list to display as a label for each point that gets displayed on the map at run time. You can also set the label placement, size, and marker type.
13. Go to the **Marker Rule** page to set rules to visualize data using markers or keep it set to "Use default marker type".
14. Go to the **Size Rule** page to set rules to visualize data using different marker sizes or keep it set to "Use default marker size". See, Use Color Rule, Marker Rule, and Size Rule for more information.
15. Go to the **Color Rule** page to set rules to visualize data using color pallets, color ranges, or custom colors for markers that gets displayed on the map or keep it set to "Use Appearance settings".
16. Go to the **Visibility** page of the dialog and make sure the layer visibility is set to **Show**. You can also select options to show or hide layer based on any expression or zoom value.
17. In the **Navigation** page of the dialog, you can optionally link the layer to a URL, bookmark, or a report.
18. The **Appearance** page of the dialog either reflects the default appearance of the polygons or the color rule settings if any.
19. In the **Font** page of the dialog, you can optionally set the font family, size, weight, style, set, and color for the label text that you had set in the **General** page of the dialog.
20. Go to the **Data Output** page and specify the Data Element Name of the layer to be used while rendering to XML

and also specify whether the layer should be included in output while rendering or not.
21. Click **OK** to close the dialog and go to the preview tab to view the map.

Line Layer

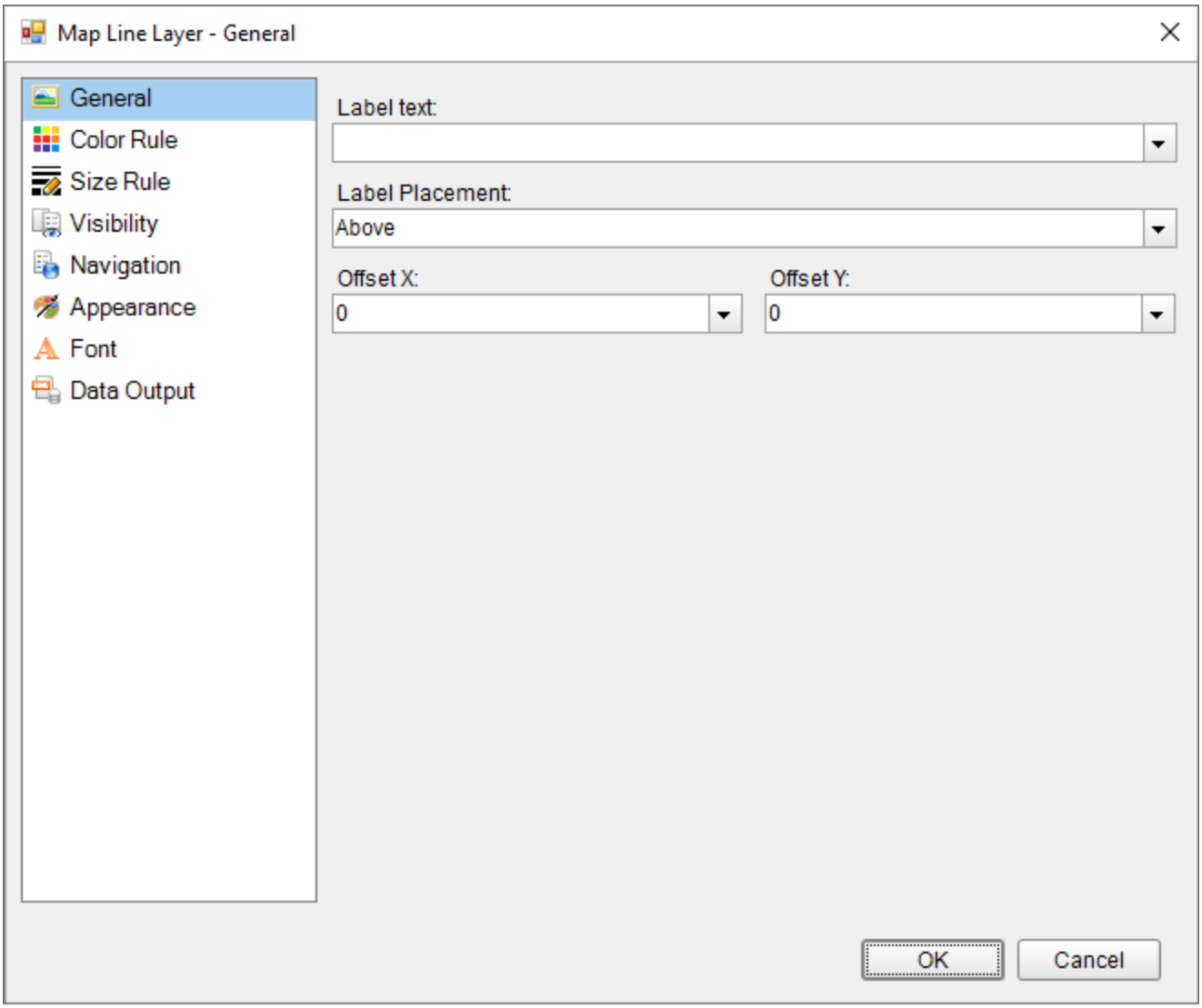
A Line layer displays routes and paths between different locations, for example, transportation route between two stores.

Use the following steps for creating a basic map using the Line layer. These steps assume that you have added a page layout template to your project and have a data connection in place. See Data Binding in Page/RDLX Reports for more information.

1. From the toolbox, drag and drop a Map control onto the design surface.
2. In the **Select a Map Template** wizard that appears, select the **New Map** template.
3. Click the map until the map panes appear.
4. Right-click inside the area labeled "Right click to add the new layer." and add either a Tile layer or a Polygon layer. One of these layers that you add serves as the geographic base to plot lines on. See, Polygon Layer or Use a Tile Layer for steps to add these layers on a Map control.
5. Right-click again inside the area labeled "right-click to add the new layer." and select **Add Line Layer**. This adds a Line layer to the map and opens the **Map Layer Data Properties** dialog.
6. In the **Map Layer Data Properties** dialog that appears, on the **General** page, import the spatial data (lines) from a shape file or use a data field from the analytical data to specify the spatial data source.
7. In case you want to visualize analytical data on the map, go to the **Analytical data** page of the dialog, select your dataset from the **Dataset** dropdown list and click the **Add** (+) button located next to the **Match** field. This makes the **Spatial field** and **Analytical field** options active. See, Add Data to a Map for more information.
8. In the **Spatial field** and **Analytical field** options set data fields that contain similar data in both Spatial and Analytical databases. Match fields are used to relate the spatial data with the analytical data.

   > **Note**: It is necessary to set match fields if you want to use a data field from the analytical data to set spatial data for the layer, or if you want to visualize analytical data on the map.

9. Go to the **Filters** page and set filters if any.
10. Click **OK** to close the dialog and return to the design surface.
11. In the layers pane, right-click on **LineLayer1** and select **Edit** to open **Map Line Layer** dialog.



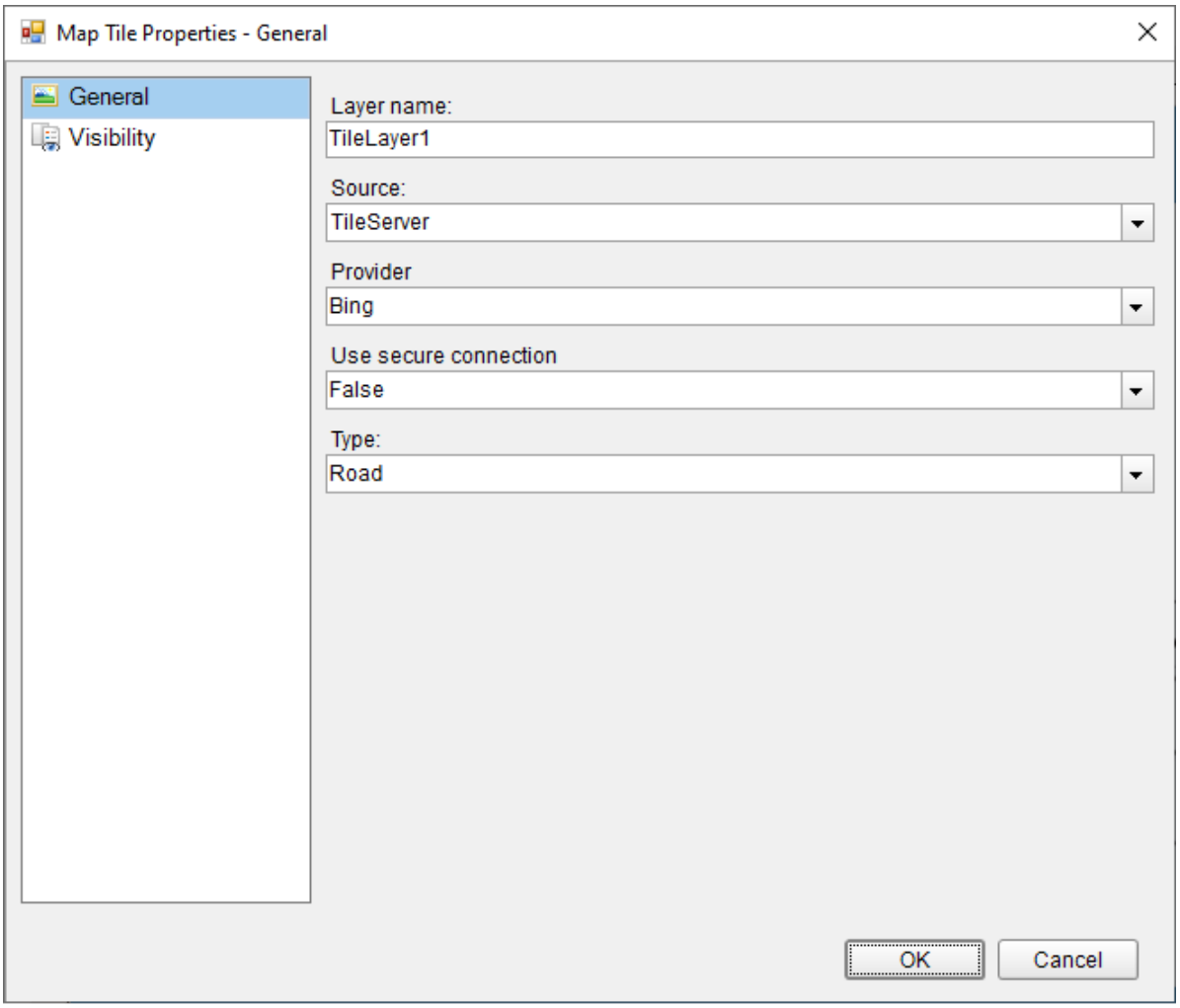
12. In the **General** page of the dialog, select any data field from the **Label Text** dropdown list to display as a label for each line that gets displayed on the map at run time. You can also set the label placement.
13. Go to the **Color Rule** page to set rules to visualize data using color pallets, color ranges, or custom colors for lines that get displayed on the map, or keep it set to "Use Appearance settings". See, Use Color Rule, Marker Rule, and Size Rule for more information.
14. Go to the **Size Rule** page to set rules to visualize data using line width or keep it set to the default line width.
15. Go to the **Visibility** page of the dialog and make sure the layer visibility is set to **Show**. You can also select options to show or hide a layer based on any expression or zoom value.
16. In the **Navigation** page of the dialog, you can optionally link the layer to a URL, bookmark, or a report.
17. Go to the **Appearance** page of the dialog and set the style, width, and color of the layer border and the layer background.
18. In the **Font** page of the dialog, you can optionally set the font family, size, weight, style, set, and color for the label text that you had set in the **General** page of the dialog.
19. Go to the **Data Output** page and specify the Data Element Name of the layer to be used while rendering to XML and also specify whether the layer should be included in output while rendering or not.
20. Click **OK** to close the dialog and go to the preview tab to view the map.

Tile Layer

A Tile layer displays the virtual earth tile background on the map.

Use the following steps for creating a basic map using the Tile layer. These steps assume that you have added a page layout template to your report and have a data connection in place. See Data Binding in Page/RDLX Reports for more information.

1. From the toolbox, drag and drop a Map control onto the design surface.
2. In the **Select a Map Template** wizard that appears, select the **New Map** template.
3. Click the map until the map panes appear.
4. Right-click inside the area labeled "Right click to add the new layer." and select **Add Tile Layer**. This adds a tile layer to the map and opens the **Map Tile Properties** dialog.



5. In the **Map Tile Properties** dialog that appears, on the **General** page, set the layer name and choose its Source and Type. The default values in these fields also work fine.
6. In the **Provider** property, choose one from the following supported tile providers:
   - **Bing**: The Microsoft Bing Map server offers static map images. This requires an Application key for authentication. The default key provided by ActiveReports is for demo purposes and can't be used by 3rd party applications. To obtain a Bing Map Key, see HowTo - Create a Bing Map Account and HowTo - Get a Bing Map Key.

     Note: After generating the key, add the following script in the ActiveReports.config file to configure embedded Bing tile provider with Application key.

Script

Paste inside the <Configuration> </Configuration> tags.

```
<MapTileServerConfiguration>
   <Timeout>5</Timeout>
   <AppID>"Your Application Key"</AppID>
</MapTileServerConfiguration>
```

⚠ **Caution:** The ActiveReports.config file should always be placed in the same folder as the EndUserDesigner.exe file for displaying a Bing tile layer on a Map.

- **Google**: The Google Map server creates your map tiles based on URL parameters sent through a standard HTTP request that returns the map tiles as an image. It is necessary to set the API key to monitor the usage of this tile server with the Google API Console. Please register with Google account with the billing information to obtain a key.

  Note: After obtaining the key, add the following script in the ActiveReports.config file to configure embedded Google tile provider with ApiKey.

  Script

  Paste inside the <Configuration> </Configuration> tags.

  ```
  <!-- Configure embedded Google tile provider with API Key -->
  <MapTileProvider Name="Google" DisplayName="Google" >
      <Settings>
        <add key="ApiKey" value="API Key" />
        <add key="Timeout" value="5000" />
      </Settings>
  </MapTileProvider>
  ```

- **MapQuest**: The MapQuest tile server provides the tiles in a format similar to Google. This tile server requires an API Key for authentication which can be obtained by registering at MapQuest.

  Note: After generating the key, add the following script in the ActiveReports.config file to configure embedded MapQuest tile provider with ApiKey.

  Script

  Paste inside the <Configuration> </Configuration> tags.

  ```
  <!-- Configure embedded MapQuest tile provider with ApiKey -->
  <MapTileProvider Name="MapQuest" DisplayName="Map Quest Tiles Provider">
      <Settings>
        <add key="ApiKey" value="API Key" />
        <add key="Timeout" value="3000" />
      </Settings>
  </MapTileProvider>
  ```

- **OpenStreetMap**: The OpenStreetMap server provides the tiles in an index based format. This tile server provides only the roadmap and returns fixed size images (256x256). Before using the OpenStreetMap server, go through the Copyright and License and Tile usage policy pages.

7. Go to the **Visibility** page of the dialog and make sure the layer visibility is set to **Show**. You can also select options to show or hide a layer based on any expression or zoom value.
8. Click **OK** to close the dialog and go to the preview tab to view the map.

Note: If you are using a proxy server connection to see the map tile images, you need to set credentials for the proxy server in

the application config file for authentication. To use your default proxy server credentials, you can do the following:

For Visual Studio Designer (IDE)

1. Find **devenv.exe.config** (the devenv.exe configuration file) in: Program Files (x86)\Microsoft Visual Studio\2017\Professional\Common7\IDE
2. In the configuration file, find the <system.net> block, and add this code:

Paste inside the <system.net></system.net> tags.

```
<defaultProxy useDefaultCredentials="true" />
```

For Visual Studio Application

1. On the menu bar, choose **Project, Add New Item** and then choose the **Application Configuration File** template.
2. In the **Name** text box, enter a name, and then click **Add** button.
3. In the configuration file, add this code:

Paste inside the <configuration></configuration> tags.

```
<system.net>
<defaultProxy useDefaultCredentials="true" />
</system.net>
```

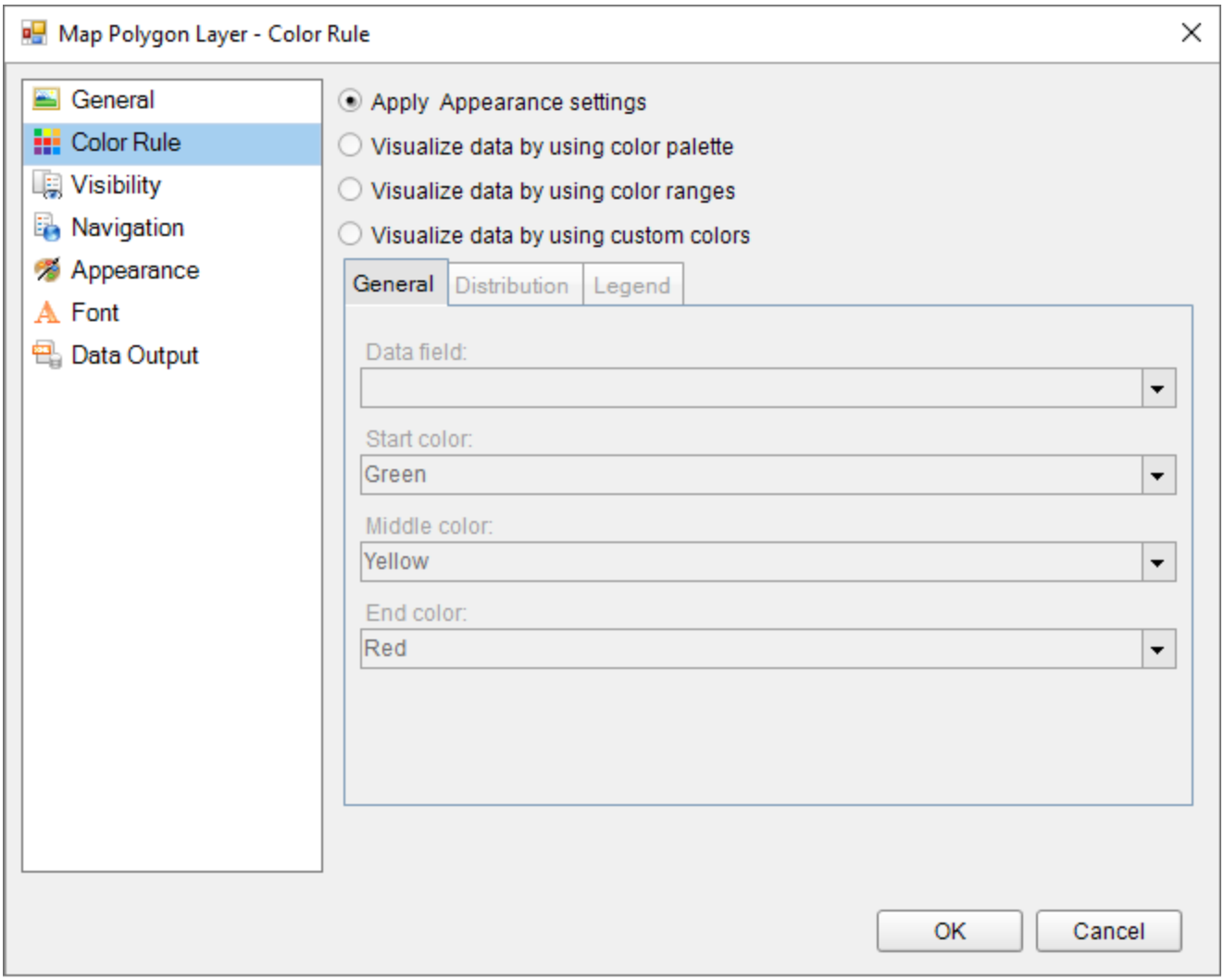
Note: ActiveReports.config file should be kept inside the project **Debug** folder and added to a Visual Studio project for displaying a Tile layer on a Map in any Viewer control.

Use Color Rule, Marker Rule, and Size Rule

You can visualize the data displayed on a map by setting rules to control color, size or marker type for all map elements on a layer. You can set three types of rules depending on the type of layer in use.

Color Rule

Color Rule is set to fill colors for map elements like polygons, markers (points or polygon center points), and lines while using a Polygon, Point, or Line layer. Color Rule provides four options:



- **Apply Appearance Settings**: Use the default appearance settings that are set in the Appearance page of the map layer dialog.
- **Visualize data by using color palette**: This option uses an in-built palette that you specify. Based on related analytical data, each map element is assigned a different color from the palette.
- **Visualize data by using color ranges**: This option, combined with the start, middle, and end colors that you specify on this page and the options that you specify on the Distribution page, divides the related analytical data into ranges. The report then assigns the appropriate color to each map element based on its associated data and the range that it falls into. For example, in a map that uses color to display temperatures on a scale of 0 to 100, low values are blue to represent cold and high values are red to represent hot.
- **Visualize data by using custom colors**: This option uses the list of custom colors that you specify. Based on related analytical data, each map element is assigned a color from the list.

Set Color Rule for polygons, lines, and markers

Using color palette

1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.

3. In the selected map layer dialog that appears, go to the **Color Rule** page.
4. In the Color Rule page, select the **Visualize data by using color palette** option.
5. In **Data field**, set the name of the field or expression that contains the analytical data that you want to visualize by color.
6. In **Palette**, set the name of the palette to use.
7. Click **OK** to close the dialog.

Using color ranges

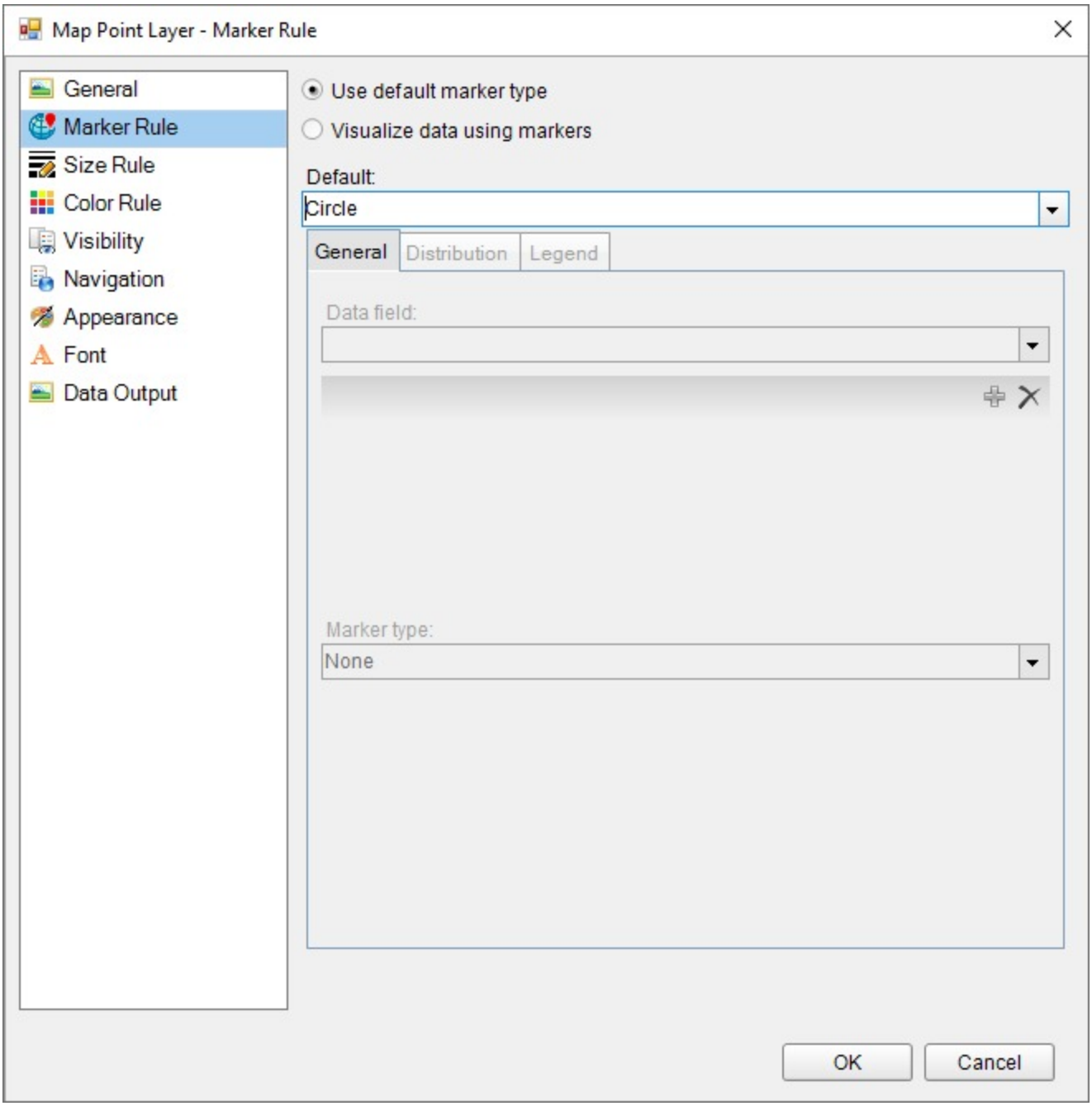
1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the **Color Rule** page.
4. In the Color Rule page, select the **Visualize data by using color ranges** option.
5. In **Data field**, set the name of the field or expression that contains the analytical data that you want to visualize by color.
6. In **Start color**, set the color to be used for the color range.
7. In **Middle color**, set the color to be used for the color range.
8. In **End color**, set the color to be used for the color range.
9. Click **OK** to close the dialog.

Using custom colors

1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the **Color Rule** page.
4. In the Color Rule page, select the **Visualize data by using custom colors** option.
5. In **Data field**, set the name of the field that contains the analytical data that you want to visualize by color.
6. Click **Add** to specify each custom color.
7. Click **OK** to close the dialog.

Marker Rule

Marker Rule is set on markers that represent points or polygon center points on a map while using a Point layer. Marker Rule support two options:



1. **Use a default marker type**: You specify one of the available marker types.
2. **Visualize data using markers**: This option uses a set of markers in the order in which you want them to be used. Marker types include Rectangle, Circle, Diamond, Triangle, Trapezoid, Star, Wedge, Pentagon, PushPin and Image.

Set Marker Rule for points

Visualize points using default marker type

1. Click the Map until the map panes appear.

2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the **Marker Rule** page.
4. In the Marker Rule page, select the **Visualize data using markers** option.
5. In **Data field**, set the name of the field that contains the analytical data that you want to visualize using different marker types.
6. Click **Add** and specify each **Marker type** in the order in which you want them to be used.
7. Click **OK** to close the dialog.

Visualize points using specific marker types

1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the **Marker Rule** page.
4. In the Marker Rule page, select the **Visualize data using markers** option.
5. In **Data field**, set the name of the field that contains the analytical data that you want to visualize using different marker types.
6. Click **Add** and specify each **Marker type** in the order in which you want them to be used.
7. Click **OK** to close the dialog.

Visualize points using Image as marker type

ActiveReports provides **Image** as one of the many available marker types to use from. You can set this marker type and use any image as a marker on a map layer. Like other marker types, you can either use it as a default marker or use it as one of the member in the markers collection.

Use Image as Default marker type

1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, on the **General** page, set **Marker Type** to **Image**. A new set of properties appears on the page.
4. In **Image Source**, choose the source of image from the provided options:
    - **External**: Select this option and set a path or url of the image file in **Image Value**.
    - **Embedded**: Choose from the list of embedded images added to your report. Once you set this option, the **Image Value** provides you the list of embedded images to choose from.
    - **Database**: Select this option and set the data field containing the image in the **Image Value** property.
5. In **MIME Type**, set the MIME type of the image chosen. In case you are using the Embedded image source the MIME Type gets set automatically as you select the image in the Image Value property.
6. Set the **Transparent Color** and the **Resize Mode**.
7. Click **OK** to close the dialog.

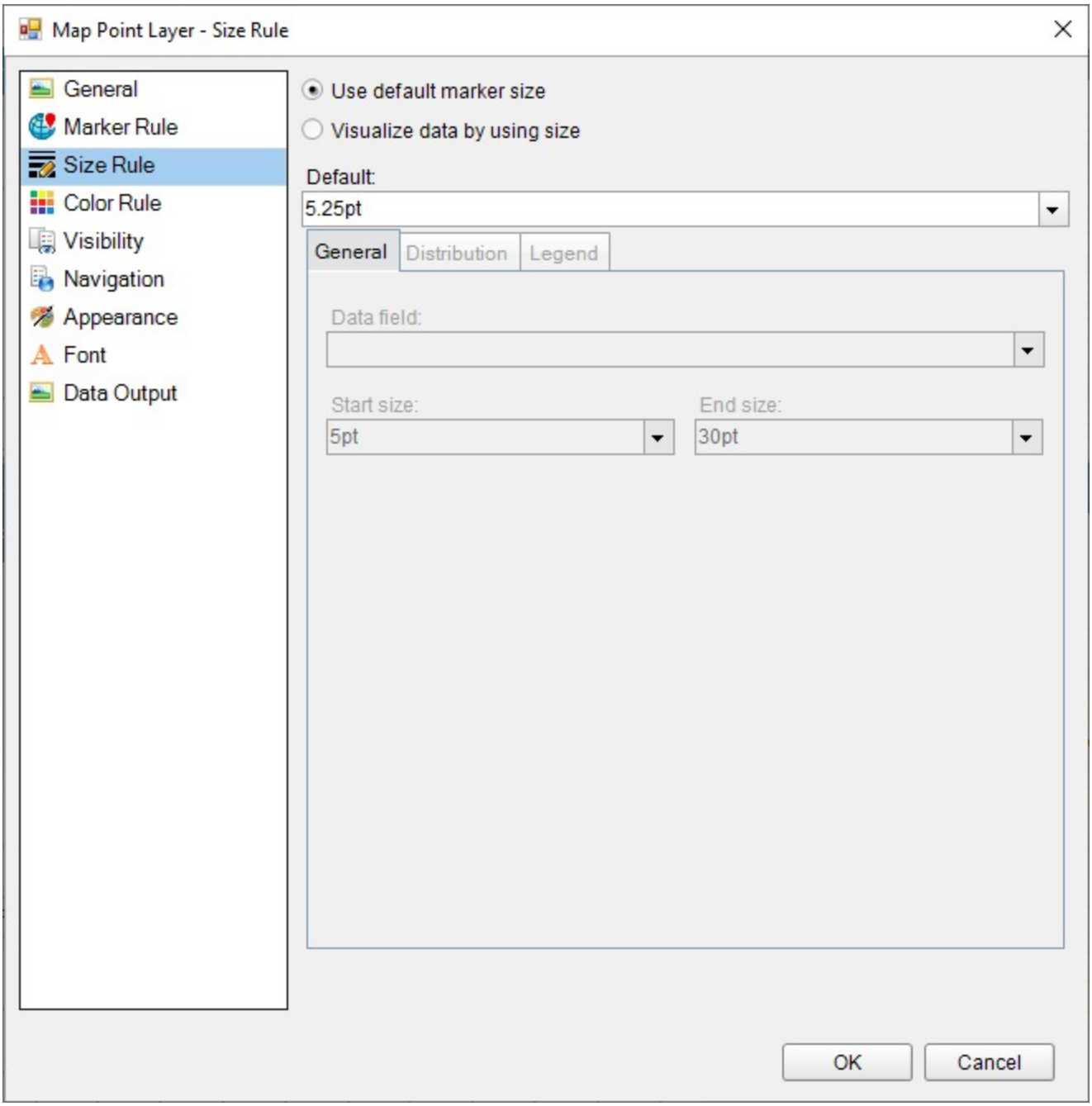
Use Image in markers collection

- Click the Map until the map panes appear.
- In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
- In the selected map layer dialog that appears, go to the **Marker Rule** page.
- In the Marker Rule page, select the **Visualize data using markers** option.
- In **Data field**, set the name of the field that contains the analytical data that you want to visualize using different marker types.

- Click **Add** and set **Marker type** to **Image**. A new set of properties for image marker types appears on the page.
- In **Image Source**, choose the source of image from the provided options:
  - **External**: Select this option and set a path or url of the image file in **Image Value**.
  - **Embedded**: Choose from the list of embedded images added to your report. Once you set this option, the **Image Value** provides you the list of embedded images to choose from.
  - **Database**: Select this option and set the data field containing the image in the **Image Value** property.
- In **MIME Type**, set the MIME type of the image chosen. In case you are using the Embedded image source the MIME Type gets set automatically as you select the image in the Image Value property.
- Set the **Transparent Color** and the **Resize Mode**.
- Click **OK** to close the dialog.

Size Rule

Size Rule is set on markers, polygon center points or line width while using a Polygon, Point or a Line layer. Size Rule support two options:



1. **Use a default marker size** or **Use a default line width**: You specify the marker size or line width in points.
2. **Visualize data by using size** or **Visualize data by using line width**: In this option, you set the minimum (start) and maximum (end) sizes or width for marker or line, specify the data field to be used for varying the marker size or line width and then specify the distribution options to apply to that data.

Set Size Rule for markers and line width

Visualize marker or line using default marker size or line width

1. Click the Map until the map panes appear.

2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the **Size Rule** page.
4. In **Default**, set default size or width for each marker or line that appears on a map.
5. Click **OK** to close the dialog.

Visualize marker or line using specific marker size or line width

1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the **Size Rule** page.
4. In the Size Rule page, select **Visualize data by using size** or **Visualize data by using line width** depending on the layer type in use.
5. In **Data field**, set the name of the field that contains the analytical data that you want to visualize using different marker sizes or line width.
6. Set S**tart size** and **End size** in case of Point Layer or **Minimum line width** and **Maximum line width** in case of Line Layer.
7. Click **OK** to close the dialog.

Distribution Options

The distribution values are used by the rules to differ the map element display values.

To create a distribution of values, you divide your data into ranges by specifying the distribution method, the number of sub-ranges, and the range start and end values.

To set distribution values for rules, follow these steps:

1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the rule page(Color Rule, Marker Rule or Size Rule) where you need to specify distribution values.
4. In the rule page of the dialog, select any option to visualize data using the selected rule type and go to the **Distribution** tab.
5. On the Distribution tab, select one of the following distribution types:
    - **EqualInterval**: Create ranges that divide the data into equal range intervals. For the example, the three ranges would be 0-2999, 3000-5999, 6000-8999. Sub-range 1: 1, 10, 200, 500. Sub-range 2: 4777. Sub-range 3: 8999.
    - **EqualDistribution**: Create ranges that divide that data so that each range has an equal number of items. For example, the three ranges would be 0-10, 11-500, 501-8999. Sub-range 1: 1, 10. Sub-range 2: 200, 500. Sub-range 3: 4777, 8999.
    - **Optimal**: Specifies ranges that automatically adjust distribution to create balanced sub-ranges. The number of Sub-ranges is determined by the algorithm.
    - **Custom**: Specify your own number of ranges to control the distribution of values. For example, you can specify your own custom ranges 0-5000 and 5001-10000.
6. In **Number of Sub-ranges**, type the number of sub-ranges to use.
7. In **Range start**, type a minimum range value. All values less than this number are the same as the range minimum.
8. In **Range end**, type a maximum range value. All values larger than this number are the same as the range maximum.
9. Click **OK** to close the dialog.

Displaying Rule results in Legend

To display rule results in legend, follow these steps:

1. Click the Map until the map panes appear.
2. In the layers pane, right-click on the added map layer and select **Edit** to open the selected map layer dialog.
3. In the selected map layer dialog that appears, go to the rule page(Color Rule, Marker Rule or Size Rule) where you need to specify displaying rule results in a legend.
4. In the rule page of the dialog, select any option to visualize data using the selected rule type and go to the **Legend** tab.
5. Select **Show in legend** checkbox and set **Legend name.**
6. In **Legend text**, enter text that specify which data should appear in the legend. Use map keywords and custom formats to help control the format of legend text. For example, #VALUE {C2} specifies a currency format with two decimal places. Following are the supported formats that you can use:

Format	Description	Example
#Value	Displays a numeric value calculated using "**(EndRangeValue - StartRangeValue)/2**" formula.	
#FROMVALUE {C0}	Displays the currency of the total value with no decimal places.	$100
#FROMVALUE {C2}	Displays the currency of the total value to two decimal places.	$40.25
#TOVALUE	Displays the actual numeric value of the data field.	100
#FROMVALUE{N0} - #TOVALUE{N0}	Displays the actual numeric values of the beginning of the range and end of the range.	10 - 500

7. Click **OK** to close the dialog.

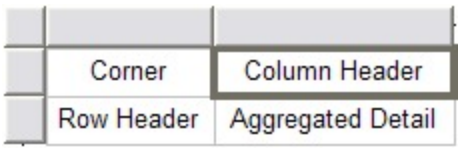
Matrix

> **Note:**
>
> - Matrix is an old-style tabular control that has been replaced with a more powerful Tablix data region.
> - Matrix is hidden by default and not supported in the WebDesigner, where it is automatically converted to Tablix.

The Matrix data region contains columns and rows in which the data is inserted and arranged. Columns and rows in a matrix can be dynamic or static. Each cell in a matrix contains a textbox by default, but as with the other data regions, the textbox can be replaced with any other report control. Also like other data regions, the matrix also repeats each report control it contains for each row in the dataset, but unlike the others, it also repeats horizontally for dynamic columns.

Cells

When you drop a matrix data region onto your report, it is initially composed of four cells.



The top left cell is the corner or label cell. You can leave this cell blank or use it to display a label for the matrix.

The top right cell is a column header and the bottom left cell is a row header. You can drag fields into these cells or use expressions to group the data.

The bottom right cell is used to aggregate the detail data. At run time, detail cells display aggregates of the intersections of columns and rows.

Data

The matrix data region is associated with a dataset. You can make this association in the **DataSetName** property in the Properties grid.

Groups

You can group the columns and rows in the matrix by opening the Property dialog, and on the **Row Groups** or **Column Groups** page, editing the **Group on** expression, or by dragging a field into the header cell.

You can nest column and row groups by right-clicking the UI above a column or to the left of row and choosing **Add Column Group** or **Add Row Group**, or simply by dragging another field onto the column or row into which you want to insert it.

To determine where new groups are displayed, when you drag a field into an existing group you can move the mouse up or down (for columns) or left or right (for rows) to nest the new group within the existing group, or to nest the existing group within the new group.

At run time, the nested group repeats within the original group. For instance, when you drag the City field from your dataset onto a header already grouped by Country, the report displays the first country with all of its cities, then the next country with all of its cities, and so on until all of the countries in the dataset are displayed with all of their cities.

Static Rows and Columns

To insert a static row or column, right-click the detail cell and choose **Add Column** or **Add Row**. This displays a column next to or a row above or below the existing one and adds another detail cell.

Aggregates

Groups in a matrix do not subtotal data by default. To add this functionality, right-click the group header and choose **Subtotal**. This adds a cell to the right of the column, or below the row. Subtotal cells have a green triangle in the upper right corner which allows you to select the subtotal and set its properties in the property grid. For more information, see Expressions.

Placement of Data

You can control where data is placed in several ways. You can change the **Direction** property of the matrix to **RTL** to cause dynamic column headers to expand left instead of right.

You can also move row headers to the right of the detail cells by changing the **GroupsBeforeRowHeaders** property of the matrix. The integer value you supply for this property equals the number of instances of the outermost column group that displays to the left of the row headers (or to the left if the **Direction** property is set to **RTL**).

Matrix Dialog

Properties for the Matrix are available in the Matrix dialog. To open it, with the Matrix control selected on the report, under the Properties Panel, click the **Property dialog** link.

The Matrix dialog lets you set properties on the report control with the following pages.

> **Note**: You can select **<Expression...>** within many of these properties to create an expression to determine the value.

General

Name: Enter a name for the matrix that is unique within the report. This name can be called in code.

Tooltip: Enter the value or expression you want to appear when a user hovers the cursor over the matrix in the viewer at run time.

Dataset name: Select a dataset to associate with the matrix. The combo box is populated with all of the datasets in the report's dataset collection.

Has own page numbering: Select to indicate whether this Matrix is in its own section with regards to pagination.

Page breaks: Select any of the following options to apply to each instance of the matrix.

- Insert a page break before this matrix
- Insert a page break after this matrix
- Fit matrix on a single page if possible

Matrix column expand: Select the direction in which columns expand.

- Left to right
- Right to left

Groups before row headers: Select the number of columns to show before the row header columns begin. The default value of **0** displays the row header column to the left.

Visibility

Initial visibility

- **Visible**: The matrix is visible when the report runs.
- **Hidden**: The matrix is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the matrix is visible. True for hidden, false for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down list where you can select the report control that users can click to show or hide this matrix.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this matrix. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Filters

Click the plus sign button to add a filter to the matrix. Use the arrow and X buttons to move or delete filters. You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the matrix.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site .
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Row Groups

The Row Groups page of the Matrix dialog allows you to add, remove, or change the order of row groups using the plus sign, X and arrow buttons.

Click the **Add** button to add a new row group to the list and set up information for each group on the following tabs.

General

Name: Enter a name for the group that is unique within the report. This property cannot be set until after a Group on expression is supplied.

Group on: Enter an expression to use for grouping the data.

Label: Enter an expression to identify an instance of the group for document map and search functions.

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

In the **Expression** box, enter an expression by which to sort the data in the group, and under **Direction**, select **Ascending** or **Descending** for the selected sort expression.

Visibility

Initial visibility

- **Visible**: The group is visible when the report runs.
- **Hidden**: The group is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the group is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down list where you can select the report control that users can click to show or hide this group.

Data Output

Element name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose **Yes** or **No** to decide whether to include this group in the XML output.

Layout

Page break at start: Inserts a page break before the group.

Page break at end: Inserts a page break after the group.

Has own page numbering: Used in conjunction with the "Page Number in Section" and "Total Pages in Section" properties, tells the report that the group constitutes a new page numbering section.

Column Groups

The Column Groups page of the Matrix dialog allows you to add, remove, or change the order of row groups using the plus sign, X and arrow buttons.

Click the **Add** button to add a new column group to the list, and set up information for each group on the following tabs.

General

Name: Enter a name for the group that is unique within the report. This property cannot be set until after a Group on expression is supplied.

Group on: Enter an expression to use for grouping the data.

Label: Enter an expression to identify an instance of the group for document map and search functions.

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.

- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

In the **Expression** box, enter an expression by which to sort the data in the group, and under **Direction**, select **Ascending** or **Descending** for the selected sort expression.

Visibility

Initial visibility

- **Visible**: The group is visible when the report runs.
- **Hidden**: The group is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the group is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. The user can click the toggle item to show or hide this band group. This enables the drop-down list where you can select the report control that users can click to show or hide this group.

Data Output

Element name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose **Yes** or **No** to decide whether to include this group in the XML output.

Data Output

Element name: Enter a name to be used in the XML output for this matrix.

Output: Choose Auto, Yes, or No to decide whether to include this matrix in the XML output. Choosing Auto exports the contents of the matrix.

Cell element name: Enter a name to be used in the XML output for the data element for the cell.

Cell element output: Choose Yes or No to decide whether to include the cell contents in the XML output.

Overflow Placeholder (Page report only)

The Overflow Placeholder control (available only in a Page report) is a rectangular placeholder for data that does not fit inside the fixed size of a List, Banded List, Tablix, or Table data region. In this case, you can link a data region to an Overflow Placeholder; the control gets its **Size** property values from the **FixedSize** of the data region it is linked with. So, data that overflows the fixed size of your tables or other data regions can span pages, but you control the layout of each page and specify where the overflow data goes with a placeholder.

You can also place multiple Overflow Placeholder controls in a report to create different looks for your data output. With multiple Overflow Placeholder controls, you should link a data region to an Overflow Placeholder control and then link that Overflow Placeholder control to another Overflow Placeholder control. Two common layouts that you can create are:

- **Columnar Report Layout**: Place the data region and the Overflow Placeholder on the same page of the report to create a layout that displays data in a columnar format.
- **Multiple Page Layout**: Place the data region on the first page of the report and Overflow Placeholder controls on subsequent pages to create a layout with overflow data on multiple pages.

> **Note:** If a page contains only an OverflowPlaceHolder with no data to display, the empty page does not render. However, if the page also contains any control with static data, the page renders.
>
> To skip rendering the page, set the **ThrowIfPlaceHoldersEmpty** property of the report's Page instance to True.

In RDLX reports, a similar report is can be created by setting the number of columns in the report's Columns property.

Shape

The Shape report control is used to display one of the available shape types on a report. You can add a shape report control to a report by dragging it from the toolbox and dropping it onto the report design surface.

In the **ShapeStyle** property of the Shape report control, you can select **Rectangle**, **RoundRect** or **Ellipse**, or you can use an expression to assign fields, datasets, parameters, constants, operations or common values to it.

You can highlight different sections or parts of a report using a shape report control. For example, you can use a Rectangle as border around different report controls or the entire page or you can use an Ellipse to highlight a note on your report.

Important Properties

By clicking on the Shape control, you can set its properties in the Properties window.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
BackgroundColor	Choose a color for the background from the Color Picker.
BackgroundImage	Add a background image from External, Embedded, or Database as Source. You can also select the MIME type of the image.
BorderColor	Choose a color for the border from the Color Picker.
BorderStyle	Select a style for the border.
BorderWidth	Enter a value to set the width of the border.
RoundingRadius	Enter a value in points to round the corner of the outer border edge of the RoundRect shape style.
ShapeStyle	Choose a shape style from Rectangle, RoundRect, or Ellipse.
Size	Enter the Width and Height of the shape.

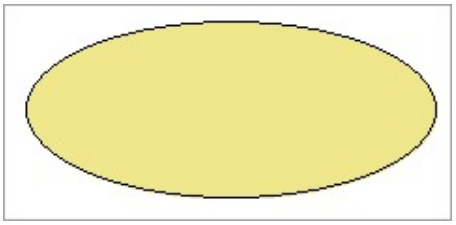
Shape Dialog Properties

You can set the Shape properties in the Shape dialog. To open it, with the Shape selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the image that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Shape Style: Choose **Rectangle**, **RoundRect** or **Ellipse** from the dropdown list, or select the **<Expression...>** option to open the Expression Editor.

Rectangle	RoundRect	Ellipse
		

Rounded Rectangle: When the Shape type is set to **RoundRect**, you can specify the radius for each corner of the shape independently. Drag the handlers ◇ available at each corner of the shape to set the value of the radius at each corner.

> **Note:** To enable specific corners, select the checkbox available near each corner of the Shape control.

Appearance

Background

Color: Select a color to use for the background of the Shape, or select the **<Expression...>** option to open the Expression Editor.

Image: Specify the background image of shape using Expression or Data Visualizer, or directly open the image file on your system.

> **Note**: If the Hatch and Gradient background styles are set using Data Visualizers, these are not displayed at design time.

Image Source: Select the location of the image for the background from External, Embedded, Database or select the **<Expression...>** option to open the Expression Editor.

MIME Type: Select the MIME type of the image.

Background repeat: Specify how the background image fills the space of the shape.

Border

Style: Select a style for the border.

Width: Enter a value in points to set the width of the border or select the **<Expression...>** option to open the Expression Editor.

Color: Select a color to use for the border, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Visibility

Initial visibility

- **Visible**: The shape is visible when the report runs.
- **Hidden**: The shape is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the shape is visible. True for hidden, false for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle shape next to another report item. This enables the drop-down box where you can specify the TextBox control which, if clicked, toggles the visibility of the shape.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this shape. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

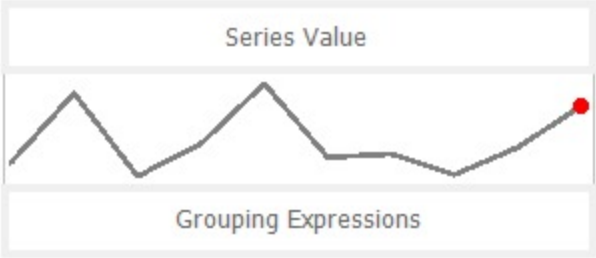
Data Output

Element Name: Enter a name to be used in the XML output for this shape report control.

Output: Choose **Auto**, **Yes**, **No,** or **Contents only** to decide whether to include this Shape in the XML output. Choosing **Auto** exports the contents of the Shape report control.

Sparkline

You can use the Sparkline report control as a simple means of displaying the trend of data in a small graph. The Sparkline displays the most recent value as the rightmost data point and compares it with earlier values on a scale, allowing you to view general changes in data over time. With the height similar to the surrounding text and the width not more than 14 letters wide, the sparkline fits well in dashboards, reports, and other documents.



Important Properties

By clicking on the Sparkline control, you can set its properties in the Properties window.

Property	Description
AccessibleDescription	Enter the alternative description of the control for use by accessibility client application. The property on exporting the report adds the 'alternative text' in PDF and 'alt' attribute in HTML.
DataSetName	Specify a dataset associated with the sparkline.
DataSetParameters	Specify a dataset parameter associated with the sparkline.
FillStyle	Choose the type of gradient for the sparkline from None, LeftRight, TopBottom, Center, DiagonalLeft, DiagonalRight, HorizontalCenter or VerticalCenter. You can also select a color to fill the sparkline from the color picker.
Filters	Add a new filter by specifying the Expression, Operator, and Value.
LineStyle	Enter a value to set the Width of the Line type sparkline. You can also choose the color of the line from the color picker.
Location	Enter a value in points to set the location of the sparkline.
MarkerColor	Select a color for the last point marker or select the Expression option to open the Expression Editor and create an expression that evaluates

	to a .NET color.
MarkerVisibility	Select whether to display a marker at the last point on the sparkline of Line type.
MaximumColumnWidth	Select the maximum width of columns in the sparkline.
Range	Select a value or enter an expression that defines the lower bound and upper bound of the wall range. By using the backdrop, you can choose the type of gradient and color of the wall range.
RangeVisibility	Select whether to display the range or not.
SeriesValue	Enter a series value of the sparkline.
Size	Enter the Width and Height of the sparkline.
SparklineType	Choose the sparkline type from Line, Columns, Whiskers, Area, or StackedBar.

Sparkline Dialog Properties

You can set the Sparkline properties in the Sparkline dialog. To open it, with the Sparkline selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the sparkline that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), back slash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Dataset Name:

Data

Value: Enter an expression to use as the sparkline value.

Name: Enter a name for the sparkline that is unique within the report. This name can be called in code. A name is created automatically if you do not enter one.

Group on: Enter an expression to use for grouping the data. If you open the expression editor, you can select a field from the dataset.

Detail Grouping: Enter an expression to use if you do not want to repeat values within the details. If you open the expression editor, you can select a field from the dataset.

Parent Group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Appearance

Sparkline Type: Choose from the following sparkline types. Each of these types has its own set of Appearance properties that appears when you select the type. See **Types of Sparklines** for more details.

- **Line**
- **Columns**
- **Whiskers**
- **Area**
- **StackedBar**

Visibility

Initial visibility

- **Visible:** The sparkline is visible when the report runs.
- **Hidden:** The sparkline is hidden when the report runs.
- **Expression:** Use an expression with a Boolean result to decide whether the sparkline is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control which, if clicked, toggles the visibility of the sparkline in the Viewer.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this sparkline. You will then be able to provide a bookmark link to this item from another report item using a **Jump to bookmark** action.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

Expression: Enter an expression by which to sort the data in the group.

Direction: Select **Ascending** or **Descending** for the selected sort expression.

Data Output

Element Name: Enter a name to be used in the XML output for this sparkline report control.

Output: Choose **Auto**, **Yes**, **No, Contents Only** to decide whether to include this Sparkline in the XML output. Choosing **Auto** exports the contents of the Sparkline report control.

Types of Sparklines

Line

The Line sparkline is widely used in financial and economic data analysis and is based on a continuous flow of data. The currency exchange rates, price change are examples of the application of this type of sparklines.

Appearance Properties

Property	Description
Last point marker is visible	Select to display a marker at the last point on the sparkline.
Marker Color	Select a color to use for the last point marker, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Line Style	
Color	Select a color to use for the line, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Width	Enter a value in points to set the width of the line.
Enable Wall Range	Select this check box to display a wall range for the sparkline. Selecting this box enables the rest of the properties in this section.
Lower Bound	Select a value or enter an expression that defines the lower bound of the wall range.
Upper Bound	Select a value or enter an expression that defines the upper bound of the wall range.
Wall Range Backdrop	
Fill Color	Select a color to use for the wall range, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Gradient	Choose the type of gradient to use for the backdrop: **None**, **LeftRight**, **TopBottom**, **Center**, **DiagonalLeft**, **DiagonalRight**, **HorizontalCenter**, or **VerticalCenter**.
Gradient End Color	Select a color to use for the end of the wall range gradient, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.

Columns

The Column sparkline is used for sports scores, cash register receipts, and other cases where previous values and the current value do not closely influence one another. This sparkline is used when dealing with discrete data points, and not a continuous flow of data as in the Line sparkline.

Appearance Properties

Property	Description
Fill Color	Select a color to use for the fill of the sparkline, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Maximum Column Width	Select the maximum width of columns in the sparkline. If blank, all columns are sized to fit.
Enable Wall Range	Select this check box to display a wall range for the sparkline. Selecting this box enables the rest of the properties in this section.
Lower Bound	Select a value or enter an expression that defines the lower bound of the wall range.
Upper Bound	Select a value or enter an expression that defines the upper bound of the wall range.
Wall Range Backdrop	
Fill Color	Select a color to use for the wall range, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Gradient	Choose the type of gradient from these choices: **None**, **LeftRight**, **TopBottom**, **Center**, **DiagonalLeft**, **DiagonalRight**, **HorizontalCenter**, or **VerticalCenter**.
Gradient End Color	Select a color to use for the end of the wall range gradient, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.

Whiskers

The Whisker sparkline is typically used in win/loss/tie or true/false scenarios. This type is similar to the Column sparkline, but it renders a tie (0 value) in a different manner. The bars in a whisker sparkline render below the baseline for a negative value, above the baseline for a positive value, and on the baseline for a zero value, for example, in a profit/loss and no profit/no loss scenario.

Appearance Properties

Property	Description
Fill Color	Select a color to use for the fill of the sparkline, or select the <**Expression...**> option to open the

	Expression Editor and create an expression that evaluates to a .NET color.
Maximum Column Width	Select the maximum width of columns in the sparkline. If blank, all columns are sized to fit.
Enable Wall Range	Select this check box to display a wall range for the sparkline. Selecting this box enables the rest of the properties in this section.
Lower Bound	Select a value or enter an expression that defines the lower bound of the wall range.
Upper Bound	Select a value or enter an expression that defines the upper bound of the wall range.
Wall Range Backdrop	
Fill Color	Select a color to use for the wall range, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Gradient	Choose the type of gradient from these choices: **None**, **LeftRight**, **TopBottom**, **Center**, **DiagonalLeft**, **DiagonalRight**, **HorizontalCenter**, or **VerticalCenter**.
Gradient End Color	Select a color to use for the end of the wall range gradient, or select the <**Expression...**> option to open the Expression Editor and create

Area

The Area sparkline is similar to the Line sparkline but visually you see the space under the line as shaded.

Appearance Properties

Property	Description
Fill Color	Select a color to use for the fill of the sparkline, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Enable Wall Range	Select this check box to display a wall range for the sparkline. Selecting this box enables the rest of the properties in this section.
Lower Bound	Select a value or enter an expression that defines the lower bound of the wall range.
Upper Bound	Select a value or enter an expression that defines the upper bound of the wall range.
Wall Range Backdrop	
Fill Color	Select a color to use for the wall range, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.
Gradient	Choose the type of gradient from these choices: **None**, **LeftRight**, **TopBottom**, **Center**, **DiagonalLeft**, **DiagonalRight**, **HorizontalCenter**, or **VerticalCenter**.
Gradient	Select a color to use for the end of the wall range gradient, or select the <**Expression...**> option to open

End Color	the Expression Editor and create an expression

StackedBar

The Stacked Bar sparkline is presented as a horizontal bar with different segment lengths marked by distinct color hues. The Stacked bar illustrates how the various segments of a part-to-whole relationship correspond to one another - the largest segment represents the highest value and the change in brightness indicates a new value on a scale.

Appearance Properties

Property	Description
Fill Color	Select a color to use for the fill of the sparkline, or select the <**Expression...**> option to open the Expression Editor and create an expression that evaluates to a .NET color.

Subreport

The Subreport control is a placeholder for data from a separate report. In ActiveReports, for better performance, we recommend using data regions instead of Subreport controls wherever possible. The reason is that the report server must process every instance of each subreport, which can become burdensome in very large reports with a large number of subreports processed many times per report. Using data regions to display separate groups of data can be much more efficient in such reports. For more information, see Report Controls.

You can also pass parameters to the subreport from the main report so that data related to the main report displays in each instance of the subreport.

> **Note**:
>
> - A Page report can use an RDLX report as the target subreport.
> - You cannot use a Section report as the target of a Subreport in a Page/RDLX report.

Subreports make sense when you need to nest groups of data from different data sources within a single data region, or when you can reuse a subreport in a number of reports. Here are some things to keep in mind while designing subreports.

- If you make changes to the subreport without changing the main report, click **Refresh** to re-run the subreport in the Preview tab.
- If you design the parent report in a standalone Report Designer application, save the parent report to the same directory as the subreport to render it in the Preview tab.
- If you set borders on the Subreport control and the body of the report hosted in it, ActiveReports does not merge the two borders.
- If the report hosted in the Subreport control cannot be found or contains no rows, only the border of the Subreport control is rendered.
- If the hosted report is found and does contain rows, the border of the hosted report body is rendered.

Subreport Dialog Properties

With the control selected on the report, in the Commands section at the bottom of the Properties window, you can click the **Property dialog** command to open the dialog.

General

Name: Enter a name for the subreport that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), back slash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Subreport: Select the **<From File...>** option to open the **Open** dialog, and then select a report to display within the Subreport control.

Use this report's theme when rendering subreport: Select this check box to have the subreport automatically use the same theme as the hosting report.

Use this report's stylesheet for the subreport: Select this checkbox to have the subreport use the same stylesheet as the parent report.

Visibility

Initial visibility

- **Visible**: The subreport is visible when the report runs.
- **Hidden**: The subreport is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the subreport is visible. True for hidden, false for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box where you can specify the TextBox control which, if clicked, toggles the visibility of the subreport.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this subreport. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Parameters

The Parameters page of the Subreport dialog allows you to enter new parameters and remove or change the order of parameters using the X and arrow buttons. For each parameter in this list, there is a **Parameter Name** and a **Parameter Value**.

Each **Parameter Name** must exactly match the name of a parameter in the target report.

For the **Parameter Value**, enter an expression to use to send information from the summary or main report to the subreport target.

Note: The following metods are not supported in the **Parameter Value** expression for a parameter that passes a value from the main report to subreport. However, this restriction does not apply to parameter default values.

- RowNumber
- RunningValue
- Lookup/LookupSet
- Previous
- CountRows
- CumulativeTotal

Data Output

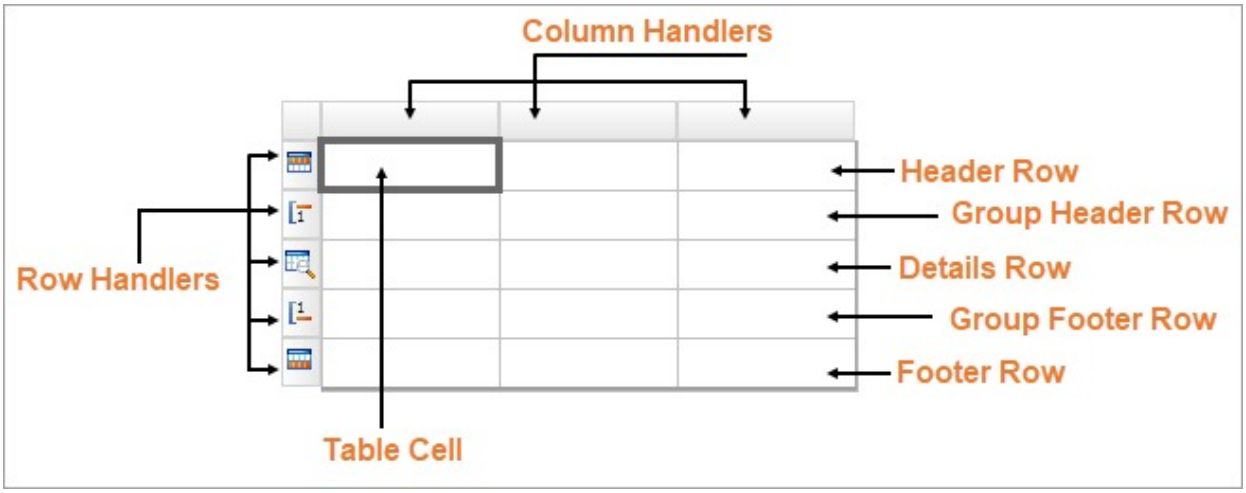
Element Name: Enter a name to be used in the XML output for this subreport.

Output: Choose **Auto**, **Yes**, or **No** to decide whether to include this subreport in the XML output. Choosing **Auto** exports the contents of the subreport.

Table

The Table data region is available in Page and RDLX reports. It is used to display the information in tabular format in Tabular Reports. The Table data region consists of columns and rows that organize data. By default, a table has three columns and three rows, a total of nine cells, where each cell is filled with a text box. At design time, you can add or remove columns, rows, and groupings to suit your needs. Also, you can embed other data regions in the table cells.

Structure



Column is a vertical group of cells in a table. If you right-click a **column handler**, you can add new columns or delete existing ones using the context menu that appears.

Row is a horizontal group of cells in a table. If you right-click a **row handler**, you can add new rows or delete existing ones using the context menu that appears.

Cell is the intersection of a row and column. By default, each table cell contains a TextBox report item. But you can replace it with other report controls, such as an Image, by drag-and-drop the corresponding item from the Toolbox into the cell.

Header row appears at the beginning of a table. If its **RepeatOnNewPage** property is set to 'True', it prints on every page taken by the table content. You could use the header row to display the title or the logo of a tabular report. A table may have several header rows.

Group header row appears at the beginning of a group. If its **RepeatOnNewPage** property is set to 'True', it prints on every page taken by the table content. You could use a group header row to display the group's field value or summary value. A group may have several header rows.

Details row repeats for each bound dataset record that passed through the dataset filters and data region filters. If a table has the grouped data, then the details row appears between the header and footer of the enclosing group instance. A table may have more than one details row.

Group footer row appears at the end of a group. If its **RepeatOnNewPage** property is set to 'True', it prints on every page taken by the table content. You could use the group footer row to display summary values. A group may have several footer rows.

Footer row appears at the end of a table. If its **RepeatOnNewPage** property is set to 'True', it prints on every page taken by the table content. You could use the footer rows to display grand totals.

Table Layout Actions

The Table data region provides context menu options to perform basic layout actions. You can access layout options for Table rows from the context menu by right-clicking on a selected row.

- Insert Row Above: Add a row above the selected row. The inserted row type is similar to the type of selected row, that is, if the selected row type is a header row, a new header row is added.
- Insert Row Below: Add a row below the selected row. The inserted row type is similar to the type of selected row, that is, if the selected row type is a header row, a new header row is added.
- Delete Rows: Delete the selected rows.
- Distribute Rows Evenly: Set the same height for multiple selected rows.
- Table Header: Show or hide table header rows.
- Table Details: Show or hide table detail rows.
- Table Footer: Show or hide table footer rows.

- Insert Group: Insert groups in a table.
- Edit Group: Edit a group in a table.
- Delete Groups: Delete groups in a table.

You can access layout options for Table columns from the context menu by right-clicking on a selected column.

- Insert Column to the Left: Add a column to the left of the selected column.
- Insert Column to the Right: Add a column to the right of the selected column.
- Distribute Columns Evenly: Set the same width for multiple selected columns.
- Add Columns: Add one or more column(s) to the right of the selected column.
- Delete Columns: Delete the selected columns.

Table Dialog Properties

You can set the Table properties in the Table dialog. To open it, with the Table selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the table that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Dataset name: Select a dataset to associate with the table. The combo box is populated with all of the datasets in the report's dataset collection.

Has own page numbering: Select to indicate whether the table has its own pagination.

Page Breaks (RDLX report):

- Insert a page break before this table
- Insert a page break after this table
- Fit table on a single page if possible

NewPage (RDLX report):

- **Next:** A default value that makes a new group start from the immediate next page of the report.
- **Odd:** A new group starts from the next odd page of the report.
- **Even:** A new group starts from the next even page of the report.

See Page Breaks in Data Regions topic for more detail.

Header and Footer: Select any of the following options.

- Repeat header row on each page
- Repeat footer row on each page
- Prevent orphaned footer on next page

Visibility

Initial visibility

- **Visible**: The table is visible when the report runs.
- **Hidden**: The table is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the Table is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. This enables the drop-down box below where you can specify the TextBox control that toggles the visibility of the Table. The user can click the toggle item to show or hide this Table.

Navigation

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this Table. You will then be able to provide a bookmark link to this item from another report item using a **Jump to bookmark** action.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

Expression: Enter an expression by which to sort the data in the group.

Direction: Select Ascending or Descending.

Groups

Click the plus sign button to add a new group to the table, and delete them using the X button. Once you add one or more groups, you can reorder them using the arrow buttons, and set up information for each group on the following tabs.

General

Name: Enter a name for the group that is unique within the report. This property cannot be set until after a Group on expression is supplied.

Group on: Enter an expression to use for grouping the data.

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right.
  Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Sorting

Click the plus sign button to enter new sort expressions, and remove them using the X button.

Expression: Enter an expression by which to sort the data in the group.

Direction: Select Ascending or Descending.

Visibility

By default, the group is visible when the report runs, but you can hide a group when certain conditions are met, or

toggle its visibility with another report item.

Initial visibility

- **Visible**: The group is visible when the report runs.
- **Hidden**: The group is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the group is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. The user can click the toggle item to show or hide this band group. This enables the drop-down list where you can select the report control that users can click to show or hide this group.

Data Output

Element Name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose **Auto**, **Yes**, **No**, or **Contents only** to decide whether to include this group in the XML output.

Layout

BreakLocation: Select from these options to decide where to insert a page break in relation to the group.

- **None**: Inserts no page break.
- **Start**: Inserts a page break before the group.
- **End**: Inserts a page break after the group.
- **StartAndEnd**: Inserts a page break before and after the group.
- **Between**: Inserts a page break between groups (at the end of a current group and the beginning of a next group).

New Page:

- **Next**: A default value that makes a new group start from the immediate next page of the report.
- **Odd**: A new group starts from the next odd page of the report.
- **Even**: A new group starts from the next even page of the report.

Include group header: Adds a group header band (selected by default).

Include group footer: Adds a group footer band (selected by default).

Repeat group header: Repeats the group header band on each page.

Repeat group footer: Repeats the group footer band on each page.

Has own page numbering: Used in conjunction with the "Page Number in Section" and "Total Pages in Section" properties, tells the report that the group constitutes a new page numbering section.

Keep together on one page if possible: Indicates that the group is kept together on one page if possible.

Prevent orphaned footer: Indicates whether the orphaned footer is displayed for the group.

Detail Grouping

Detail grouping is useful when you do not want to repeat values within the details. When a detail grouping is set, the value repeats for each distinct result of the grouping expression instead of for each row of data. For example, if you use the Customers table of the NorthWind database to create a list of countries without setting the details grouping, each country is listed as many times as there are customers in that country. If you set the details grouping to `=Fields!Country.Value` each country is listed only once.

> **Note**: If the detail grouping expression you use results in a value that is distinct for every row of data, a customer number for example, you will see no difference in the results.

The Detail Grouping page has the following tabs.

General

Name: Enter a name for the group that is unique within the report. This property cannot be set until after a Group on expression is supplied.

Group on: Enter an expression to use for grouping the data.

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Parent group: For use in recursive hierarchies. Enter an expression to use as the parent group.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right.
  For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right.
  Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Visibility

By default, the group is visible when the report runs, but you can hide a group when certain conditions are met, or toggle its visibility with another report item.

Initial visibility

- **Visible**: The group is visible when the report runs.
- **Hidden**: The group is hidden when the report runs.
- **Expression**: Use an expression with a Boolean result to decide whether the group is visible. True for hidden, False for visible.

Visibility can be toggled by another report item: Select this check box to display a toggle image next to another report item. The user can click the toggle item to show or hide this band group. This enables the drop-down list where you can select the report control that users can click to show or hide this group.

Data Output

Element Name: Enter a name to be used in the XML output for this group.

Collection: Enter a name to be used in the XML output for the collection of all instances of this group.

Output: Choose **Yes** or **No** to decide whether to include this group in the XML output.

Layout

BreakLocation: Select from these options to decide where to insert a page break in relation to the group.

- **None**: Inserts no page break.
- **Start**: Inserts a page break before the group.
- **End**: Inserts a page break after the group.
- **StartAndEnd**: Inserts a page break before and after the group.
- **Between**: Inserts a page break between groups (at the end of a current group and the beginning of a next group).

NewPage:

- **Next:** A default value that makes a new group start from the immediate next page of the report.
- **Odd:** A new group starts from the next odd page of the report.
- **Even:** A new group starts from the next even page of the report.

See Page Breaks in Data Regions topic for more detail.

Has own page numbering: Used in conjunction with the "Page Number in Section" and "Total Pages in Section" properties, tells the report that the group constitutes a new page numbering section.

Filters

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Data Output

The Data Output page of the Table dialog allows you to control the following properties when you export to XML.

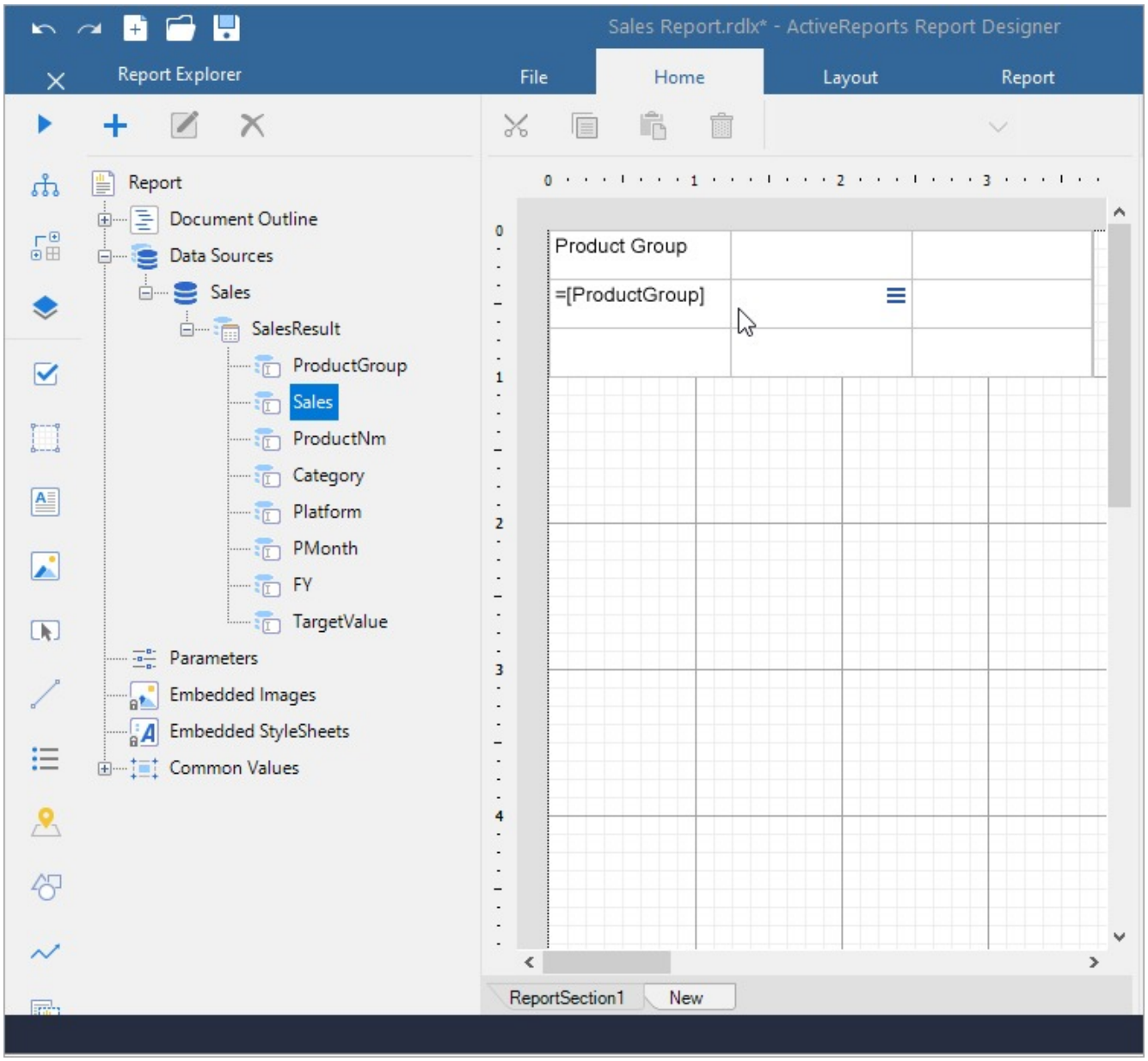
- **Element Name**: Enter a name to be used in the XML output for this Table.
- **Output**: Choose **Auto**, **Yes**, or **No** to decide whether to include this Table in the XML output. Choosing **Auto** exports the contents of the Table.
- **Detail element name**: Enter a name to be used in the XML output for the data element for instances of the table. This name is ignored if you have specified a details grouping.
- **Detail collection name**: Enter a name to be used in the XML output for the data element for the collection of all instances of the detail grouping.
- **Data element output**: Choose **Yes** or No to decide whether to include the details in the XML output.

Adding Data to Table

Once you place the Table data region on a report, you can add data to its cells. As with any data region, you can drag fields from your Fields list onto cells in the table. Although the default report control within each cell of the table is a text box, you can replace it with any other report control or a data region. When you drag a field into a cell in the details row, ActiveReports

automatically provides a label in the table header. As with all report controls, you can use expressions to further manipulate the data within the cells of the table. For more information, see Expressions.

1. From the Toolbox, drag-drop the Table data region onto the design area.
2. From the Report Explorer, expand the DataSet and drag-drop data fields onto the cells in the details row.



Alternatively, in the table, select a TextBox in the Details row and, from the Fields Selection Adorner, select a field from the list.

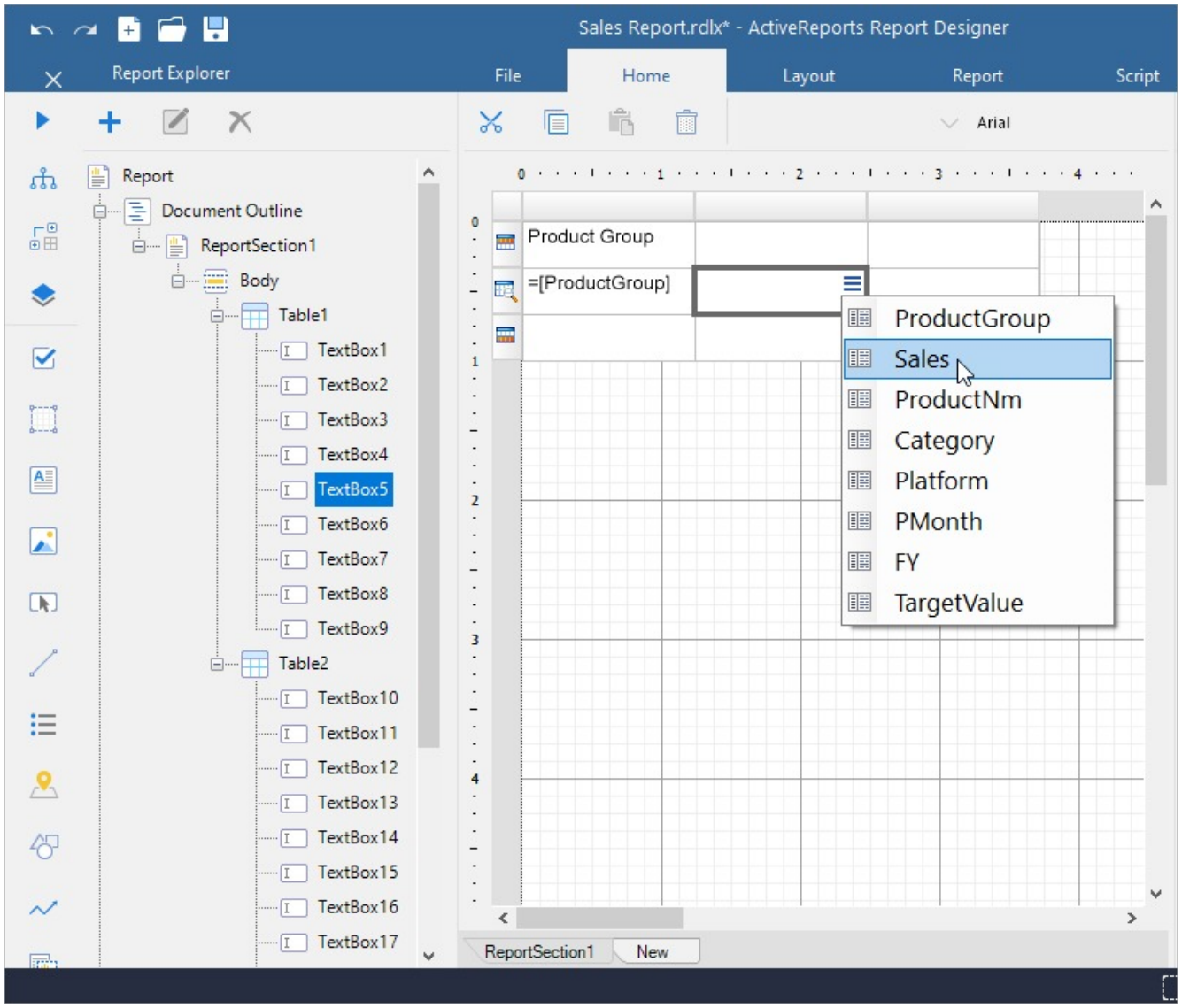


Table Features

Table Grouping

Groups provide a way to organize data and make data analyses easy. In a Table data region, you can also create a document map based on the groups by specifying the group expression in the Document map label option. For information on document map and navigation, see the Document Map topic for more details.

Let us understand grouping using the report shown. The report consists of a Table that displays Movie details – Title, MPAA, and User Rating fields. The report connects to the 'Movie' table from 'Reels.db' data source on GitHub.

Title	MPAA	User Rating
=[Title]	=[MPAA]	=[UserRating]

Title	MPAA	User Rating
101 Dalmatians	G	9
2 Fast 2 Furious	PG-13	7.8
50 First Dates	PG-13	8.6
8 Mile	R	9.1
A Beautiful Mind	PG-13	9.3
A Bug's Life	G	8.7
A Few Good Men	R	8.8
A League of Their Own	PG	9
A Time to Kill	R	6
Ace Ventura: When Nature Calls	PG-13	7.1
Air Force One	R	6
Airport	G	9.2
Aladdin	G	5.1
American Beauty	R	8.4
American Graffiti	PG	7.5

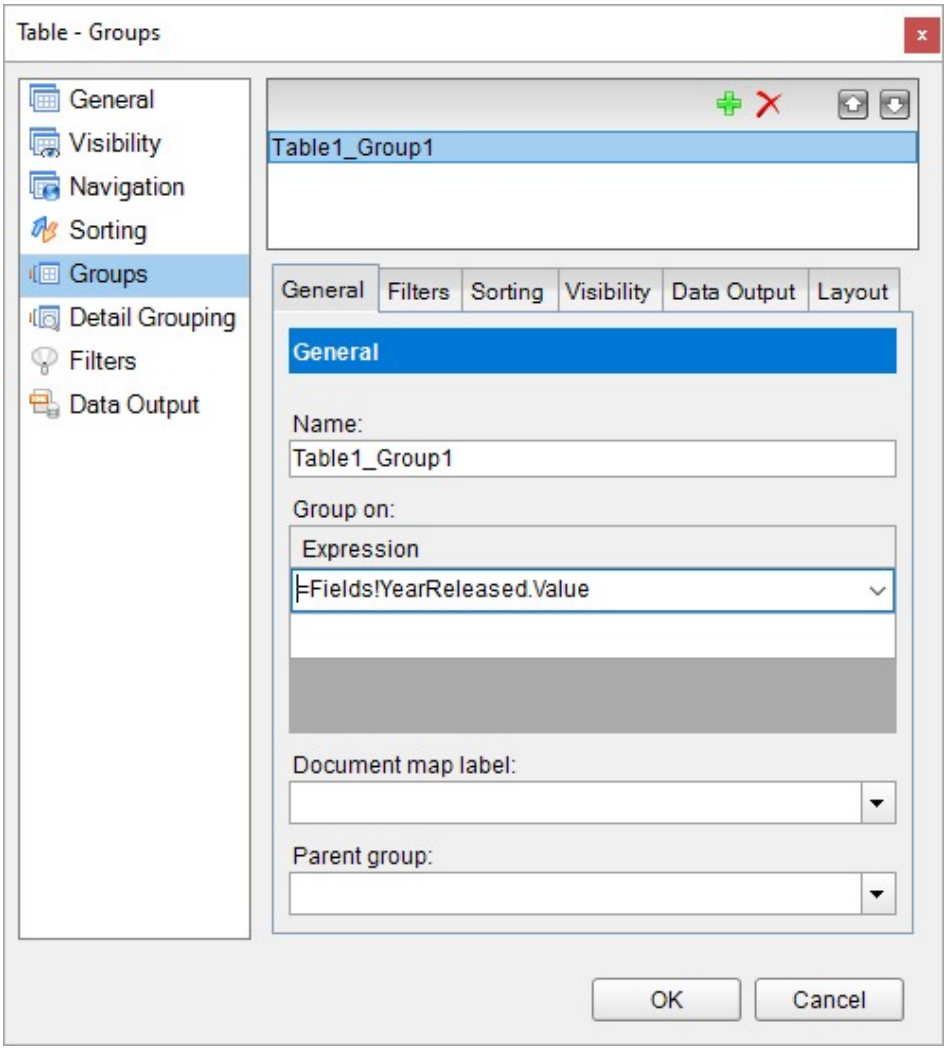
You can group the table data according to the other dataset field, 'Year Released', to group the details based on the year when a movie was released.

The following steps add a group in the table using **Group On** expression.

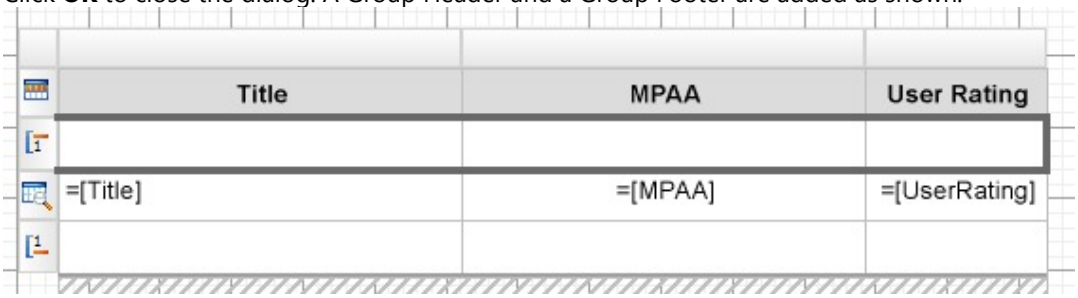
1. With the Table data region selected, under the Properties window, click the **Property dialog** link to open the Table dialog.

   > **Note**: Alternatively, you can add a group by selecting **Insert Group** in the context menu of the Table data region, and adding a group in the **Table - Groups** dialog that opens.

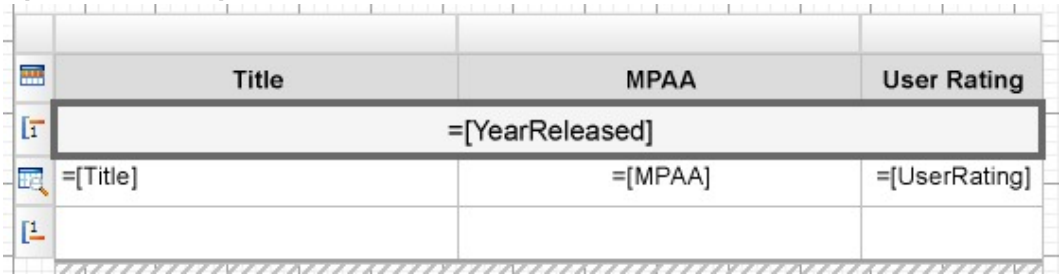
2. Go to the **Groups** page and click Add.
3. Under **Group on**, enter the expression on which you want to group the data. For e.g., `=Fields!YearReleased.Value`.



4. Click **OK** to close the dialog. A Group Header and a Group Footer are added as shown.



5. Merge the cells in the **Group Header** row and enter the expression in the **Value** property as `=[YearReleased]`



This makes the group value appear and span above each group.

6. Preview the report. Here's how the table will look like.

Title	MPAA	User Rating
1997		
Air Force One	R	6
As Good as It Gets	PG-13	9.5
Batman & Robin	PG-13	5.4
Con Air	R	8.9
Contact	PG	7.6
Face/Off	R	8.9
George of the Jungle	PG	7.4
Good Will Hunting	R	8.7
Liar Liar	PG-13	7.2
Men in Black	PG-13	7.8
My Best Friend's Wedding	PG-13	6
Scream 2	R	7
The Lost World: Jurassic Park	PG-13	9.6
Titanic	PG-13	9.1
Tomorrow Never Dies	PG-13	9.2
1977		
Close Encounters of the Third Kind	PG	5.1
Saturday Night Fever	R	9.7
Smokey and the Bandit	PG	8
Star Wars	PG	8
2004		
50 First Dates	PG-13	8.6
Collateral	R	9.5
Dodgeball: A True Underdog Story	PG-13	9.7

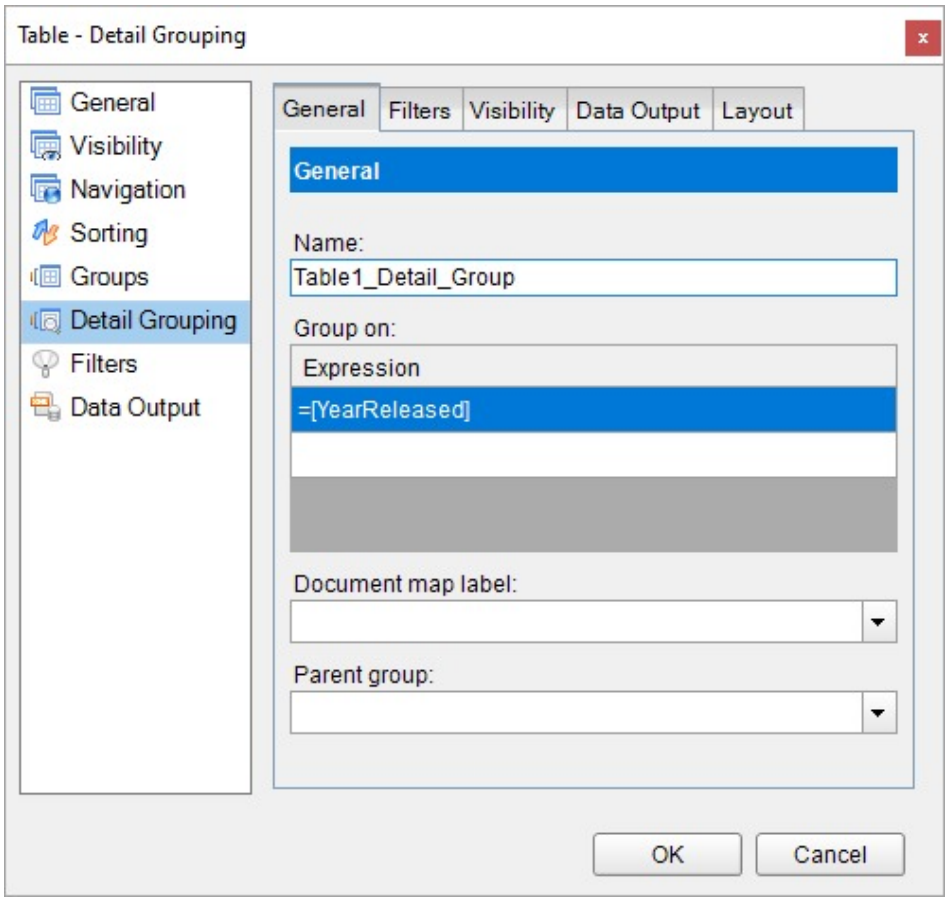
Detail Grouping

Another way of grouping the table data is by grouping the Details row without adding a group. The **Detail Grouping** groups the data in such a way that only one row per distinct grouping value or expression is displayed for each row of data in Details row. Also, it does not add a Group Header and Group Footer. Detail grouping is useful when you do not want the values to repeat within the details as many times as they are present in the data source.

Let us use Detail Grouping to group the table data. For example, we have a report with the Table that displays Movie details – Title, MPAA, and User Rating fields. The report connects to the 'Movie' table from 'Reels.db' data source on GitHub.

Title	MPAA	User Rating
=[Title]	=[MPAA]	=[UserRating]

1. With the Table selected on the report, under the Properties window, click the **Property dialog** link to open the Table dialog.
2. Go to the **Detail Grouping** page and, under **Group on**, enter the expression on which you want to group the data. For e.g.,
   `=Fields!YearReleased.Value`.



3. Click **OK** to close the dialog.
4. Preview the report. Here's how the table will look like.
   Note that the report size is smaller after applying the Details Grouping since the details row is shown once for every grouping value.

Title	MPAA	User Rating
Aladdin	G	5.1
Back to the Future	PG	6.7
Bambi	Approved	8.2
Batman	PG-13	8
Blazing Saddles	R	5.5
Butch Cassidy and the Sundance Kid	M	6.4
E.T. the Extra-Terrestrial	PG	7.3
Forrest Gump	PG-13	5.4
Ghost Busters	PG	6.5
Gone with the Wind	Approved	9.3
Grease	PG	9.3
Harry Potter and the Sorcerer's Stone	PG	5.4

Merge Cells

You can combine adjacent cells in horizontal (same row) or vertical (same column) direction into a single cell. Vertical cell merging is possible within the same row type, that is, within the Header, Group Header, Footer, Group Footer, or Details row.

To merge cells,

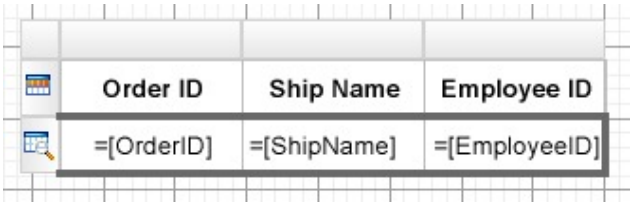
1. Select the cells (Ctrl+Click) and the right-click.
2. From the context menu, select **Merge Cells**.

Auto Merge

The **AutoMergeMode** property lets you set the mode to merge the adjacent cells (text boxes) in details row with the same value. This property takes Always, Never, and Restricted values. The details row with the same data values and with AutoMergeMode property set to:

- **Always** - are merged.
- **Never** - are not merged.
- **Restricted** - are merged only if the corresponding cells in the previous columns are similarly merged. If for example, cells in Column 2 (with same data values) are set 'Restricted' and the corresponding cells (with same the data values) in previous column, that is Column 1, are set 'Never', then cells in Column 2 are not merged.

The following steps take you through how to add automatic merge to the cells in the following table. The report connects to the 'Orders' table from 'NWIND.db' data source on GitHub.



Order ID	Ship Name	Employee ID
10272	Rattlesnake Canyon Grocery	6
10273	QUICK-Stop	3
10274	Vins et alcools Chevalier	6
10275	Magazzini Alimentari Riuniti	1
10276	Tortuga Restaurante	8
10277	Morgenstern Gesundkost	2
10278	Berglunds snabbköp	8
10279	Lehmanns Marktstand	8
10280	Berglunds snabbköp	2
10281	Romero y tomillo	4
10282	Romero y tomillo	4
10283	LILA-Supermercado	3
10284	Lehmanns Marktstand	4
10285	QUICK-Stop	1
10286	QUICK-Stop	8
10287	Ricardo Adocicados	8

1. Select the cell with `EmployeeID` field and set the **Layout > AutoMergeMode** property to 'Restricted'. This merges employee ids depending on whether the corresponding ship names (cells in the previous column) are merged.
2. Select the cell with `ShipName` field and set the **Layout > AutoMergeMode** property to 'Always'. This merges the cells with similar ship names.
3. Preview the report. Here's how the table will look like.

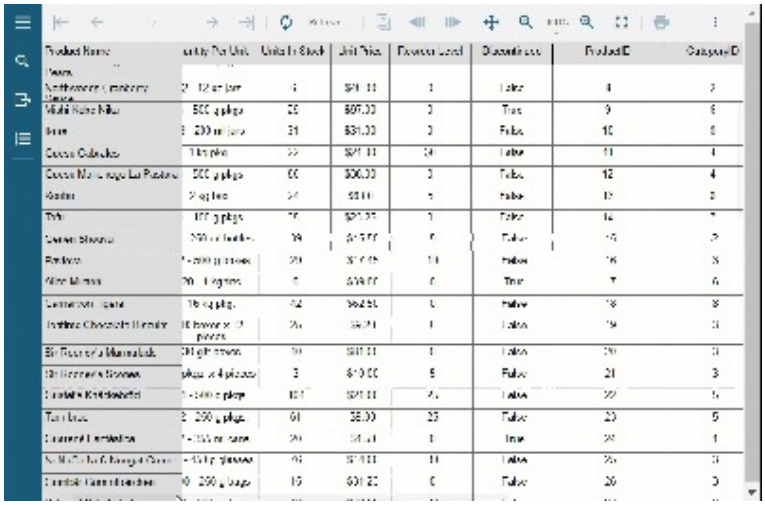
Order ID	Ship Name	Employee ID
10272	Rattlesnake Canyon Grocery	6
10273	QUICK-Stop	3
10274	Vins et alcools Chevalier	6
10275	Magazzini Alimentari Riuniti	1
10276	Tortuga Restaurante	8
10277	Morgenstern Gesundkost	2
10278	Berglunds snabbköp	8
10279	Lehmanns Marktstand	8
10280	Berglunds snabbköp	2
10281	Romero y tomillo	4
10282		
10283	LILA-Supermercado	3
10284	Lehmanns Marktstand	4
10285	QUICK-Stop	1
10286		8
10287	Ricardo Adocicados	8

Freeze Rows and Columns (RDLX Reports)

When you use a Table data region containing a large amount of data in an RDLX report, the user must scroll to see all of the data. On scrolling the column or row headers out of sight, the data becomes difficult to understand. To resolve this problem, you can use the **FrozenRows** and **FrozenColumns** properties that take effect in the **JSViewer** in **Galley** mode, and allow you to freeze headers so that they remain visible while scrolling through the data region. You can freeze as many rows or columns as you have headers in the data region.

- If your data stretches downward, set the **FrozenRows** property to a value to float the column headers when scrolling.
- If your data stretches to the right, set the **FrozenColumns** property to a value to float the row headers when scrolling.
- If your data stretches both downward and to the right, set both **FrozenRows** and **FrozenColumns** properties.

Here's how a report with one frozen row and one frozen column looks like in JS Viewer.



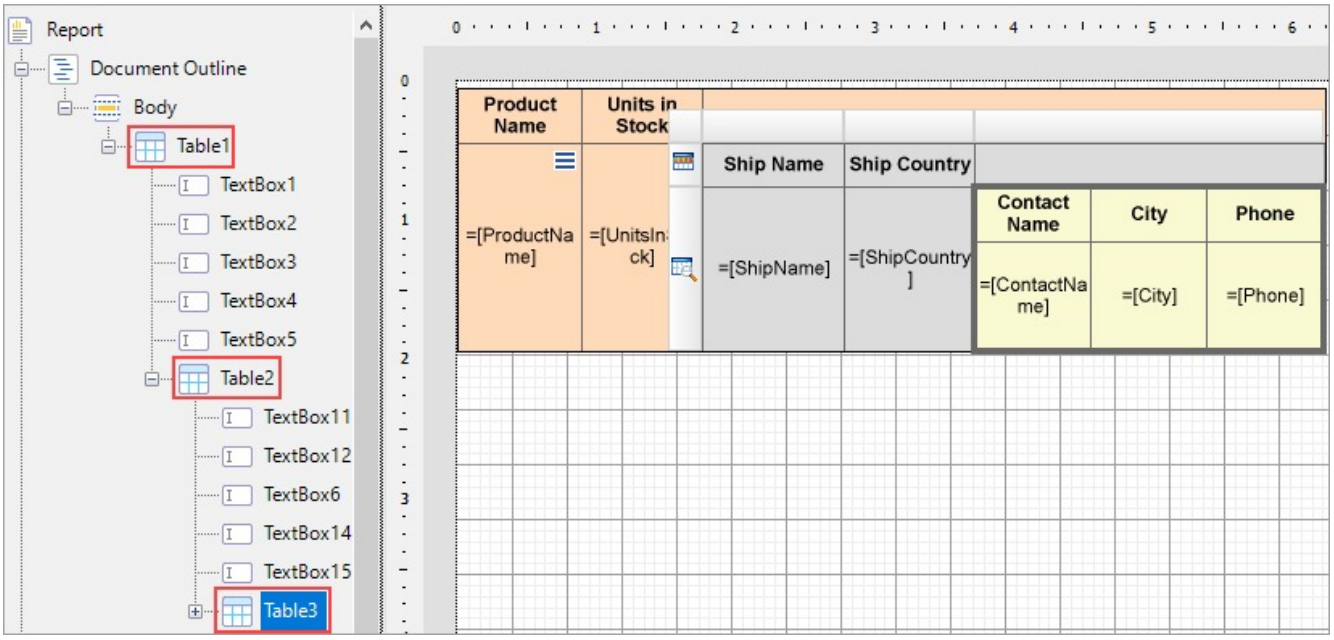
If any header cells that you want to freeze are merged, you should not set the **FrozenRows** or **FrozenColumns** property to a value that would split a merged cell.

Nesting in Tables

To display data from different datasets, you can use nested tables that are bound to different datasets. You can do any one of the following to achieve this:

- add a filter to a child table (**Filters** property) which filters the data in the table data region based on the common field in two datasets
- create a parameter for the child table (**DataSetParameters** property) that is passed to the dataset query and filters the dataset fields based on the common field in the two datasets

For example, the following report shows three nested Table data regions that fetch data from three different datasets.



The report connects to the 'NWIND.db' data source on GitHub.

Table1 is bound to 'Products' dataset, **Table2** is bound to 'Invoices' dataset, and **Table3** is bound to 'Customers' dataset. The

parent table, Table1, shows the Product Name and Units In Stock information from the **Products** dataset. For each ProductID, child table, Table2 shows the Ship Name and Ship Country information from the **Invoices** dataset. And for each CustomerID, the child table, Table3, shows the Contact Name, City, and Phone information from the **Customers** dataset.

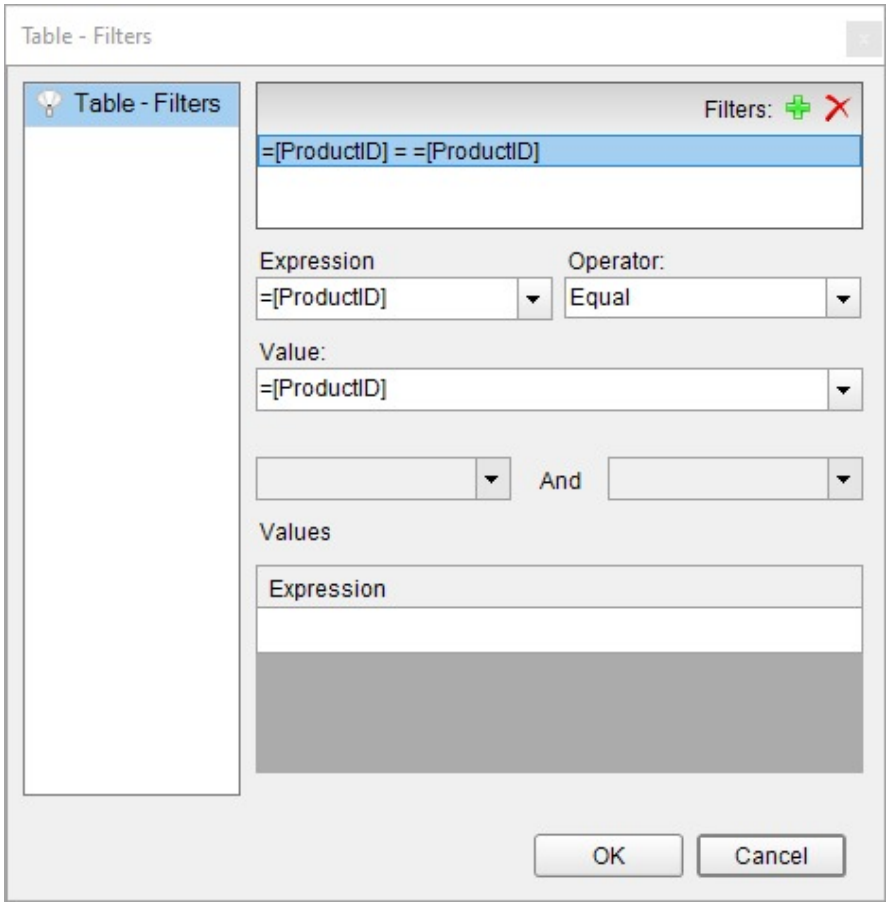
Product Name	Units in Stock					
Chai	39	**Ship Name**	**Ship Country**			
		QUICK-Stop	Germany	**Contact Name**	**City**	**Phone**
				Horst Kloss	Cunewalde	0372-035188
		Rattlesnake Canyon Grocery	USA	**Contact Name**	**City**	**Phone**
				Paula Wilson	Albuquerque	(505) 555-5939
		Lonesome Pine Restaurant	USA	**Contact Name**	**City**	**Phone**
				Fran Wilson	Portland	(503) 555-9573
		Die Wandernde Kuh	Germany	**Contact Name**	**City**	**Phone**
				Rita Müller	Stuttgart	0711-020361
		Pericles Comidas clásicas	Mexico	**Contact Name**	**City**	**Phone**
				Guillermo Fernández	México D.F.	(5) 552-3745

Let us create this report using both methods.

Using a filter

For the child tables, add a filter using the Filters property as elaborated below:

1. Select **Table2** and go to the **Filters** property.
2. In the **Table - Filters**, add a filter with the expression `=[ProductID]==[ProductID]`, the common field in the datasets of the parent table and the child table.



Similarly,

1. Select **Table3** and go to the **Filters** property.
2. In the **Table - Filters**, add a filter with the expression `=[CustomerID]=[CustomerID]`, the common field in the datasets of the parent table and the child table.

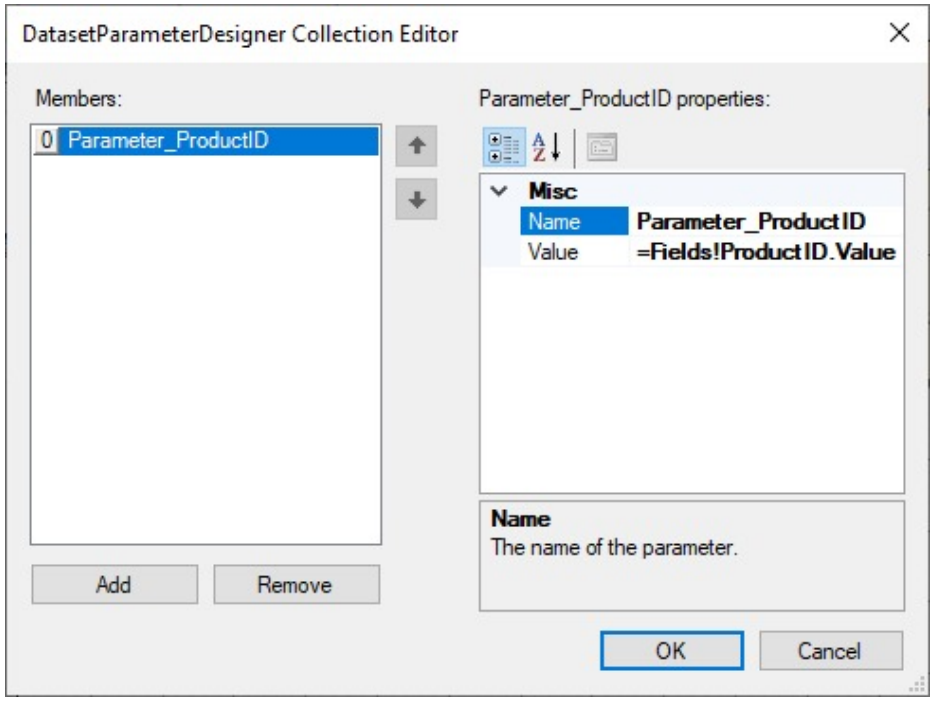
Using a parameter

For each child table, we create a parameter to pass to the dataset and then modify the dataset query to filter the dataset fields.

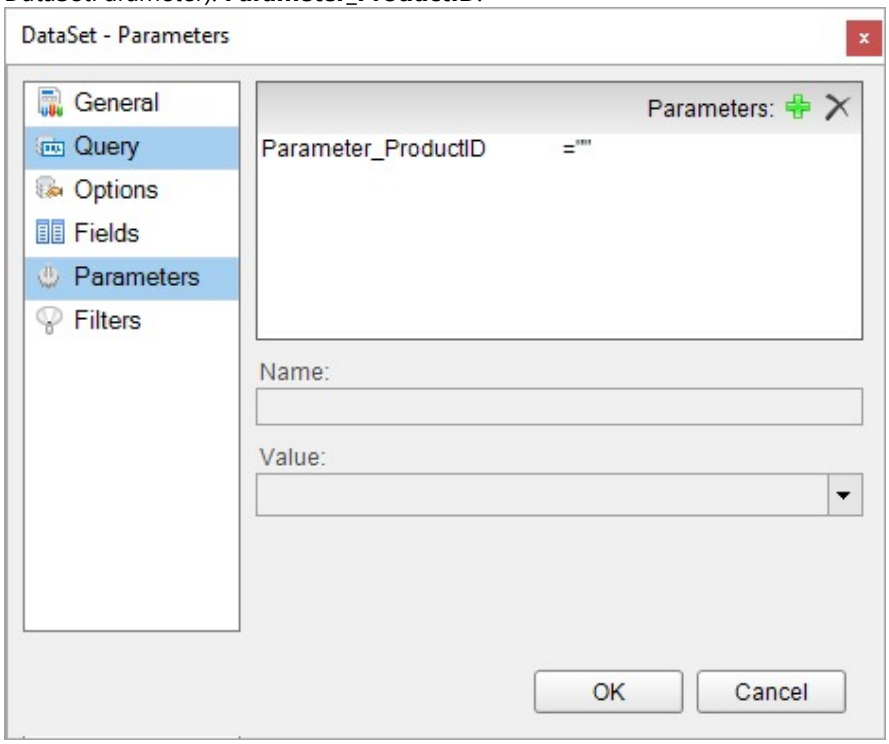
> **Note:** JSON dataset does not have the parameters collection. Hence, the DataSetParameters from the data region cannot interact with such a dataset.

The following steps elaborate the procedure.

1. Select Table2 and go to the **DataSetParameters** property.
2. Add a new parameter with **Name:** Parameter_ProductID and **Value:** `=Fields!ProductID.Value`.



3. Go to the dataset to which the Table2 is bound, that is, **Invoices**.
4. In the DataSet designer dialog, go to **Parameters** and add a new parameter (same name as DataSetParameter): **Parameter_ProductID**.



5. Go to **Query** and add a substring with the parameter to the existing dataset query so the updated query is:
   `select * from Invoices where ProductID = @Parameter_ProductID`

Similarly,

1. Select Table3 and go to the **DataSetParameters** property.

2. Add a new parameter with **Name:** Parameter_CustomerID and **Value:** `=Fields!CustomerID.Value`.
3. Go to the dataset to which Table3 is bound, that is, **Customers**.
4. In the DataSet designer dialog, go to **Parameters** and add a new parameter (same name as DataSetParameter): Parameter_CustomerID.
5. Go to **Query** and add a substring with the parameter to the existing dataset query so the updated query is:
   ```
   select * from Invoices where ProductID = @Parameter_CustomerID
   ```

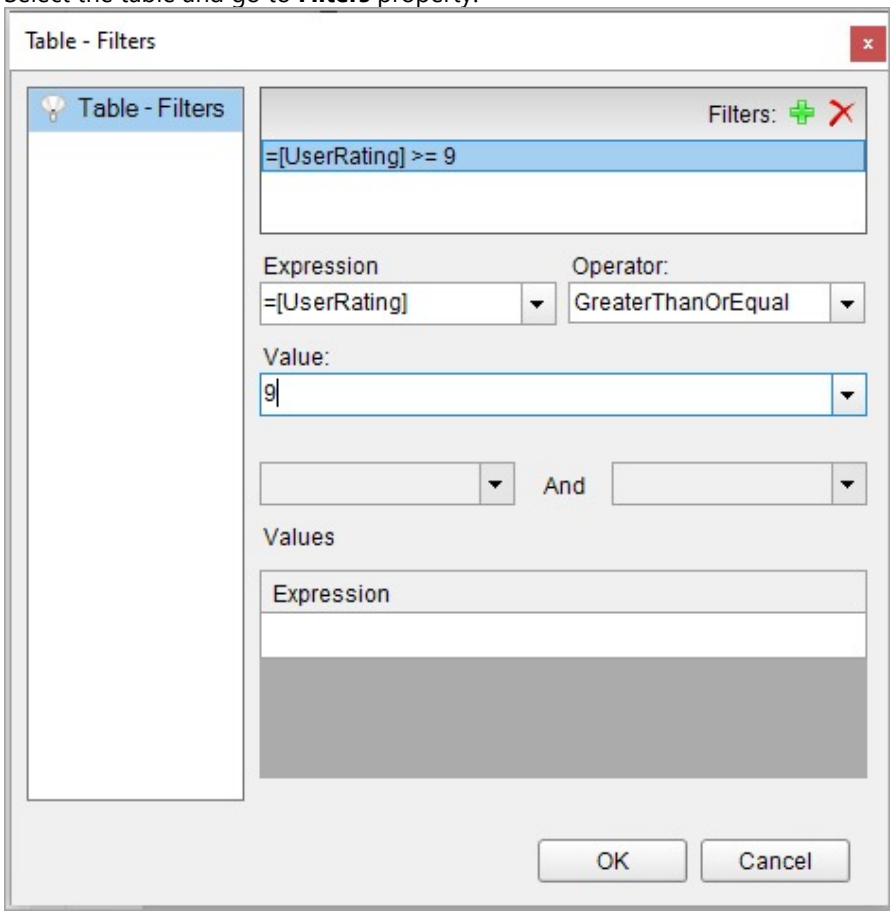
Filter, Sort, and Interactive Sort

Filter

Filtering allows you to set filters on a large set of data that has already been retrieved from the data source and use them with datasets or data regions to limit the information you want to display on your report. Although not as efficient performance-wise as query parameters which filter data at the source, there are still scenarios that demand filters. The obvious case is when the data source does not support query parameters.

To apply a filter in a Table data region,

1. Select the table and go to **Filters** property.



2. In the Table - Filters dialog, click **Add**.

   You can also first go to the **Property dialog** and then select the **Filters** page.

3. In the **Expression** field, select the data field expression (`=[UserRating]`) available from the dataset to which the table is bound.
4. Select the filter **Operator** (GreaterThanOrEqual).

5. Enter the **Value** (say 9) to which the expression should evaluate to filter the data.
6. Click OK.
7. Preview the report. Here's how the table will look like with the 'User Rating' column showing a rating above '9'.

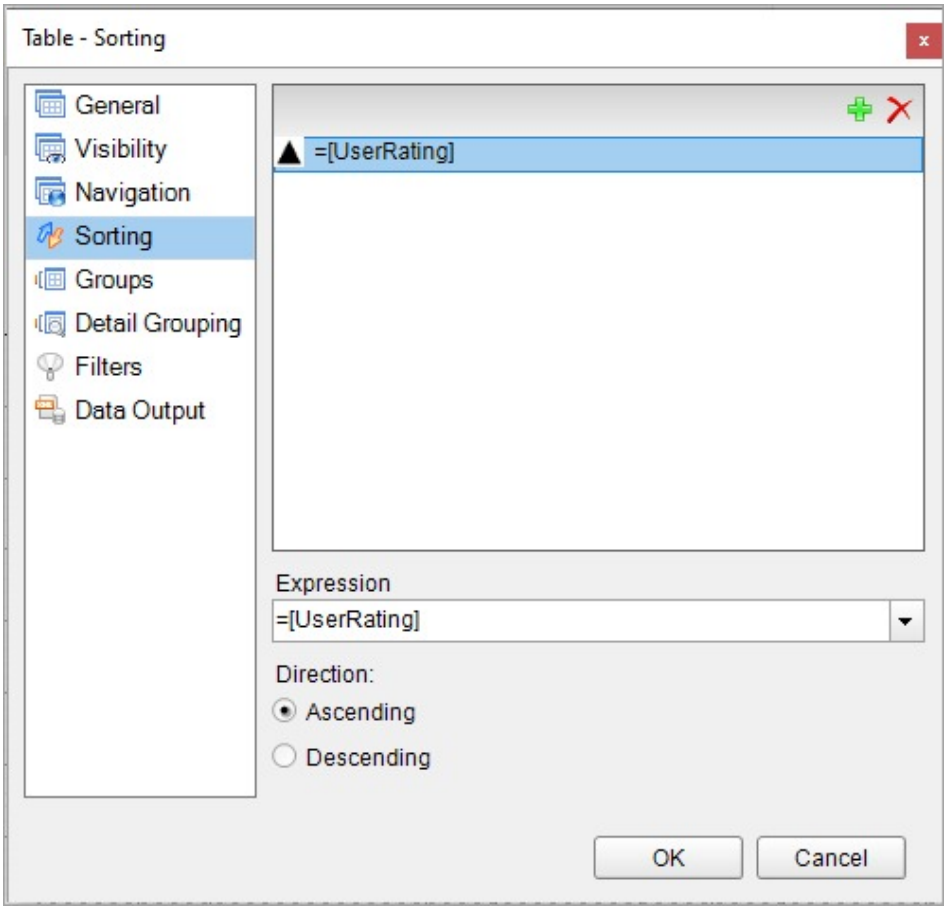
Title	MPAA	User Rating
Titanic	PG-13	9.1
Star Wars: Episode I - The Phantom Menace	PG	9.1
The Lord of the Rings: The Fellowship of the Ring	PG-13	10
The Sixth Sense	PG-13	9.4
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	PG	9
Harry Potter and the Goblet of Fire	PG-13	9.5
How the Grinch Stole Christmas	PG	9.8
Jaws	PG	9.6
My Big Fat Greek Wedding	PG	9.3
The Lost World: Jurassic Park	PG-13	9.6
Mission: Impossible II	PG-13	9.9
X2	PG-13	9.7

Sort

Sorting helps better organize and present data in your report, you can sort alphabetically or numerically in ascending or descending order.

To apply sort in a Table data region,

1. Select the table and click the **Property dialog** link to open the Table dialog.



2. Go to the **Sorting** page and click **Add**.
3. In the **Expression** field, select a value to sort the report data (for example, `=[Title]`).
4. Select the sorting direction by clicking **Ascending** or **Descending** options.
5. Click OK.
6. Preview the report. Here's how the table will look like with the 'User Rating' column sorted in ascending order.

Title	MPAA	User Rating
Raiders of the Lost Ark	R	5.1
Aladdin	G	5.1
The Jungle Book	G	5.1
Star Wars: Episode V - The Empire Strikes Back	PG	5.3
One Hundred and One Dalmatians	G	5.3
The Godfather	R	5.3
Forrest Gump	PG-13	5.4
Harry Potter and the Sorcerer's Stone	PG	5.4
Blazing Saddles	R	5.5
Top Gun	PG	5.6
Love Story	PG	5.8

Interactive Sort

You can add **interactive sort** or user sort on a TextBox control to allow sorting columns of data within a data region on a published report.

The interactive sorting feature is set through the **Interactive Sort** page in the TextBox dialog.

Once you set interactive sorting on a TextBox control while viewing the report in the Viewer or in the Preview Tab, the textbox control displays a sort icon inside it. A user can sort data that appears inside the textbox in ascending or descending order by clicking the icons.

On the **Interactive Sort** page of the TextBox dialog you can find the following fields available for entering values:

- **Sort Expression:** An expression specifying the sort value for data contained in the column.
- **Data region or group to sort:** Select the grouping level or data region within the report to sort. The default value is 'Current scope', but you may also opt 'Choose data region or grouping'.
- **Evaluate sort expression in this scope:** Select the grouping level within the report on which to evaluate an aggregate sorting expression. The default value is 'Current scope', but you may also opt 'Choose data region or grouping'.

Title	MPAA	User Rating
Titanic	PG-13	9.1
Star Wars	PG	8
Shrek 2	PG	7.1
E.T. the Extra-Terrestrial	PG	7.3
Star Wars: Episode I - The Phantom Menace	PG	9.1
Spider-Man	PG-13	8.6
Star Wars: Episode III - Revenge of the Sith	PG-13	8.5
The Lord of the Rings: The Return of the King	PG-13	6.5
Spider-Man 2	PG-13	8.8
The Passion of the Christ	R	8.8
Jurassic Park	PG-13	6.8

Title	MPAA	User Rating
101 Dalmatians	G	9
2 Fast 2 Furious	PG-13	7.8
50 First Dates	PG-13	8.6
8 Mile	R	9.1
A Beautiful Mind	PG-13	9.3
A Bug's Life	G	8.7
A Few Good Men	R	8.8
A League of Their Own	PG	9
A Time to Kill	R	6
Ace Ventura: When Nature Calls	PG-13	7.1
Air Force One	R	6

Title	MPAA	User Rating
You've Got Mail	PG	9.6
xXx	PG-13	8.3
X-Men	PG-13	9.1
X2	PG-13	9.7
Wo hu cang long	PG-13	8.6
Wild Wild West	PG-13	6.7
Who Framed Roger Rabbit	PG	5.8
What Women Want	PG-13	8.6
What Lies Beneath	PG-13	9.9
Wedding Crashers	R	7.8
Wayne's World	PG-13	8.4

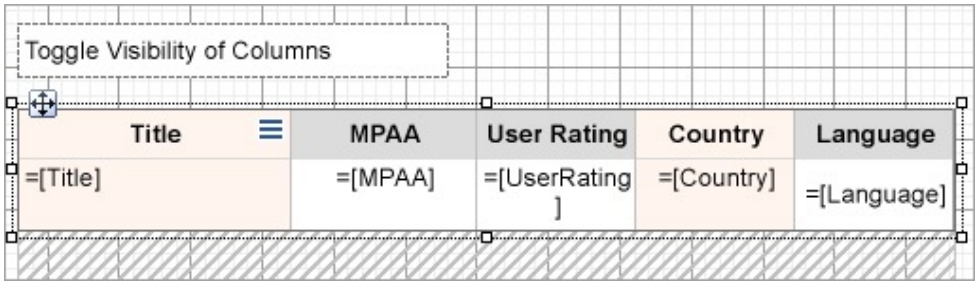
Dynamic Columns

The dynamic columns in tables are the columns that grow or shrink depending on the visibility of other columns. When a Table has hidden column(s), the other column(s) can be made dynamic by setting the column's **AutoWidth** property to 'Proportional'.

So, for example, in the following table, we have five columns: Title, MPAA, User Rating, Country, and Language. The visibility of columns MPAA, User Rating, and Language is being toggled by a text box. We want the columns Title and Country to grow or shrink when the visibility of the other three columns is toggled.

Title	MPAA	User Rating	Country	Language
Titanic	PG-13	9.1	USA	English
Star Wars	PG	8	USA	English
Shrek 2	PG	7.1	USA	English
E.T. the Extra-Terrestrial	PG	7.3	USA	English
Star Wars: Episode I - The Phantom Menace	PG	9.1	USA	English
Spider-Man	PG-13	8.6	USA	English
Star Wars: Episode III - Revenge of the Sith	PG-13	8.5	USA	English
The Lord of the Rings: The Return of the King	PG-13	6.5	USA	Old English
Spider-Man 2	PG-13	8.8	USA	English

The above report can be created as elaborated below:



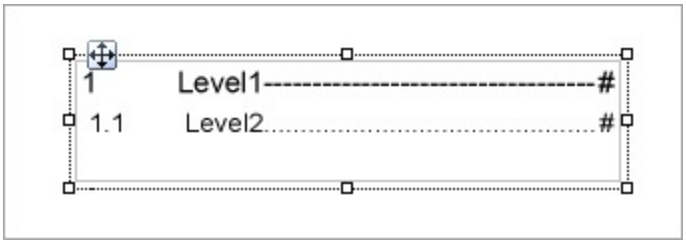
1. First set the visibility of columns MPAA, User Rating, and Language and set the toggle control:
    1. Select the columns MPAA, User Rating, and Language (Ctrl+Click the column handlers).
    2. From the Properties window, set the **Visibility > Hidden** to 'False' and **ToggleItem** to the name of the textbox from which we will toggle the visibility of these columns.
2. Now make columns Title and Country dynamic:
    1. Select the columns Title and Country.
    2. From the Properties window, set their **AutoWidth** property to 'Proportional'.
3. Preview the report.
   The column width of the visible columns is expanded to take up the width of the Table data region. This way, regardless of how many columns a table displays, the width stays the same.

Table of Contents

The TableOfContents report control is available in Page and RDLX reports. It is used to display the document map, an organized hierarchy of the report heading levels and labels along with their page numbers in the body of a report.

The TableOfContents control allows you to quickly understand and navigate the data inside a report in all viewers that are supported in ActiveReports. Unlike the Document Map that is only available in the Viewers and cannot be rendered or printed, you can use the TableOfContents control to embed the table of contents structure in the report body for printing and rendering purposes.

Structure



Important Properties

Property	Description
Levels	Contains the collection of TableOfContents levels and allows you to access the **LevelDesigner Collection Editor** dialog, where you can set up the report TableOfContents levels and their properties
MaxLevel	Restricts the maximum number of levels in the document map.
StyleName	Allows you to apply the selected styles from a style sheet. These styles can be applied to the TableOfContents report control using the **StyleName** property or to Table Of Contents levels using the **LevelDesigner Collection Editor** dialog
OverflowName (Page report)	Specify the OverflowPlaceHolder control name to link it with the TableOfContents control.
FixedHeight (Page report)	Allows you to set the maximum height of the TableOfContents control on each page, similar to the FixedSize property that is available with other report controls.

Table of Contents Dialog Properties

General

Name: Enter a name for the table of contents that is unique within the report. This name can be called in code. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Visibility

By default, the Table of Contents is visible when the report runs, but you can hide it, hide it only when certain conditions are met, or toggle its visibility with another report control.

Initial visibility

Visible: The Table of Contents is visible when the report runs.

Hidden: The Table of Contents is hidden when the report runs.

Expression: Use an expression with a Boolean result to decide whether the Table of Contents is visible. True for hidden, false for visible.

Visibility can be toggled by another report control: Select this check box to display a toggle TableOfContents control next to another report control. This enables the drop-down box where you can specify the TextBox control which, if clicked, toggles the visibility of the TableOfContents control.

Appearance

Border

Style: Select a style for the border.

Width: Enter a value in points to set the width of the border.

Color: Select a color to use for the border, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Background

Color: Select a color to use for the background.

Data Output

Element Name: Enter a name to be used in the XML output for the TableOfContents control.

Output: Choose **Auto**, **Yes**, **No,** or **Contents only** to decide whether to include this Table of Contents in the XML output. Choosing **Auto** exports the contents of the TableOfContents control.

LevelDesigner Collection Editor

Appearance

BackgoundColor: Select a color to use for the background of the TableOfContents level.

Color: Select the color of the text.

Font: Select the font to render the TableOfContents level text.

> **Style:** Choose Normal, Italic, or select the <Expression...> option to open the Expression Editor and create an expression.
>
> **Family:** Choose the font family name.
>
> **Size:** Choose the size in points for the font.
>
> **Weight:** Choose from Lighter, Thin, ExtraLight, Light, Normal, Medium, SemiBold, Bold, ExtraBold, Heavy, and Bolder, or select the <Expression...> option to open the Expression Editor and create an expression.

Padding: Specify left, right, top, and bottom values for the padding to apply to a TableOfContents level.

StyleName: Select a style to apply to the TableOfContents level.

TextAlign: Specify the horizontal alignment of the text.

TextDecoration: Choose from None, Underline, Overline, and LineThrough, or select the <Expression...> option to open the Expression Editor and create an expression.

Data

DataElementName: Enter a name to be used in the XML output for this TableOfContents level.

General

DisplayFillCharacters: Specifies whether to display a leading character. The Default value is **True**.

DisplayPageNumber: Specifies whether to display a page number. The Default value is **True**.

FillCharacter: Use the expression to specify a fill character for a leading character.

Layout

TextIndent: Specify the text indent.

Misc

Name: Specify a name for the TableOfContents level.

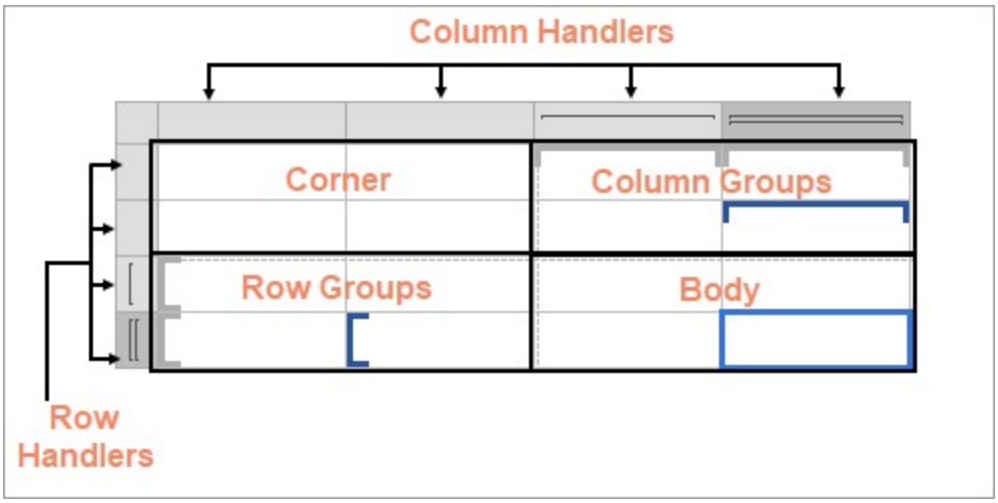
Tablix

A Tablix data region displays data in cells that are arranged in rows and columns. It provides enhanced layout capabilities ranging from the creation of simple tables to advanced matrices. Tablix is essentially a combination of two data regions, the table, and the matrix. Therefore, it provides all the features of a table and a matrix along with added capabilities including support for multiple adjacent groups on rows or columns and improved layout flexibility with

stepped group layouts.

By default, each tablix cell contains a TextBox control, and the function for each cell is determined by its location. You can change the layout of the Tablix data region using the **LayoutDirection** property.

Structure



The Tablix data region is composed of four areas denoted by dotted lines on the design surface - the corner, the row group area, the column group area, and the body.

Corner

The Corner element is located in the upper-left corner, or upper-right corner if a tablix has the **LayoutDirection** property set to Rtl. The layout direction applies at preview time only. This area is automatically expanded horizontally or vertically when you add a new column or row groups. You can merge cells inside the tablix corner and insert a data visualizer such as a TextBox or Image.

A corner may contain only **static cells** that are rendered only once in Tablix.

Column Group

A Column group is represented by square brackets above the columns. Tablix column groups are located in the upper-right corner (upper-left corner for the Rtl layout). A column group represents a member of the column groups hierarchy and displays the column group instance values.

Row Group

A Row group is represented by square brackets on the left side of the rows. Tablix row groups are located on the lower-left corner (lower right for the Rtl layout). A row group represents a member of the row groups hierarchy and displays row group instance values.

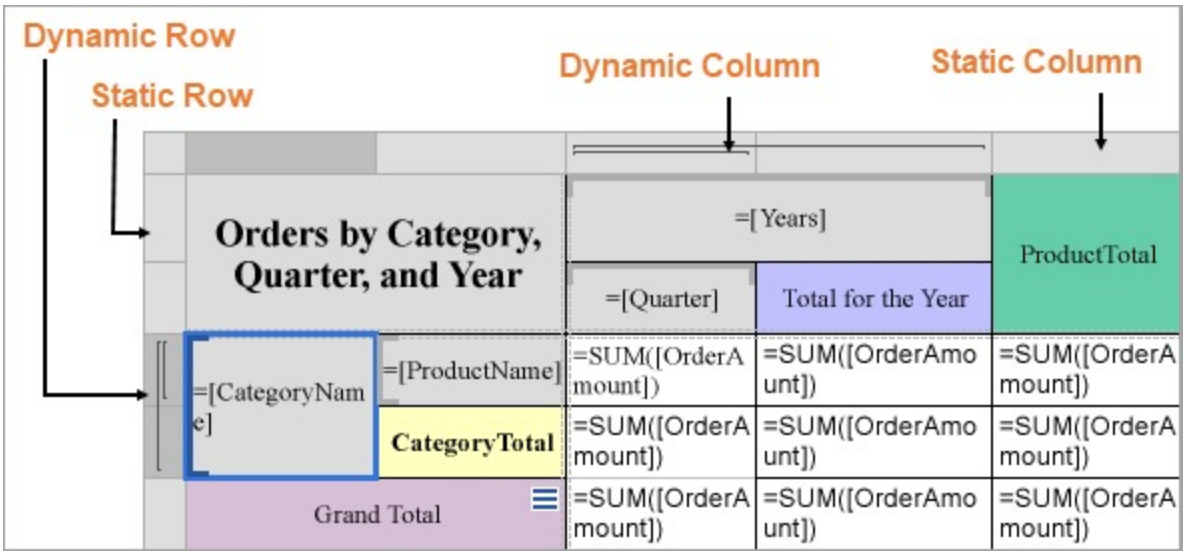
Body

The Body element is used for displaying aggregated data with respect to row and column grouped data in the report. The body may contain only **static cells** that are rendered only once in Tablix.

Static and Dynamic Rows and Columns

Rows or columns in the Tablix data region can be **static** or **dynamic**. The Tablix data region contains multiple rows and columns that provide a grid-type layout, where you can add or remove static or dynamic rows and columns in order to display your data efficiently.

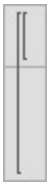
- **Static Rows and Columns** - A static row or column is not associated with any group data. When the report runs, a static row or column is rendered only once. Labels and totals are displayed using static rows or columns in the Tablix data region.
- **Dynamic Rows and Columns** - A dynamic row or column is associated with one or more groups and renders once for every unique value in the group. You can also create dynamic group rows or columns by adding a row group or a column group.



Row and Column Handlers

When you select a Tablix data region, the row and the column handles appear. These handles help you to work with a Tablix and visually specify the type of data added in your tablix layout.

The following table shows the different types of handles that appear in a Tablix data region.

Handle Icon	Description
	Row or column with one outer group.
	One outer group and one inner group.
	One outer group with an extra row for totals and one inner group.

Tablix Layout Actions

The Tablix data region provides context menu options to perform basic layout actions. You can access layout options for Tablix rows from the context menu by right-clicking on a selected row.

- **Insert Row:** Select from the following options to insert a row inside or outside of the selected group cell.
  - **Inside Group:** If a row group contains groups having distinct values, then as many rows are inserted as there are groups.
    - **Above**: Inserts a row above for each unique value of the row group.
    - **Below**: Inserts a row below for each unique value of the row group.
  - **Outside Group**: If a row group contains nested groups consisting of child and parent groups, then as many rows are inserted as there are parent groups.
    - **Above**: Inserts a row above for each unique value of the parent row group.
    - **Below**: Inserts a row below for each unique value of the parent row group.
- **Delete Row:** Delete the selected rows.
- **Distribute Rows Evenly**: Set the same height for multiple selected rows.
- **Add Row Group:** Select from the following options to insert row groups in a tablix.
  - **Parent Group**: To insert a parent row group.
  - **Child Group**: To insert a child row group.
  - **Adjacent Above**: To insert an adjacent row group above the selected row group.
  - **Adjacent Below**: To insert an adjacent row group below the selected row group.
- **Row Group:** Select the Delete Group option to delete a row group.

You can access layout options for Tablix columns from the context menu by right-clicking on a selected column.

- **Insert Column**: Select from the following options to insert a column inside or outside of the selected group cell.
  - **Inside Group**: If a column group contains groups having distinct values, then as many columns are inserted as there are groups.
    - **Left**: Inserts a column to the left for each unique value of the column group.
    - **Right**: Inserts a column to the right for each unique value of the column group.
  - **Outside Group**: If a column group contains nested groups consisting of child and parent groups, then as many columns are inserted as there are parent groups.
    - **Left**: Inserts a column to the left for each unique value of the parent column group.
    - **Right**: Inserts a column to the right for each unique value of the parent column group.
- **Delete Column**: Delete the selected columns.
- **Distribute Columns Evenly:** Set the same width for multiple selected columns.
- **Add Column Group:** Select from the following options to insert column groups in a tablix.
  - **Parent Group**: To insert a parent column group.
  - **Child Group**: To insert a child column group.
  - **Adjacent Left**: To insert an adjacent column group to the left of the selected column group.
  - **Adjacent Right**: To insert an adjacent column group to the right of the selected column group.
- **Column Group:** Select the Delete Group option to delete a column group.

Group Editor

The Group Editor window gets displayed or hidden by the Group Editor icon, located on the left of the Design area. The Group Editor contains the following groups:

- **Row Groups:** The Row Groups section in the Group Editor displays all the groups that are applied in the row

group area of the Tablix data region.
Category Name
The left-center cell of the Tablix data region **=[CategoryName]** represents the **Category** group in the Group Editor window. This group displays the category names for products.
Product Name
The left-center cell of the Tablix data region **=[ProductName]** represents the **Product** group in the Group Editor window. This group displays the product names.

- **Column Groups:** The Column Groups section in the Group Editor window displays all the groups that are applied in the column group area of the Tablix data region.
  Year
  The center-left cell of the Tablix data region **=[Years]** represents the **Years** group in the Group Editor window. This group displays years of orders.
  Quarter
  The center-right cell of the Tablix data region **=[Quarter]** represents the **Quarter** group in the Group Editor window. This group displays quarters for the orders.
- **Static Cells:** Static cells in the Row Groups and Column Groups are not represented in the Group Editor window because these cells are not associated with any grouped data. Static row and column cells are used to display labels and totals in a Tablix data region.

The static column cell displays the label **YEARS** and **CATEGORY NAMES** in the Tablix data region. The static row cell displays the label **Total** in the Tablix data region.

The image below displays the subjects in a row group. Nested column groups display practical and theory scores for the students. The total row displays the total scores for all of the subjects.

Orders by Category, Quarter, and Year		1994			1995		
		Q 3	Q 4	Total for the Year	Q 1	Q 2	Q 3
Grains/Cereals	Singaporean Hokkien Fried	$98,00	$302,40	$400,40	$448,00	$1 484,00	$1 862,00
	Gustaf's Knäckebröd	$100,80		$100,80	$201,60	$504,00	$3 003,00
	Ravioli Angelo	$1 045,20	$1 029,60	$2 074,80	$546,00	$97,50	$585,00
	Gnocchi di nonna Alice	$60,80	$1 702,40	$1 763,20	$6 870,40	$8 177,60	$8 626,00
	Wimmers gute Semmelknödel	$239,40	$2 261,00	$2 500,40	$2 872,80	$1 010,80	$2 094,75
	Filo Mix		$156,80	$156,80	$308,00	$42,00	$931,00
	Tunnbröd		$468,00	$468,00	$720,00	$1 141,20	$90,00
	CategoryTotal	$1 544,20	$5 920,20	$7 464,40	$11 966,80	$12 457,10	$17 191,75
Dairy Products	Mozzarella di Giovanni	$1 786,40	$3 058,00	$4 844,40	$3 808,60	$2 609,00	$4 906,80
	Queso Cabrales	$168,00	$1 646,40	$1 814,40	$1 209,60	$2 457,00	$1 554,00
	Geitost	$258,00	$16,00	$274,00	$398,00	$196,50	$170,00
	Camembert Pierrot	$3 155,20	$3 889,60	$7 044,80	$4 651,20	$6 426,00	$6 086,00

Tablix Features

Column and Row Groups

Groups categorize the report data using a specified expression. You can add a group by using the context menu options or in the Group Editor component that is part of the designer.

In Tablix, you can add row/column groups in the following ways:

- **Parent-child groups**: To depict the hierarchical relation.
- **Adjacent groups**: To show the side-by-side grouping of report data.

Important Group and Layout Properties for Tablix Groups

- **Filters**: The filters to apply on the group.
- **GroupExpressions**: Specifies the group expression for the group.
- **NewSection**: Determines whether each group instance has its page numbering.
- **PageBreak**: Indicates how the rendering engine inserts a page break in relation to the group.
    - **BreakLocation:** Determines the location of page breaks generated by group instances.
        - None: no page breaks are generated.
        - Start: each group instance inserts the page break before printing its content.

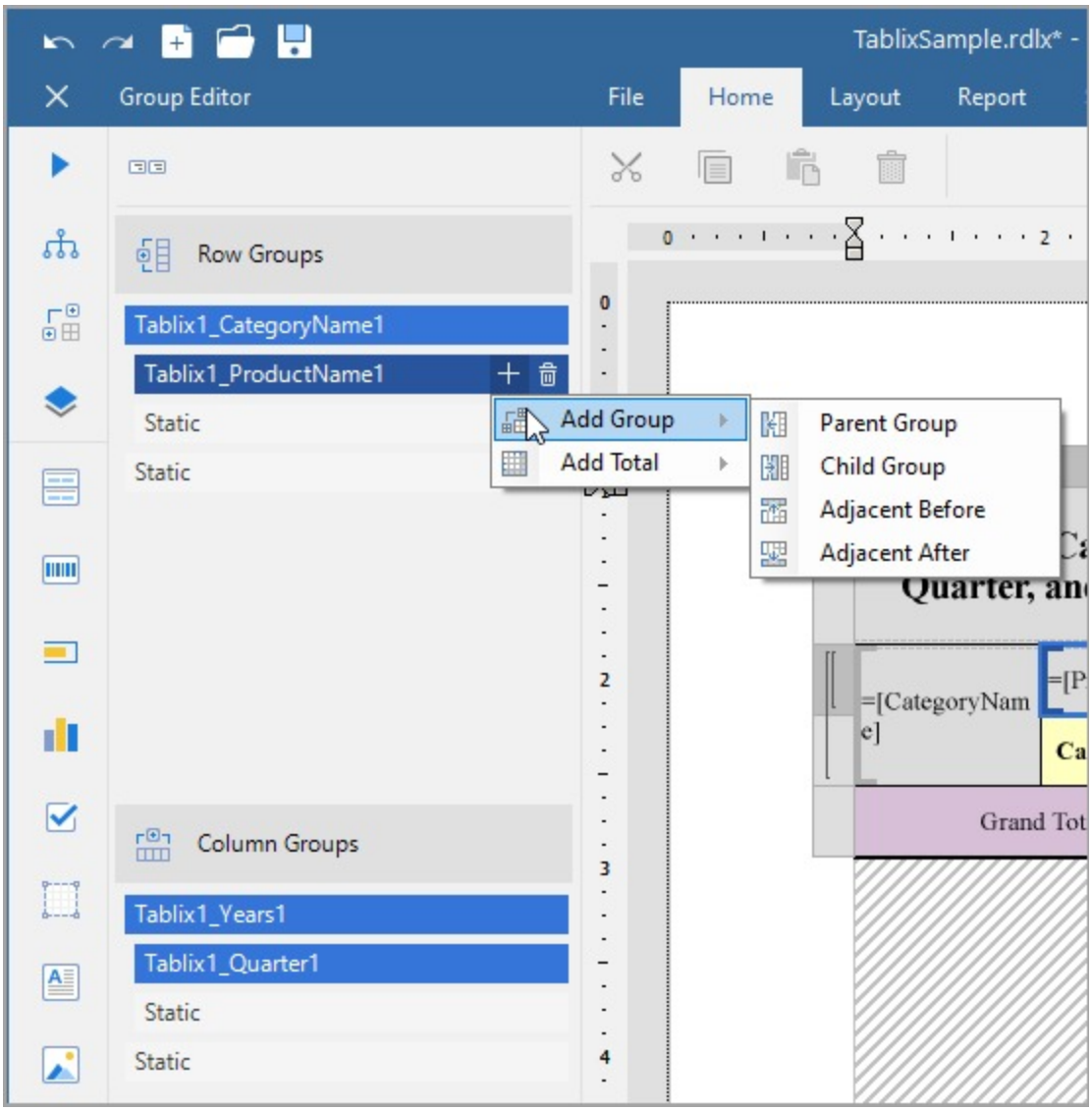
        - End: each group instance inserts the page break after printing its content.
        - StartAndEnd: the combination of the Start and End options.
        - Between: each group instance starts on the new page.
    - **Disabled**: Indicates whether the page break properties should be ignored. The expression allows you to conditionally prevent page breaks from being inserted by the aforementioned property.
    - **NewPage**: Indicates on which page the content to start after the page break.
        - **Next**: A default value that makes a new group start from the immediate next page of the report.
        - **Odd**: A new group starts from the next odd page of the report.
        - **Even**: A new group starts from the next even page of the report.

  See Page Breaks in Data Regions topic for more detail.

- **KeepTogether**: Setting it True ensures that the group instance always appears on a single page if it fits.

To add a group

1. Select the Tablix data region in the design area.
2. Right-click and select any group action in the context menu that appears.
3. Or, with a Tablix selected, click **Group Editor** on the left of the Designer.



Totals

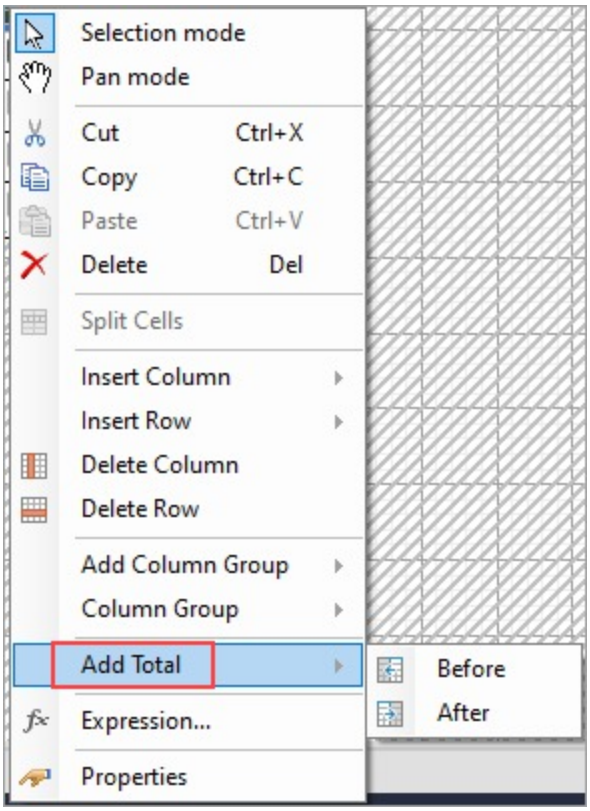
The intersection of a Row and Column Group displays one or more summary values. For example, in a report output, Row Groups may display **media types**, Column Groups - **quarters**, and their intersections may show the **Sales** for each media type in each quarter.

A tablix can display two types of totals.

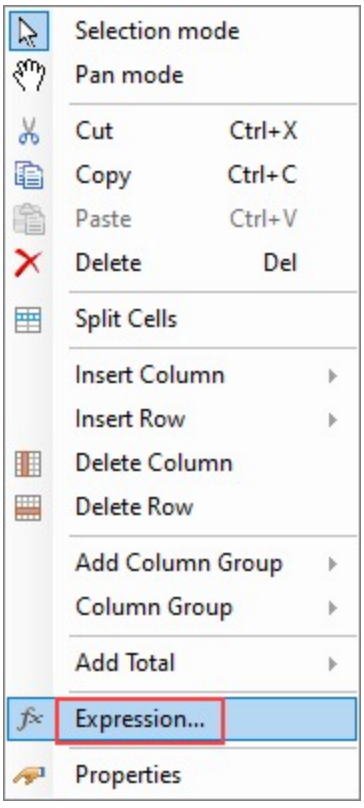
- The **Grand Total** appears at the beginning or at the end of all the group instances.
- The **Subtotal** appears at the beginning or at the end of each group instance.

To add a total,

1. Right-click a Tablix cell and select **Add Total > Before** or **Add Total > After** menu item.
   If a group does not have a parent, then the grand total will be inserted. Otherwise, the subtotal for the parent group will be created.



2. To insert additional totals, you can add rows and columns and set an expression to calculate sum for the newly added cells.



Merge Cells

Tablix cells with the same value in a row group or a column group area are automatically merged. In the case of static cells, you can combine adjacent cells in horizontal (same row) or vertical (same column) direction into a single cell. For example, you may want a column header to span across the columns.

To merge cells,

1. Select the cells (Ctrl+Click) and then right-click.
2. From the context menu, select **Merge Cells**.

Auto Merge

The **AutoMergeMode** property lets you set the mode to merge the adjacent cells (text boxes) in a row group with the same value. This property takes Always, Never, and Restricted values. The row groups with the same data values and with AutoMergeMode property set to:

- **Always** - are merged.
- **Never** - are not merged.
- **Restricted** - are merged only if the corresponding cells in the previous columns are similarly merged. If for example, cells in Column 2 (with the same data values) are set 'Restricted' and the corresponding cells (with the same data values) in previous column, that is Column 1, are set 'Never', then cells in Column 2 are not merged.

Let us take an example. In the following simple Tablix data region, 'District' column is **Outside Group - Right** to the 'Region' column group. We want the 'District' values to merge in case the 'Region' values are the same.
The tablix without AutoMergeMode set in any cell looks as follows:

Store Name	Region	District
HQ	No Region	No District
Store #1000	North West	Portland
Store #1001	Canada West	Victoria
Store #1002	North West	Seattle
Store #1003	Canada West	Vancouver
Store #1004	Canada West	Vancouver
Store #1005	Canada West	Victoria
Store #1006	North West	Portland
Store #1007	Canada West	Vancouver

Let us set the **AutoMergeMode** property for the 'District' and the 'Region' values.

1. Select the cell with `District` field in the row group and set the **Layout > AutoMergeMode** property to 'Restricted'. This merges districts depending on whether the corresponding regions (cells in the previous column) are merged.
2. Select the cell with `Region` field in the row group and set the **Layout > AutoMergeMode** property to 'Always'. This merges the cells with similar regions.

3. Preview the report. Here's how the tablix will look like.

Store Name	Region	District
HQ	No Region	No District
Store #1000	North West	Portland
Store #1001	Canada West	Victoria
Store #1002	North West	Seattle
Store #1003	Canada West	Vancouver
Store #1004		
Store #1005		Victoria
Store #1006	North West	Portland
Store #1007	Canada West	Vancouver

Repeat to Fill (Page report)

If you set the **RepeatToFill** property to True (the default value is False), the Tablix will fill extra space with empty rows.

Setting the **RepeatToFill** property to 'True' (default value is 'False') fills the tablix with empty rows to reach the Fixed Height of the tablix. So, each page of the report displays the tablix with the same height. For example, the following image shows the page of a report where **RepeatToFill** property for the Tablix data region is set to 'False':

USA	Beaverton	DVD	$563,08	$854,94	$1 033,46	$481,12	$2 932,60
		HD-DVD	$64,90	$59,98	$189,74	$33,95	$348,57
		LaserDisc	$403,85	$786,90	$583,61	$641,59	$2 415,95
		VHS	$714,17	$775,68	$948,34	$475,46	$2 913,65
		City Total	$1 746,00	$2 477,50	$2 755,15	$1 632,12	$8 610,77
	Issaquah	DVD	$335,65	$1 104,47	$650,30	$1 848,63	$3 939,05
		HD-DVD		$100,93	$35,99	$85,85	$222,77
		LaserDisc	$209,62	$961,01	$591,73	$801,48	$2 563,84
		VHS	$389,01	$1 024,18	$445,57	$1 284,09	$3 142,85
		City Total	$934,28	$3 190,59	$1 723,59	$4 020,05	$9 868,51
	W. Linn	DVD	$515,38	$864,10	$930,74	$914,54	$3 224,76
		HD-DVD	$29,95	$166,83	$33,95	$115,84	$346,57
		LaserDisc	$661,53	$457,24	$980,70	$649,71	$2 749,18
		VHS	$944,26	$1 133,93	$822,60	$955,62	$3 856,41
		City Total	$2 151,12	$2 622,10	$2 767,99	$2 635,71	$10 176,92
	Country Total		$4 831,40	$8 290,19	$7 246,73	$8 287,88	$28 656,20
Total			$12 492,68	$14 783,74	$15 768,20	$18 266,29	$61 310,91

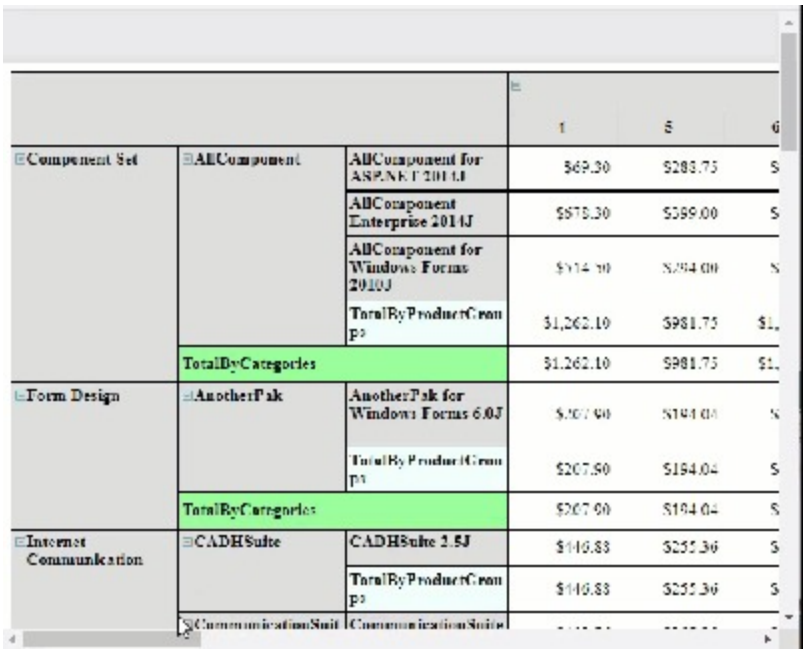
Then, in the Properties window, set the **RepeatToFill** property to 'True'. At the report preview, you will see that the tablix now has additional empty rows on the same page of the report.

USA	Beaverton	DVD	$563,08	$854,94	$1 033,46	$481,12	$2 932,60
		HD-DVD	$64,90	$59,98	$189,74	$33,95	$348,57
		LaserDisc	$403,85	$786,90	$583,61	$641,59	$2 415,95
		VHS	$714,17	$775,68	$948,34	$475,46	$2 913,65
		City Total	$1 746,00	$2 477,50	$2 755,15	$1 632,12	$8 610,77
	Issaquah	DVD	$335,65	$1 104,47	$650,30	$1 848,63	$3 939,05
		HD-DVD		$100,93	$35,99	$85,85	$222,77
		LaserDisc	$209,62	$961,01	$591,73	$801,48	$2 563,84
		VHS	$389,01	$1 024,18	$445,57	$1 284,09	$3 142,85
		City Total	$934,28	$3 190,59	$1 723,59	$4 020,05	$9 868,51
	W. Linn	DVD	$515,38	$864,10	$930,74	$914,54	$3 224,76
		HD-DVD	$29,95	$166,83	$33,95	$115,84	$346,57
		LaserDisc	$661,53	$457,24	$980,70	$649,71	$2 749,18
		VHS	$944,26	$1 133,93	$822,60	$955,62	$3 856,41
		City Total	$2 151,12	$2 622,10	$2 767,99	$2 635,71	$10 176,92
	Country Total		$4 831,40	$8 290,19	$7 246,73	$8 287,88	$28 656,20

Freeze Rows and Columns (RDLX Reports)

When you use a Tablix data region containing a large amount of data in an RDLX report, the user must scroll to see all of the data. On scrolling the column or row headers out of sight, the data becomes difficult to understand. To resolve this problem, you can use the **FrozenRows** and **FrozenColumns** properties that take effect in the JSViewer in Galley mode, and allow you to freeze headers so that they remain visible while scrolling through the data region. You can freeze as many rows or columns as you have headers in the data region.

- If your data stretches downward, set the FrozenRows property to a value to float the column headers when scrolling.
- If your data stretches to the right, set the FrozenColumns property to a value to float the row headers when scrolling.
- If your data stretches both downward and to the right, set both FrozenRows and FrozenColumns properties.



			1	5	6
Component Set	AllComponent	AllComponent for ASP.NET 2014J	$69.30	$283.75	$
		AllComponent Enterprise 2014J	$575.30	$399.00	$
		AllComponent for Windows Forms 2010J	$514.30	$294.00	$
		TotalByProductGroup	$1,262.10	$981.75	$1,
	TotalByCategories		$1,262.10	$981.75	$1,
Form Design	AnotherPak	AnotherPak for Windows Forms 6.0J	$207.90	$194.04	$
		TotalByProductGroup	$207.90	$194.04	$
	TotalByCategories		$207.90	$194.04	$
Internet Communication	CADHSuite	CADHSuite 2.5J	$146.88	$255.36	$
		TotalByProductGroup	$146.88	$255.36	$
	CommunicationSuit	CommunicationSuite			

If any header cells that you want to freeze are merged, you should not set the **FrozenRows** or **FrozenColumns** property to a value that would split a merged cell.

TextBox

The Textbox report control is the most commonly used report control that displays textual data in any report.

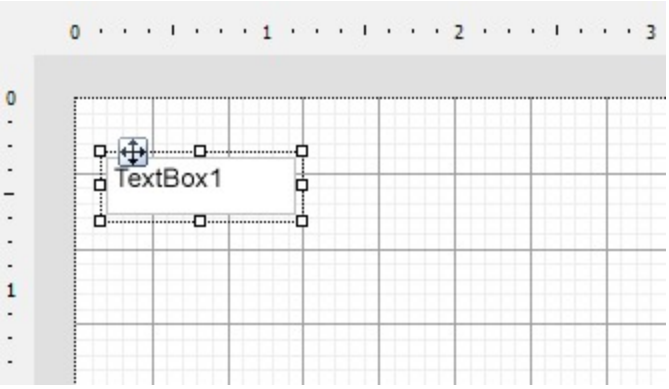
In Page and RDLX reports, the TextBox by default appears in each cell of a Table or Tablix data region. Also, a TextBox is what is created when you drag a field from the data set onto the report. In the **Value** property of the TextBox, you can enter static text or an expression. To enter a text directly into the TextBox, just double-click inside the control on the design surface of the report. An expression in TextBox can display fields from a database, calculate a value, or visually display data.

In the Properties window, there are a number of properties that you can use to control the appearance and behavior of the TextBox. For example, you can set the **Action** property to have the viewer jump to a bookmark within the report, another report, or a URL when a user clicks the TextBox at run time. The **Data Element** properties allow you to control how and whether the TextBox displays in XML exports.

By default, in RDLX reports, the TextBox can grow vertically to accommodate the data it displays, and it cannot shrink smaller than it appears at design time. To change this behavior, set the **CanShrink** and **CanGrow** properties in the Properties grid.

Edit Mode

You can double-click in the TextBox control to enter edit mode and enter text directly in the control, or you can enter text in the Properties window or code through the **Value** property.



You can format text in the TextBox control in edit mode using the ActiveReports toolbar, or you can modify properties in the Properties window. Formats apply to all of the text in the control. Text formatting changes in the Properties window immediately appear in the control, and changes made in the toolbar are immediately reflected in the Properties window.

Important Properties

Clicking the TextBox control reveals its properties in the Properties window.

Property	Description
RDLX report	
CanGrow	Determines whether ActiveReports should increase the height of the control based on its content.
CanShrink	Determines whether ActiveReports should decrease the height of the control based on its value.
CharacterSpacing	Gets or sets a character spacing in points.
LineSpacing	Gets or sets a line spacing in points.
MinCondenseRate	Specifies the minimal rate of the text horizontal scaling in percentages. Should be between 10 and 100.
ShrinkToFit	Determines whether ActiveReports decreases the font size when text values exceed available space.
TextJustify	Specifies text justification with TextAlign set to Justify.
Value	A field, constant, or expression, which value is displayed in the textbox.
VerticalAlignment	Gets or sets the position of the textbox's text vertically within the bounds of the control.
WrapMode	Indicates whether a multi-line textbox control automatically wraps words or characters to the beginning of the next line when necessary.
Page report	
CharacterSpacing	Gets or sets a character spacing in points.
LineSpacing	Gets or sets a line spacing in points.
MinCondenseRate	Specifies the minimal rate of the text horizontal scaling in percentages. Should be between 10 and 100.

ShrinkToFit	Determines whether ActiveReports decreases the font size when text values exceed available space.
TextJustify	Specifies text justification with TextAlign set to Justify.
Value	A field, constant, or expression, which value is displayed in the textbox.
VerticalAlignment	Gets or sets the position of the textbox's text vertically within the bounds of the control.
WrapMode	Indicates whether a multi-line textbox control automatically wraps words or characters to the beginning of the next line when necessary.

TextBox Dialog Properties

You can set the TextBox properties in the TextBox dialog. To open it, with the TextBox selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the textbox that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tooltip: A textual label for the report item used to include TITLE or ALT attributes in HTML reports.

Value: A field, constant, or expression, which value is displayed in the textbox.

> **Note:** When the group or dataset breaks to a new page, the first instance of the repeated value is printed.

Visibility

Initial visibility allows you to select from the following options:

- **Visible:** The textbox is visible when the report runs.
- **Hidden:** The textbox is hidden when the report runs.
- **Expression:** Use an expression with a Boolean result to decide whether the textbox is visible. For example, on a "Free Shipping" textbox, you could use the expression to see whether the ShippingCountry is international. A value of True hides the textbox, False shows it.

Visibility can be toggled by another report item: If you select this check box, it enables the drop-down box where you can specify the TextBox control that users can click to toggle the visibility of the textbox.

Initial appearance of the toggle image: allows you to select from the following options:

- **Expanded:** The toggle image shows as a minus sign, and all instances of this textbox are visible.
- **Collapsed:** The toggle image shows as a plus sign, and all instances of this textbox are hidden.
- **Expression:** Use an expression with a Boolean result to decide whether the toggle image is expanded. A value of True expands the toggle image, False collapses it.

Navigation

Select one of the following actions to perform when a user clicks on the textbox.

None: The default behavior is to do nothing when a user clicks the textbox at run time.

Jump to report: For drill-through reporting, select this option and provide the name of a local report, the relative path of a report in another folder, or the full path of a report on another server. You can also use expressions to create drill-through links.

Parameters: Supply parameters to the targeted report by entering the **Name** of each parameter, the **Value** to send to the targeted report, or whether to **Omit** the parameter. Note that parameter names you supply must exactly match parameters in the target report. You can remove or change the order of parameters using the X and arrow buttons.

Jump to bookmark: Select this option and provide a valid Bookmark ID to allow the user to jump to the report control with that Bookmark ID.

Jump to URL: Select this option and provide a valid URL to create a hyperlink to a Web page.

Apply Parameters: Select the Name, the Type, and the Value of the parameter to set a parameter value through user action. See Actionable Parameters for more information.

Document map label: Enter an expression to use as a label to represent this item in the table of contents (document map).

Bookmark ID: Enter an expression to use as a locator for this textbox. You will then be able to provide a bookmark link to this item from another report control using a **Jump to bookmark** action.

Appearance

Border

Style: Select a style for the border.

Width: Enter a value in points to set the width of the border.

Color: Select a color to use for the border, or select the **<Expression...>** option to open the Expression Editor and create an expression that evaluates to a .NET color.

Background

Color: Select a color to use for the background of the textbox.

Image: Enter an image to use for the background of the textbox.

Image Source: Select whether the image comes from a source that is **External**, **Embedded**, or **Database**, or select the **<Expression...>** option to open the Expression Editor.

MIME Type: Select the MIME type of the chosen image.

Background repeat: Select from the options **Repeat**, **NoRepeat**, **RepeatX**, **RepeatY**, or select the **<Expression...>** option to open the Expression Editor.

Note: The Background Color and Background Image properties allow you to choose the **<Data Visualizer...>** option as well to launch the dialog that let you build a data visualization expression.

Font

Family: Select a font family name or a theme font.

Size: Choose the size in points for the font or use a theme.

Style: Choose **Normal** or **Italic** or select a theme.

Weight: Choose from **Lighter**, **Thin**, **ExtraLight**, **Light**, **Normal**, **Medium**, **SemiBold**, **Bold**, **ExtraBold**, **Heavy**, or **Bolder**.

Color: Choose a color to use for the text.

Decoration: Choose from **None**, **Underline**, **Overline**, or **LineThrough**.

Format

Format code: Select one of the common numeric formats provided or use a custom .NET formatting code to format dates or numbers. For more information, see MSDN's Formatting Types topic.

Line Spacing: This property sets the space between lines of text.

Character Spacing: This property sets the space between characters.

Textbox height

Can increase to accommodate contents (RDLX report): Select this check box to set CanGrow to True.

Can decrease to accommodate contents (RDLX report): Select this check box to set CanShrink to True.

Can shrink text to fit fixed size control: Select this check box to set ShrinkToFit to True.

Text direction and writing mode

Direction: Choose **LTR** for left to right, or **RTL** for right to left.

Mode: Choose **lr-tb** for left right top bottom (normal horizontal text) or **tb-rl** for top bottom right left (vertical text on its side).

Upright: Indicates whether the text is written horizontally in the vertical text. Choose from None, Digits, DigitsAndLatinLetters, or the **<Expression...>** option.

Angle: Enter the number of degrees to rotate the text in a counter-clockwise direction. Enter a negative number to rotate the text in a clockwise direction.

Minimal rate of text horizontal shrinking (in %): Specify the percentage to which the text should be shrunk horizontally.

Alignment

Vertical alignment: Choose **Top**, **Middle**, **Bottom**, or the **<Expression...>** option.

Horizontal alignment: Choose **General**, **Left**, **Center**, **Right**, **Justify**, or the **<Expression...>** option.

Justify method: Set Horizontal alignment to **Justify** to enable this property. Choose **Auto**, **Distribute**, **DistributeAllLines**, or the **<Expression...>** option.

Wrap mode

- **NoWrap:** No text wrap.
- **WordWrap:** Wrap words from one line to another.
- **CharWrap:** Per-character wrapping mode.

Padding

- **Top**: Set the top padding in points.
- **Left**: Set the left padding in points.
- **Right**: Set the right padding in points.
- **Bottom**: Set the bottom padding in points.

Interactive Sort

Select the checkbox next to **Add an interactive sort action to this textbox** to enable the following controls which allow end users to sort the report data in the viewer.

Sort expression: Enter an expression to use for determining the data to sort.

Data region or group to sort: Select the grouping level or data region within the report to sort. The default value is **Current scope**, but you may also elect to choose an alternate data region or grouping.

Evaluate sort expression in this scope: Select the grouping level within the report on which to evaluate an aggregate sorting expression. The default value is **Current scope**, but you may also elect to choose an alternate data region or grouping.

Data Output

Element Name: Enter a name to be used in the XML output for this textbox.

Output: Choose **Auto**, **Yes**, or **No** to decide whether to include this textbox in the XML output. **Auto** exports the contents of the textbox only when the value is not a constant.

Render as: Choose **Auto**, **Element**, or **Attribute** to decide whether to render text boxes as Attributes or Elements in the exported XML file. **Auto** uses the report's setting for this property.

> **Attribute example:** `<table1 textbox3="Report created on: 7/26/2021 1:13:00 PM">`
>
> **Element example:** `<table1> <textbox3>Report created on: 7/26/2021 1:13:28 PM</textbox3>`

Data Fields

When you drag a field from a dataset and drop it onto the report surface, a TextBox report control with an expression is automatically created. The type of expression that is created depends upon the context where you drop the field. The following table describes the various contexts and expressions created if you drag a field named **SalesAmount** onto the report.

Expressions created for fields in different contexts

> **Note**: The expression created is different for a field with a string or unknown data type. In these cases, the **First** aggregate is used in place of the **Sum** aggregate in the expressions below. At run time, the first value found within the scope is displayed instead of a summary.

Context	Expression	Run-Time Behavior
Directly on the report surface	`=Sum(Fields!SalesAmount.Value)`	Displays a summary of the sales amount for the entire dataset.
List data region	`=Fields!SalesAmount.Value`	Displays a value for each row of data, in a list running down the page.
BandedList data region, header, or footer band	`=Sum(Fields!SalesAmount.Value)`	Displays a summary of the sales amount for the dataset associated with the BandedList.
BandedList data region, detail band	`=Fields!SalesAmount.Value`	Displays a value for each row of data, in a list running down the page.
BandedList data region, group header, or footer band	`=Sum(Fields!SalesAmount.Value)`	Displays a summary of the sales amount for the grouping.
Table data region, header, or footer row	`=Sum(Fields!SalesAmount.Value)`	Displays a summary of the sales amount for the dataset associated with the Table.
Table data region, detail row	`=Fields!SalesAmount.Value`	Displays a value for each row of data, in a list running down the page.
Table data region, group header or footer row	`=Sum(Fields!SalesAmount.Value)`	Displays a summary of the sales amount for the grouping.
Tablix data region, corner cell	none	Displays a blank cell. You can add a label or even use this area to embed other report control.
Tablix data region, column group cell	`=Fields!SalesAmount.Value`	Displays the value at the top of a new column for each row of data running to the right.
Tablix data region, row group cell	`=Fields!SalesAmount.Value`	Displays the value to the left of a new row for each row of data running down the page.
Tablix data region, body cell	`=Sum(Fields!SalesAmount.Value)`	Displays a summary of the sales amount for the intersection of the column and row.

TextBox Features

Text Justification

The **TextJustify** property of a Textbox control provides you justification options for aligning your text within a control. It is important to note that the **TextAlign** property must be set to Justify for **TextJustify** property to affect the text layout.

You can choose from the following values of the TextJustify property:

Auto

Results in Standard MSWord like justification where space at the end of a line is spread across other words in that line. This is the default value.

Distribute

Spaces individual characters within a word, except for the last line.

DistributeAllLines

Spaces individual characters within words and also sets the justification on the last line according to the length of other lines.

To set text justification,

1. Select the **TextBox** control to view its properties in the Properties window.
2. In the Properties window, set the **TextAlign** property
3. Go to the **TextJustify** property and from the drop down list select any one option.

Text justification is supported when you preview a report in the Viewer, print a report, export a report in PDF, and TIFF formats, or render a report in Word, HTML, PDF and Image formats using rendering extensions.

Shrink Text to Fit in a Control

When working with the Textbox control in a Page report and RDLX report, you can use the **ShrinkToFit** property to reduce the size of the text so that it fits within the bounds of the control. The text shrinks at run time, so you can see the reduced font size when you preview, print or export the report.

The following image illustrates the result when the **ShrinkToFit** property is set to True on 'Title' field.

Title	Year Released	User Rating
Terminator 2: Judgment Day	1991	9.4
Terminator 3: Rise of the Machines	2003	9
The Aviator	2004	9.4
The Bourne Identity	2002	9.5
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	2005	9
The Flintstones	1994	9.3
The Fugitive	1993	9.3
The Grudge	2004	9.7
The Hunt for Red October	1990	9.4
The Karate Kid, Part II	1986	9.3
The Longest Yard	2005	9.9
The Lord of the Rings: The Fellowship of the Ring	2001	10
The Lost World: Jurassic Park	1997	9.6
The Matrix	1999	9.5
The Patriot	2000	9
The Perfect Storm	2000	9.1
The Polar Express	2004	9.9

You can use other text formatting properties in combination with the **ShrinkToFit** property.

⚠ **Caution**:

- When both **CanGrow** and **ShrinkToFit** are set to True, CanGrow setting is ignored and only ShrinkToFit is applied.
- When **ShrinkToFit** is set to True and **Angle** is set to a value other than 0, the ShrinkToFit property is ignored.
- **ShrinkToFit** property does not work when the WritingMode property is set for a control.
- Common value with Page number (in data region or report header/footer) appears clipped in exported files (HTML, MHT, Word, and Excel) even when **ShrinkToFit** property for the TextBox is set to True.

On exporting a report, various file formats handle **ShrinkToFit** differently. ShrinkToFit gets exported in all formats except **Text**. While rendering a Page report or RDLX report using rendering extensions, ShrinkToFit is not supported in **XML**. However, all other rendering extensions allow ShrinkToFit to display as it is.

Multi-line in TextBox

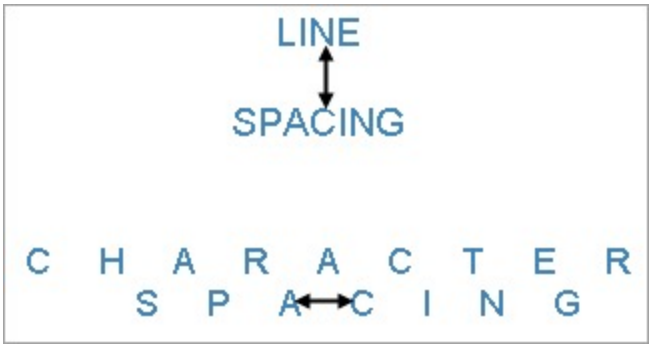
You can display multi-line text in TextBox and some other controls such as CheckBox for Page/RDLX reports.

In a Page/RDLX report, with your control in edit mode, insert line breaks at the desired location using the **Enter** key or **Ctrl** + **Enter** key to create multi-line text. You can also insert line breaks in the Expression Editor through the **Value** property of the control.

> **Note**: In edit mode, scrollbars appear automatically to fit multi-line content within a control. However, these are not displayed at preview, so you may need to adjust the **Size** property of the control to display all of the text.

Line Spacing and Character Spacing

In edit mode, scrollbars appear automatically to fit multi-line content within a control. However, these are not displayed at preview, so you may need to adjust the **Size** property of the control to display all of the text.



To set Line or Character spacing,

1. On the design surface, click the TextBox control to display it in the Properties window.
2. In the Properties window, click the **Property dialog** command at the bottom to open the control dialog.
3. In the TextBox dialog, go to the **Format** page and set the Line Spacing or Character Spacing values in points.

Line and character spacing are supported when you preview a report, print a report, or export a report. It is also supported while rendering a report through rendering extensions in Word, HTML, PDF and Image formats.

Keyboard Shortcuts

In the edit mode, you can use the following keyboard shortcuts.

Key Combination	Action
Enter	New line.
Alt + Enter	Saves modifications and exits edit mode.
Esc	Cancels modifications and exits edit mode.

In Visual Studio Integrated Designer, you can disable this feature in the **EditModeEntering Event (on-line documentation)** and **EditModeExit Event (on-line documentation)**.

Section Report

When you work on **Section Reports** in ActiveReports Designer, you get a toolbox group with a set of report controls

that can be used to create section reports. These controls can be easily dragged from the toolbox and dropped on to the reports. All the controls in the toolbox have been designed to help you create purposeful section reports.

The following controls are most commonly used to create Section Reports:

Barcode

Learn how to use BarCode control in Section Reports and choose from several barcode styles available.

Chart

Learn about

CheckBox

Learn how to use checkbox in Section Reports.

Cross Section Controls

Learn about cross section controls and how to use them in section reports.

InputFieldCheckBox

Learn how to use InputFieldCheckBox control in Section Reports.

InputFieldText

Learn how to use InputFieldText report control for editable text fields in an exported PDF reports.

Label

Learn how to use labels in Section Reports to display descriptive text for controls.

Line

Learn how to use lines to draw boundaries or highlights specific areas in a report.

OleObject

Learn about

Picture

Learn how to use Picture control and displays images files on the screen in section reports.

Report Info

Learn how to use the ReportInfo control to quickly display page numbers, page counts and report dates in section reports.

RichTextBox

Learn how to use the RichTextBox control to enter rich text in the form of formatted text, tables, hyperlinks, images, etc. in section reports.

Shape

Learn how to use shapes in section reports.

SubReport

Learn how to use the subreport control to connect the separate report to the subreport control in section reports.

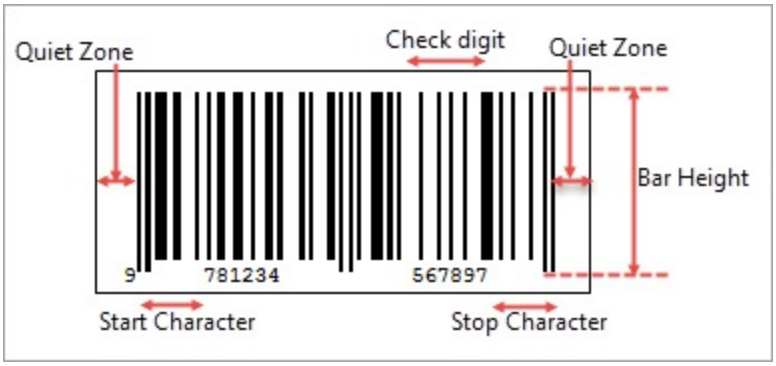
TextBox

Learn how to use textbox control in Section Reports.

Barcode

The Barcode report control, consisting of parallel bars and lines, is used to represent the data in a machine-readable format. This enables accurate scanning of sensitive information quickly and efficiently, and saves you the time and expense of finding and integrating a separate component.

Structure



Quiet zone is the left and right ends of the barcode. Both of the ends must be at least 10 times as wide as the minimum element width for proper scanning of the barcode data.

Start character indicates the start of the barcode data.

Stop character indicates the end of the barcode data.

Check digit is a numeric value used to check for read errors. It is located right after the barcode data.

Bar height is recommended to be greater than 15% of the barcode length. If the bar height is not high enough, it may lead to unstable readings of the barcode by the laser.

Barcode Types

ActiveReports supports all of the most popular symbologies.

Symbology Name	Example	Description
Ansi39		ANSI 3 of 9 (Code 39) uses uses upper case alphabets (A to Z), numerals (0 to 9), - , * $ / + %. This is the default barcode style.
Ansi39x		ANSI Extended 3 of 9 (Extended Code 39) uses the complete ASCII character set.
Aztec		Aztec is a two-dimensional barcode symbology that supports all the ASCII characters from 0 to 255.

BC412		Data BC412 uses 35 characters, numerals (0 to 9), and upper case alphabets (A to Z). It is designed for semiconductor wafer identification.
Codabar		Data that can be encoded are numerals (0 to 9) and symbols ("-", "$", ":", "/", "+" and ";"). For the Start Character and the Stop Character, A, B, C or D can be selected.
Code_11		Encodes the numerals (0 to 9), the hyphen (-), and start/stop characters. It is primarily used in labeling telecommunications equipment.
Code_128_A		Code 128 A uses control characters, numerals (0 to 9), punctuation, and upper case alphabets (A to Z).
Code_128_B		Code 128 B uses punctuation, numerals (0 to 9), upper case, and lower case alphabets.
Code_128_C		Code 128 C uses only numerals (0 to 9).
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_2_of_5		Code 2 of 5 uses only numerals (0 to 9).
Code_93		Code 93 uses upper case alphabets (A to Z), % $ * / , + -, and numerals (0 to 9).
Code25intlv		Interleaved 2 of 5 uses only numerals (0 to 9).
Code39		Code 39 uses numerals (0 to 9), % * $ /. , - +, and upper case alphabets (A to Z).

Code39x		Extended Code 39 uses the complete ASCII character set.
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.
Code93x		Extended Code 93 uses the complete ASCII character set.
DataMatrix		Data Matrix is a high-density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern.
EAN_13		EAN-13 uses only numerals (12 numbers and a check digit). It takes only 12 numbers as a string to calculate a check digit (CheckSum) and add it to the thirteenth position. The check digit is an additional digit used to verify that a bar code has been scanned correctly. The check digit is added automatically when the CheckSum property is set to True.
EAN_8		EAN-8 uses only numerals (7 numbers and a check digit).
EAN128FNC1*		EAN-128FNC1 is an alphanumeric one-dimensional representation of Application Identifier (AI) data for marking containers in the shipping industry.
GS1QRCode		GS1QRCode is a subset of the QR Code. The GS1 QR Code is a 2D symbol that denotes the Extended Packaging URL for a trade item. It is processed to obtain one URL address associated with the trade item identified by the Global Trade Item Number (GTIN). GS1 QR Code requires the mandatory association of the GTIN

		and Extended Packaging URL. GS1 QR Code allows encoding GS1 System Application Identifiers (AI) into QR Code 2D barcodes. **Limitation:** Kanji, CN, JP and Korean characters.
HIBCCode128		HIBCCode128 barcode uses the Code128 symbology. It encodes 'Primary Data' and 'Secondary Data' using slash (/) as a delimiter. It is used in the health care products industry for identification purposes.
HIBCCode39		HIBCCode39 barcode uses the Code39 symbology, with the Code39OptionalCheckDigit property set to True. It encodes Primary Data and Secondary Data using slash (/) as a delimiter. It is used in the health care products industry for identification purposes.
IATA_2_of_5		IATA_2_of_5 is a variant of Code_2_of_5 and uses only numerals with a check digit.
IntelligentMail		Intelligent Mail, formerly known as the 4-State Customer Barcode, is a 65-bar code used for domestic mail in the U.S.
IntelligentMailPackage		IntelligentMailPackage is more efficient in terms of processing and tracking mails than Intelligent Mail barcode.
ISBN		International Standard Book Number barcode is a special form of the EAN-13 code and is used as a unique 9-digit commercial book identifier.
ISMN		Internationally Standard Music Number barcode is a special form of the EAN-13 code. It is used for marking printed musical publications.
ISSN		International Standard Serial Number barcode is a special form of the EAN-13 code. It is used to identify serial publications, publications that are issued

		in numerical order, such as the volumes of a magazine.
ITF14		Interleaved Two of Five code is used to mark cartons that contain goods with an EAN-13 code. One digit is added in front of the EAN-13 code to mark the packing variant.
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.
Matrix_2_of_5		Matrix 2 of 5 is a higher density barcode consisting of 3 black bars and 2 white bars.
MaxiCode		MaxiCode is a special polar barcode that uses 256 characters. It is used to encode a specific amount of data.
MicroPDF417		MicroPDF417 is two-dimensional (2D), multi-row symbology, derived from PDF417. Micro-PDF417 is designed for applications that need to encode data in a two-dimensional (2D) symbol (up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits) with the minimal symbol size. MicroPDF417 allows you to insert an FNC1 character as a field separator for variable-length Application Identifiers (AIs). To insert an FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at run time.

MicroQRCode		MicroQRCode is a two-dimensional (2D) barcode that is designed for applications that use a small amount of data. It can handle numeric and alphanumeric data as well as Japanese kanji and kana characters. This symbology can encode up to 35 numeric characters.
MSI		MSI Code uses only numerals (0 to 9).
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. This symbology can encode up to 35 alphanumeric characters or 2,710 numeric characters.
Pharmacode		Pharmacode represents only numeric data from 3 to 131070. It is a barcode standard used in the pharmaceutical industry for packaging. It is designed to be readable despite printing errors.
Plessey		Plessey uses hexadecimal digits to encode. It is a one-dimensional barcode used mainly in libraries.
PostNet		PostNet uses only numerals (0 to 9) with a check digit.
PZN		Pharmaceutical Central/General Number uses the same encoding algorithm as Code 39 but can carry only digits – 0123456789. The number of digits supported for encoding are 6 or 7. The letters 'PZN' and checksum digit are automatically added. It is mainly used to identify medicine and health-care products in Germany and other German-speaking countries.

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.
RM4SCC		Royal Mail (RM4SCC) uses only letters and numerals (with a check digit). This is the barcode used by the Royal Mail in the United Kingdom.
RSS14 (GS1 DataBar)		RSS14 is a 14-digit Reduced Space Symbology that uses EAN.UCC item identification for point-of-sale omnidirectional scanning. The RSS family of barcodes is also known as **GS1 DataBar**.
RSS14Stacked (GS1 DataBar Stacked)		RSS14Stacked uses the EAN.UCC information with Indicator digits as in the RSS14Truncated, but stacked in two rows for a smaller width. RSS14Stacked allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.
RSS14StackedOmnidirectional (GS1 DataBar Stacked Omnidirectional)		RSS14StackedOmnidirectional uses the EAN.UCC information with omnidirectional scanning as in the RSS14, but stacked in two rows for a smaller width.
RSS14Truncated (GS1 DataBar Truncated)		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.
RSSExpanded (GS1 DataBar Expanded)		RSSExpanded uses the EAN.UCC information as in the RSS14, but also adds AI elements such as weight and best-before dates. RSSExpanded allows you to

		insert an FNC1 character as a field separator for variable-length Application Identifiers (AIs). To insert an FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at run time.
RSSExpandedStacked (GS1 DataBar Expanded Stacked)		RssExpandedStacked uses the EAN.UCC information with AI elements as in the RSSExpanded, but stacked in two rows for a smaller width. RSSExpandedStacked allows you to insert an FNC1 character as a field separator for variable-length Application Identifiers (AIs). To insert FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at run time.
RSSLimited (GS1 DataBar Limited)		RSS Limited 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. RSSLimited allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.
SSCC_18		SSCC_18 is an 18-digit Serial Shipping Container Code. It is used to identify individual shipping containers for tracking purposes.
Telepen		Telepen has 2 different modes - alphanumeric-only and numeric-only. Both modes require a start character, a check digit, and a stop character. It is mainly used in manufacturing industries.
UCCEAN128		UCCEAN-128 complies to GS1-128 standards. GS1-128 uses a series of Application Identifiers to encode data. This barcode uses the complete ASCII character set. It also uses the FNC1 character as the first character position. Using AI's, it encodes best before dates, batch numbers, weights, and more such attributes. It is also used in HIBC

		applications.
UPC_A		UPC-A uses only numerals (11 numbers and a check digit).
UPC_E0		UPC-E0 uses only numerals. 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 numerals. Used typically for shelf labeling in the retail environment. The length of the input string for U.P.C. E1 is six numeric characters.

Note that the following barcodes support FNC1 characters:

- Aztec
- UCCEAN128
- MicroPDF417
- RSSExpanded
- RSSExpandedStacked

*The barcode EAN128FNC1 is now obsolete; you should use **UCCEAN128** instead. The UCCEAN128 provides similar functionality with better performance and some default properties. You should choose EAN128FNC1 only if you do not need the default properties.

Important Properties

Clicking the barcode control reveals its properties in the Properties window.

Property	Description
BarHeight	Set the height, in inches, of the barcode's bars. If the bar height exceeds the height of the control, this property is ignored.
CaptionGrouping	Gets or sets a value indicating whether to add spaces between groups of characters in the caption to make long numbers easier to read. This property is only available with certain styles of barcode and is ignored with other styles.
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.
Font	Set the font for the caption. Only takes effect if you set the CaptionPosition property to a value other than None.
NarrowBarWidth	Also known as the X dimension, this is a value defining the width of the narrowest part of the barcode. Before using an extremely small value for this width, ensure that the scanner can read it. This value is specified in pixels (for example, 10 pixels).
NWRatio	Also known as the N dimension, this is a value defining the multiple of the ratio between the narrow and wide bars in symbologies that contain bars in only two widths. For example, if it is a 3 to 1 ratio, this value is 3.
QuietZone	Sets an area of blank space on each side of a barcode that tells the scanner where the symbology starts and stops. You can set separate values for the Left, Right, Top, and Bottom.
Rotation	Sets the amount of rotation to use for the barcode. You can select from None, Rotate90Degrees, Rotate180Degrees, or Rotate270Degrees.
SupplementOptions	Sets the 2/5-digit add-ons for EAN/UPC symbologies. You can specify Text, DataField, BarHeight, CaptionPosition, and Spacing for the supplement.
BackColor	Select a background fill color for the barcode.
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.
ForeColor	Select a color for the barcode and caption.
Style	Sets the symbology used to render the barcode. See the table below for details about each style.
AutoSize	When set to True, the barcode automatically stretches to fit the control. ⚠ **Caution:** This property works only with few barcode types and can break layout, so should be used very carefully.
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 Specific Properties

Aztec Options

Aztec options are available for the Aztec barcode style.

Error Correction: Indicates whether to allow code recovery if the barcode image is partly damaged, the value is ranging from 10% to 90%.

Layers: Indicates the number of the barcode layers.

Encoding: Select the barcode encoding from the drop-down list.

Code49 Options

Code49 options are available for the Code49 barcode style.

Use Grouping: Indicates whether to use grouping for the Code49 barcode. The possible values are **True** or **False** (default). If Grouping is set to True, any value not expressed by a single barcode is expressed by splitting it into several barcodes.

GroupNumber: Enter a number between 0 (default) and 8 for the barcode grouping. 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.

DataMatrix Options

DataMatrix options are available for the DataMatrix barcode style.

EccMode: Select the Ecc mode from the drop-down list. The possible values are **ECC000**, **ECC050**, **ECC080**, **ECC100**, **ECC140** or **ECC200**.

Ecc200 Symbol Size: Select the size of the ECC200 symbol from the drop-down list. The default value is **SquareAuto**.

Ecc200 Encoding Mode: Select the encoding mode for ECC200 from the drop-down list. The possible values are **Auto**, **ASCII**, **C40**, **Text**, **X12**, **EDIFACT**, or **Base256**.

Ecc000_140 Symbol Size: Select the size of the ECC000_140 barcode symbol from the drop-down list.

Structured Append: Select whether the barcode symbol is part of the structured append symbols. The possible values are **True** or **False**.

Structure Number: Enter the structure number of the barcode symbol within the structured append symbols.

File Identifier: Enter the file identifier of a related group of the structured append symbols. If you set the value to 0, the file identifier symbols are calculated automatically.

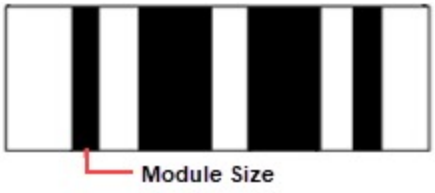
Encoding: Select the barcode encoding from the drop-down list.

EAN128FNC1 Options

EAN128FNC1 options are available for the EAN128FNC1 barcode style.

DPI: Specify the printer resolution.

Module Size: Enter the horizontal size of the barcode module.



Bar Adjust: Enter the adjustment size by dot units, which affects the size of the module and not the entire barcode.

GS1DataMatrix Options

GS1DataMatrix options are available for the GS1DataMatrix barcode style.

EccMode: Select the Ecc mode from the drop-down list. The possible values are **ECC000**, **ECC050**, **ECC080**, **ECC100**, **ECC140** or **ECC200**.

Ecc200 Symbol Size: Select the size of the ECC200 symbol from the drop-down list. The default value is **SquareAuto**.

Ecc200 Encoding Mode: Select the encoding mode for ECC200 from the drop-down list. The possible values are **Auto**, **ASCII**, **C40**, **Text**, **X12**, **EDIFACT**, or **Base256**.

Ecc000_140 Symbol Size: Select the size of the ECC000_140 barcode symbol from the drop-down list.

Structured Append: Select whether the barcode symbol is part of the structured append symbols. The possible values are **True** or **False**.

Structure Number: Enter the structure number of the barcode symbol within the structured append symbols.

File Identifier: Enter the file identifier of a related group of the structured append symbols. If you set the value to 0, the file identifier symbols are calculated automatically.

Encoding: Select the barcode encoding from the drop-down list.

GS1QRCode Options

GS1QRCode options are available for the GS1QRCode barcode style.

ErrorLevel: Select the error correction level for the barcode from the drop-down list. Valid values are **M**, **L**, or **Q**. The available Error Level values change depending on the version you select.

Version: Enter the version of the MicroQRCode barcode style. Valid values are **M1**, **M2**, **M3**, or **M4.** Version M4 stores maximum amount of data.

Mask: Select the pattern for the barcode masking from the drop-down list. Valid values are **Mask00**, **Mask01**, **Mask10**, or **Mask11**.

Encoding: Select the barcode encoding from the drop-down list.

Composite Options

GS1Composite options are available for the RSS14Stacked and RSSLimited barcode styles.

Type: Select the type of the composite barcode from the drop-down list. The possible values are **None** or **CCA**. CCA (Composite Component - Version A) is the smallest variant of the 2-dimensional composite component.

Value: Enter the expression to set the value of the composite barcode.

MaxiCode Options

MaxiCode option is available for the MaxiCode barcode.

Mode: Select the mode of the MaxiCode barcode. The available values are Mode2 to Mode6.

MicroPDF417 Options

MicroPDF417 options are available for the MicroPDF417 barcode style.

Compaction Mode: Select the type of the compaction mode from the drop-down list. The possible values are **Auto**, **TextCompactionMode**, **NumericCompactionMode,** or **ByteCompactionMode**.

Version: Select the version from the drop-down box to set the symbol size.

Segment Index: The segment index of the structured append symbol. The valid value is from 0 to 99998, and

less than the value in Segment Count.

Segment Count: The segment count of the structured append symbol. The valid value is from 0 to 99999.

File ID: The file id of the structured append symbol. The valid value is from 0 to 899.

MicroQRCode Options

MicroQRCode options are available for the MicroQRCode barcode style.

ErrorLevel: Select the error correction level for the barcode from the drop-down list. Valid values are **M**, **L**, or **Q**. The available Error Level values change depending on the version you select.

Version: Enter the version of the MicroQRCode barcode style. Valid values are **M1**, **M2**, **M3**, or **M4.** Version M4 stores maximum amount of data.

Mask: Select the pattern for the barcode masking from the drop-down list. Valid values are **Mask00**, **Mask01**, **Mask10**, or **Mask11**.

Encoding: Select the barcode encoding from the drop-down list.

PDF417 Options

PDF417 options are available for the Pdf417 barcode style.

Columns: Enter column numbers for the barcode. Values for this property range from 1 to 30. The default value is -1 which automatically determines column numbers.

Rows: Enter row numbers for the barcode. Values range between 3 and 90. The default value is -1 which automatically determines row numbers.

ErrorLevel: Enter 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.

PDF 417 Barcode Type: Select the PDF417 barcode type from the drop-down list. The possible values are **Normal** or **Simple**. Simple is the compact type in which the right indicator is neither displayed nor printed.

QRCode Options

QRCode options are available for the QRCode barcode style.

Model: Select the model for the QRCode barcode style from the drop-down list. The possible values are **Model1**, the original model or **Model2**, the extended model. For **GS1QRCode**, Model1 is not supported.

ErrorLevel: Select the error correction level for the barcode from the drop-down list. The possible values are **L** (7% restorable), **M** (15% restorable), **Q** (25% restorable), and **H** (30% restorable). The higher the percentage, the larger the barcode becomes.

Version: Enter the version of the QRCode barcode style. 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.

Mask: Select the pattern for the barcode masking from the drop-down list. 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) mod 2 = 0

- Mask001 i mod 2 = 0
- Mask010 j mod 3 = 0
- Mask011 (i+j) mod 3 = 0
- Mask100 ((i div 2)+(j div 3)) mod 2 = 0
- Mask101 (ij) mod 2 + (ij) mod 3 = 0
- Mask110 ((ij) mod 2 +(ij) mod 3) mod 2 = 0
- Mask111 ((ij)mod 3 + (i+j) mod 2) mod 2 = 0

Use Connection: Select whether to use the connection for the barcode. The possible values are **True** or **False**. This property is used in conjunction with the ConnectionNumber property. This property does not apply to the GS1QRCode barcode.

ConnectionNumber: Enter the connection number for the barcode. 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. This property does not apply to the GS1QRCode barcode.

Encoding: Select the barcode encoding from the drop-down list.

RssExpandedStacked Options

RssExpandedStacked options are available for the RSSExpandedStacked barcode style.

Row Count: Enter the number of the barcode stacked rows.

UPC Supplementary Options

Supplementary options are available for UPC_A, UPC_E0, UPC_E1, EAN_13, and EAN_8 barcode styles.

Supplement DataField: Select the data field for the barcode supplement.

Supplement Value: Enter the expression to set the value of the barcode supplement.

Caption Location: Select the location for the supplement caption from the drop-down list. The possible values are **None**, **Above**, or **Below**.

Supplement Bar Height: Enter the bar height for the barcode supplement.

Supplement Spacing: Enter the spacing between the main and supplement barcodes.

Barcode Dialog Properties

You can set the Barcode properties in the Barcode dialog. To open it, with the Barcode selected in the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the control that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

DataField: Select a field from the data source to bind to the control.

Text: Enter static text to show in the textbox. If you specify a DataField value, this property is ignored.

Autosize: Clear this check box to prevent the barcode from automatically resizing to fit the control.

> ⚠ **Caution:** This property works only with few barcode types and can break layout, so should be used very carefully.

Caption

Location: Select a value to indicate whether and where to display a caption for the barcode. You can select from Above, Below, or None.

Text alignment: Select a value to indicate how to align the caption text. You can select from Center, Near, or Far.

Barcode Settings

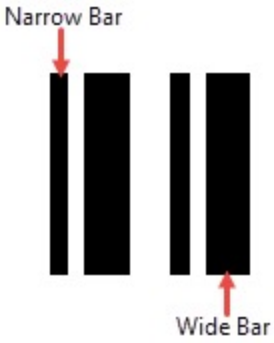
Symbology: Enter the type of barcode to use. ActiveReports supports all of the most popular symbologies.

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

When you choose a symbology which offers supplemental options, the additional options appear after the general collection of properties.

Bar Height: Enter a value in inches (for example, .25in) for the height of the barcode.

Narrow Bar Width (also known as X dimension): Enter a value in points (for example, 0.8pt) for the width of the narrowest part of the barcode. Before using an extremely small value for this width, ensure that the scanner can read it.



> **Tip**: For accurate scanning, the quiet zone should be ten times the Narrow Bar Width value.

Narrow Width Bar Ratio: Enter a value to define the multiple of the ratio between the narrow and wide bars in symbologies that contain bars in only two widths. For example, if it is a 3 to 1 ratio, this value is 3. Commonly used values are 2, 2.5, 2.75, and 3.

QuietZone

A quiet zone is an area of blank space on either side of a barcode that tells the scanner where the symbology starts and stops.

Left: Enter a size in inches of blank space to leave to the left of the barcode.

Right: Enter a size in inches of blank space to leave to the right of the barcode.

Top: Enter a size in inches of blank space to leave at the top of the barcode.

Bottom: Enter a size in inches of blank space to leave at the bottom of the barcode.

> **Note**: The units of measure listed for all of these properties are the default units of measure used if you do not specify. You may also specify **cm**, **mm**, **in**, **pt**, or **pc**.

Checksum

A checksum provides greater accuracy for many barcode symbologies.

Compute Checksum: Select whether to automatically calculate a checksum for the barcode.

> **Note**: If the symbology you choose requires a checksum, setting this value to **False** has no effect.

For more information, see the section on **Barcode Specific Properties**.

Appearance

Fore color: Select a color to use for the bars in the barcode.

Background color: Select a color to use for the background of the control.

Rotation: Select a value indicating the degree of rotation to apply to the barcode. You can select from None, Rotate90Degrees, Rotate180Degrees, or Rotate270Degrees.

Font

Name: Select a font family name to use for the caption.

Size: Choose the size in points for the font.

Style: Choose from **Normal** or **Italic**.

Weight: Choose from **Normal** or **Bold**.

Decoration: Select check boxes for **Underline** and **Strikeout**.

GDI Charset: Enter a value to indicate the GDI character set to use. For a list of valid values, see MSDN Font.GDICharSet Property.

GDI Vertical: Select this checkbox to indicate that the font is derived from a GDI vertical font.

Chart

In ActiveReports, you can use the **Chart** data region to present data graphically in a report. The chart offers you 17 core chart types along with all of their variations, plus access to properties that control every aspect of your chart's appearance.

The Chart data region presents a series of points in different ways depending upon the chart type you choose. Some chart types display multiple series of data points in a single chart. Add more information to your chart by configuring data points, axes, titles, and labels. You can modify all of these elements in the Properties Panel.

When you first drop a Chart data region onto a report, the Chart Wizard appears, and you can set up your chart type, appearance, series, titles, axes, and legend on the pages of the wizard. You can specify a data source on the Series page.

Important Properties

Property	Description
BlackAndWhiteMode	Gets or sets a value indicating whether the chart is drawn in black and white using hatch patterns and line dashing to designate colors.
AutoRefresh	Gets or sets a value indicating whether the chart is automatically refreshed (redrawn) after every property change.

Backdrop	Gets or sets the chart's background style.
ChartAreas	Opens the ChartArea Collection Editor where you can set properties such as axes and wall ranges, and you can add more chart areas.
ChartBorder	Gets or sets the chart's border style.
ColorPalette	Gets or sets the chart's color palette.
DataSource	Gets or sets the data source for the chart.
GridLayout	Gets or sets the layout of the chart's areas in columns and rows.
Legends	Opens the Legend Collection Editor where you can set up the chart's legends.
Series	Opens the Series Collection Editor where you can set up the series collection for the chart.
Titles	Opens the Titles Collection Editor where you can set up titles in the header and footer of the chart.
UIOptions	Gets or sets user interface features for the chart. Choose from None, ContextCustomize, UseCustomTooltips, or ForceHitTesting.
Culture	Gets or sets the chart's culture used for value output formatting.
ImageType	Sets or returns the image generated by the chart. Choose from Metafile or PNG.

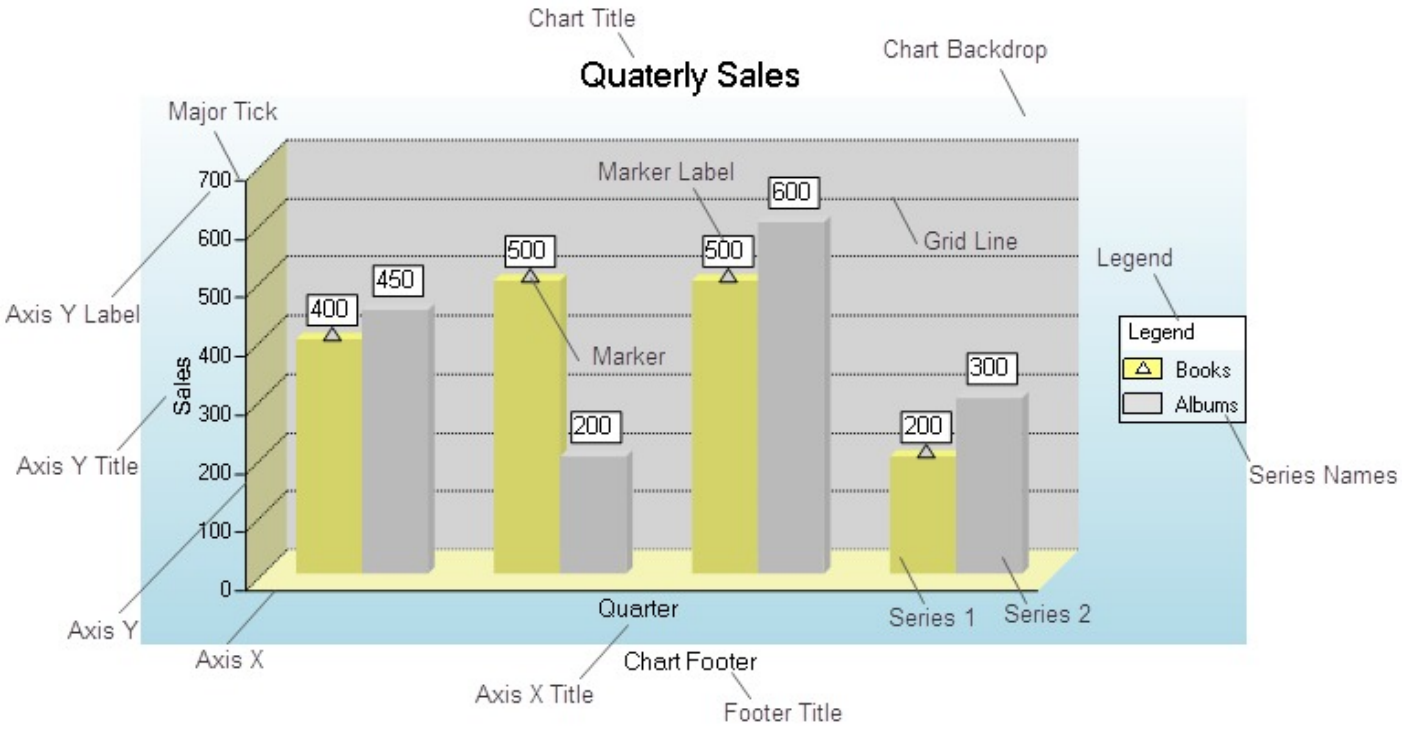
Chart Commands and Dialogs

With the control selected on the report, in the Commands section at the bottom of the Properties window, you can click any of the commands to open a dialog. Commands in this section include:

- **Clear Chart** clears all of the property settings from the chart so that you can begin with a clean slate. You are given an opportunity to cancel this action.
- **Load** allows you to load a saved XML file containing a chart that you created using the Chart data region.
- **Save As** allows you to save the current chart to an XML file that you can load into a chart on any Section Report.
- **Customize** opens the main Chart Designer dialog where you can access Chart Areas, Titles, Series, Legends, and Appearance tabs. This dialog has access to more of the customizable areas than the wizard, but all of the properties in this dialog are also available in the Properties window.
- **Wizard** reopens the Chart Wizard that appears by default when you first drop a Chart data region onto a report.
- **Data Source** opens the Chart Data Source dialog where you can build a connection string and create a query.

Chart Elements

The Chart elements help you to easily analyze the visual information and interpret numerical and relational data. The following image illustrates the elements that make up the Chart data region.



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.

Footer Title

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

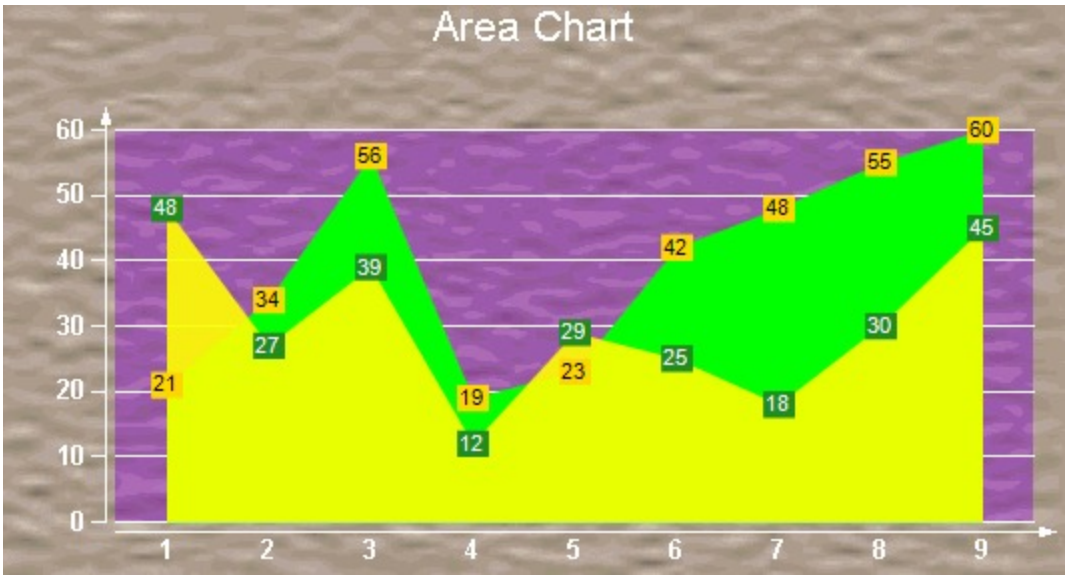
Area Chart

The Area Chart displays quantitative data and is based on the Line chart. In Area charts, the space between axis and line are commonly emphasized with colors, textures and hatchings.

2D Area Charts

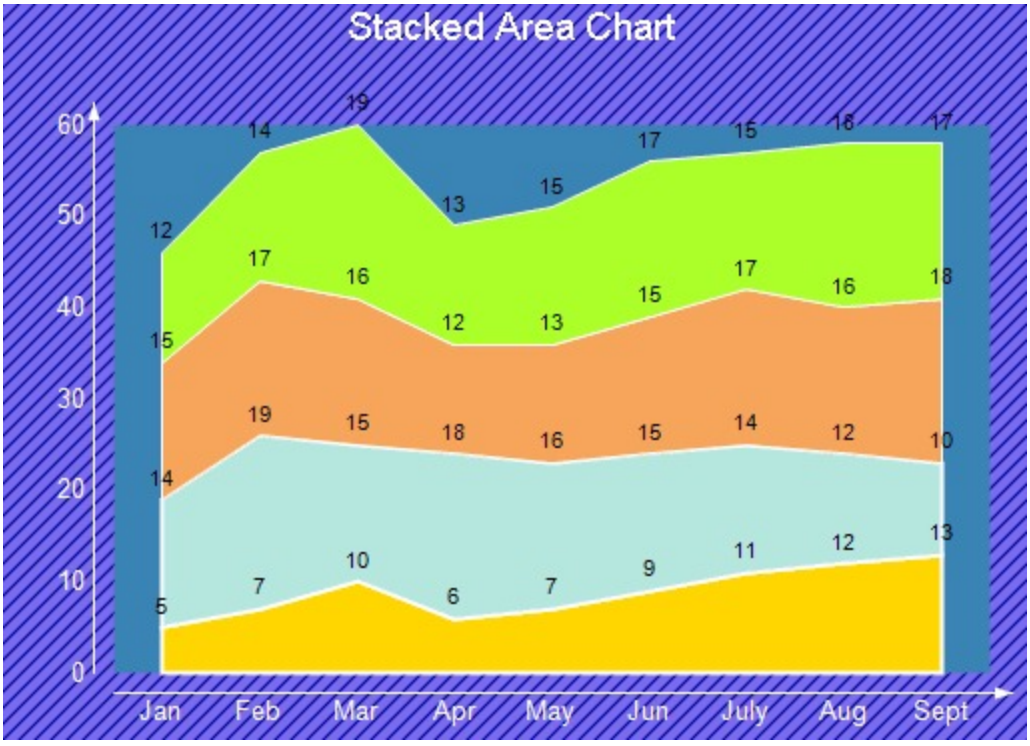
This section describes 2D charts that fall under the Area Chart category (ChartType Area2D).

Area Chart



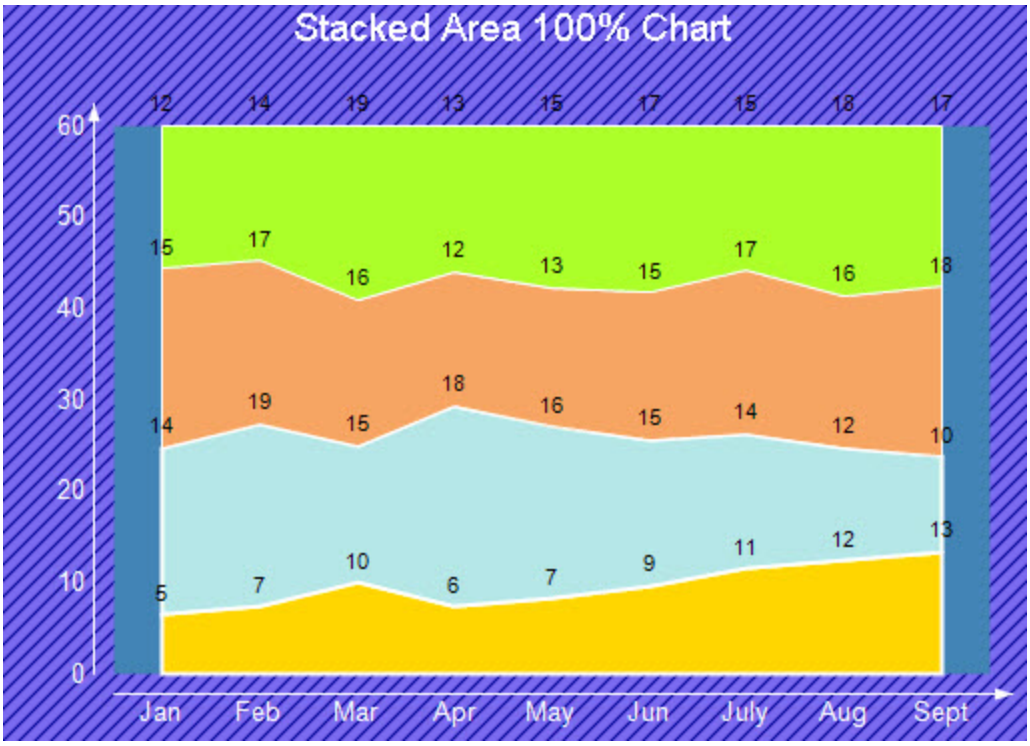
An area chart is used to compare trends over a period of time or across categories.

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.

Stacked Area 100% Chart

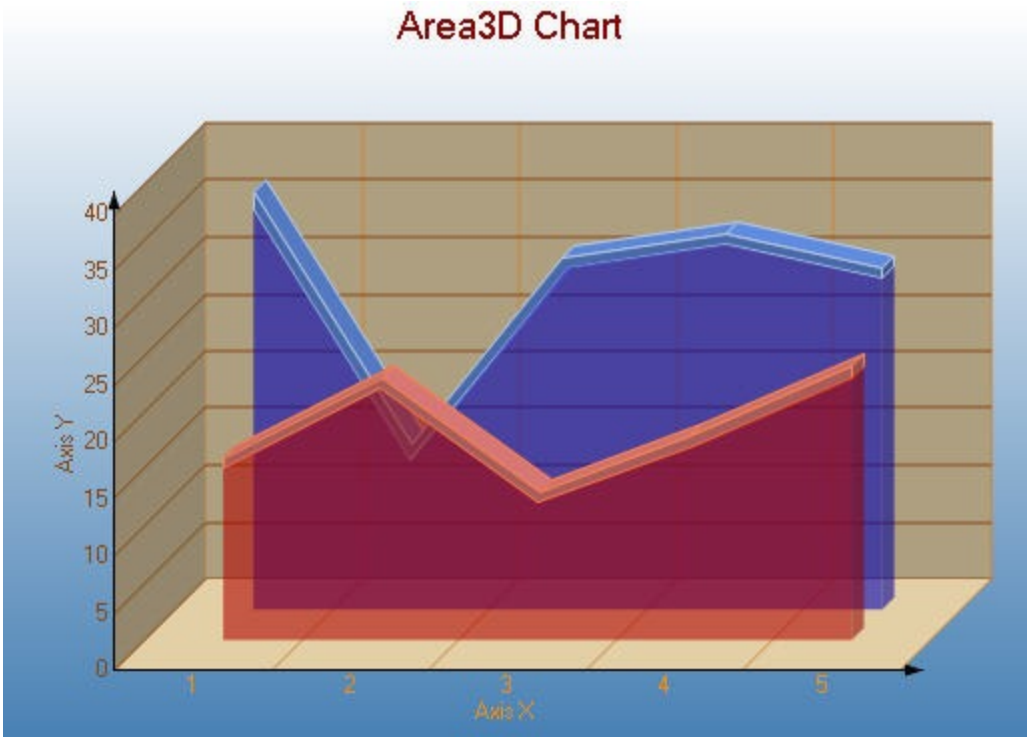


A stacked area chart (100%) is an 100% area chart with two or more data series stacked one on top of the other. Use this chart to show how each value contributes to a total.

3D Area Charts

This section describes 3D charts that fall under the Area Chart category (ChartType Area3D).

Simple Area 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.

> **Note:** To view a chart in 3D, in the **ChartAreas** property open the **ChartArea Collection Editor** and set the **ProjectionType** property to Orthogonal.

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.

Stacked Area 100% Chart

A stacked area chart (100%) is an 100% 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.

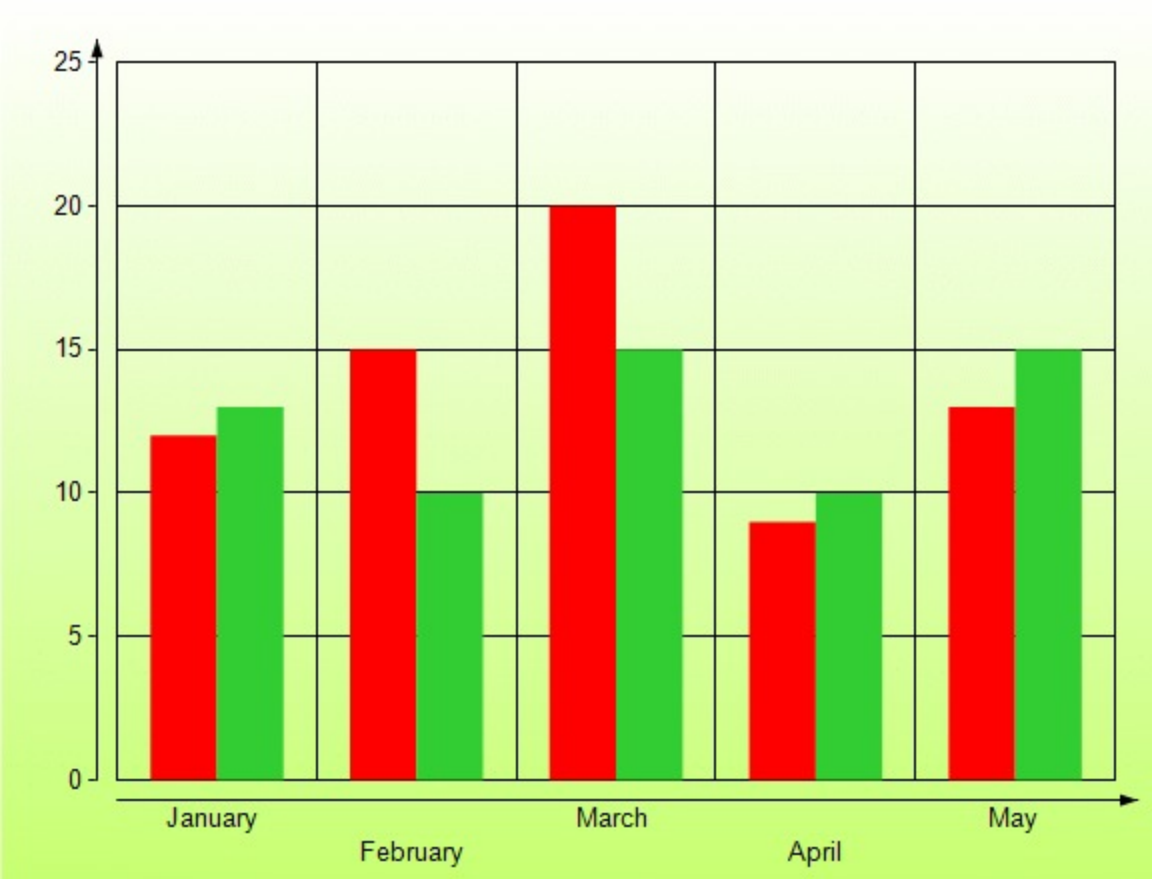
Bar Chart

The **Bar Chart** is a chart with rectangular bars where the lengths of bars are proportional to the values they represent. The bars can be plotted vertically or horizontally.

2D Bar Charts

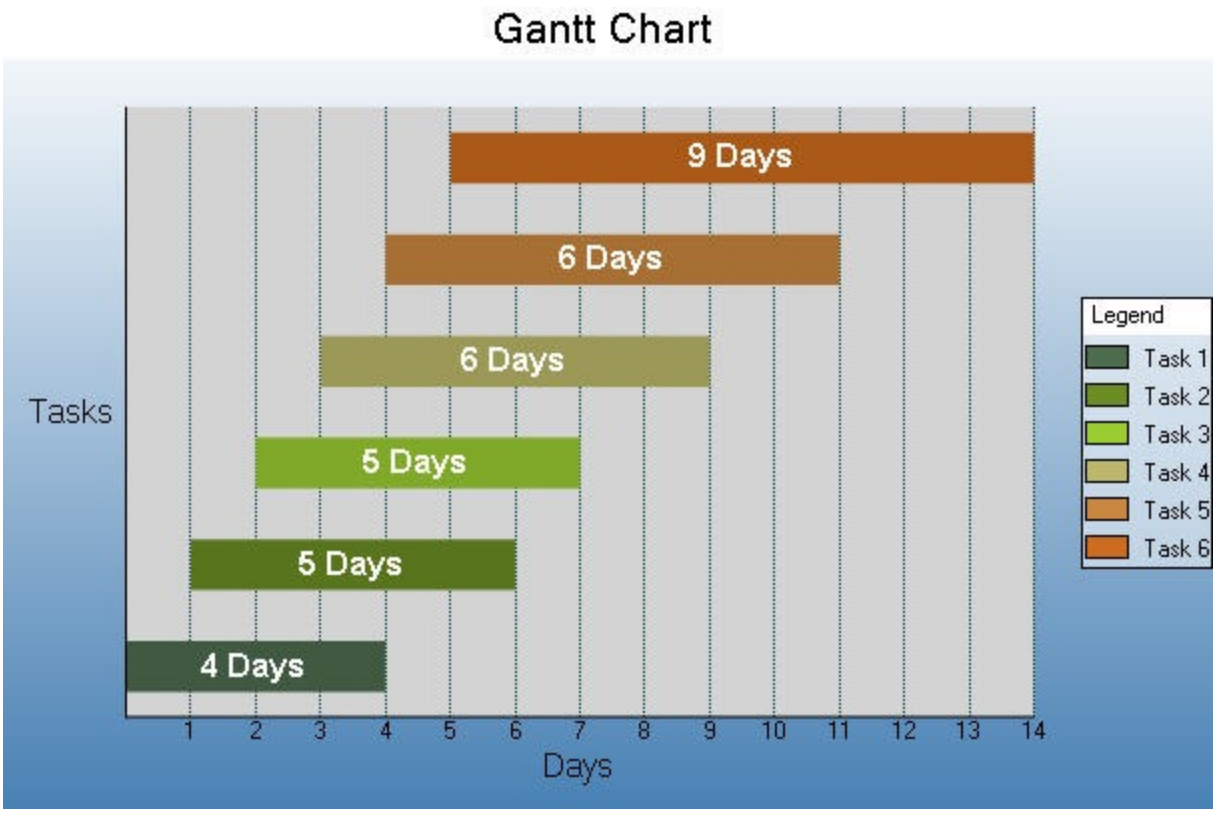
Given below is the list of 2D charts that falls under the Bar Chart category (Chart Type).

Bar Chart



In a Bar Chart, values are represented by the height of the bar shaped marker as measured by the y-axis. Category labels are displayed on the x-axis. Use a bar chart to compare values of items across categories.

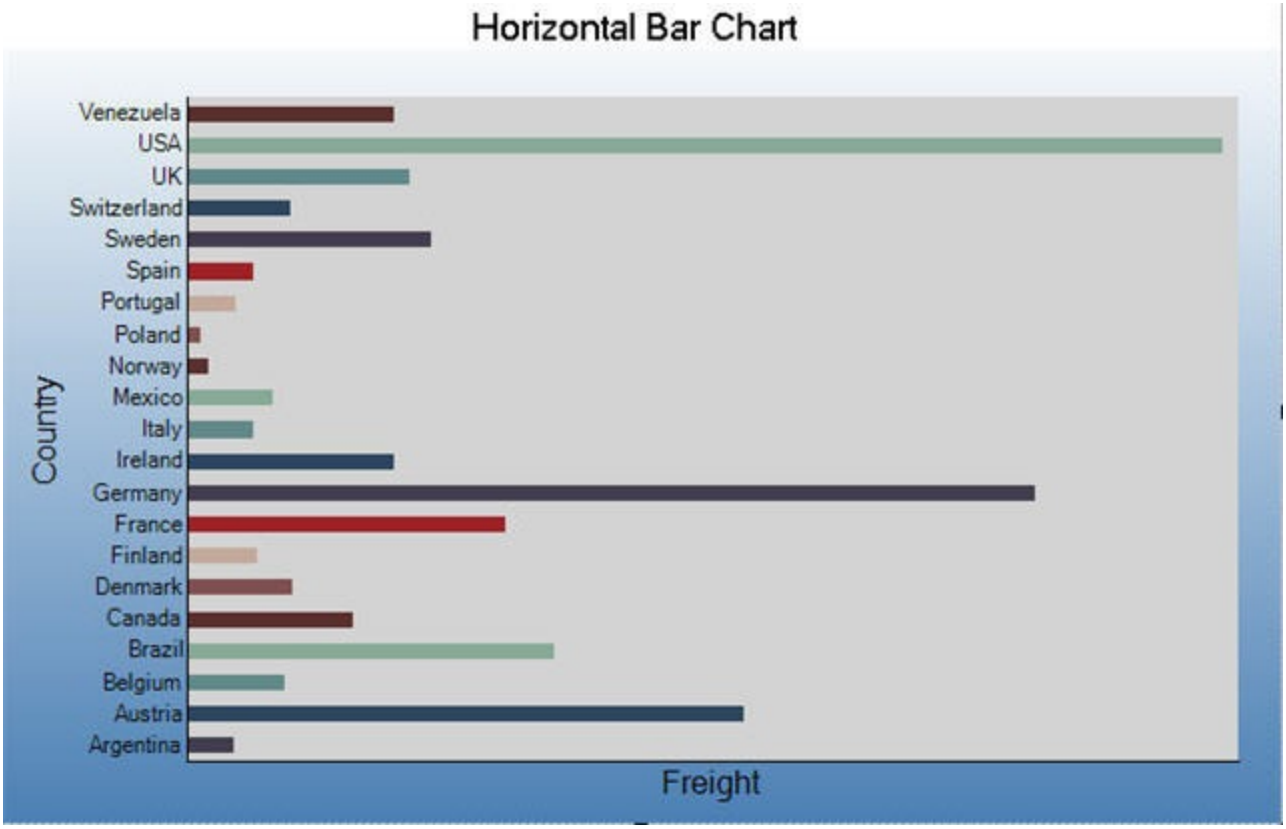
Gantt Chart



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.

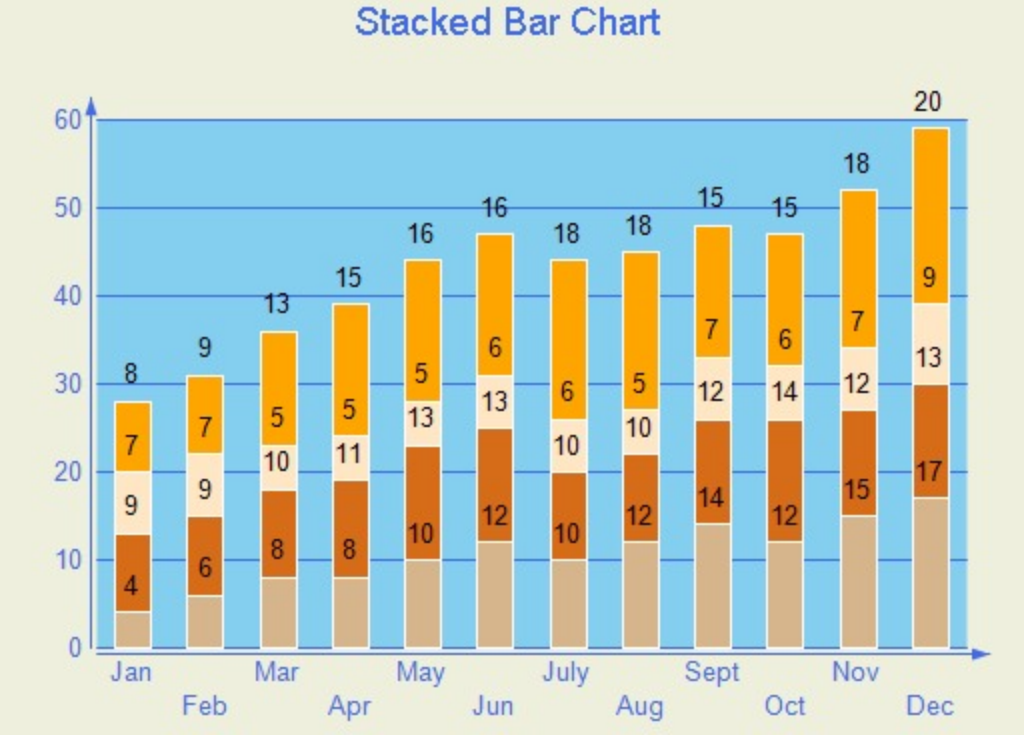
> ⚠ **Caution**: In a Gantt chart, the X and Y axes are reversed. AxisX is vertical and AxisY is horizontal.

Horizontal Bar Chart



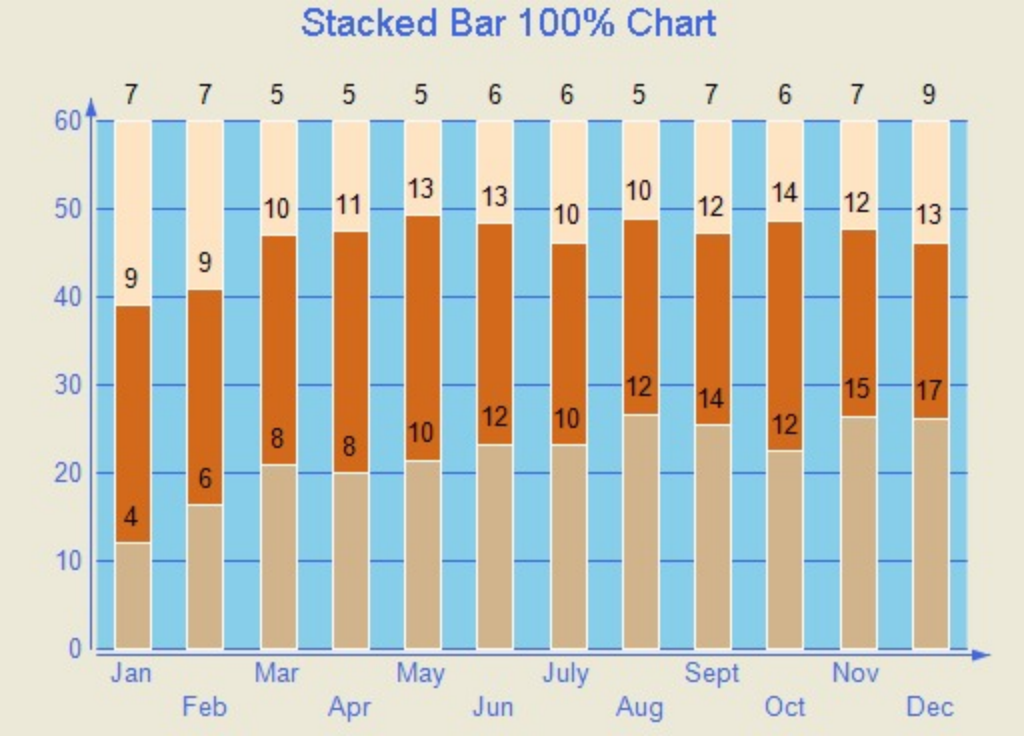
In a Horizontal Bar Chart, both the axes are swapped and therefore the bars appears horizontally. Although, values are represented by the height of the bar shaped marker as measured by the y-axis and the Category labels are displayed on the x-axis. Use a horizontal bar chart to compare values of items across categories.

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.

Stacked Bar Chart 100%



A StackedBAR110Pct 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

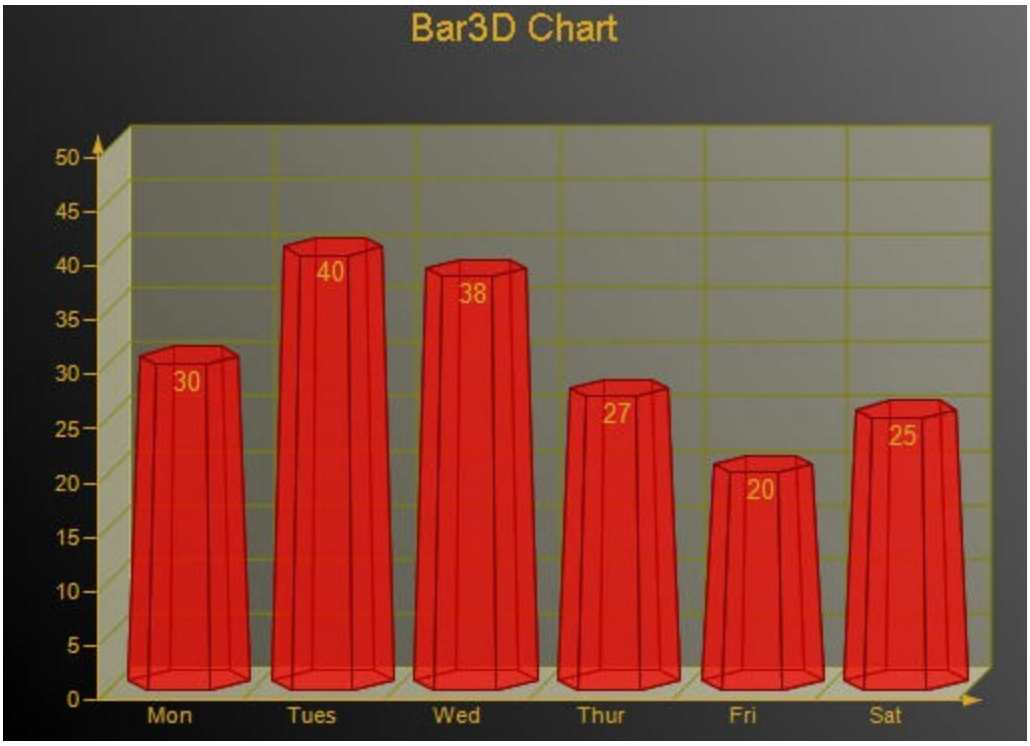
contribution to the total.

3D Bar Charts

Given below is the list of 3D charts that fall under the Bar Chart category.

> **Note:** To view a chart in 3D, open the **ChartArea Collection Editor** in the **ChartAreas** property and set the **ProjectionType** property to Orthogonal.

Simple Bar Chart



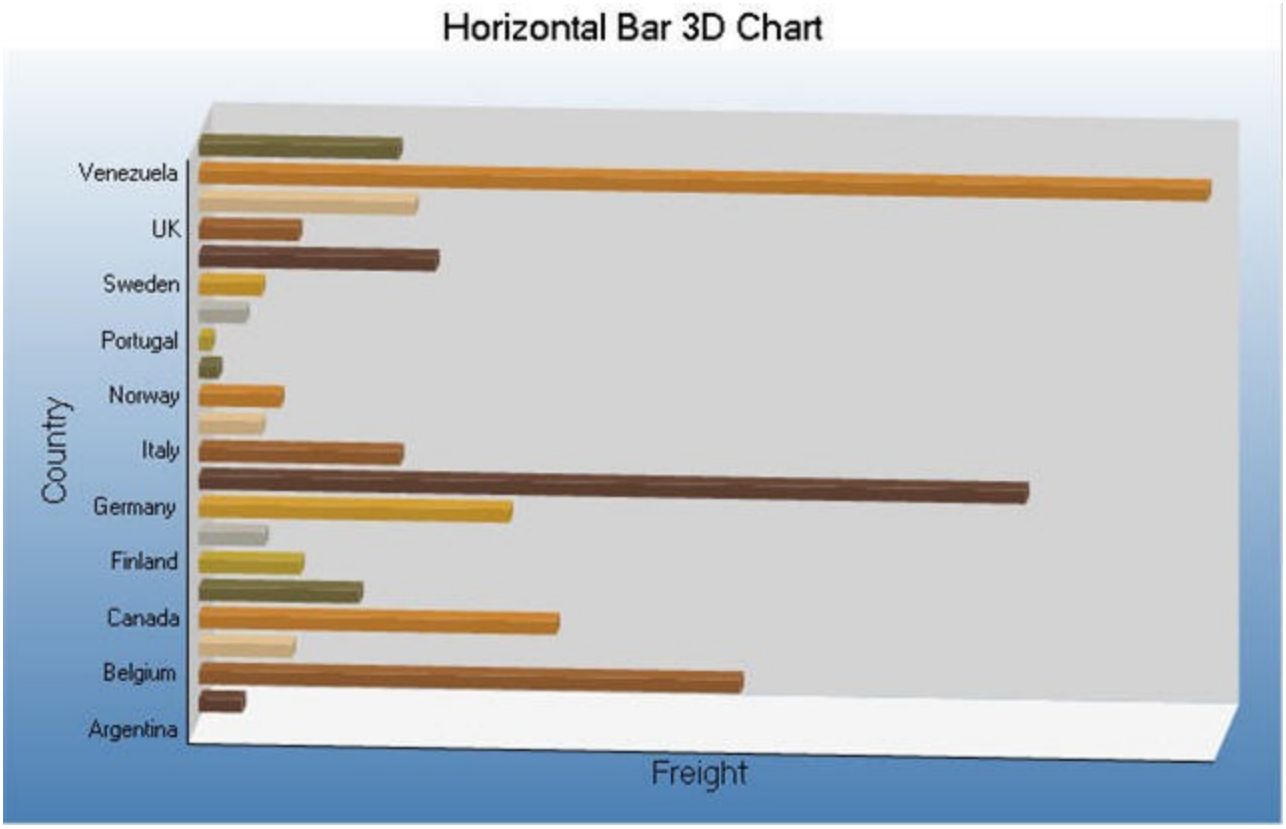
In a Bar Chart, values are represented by the height of the bar shaped marker as measured by the y-axis. Category labels are displayed on the x-axis. Use a 3D bar chart to compare values of items across categories, allowing the data to be viewed in a convenient 3D format.

Gantt Chart

The 3D gantt chart displays a gantt.

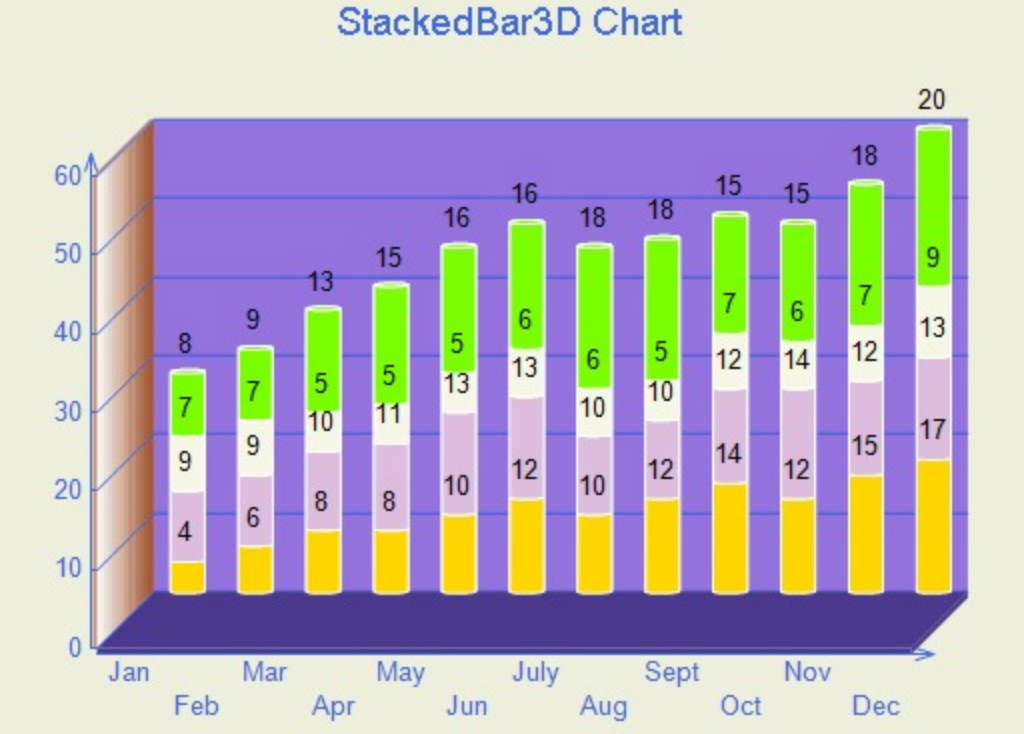
> **Note**: In a 3D Gantt chart the X and Y axes are reversed. AxisX is vertical and AxisY is horizontal.

Horizontal Bar Chart



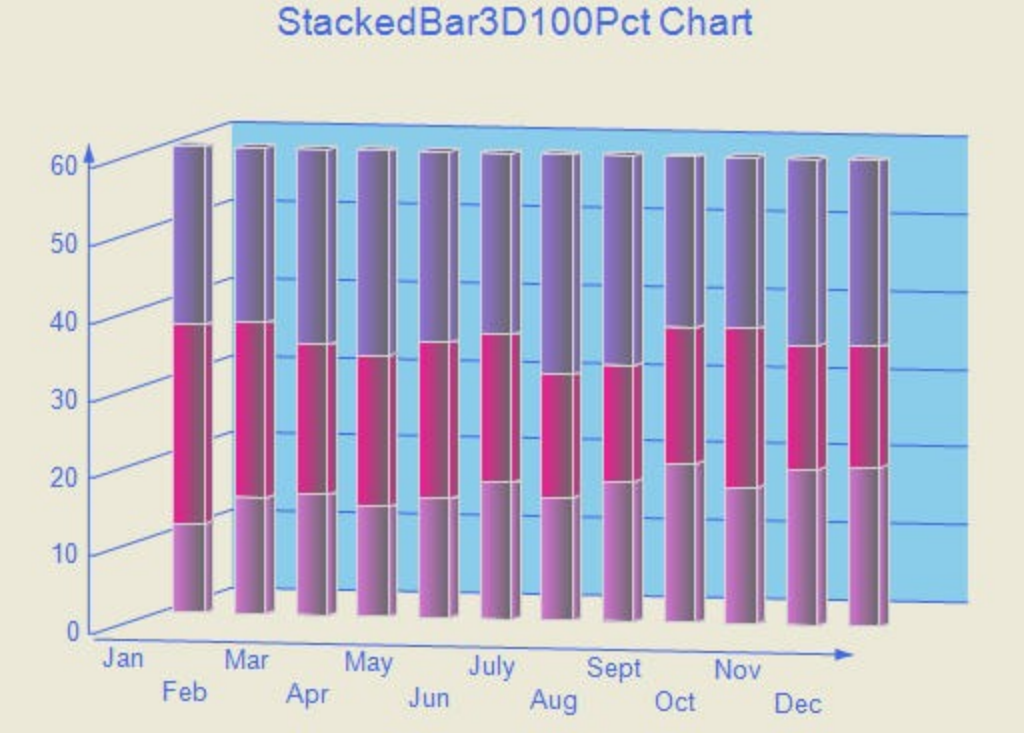
In a Horizontal Bar Chart, both the axes are swapped and therefore the bars appears horizontally. Although, values are represented by the height of the bar shaped marker as measured by the y-axis and the Category labels are displayed on the x-axis. Use a horizontal 3D bar chart to compare values of items across categories, allowing the data to be viewed in a convenient 3D format.

Stacked Bar 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.

Stacked Bar Chart 100%



A Stacked Bar 3D 100% chart is a bar chart with two or more data series in 3D 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.

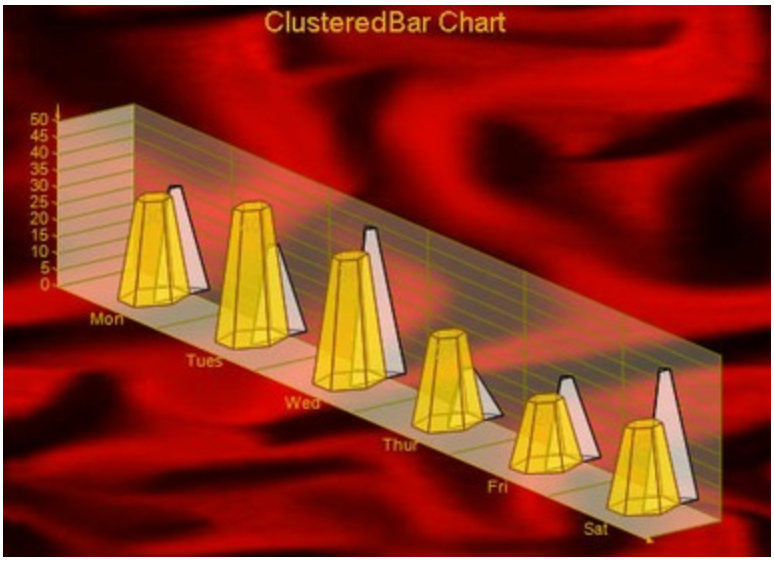
Bar/Cylinder Chart

It is almost similar to a bar chart and values are represented by the height of the bar shaped marker as measured by the y-axis. Category labels are displayed on the x-axis. The only difference is that in a Bar/Cylinder Chart the data is represented through cylindrical shaped markers.

Bar/Pyramid Chart

In a Bar/Pyramid Chart the data is represented through pyramid shaped bars and values are represented by the height of the bars as measured by the y-axis. Category labels are displayed on the x-axis.

Clustered Bar Chart



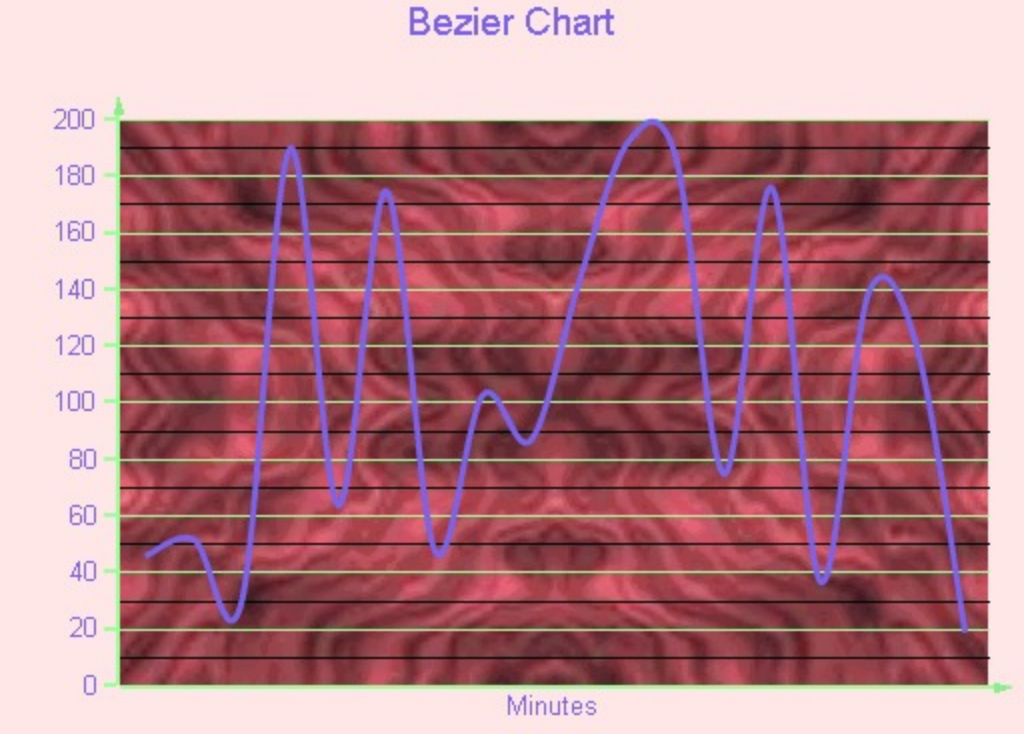
Use a 3D clustered bar chart to compare values of items across categories, allowing the data to be viewed in a convenient 3D format.

Line Chart

The **Line Chart** is a type of chart that displays information as a series of data points, connected by straight line segments.

2D Line Chart

Bezier Chart



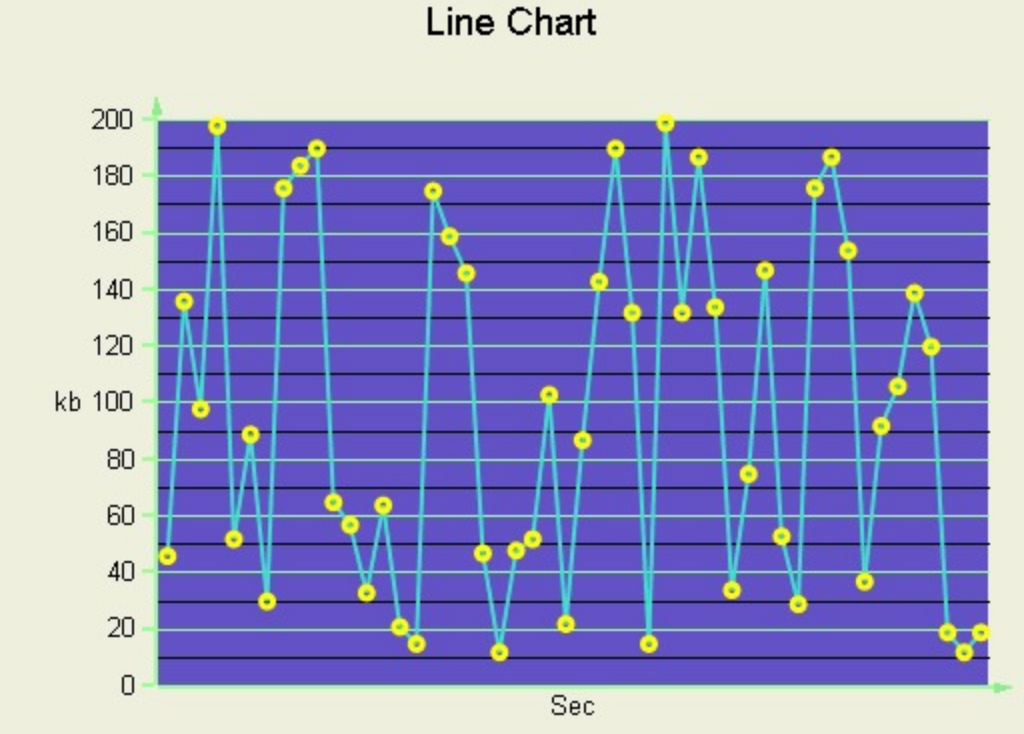
Use a Bezier or spline chart to compare trends over a period of time or across categories. It is a line chart that plots curves through the data points in a series.

Bezier XY Chart

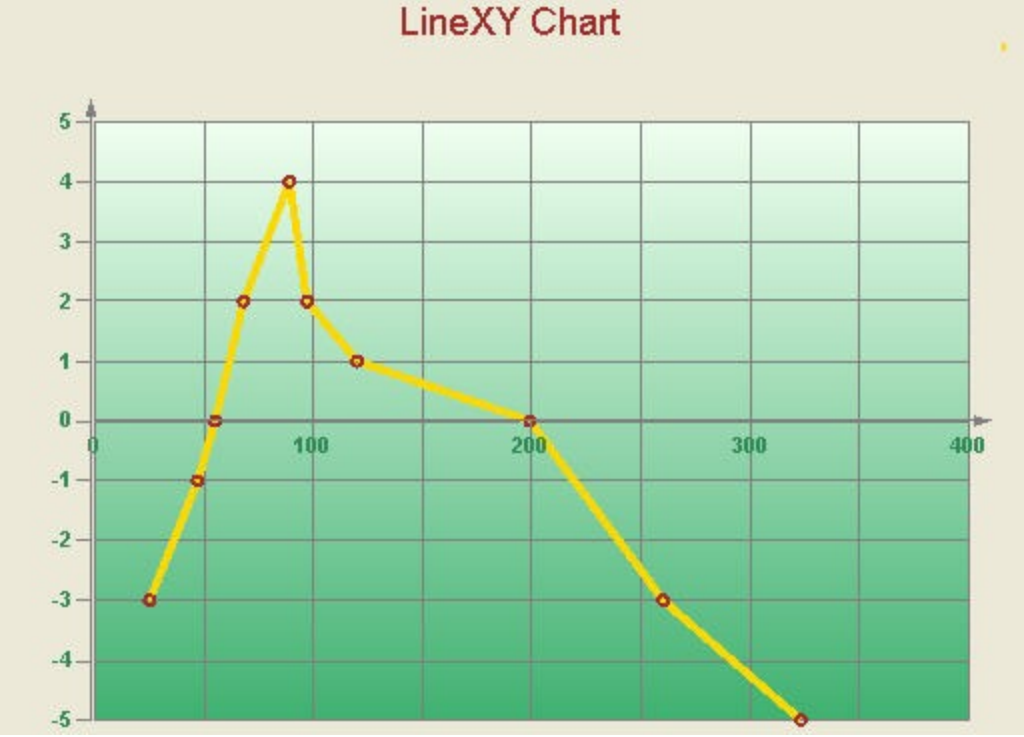
A Bezier XY chart connects DataPoints on X and Y with curved lines.

Simple 2D Line Chart

Use a 2D line chart to compare trends over a period of time or in certain categories in a 2D format.



Line XY Chart



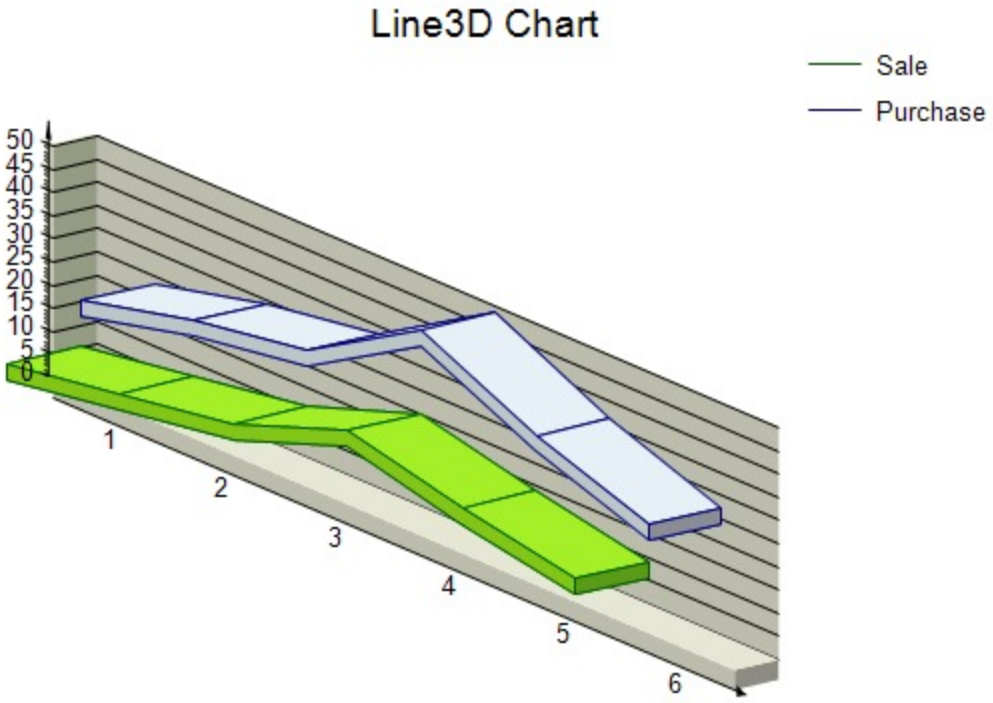
A line XY chart plots points on the X and Y axes as one series and uses a line to connect points to each other.

3D Line Chart

Bezier Chart

Render a Bezier or Spline chart in 3D format.

Line Chart



Use a 3D line chart to compare trends over a period of time or in certain categories in a 3D format.

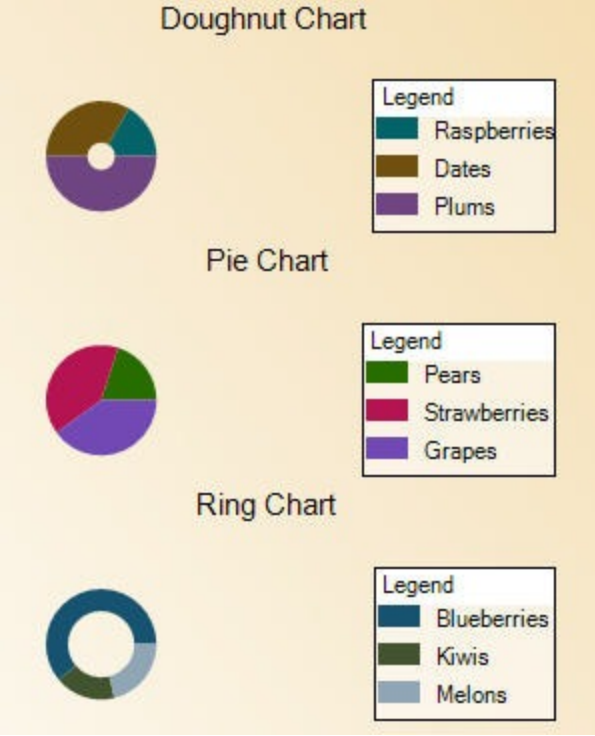
> ⚠ **Caution**: To view a chart in 3D, open the **ChartArea Collection Editor** in the **ChartAreas** property and set the **ProjectionType** property to Orthogonal.

Pie and Doughnut Charts

A **Pie Chart** is a circular chart divided into sectors to illustrate proportion. A **Doughnut Chart** is functionally identical to a **Pie Charts.** It also has single-series and multi-series versions, with the only difference that it has a hole in the middle.

2D Pie and Doughnut Charts

Doughnut Chart



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

> ⚠ In order to show each section of the pie in a different color, set the **Background** property for each data point.

Funnel Chart

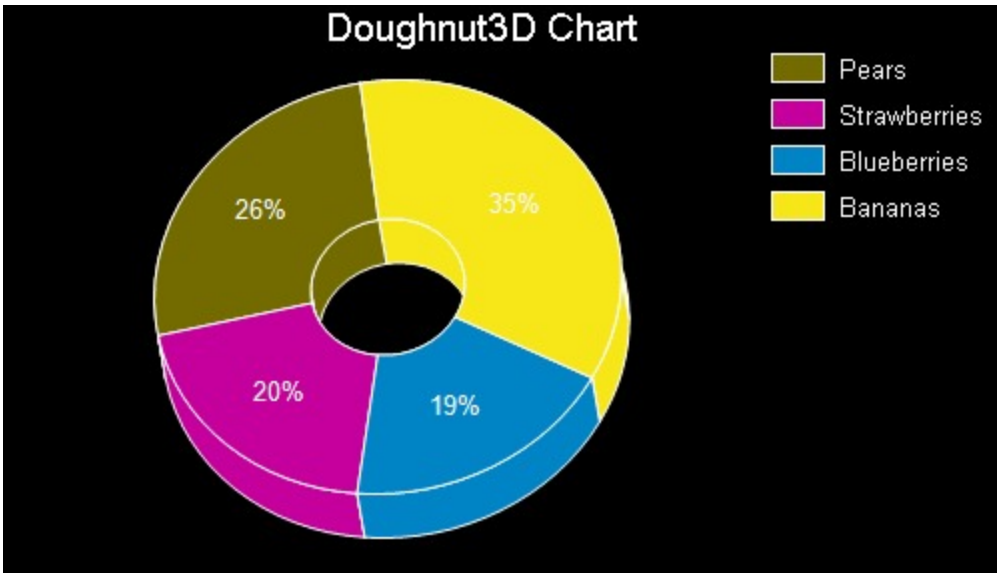
A funnel chart shows how the percentage of each data item contributes as a whole

Pyramid Chart

A Pyramid chart shows how the percentage of each data item contributes as a whole.

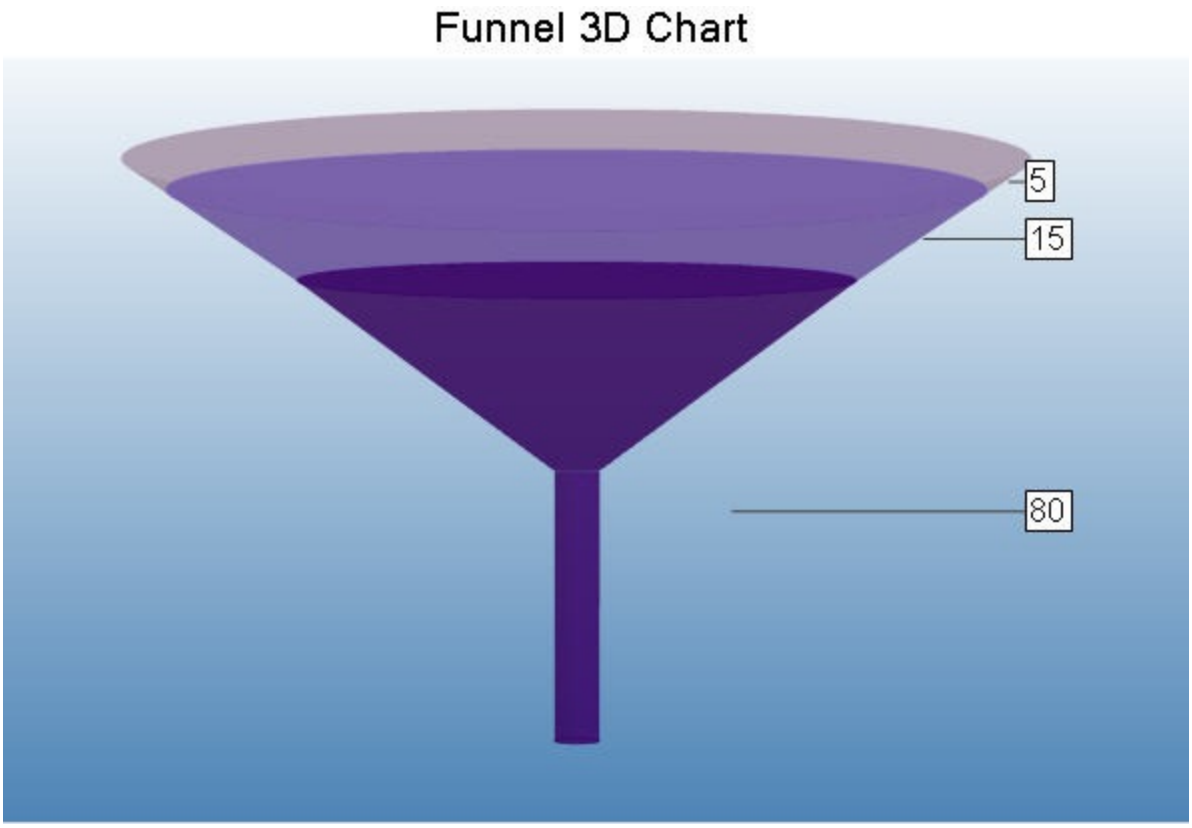
3D Pie and Doughnut Charts

Doughnut Chart



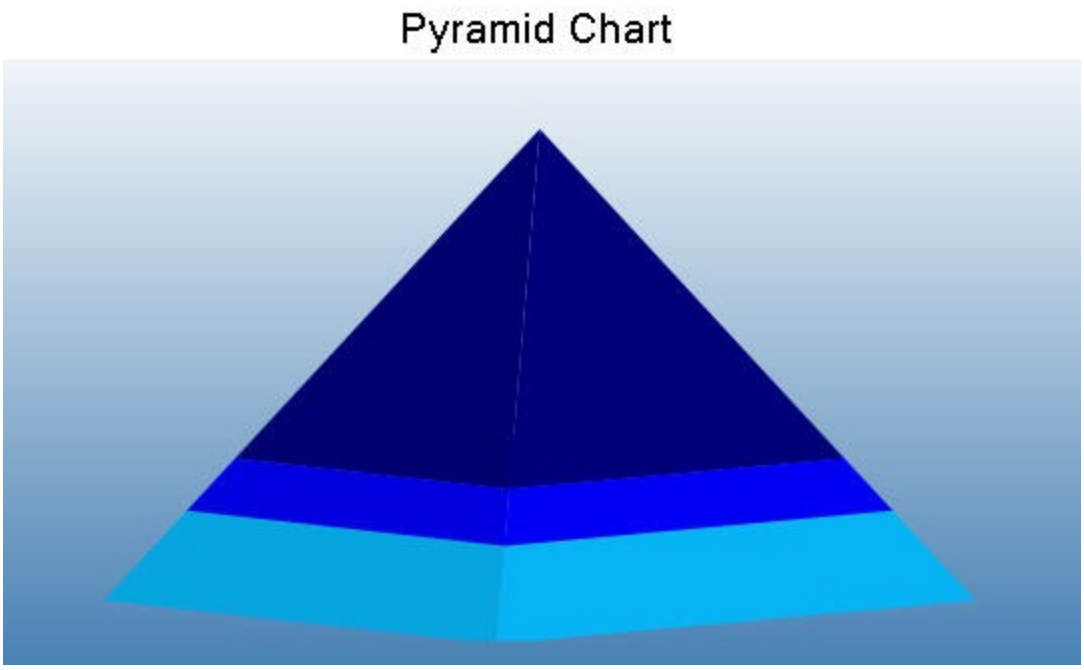
A 3D doughnut chart shows how the percentage of each data item contributes to a total percentage, allowing the data to be viewed in a 3D format.

Funnel Chart



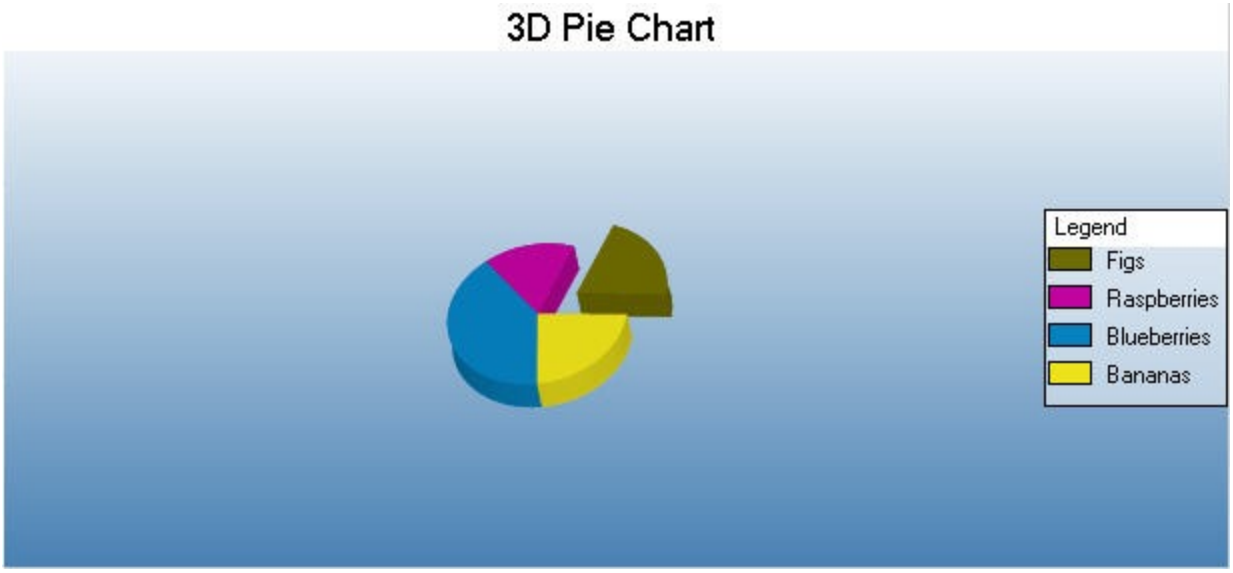
A 3D funnel chart shows how the percentage of each data item contributes to the whole, allowing the data to be viewed in a 3D format.

Pyramid 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 3D format.

Pie Chart



This type of chart displays the contribution of each value to a total.

Ring Chart

This chart type uses rings (inner and outer) to represent data.

Financial Chart

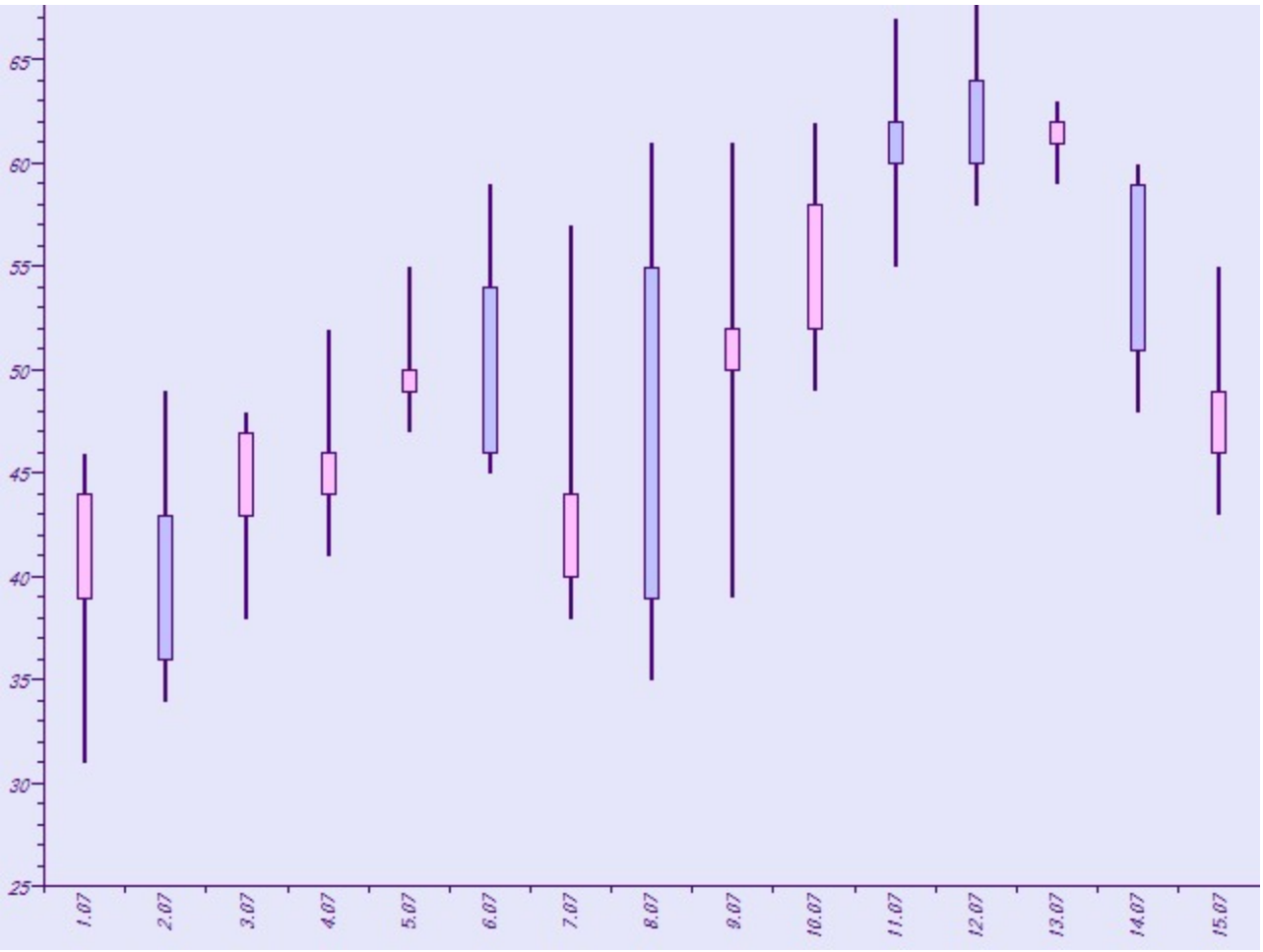
Financial charts are those which are specific to representing data related to financial activities. The Chart data region

can draw a number of financial chart types:

- Candle, High Low, High Low Open Close
- Kagi, Renko
- Point and Figure, Three Line Break

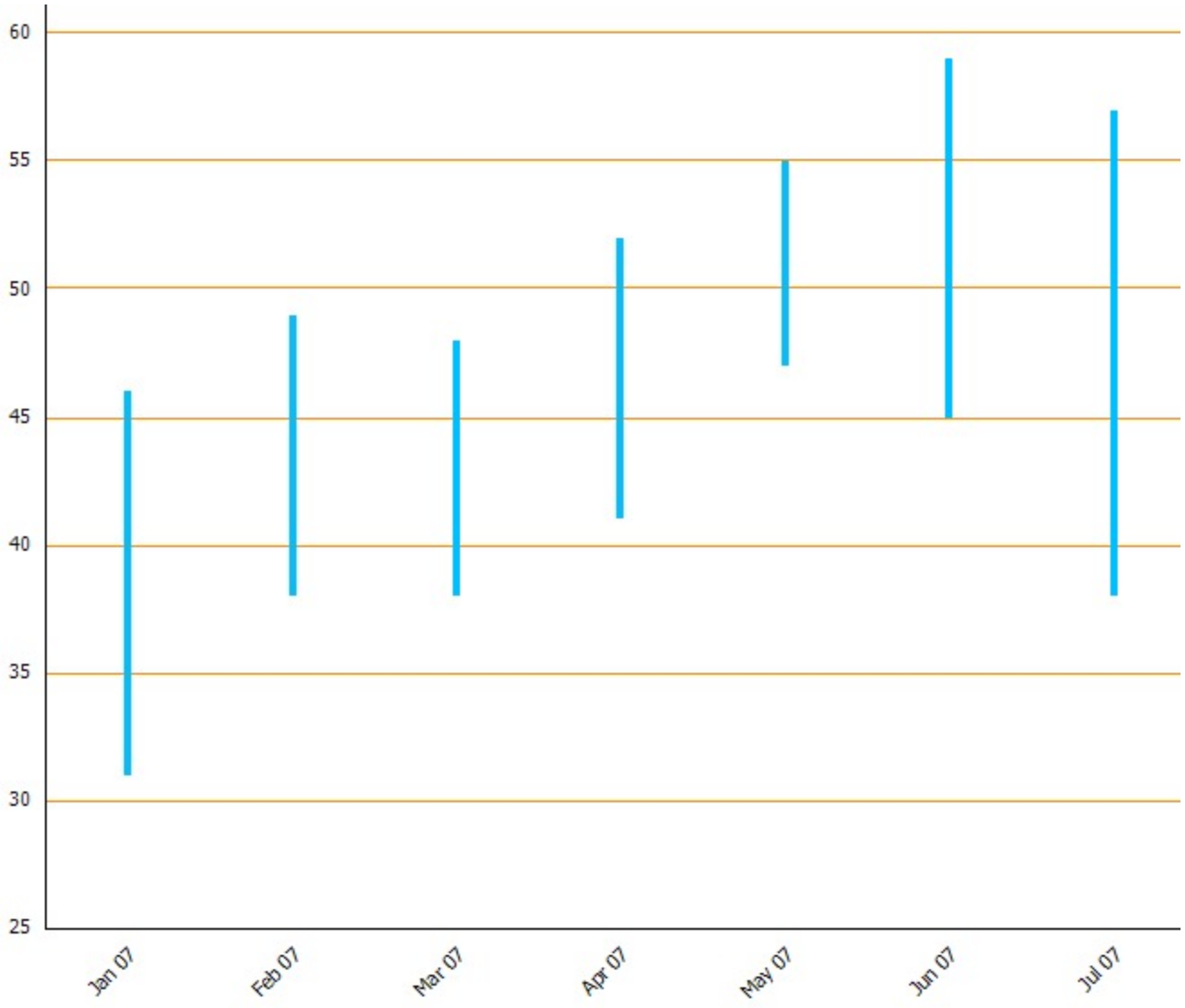
2D Financial Chart

Candle



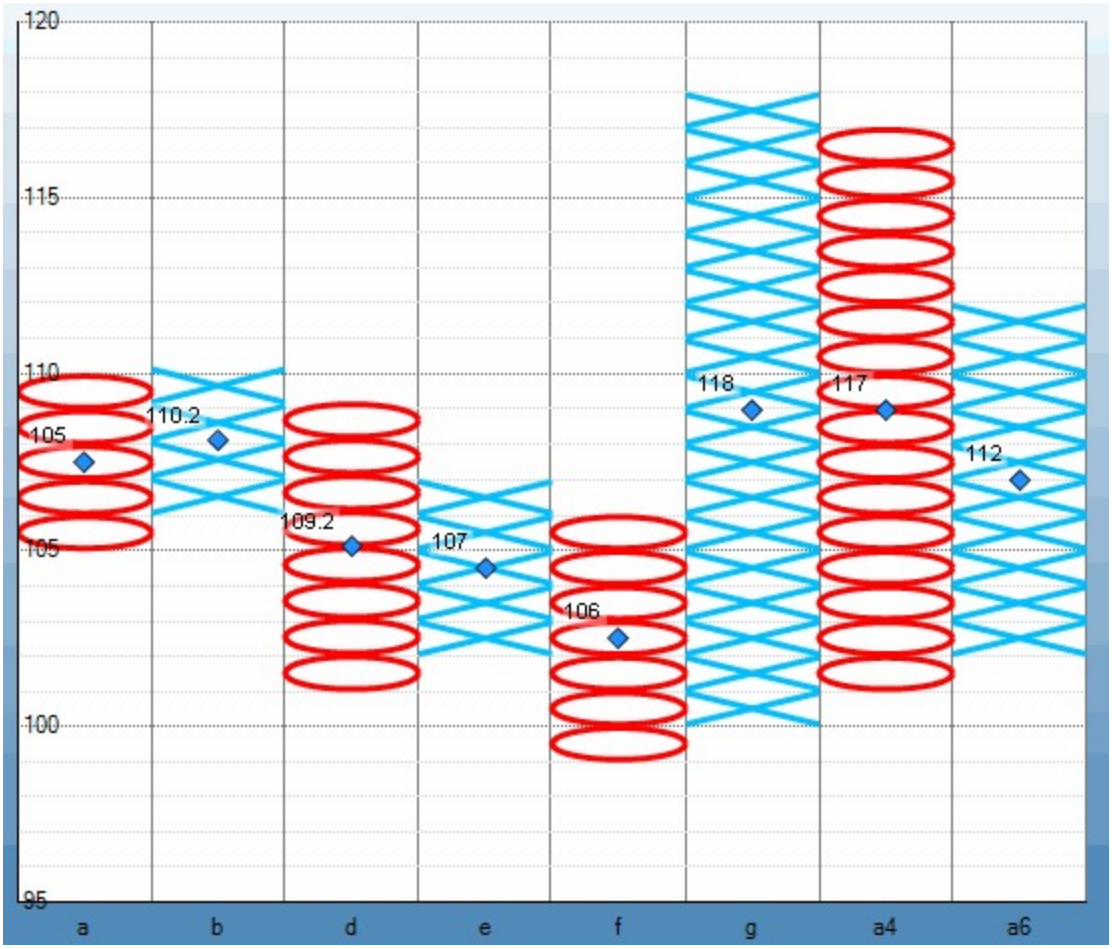
A candle chart displays stock information, using High, Low, Open and Close values. The size of the wick line is determined by the High and Low values, while the size of the bar is determined by the Open and Close values. The bar is displayed using different colors, depending on whether the price of the stock has gone up or down.

HiLo Chart



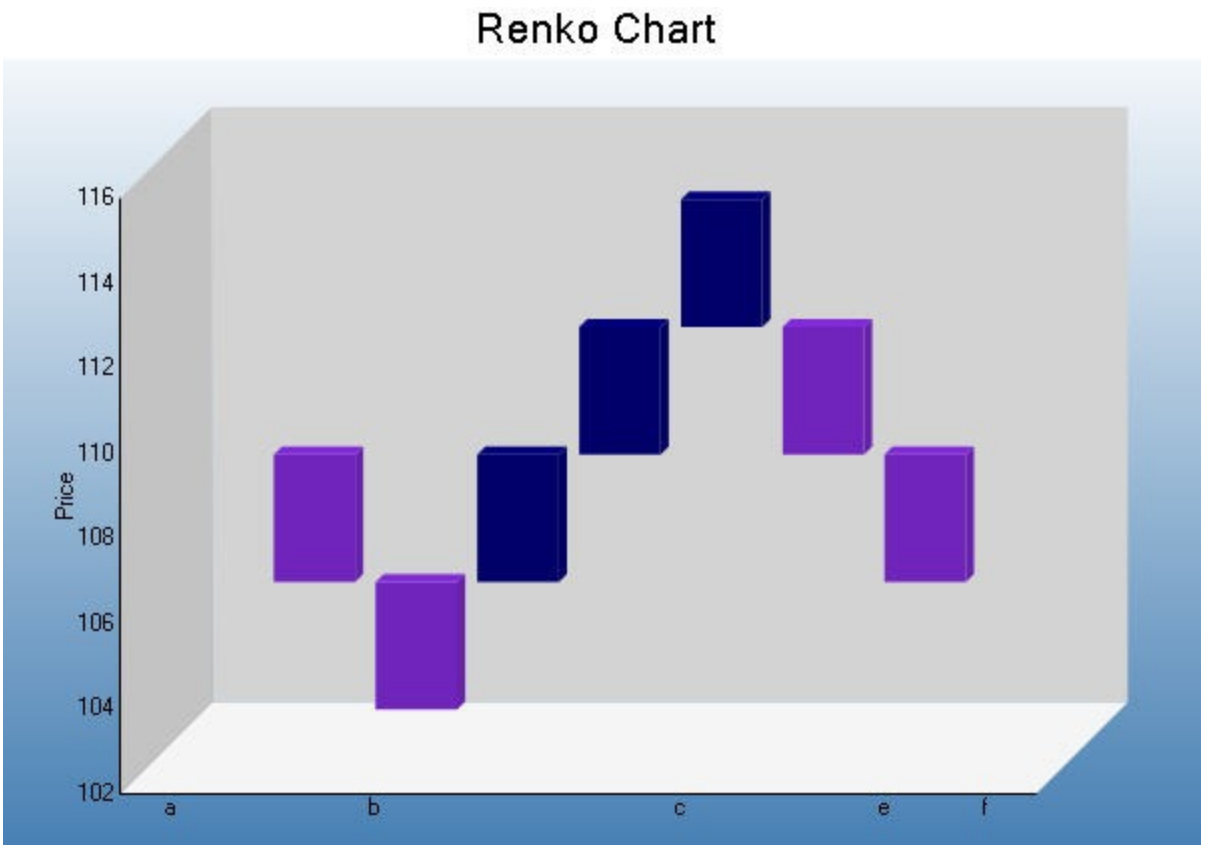
A HiLo chart displays stock information using High and Low, or Open and Close, values. The length of the HiLo line is determined by the High and Low values, or the Open and Close values.

Point and Figure Chart



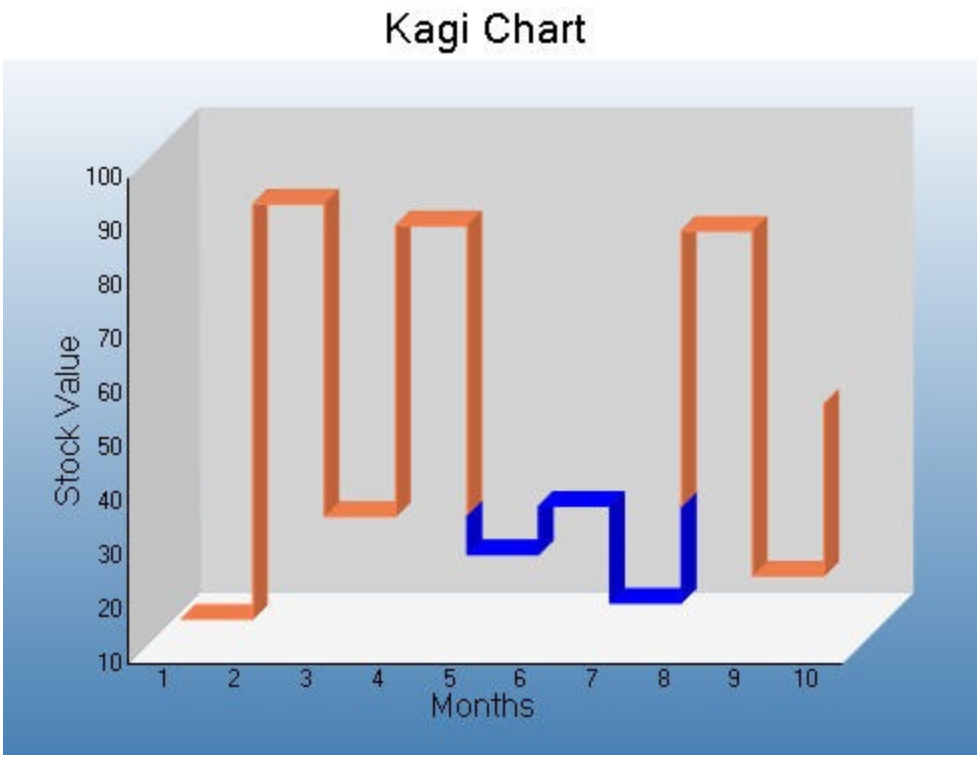
The point and figure chart uses stacked columns of X's to indicate that demand exceeds supply and columns of O's to indicate that supply exceeds demand to define pricing trends. A new X or O is added to the chart if the price moves higher or lower than the BoxSize value. A new column is added when the 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.

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.

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).

Stock Chart

In a stock chart, series are displayed as a set of lines with markers for high, low, close, and open values. Values are represented by the height of the marker as measured by the y-axis. Category labels are displayed on the x-axis.

Stock chart is a visual representation of data related to the stock market. It may be used to represent data like stock prices and stock activities.

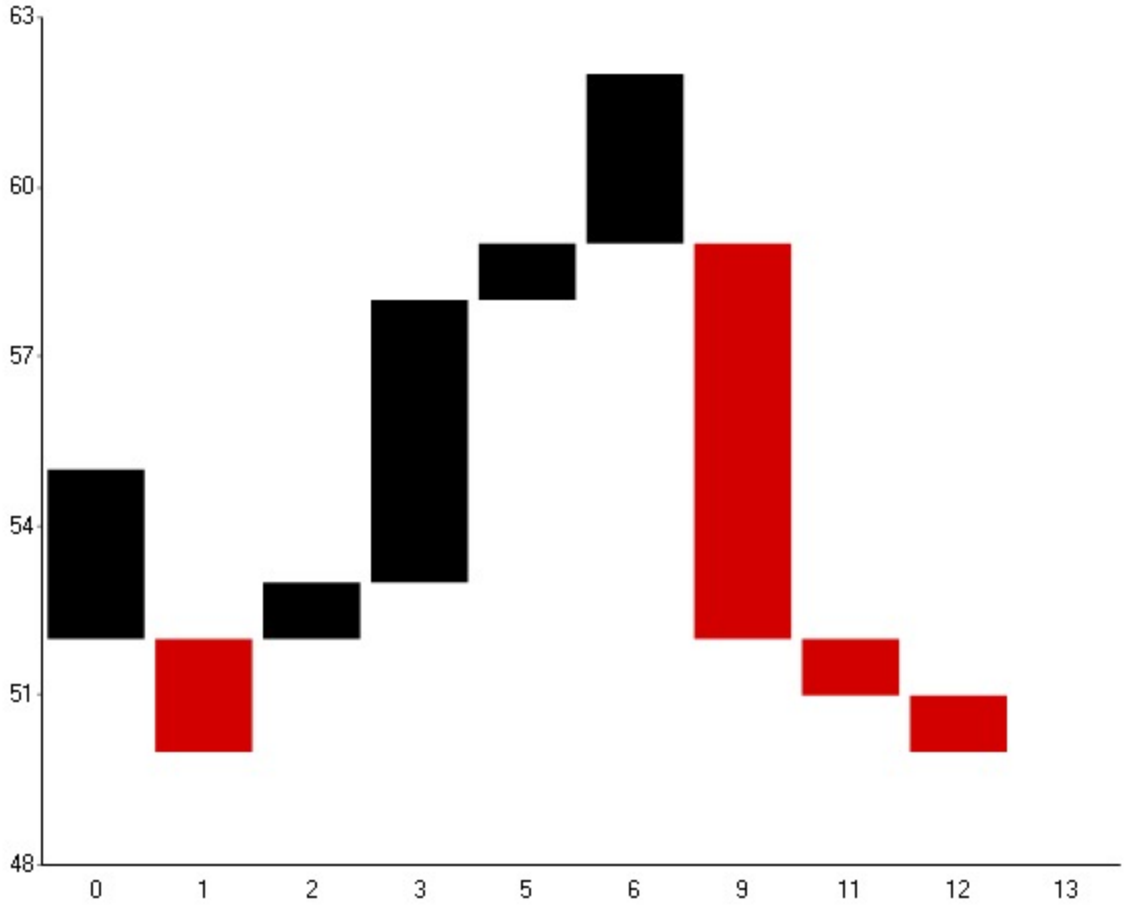
Stock Close Only Chart

A Stock chart is a visual representation of data related to the stock market. This type of chart requires four series of values in the correct order (open, high, low, and then close).

Stock Open Only Chart

A Stock chart is a visual representation of data related to the stock market. This type of chart requires four series of values in the correct order (open, high, low, and then close), low, and then close).

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.

3D Financial Chart

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).

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.

Three Line Break Chart

Three line break 3D chart is a chart rendered in 3D.

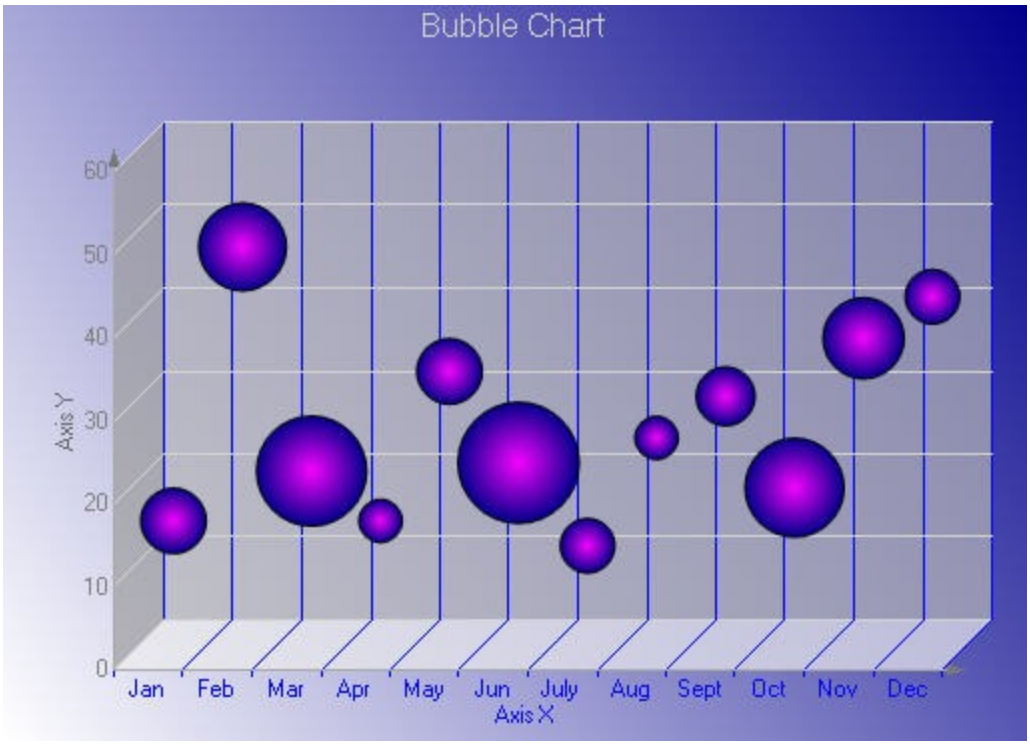
Point and Bubble Charts

Point or **Bubble** charts represent data by means of points and bubbles.

The ActiveReports Chart control can draw a number of point/bubble chart types, like, Bubble, BubbleXY, PlotXY, and Scatter.

Given below is the list of 2D charts that fall under the Point/Bubble Chart category.

Bubble Chart

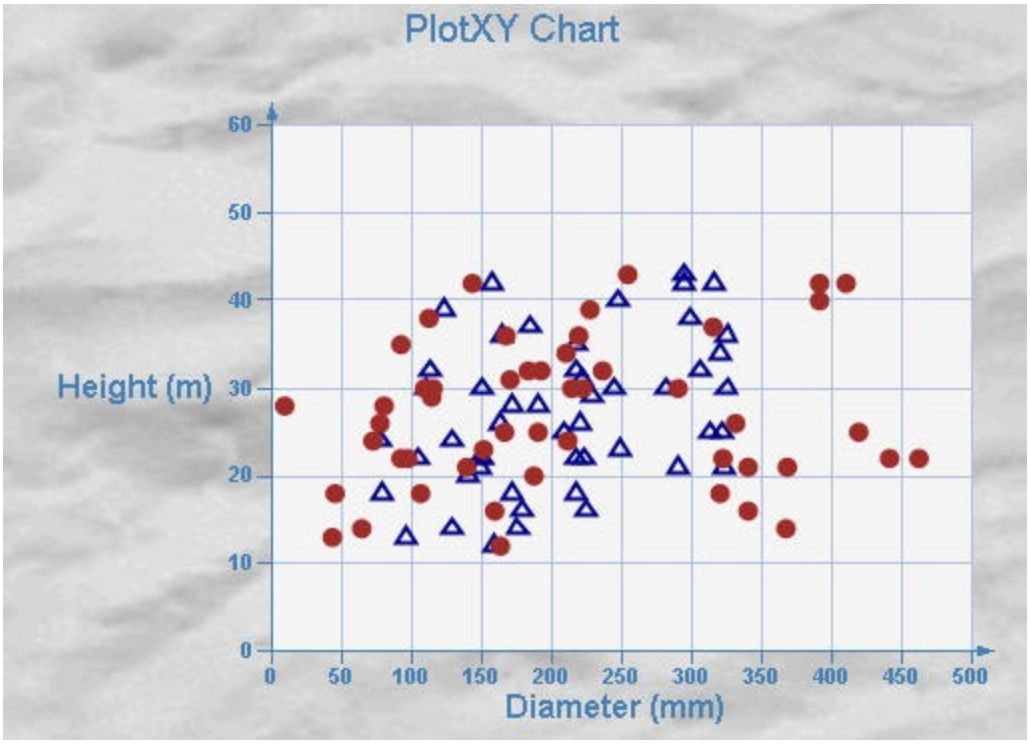


The Bubble chart is an XY chart in which bubbles represent data points. The first Y value is used to plot the bubble along the Y axis, and the second Y value is used to set the size of the bubble. The bubble shape can be changed using the series Shape property.

Bubble XY Chart

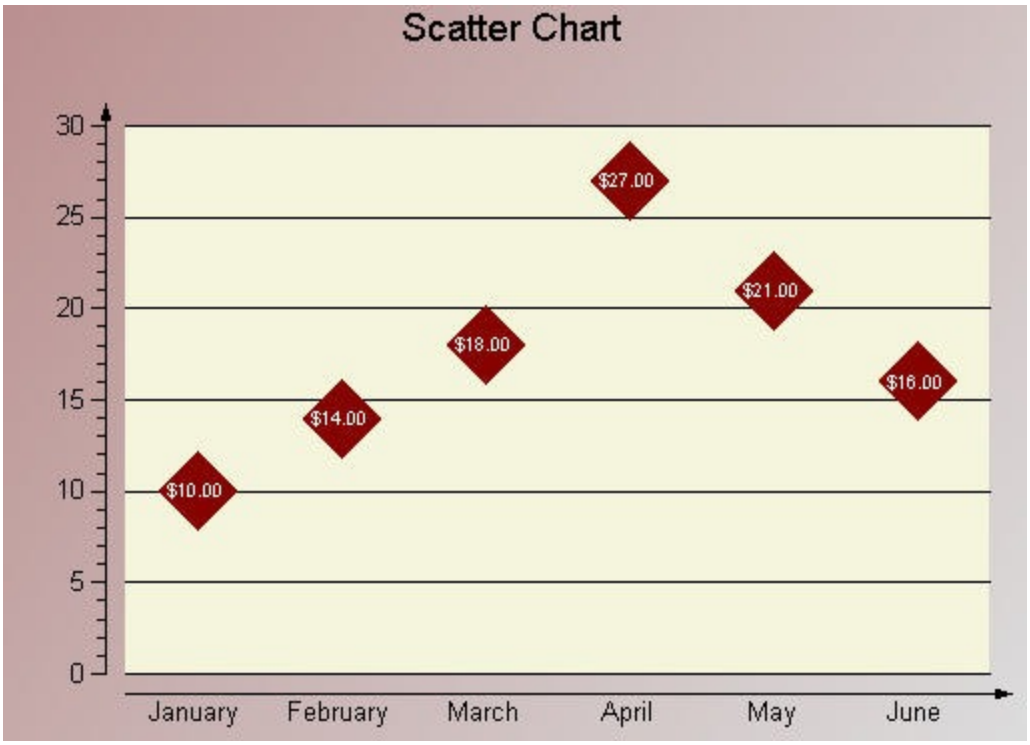
The Bubble XY chart is an XY chart in which bubbles represent data points. The BubbleXY uses a numerical X axis and plots the x values and first set of Y values on the chart. The second Y value is used to set the size of the bubble.

Plot XY Chart



A plot XY chart shows the relationships between numeric values in two or more series sets of XY values.

Scatter Chart



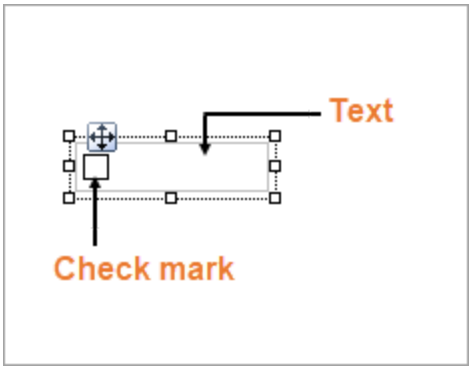
Use a scatter chart to compare values across categories.

CheckBox

You can use the CheckBox control to represent a Boolean value in a report. By default, the CheckBox control appears as a small box inside an empty textbox. If the **Checked** value is set to True, the small box appears with a check mark; if False, the box is empty. By default, the checkbox is empty.

> **Note**: There is a difference in how the text is rendered in the CrossPlatform and in the GDI compatibility modes. Some text in the CrossPlatform mode (for example, text without spacing, text with non-ASCII characters, etc.) may not appear correctly in the report preview. For the correct text rendering, you must manually update the report layout (for example, change the control's size).

Structure



Important Properties

Clicking the CheckBox control reveals its properties in the Properties window.

Property	Description
Checked	Gets or sets a value indicating whether the check box is in the checked state. You can also set the **Checked** property of the check box in code or bind it to a Boolean database value.
Text	Gets or sets the printed caption of the check box.
CheckAlignment	Gets or sets the alignment of the check box text within the control drawing area.
DataField	Gets or sets the field from the data source to bind to the control.

You can double-click the CheckBox control to enter edit mode and enter text directly in the control, or you can enter text in the Properties window or you can assign data to display in code through the **Text** property.

In edit mode, using the toolbar you can format text in the CheckBox control using the toolbar or you modify properties in the Properties window. Formats apply to all of the text in the control. Text formatting changes in the Properties window immediately appear in the control, and changes made in the toolbar are immediately reflected in the Properties window.

CheckBox Dialog Properties

You can set the CheckBox properties in the CheckBox dialog. To open it, with the CheckBox selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the checkbox that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

DataField: Select a field that returns a Boolean value from the data source to bind to the control. The value of this field determines how to set the Checked property at run time.

Text : Enter an expression or a static label, or choose a field expression from the drop-down list. You can access the expression editor by selecting **<Expression...>** in the list. The value of this expression or text is displayed in the report to the right of the checkbox.

Check Alignment: Drop down the visual selector to choose the vertical and horizontal position for the check box within the control.

Checked: Select this check box to have the CheckBox control appear with a check mark in the box.

Appearance

Background

Color: Select a color to use for the background of the checkbox.

Font

Name: Select a font family name or a theme font.

Size: Choose the size in points for the font.

Style: Choose **Normal** or **Italic**.

Weight: Choose from **Normal** or **Bold**.

Color: Choose a color to use for the text.

Decoration: Select check boxes for **Underline** and **Strikeout**.

GDI Charset: Enter a value to indicate the GDI character set to use. For a list of valid values, see MSDN Font.GDICharSet Property.

GDI Vertical: Select this checkbox to indicate that the font is derived from a GDI vertical font.

Alignment

Alignment: Alignment of the text inside the checkbox.

Wrap mode: Choose NoWrap, WordWrap, or CharWrap.

Padding: Amount of space (in Int32) to leave around report control.

Top padding: Set the top padding.

Left padding: Set the left padding.

Right padding: Set the right padding.

Bottom padding: Set the bottom padding.

Keyboard Shortcuts

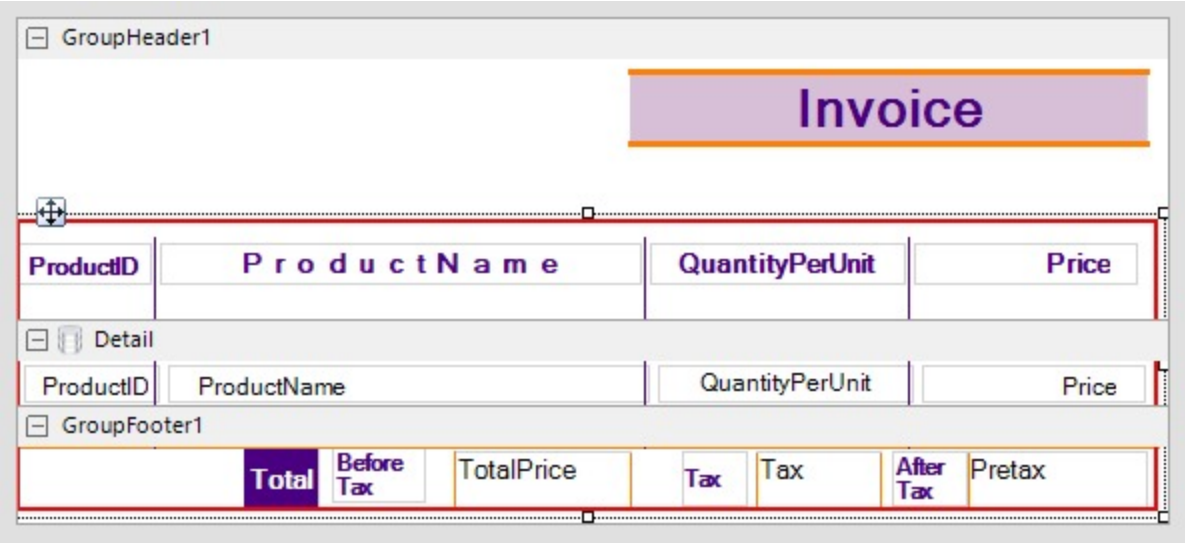
In the edit mode, you can use the following keyboard shortcuts.

Key Combination	Action
Enter	New line.
Alt + Enter	Saves modifications and exits edit mode.
Esc	Cancels modifications and exits edit mode.

In Visual Studio Integrated Designer, you can disable this feature in the **EditModeEntering Event (on-line documentation)** and **EditModeExit Event (on-line documentation)**.

CrossSectionBox

In section reports, you can use the CrossSectionBox report control to display a frame that runs from a header section through its related footer section, spanning all details that come between.



You can place the CrossSectionBox control in the header section in the designer. On adding the control in the footer section, it automatically associates itself with the related header section in the Report Explorer. The control automatically spans intervening sections to end in the related footer section. Move and sizing handles for cross-section controls are not available; you need to use Properties panel instead.

Important Properties

Clicking the four-way arrow selects the control and reveals its properties.

Property	Description
Radius	Sets the radius of each corner for the RoundRect shape type. You can select Default, TopLeft, TopRight, BottomLeft or BottomRight. Selecting Default sets the radius of all the corners of the CrossSectionBox control to a specified percentage. Default value is 0 (pt).
BackColor	Sets the back color.

⚠ **Caution**:

- The CrossSectionBox and CrossSectionLine are not supported in multi-column reports where a GroupHeader section has the **ColumnLayout** property set to True.
- The CrossSectionBox and CrossSectionLine controls are drawn across multiple sections. Therefore, it is not possible to change the control's appearance, etc. in the event of a section. The appearance of the CrossSection control can be dynamically changed only in the ReportStart event.

CrossSectionBox Dialog Properties

You can set the CrossSectionBox properties in the CrossSectionBox dialog. To open it, with the CrossSectionBox selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the control that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

Appearance

Line style: Select a line style to use for the borderline. You can set it to Transparent, Solid, Dash, Dot, DashDot, DashDotDot, or Double.

Line weight: Enter the width for the borderline.

Line color: Select a color to use for the borderline.

Background color: Select a color to use for the background of the picture control.

CloseBorder: Select the checkbox to close the borders.

> **Note:** The **CloseBorder** property is only available for CrossSectionBox control placed in Section report's Group Header and Group Footer.

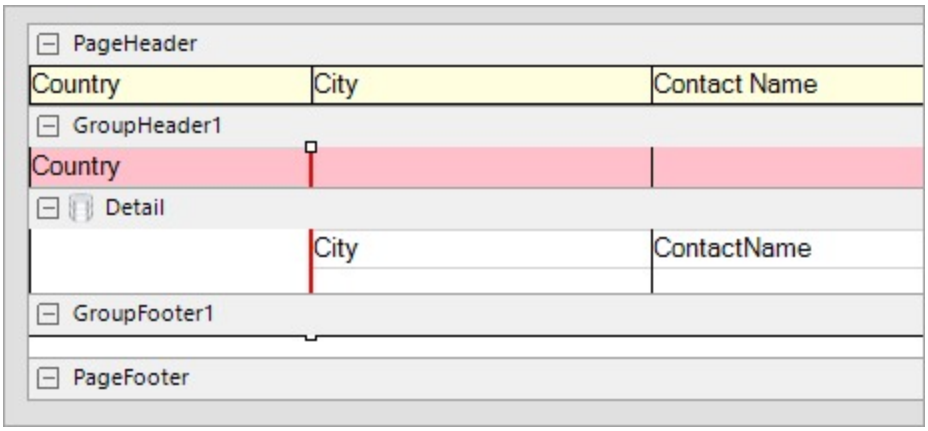
Rounded Rectangle: Specify the radius for each corner of the CrossSectionBox individually. Drag the handlers available at each corner of the CrossSectionBox to set the value of the radius at each corner.

> **Note:** To enable specific corners, check the checkbox available near each corner of the CrossSectionBox control.

- **Use the same radius on specified corners**: Select this option to apply the same radius to all selected corners of the CrossSectionBox.
- **Use different radius on specified corners:** Select this option to apply a different radius to each selected corner of the CrossSectionBox.

CrossSectionLine

In section reports, you can use the CrossSectionLine report control to display vertical lines that run from a header section through its related footer section, spanning all details that come between.



You can place the CrossSectionLine control in the header section in the designer. On adding the control in the footer section, it automatically associates itself with the related header section in the Report Explorer. The control automatically spans intervening sections to end in the related footer section. Move and sizing handles for cross-section controls are not available; you need to use Properties panel instead.

Important Properties

Clicking the four-way arrow selects the control and reveals its properties.

Property	Description
LineColor	Allows you to get or set the color of the line.
LineStyle	Allows you to select the line style from solid, dash, dotted, or double.
LineWeight	Allows you to specify the thickness of the line.

⚠ **Caution**:

- The CrossSectionBox and CrossSectionLine controls do not render properly in multi-column reports, that is, those in which a GroupHeader section has the **ColumnLayout** property set to True.
- The CrossSectionBox and CrossSectionLine controls are drawn across multiple sections. Therefore, it is not possible to change the control's appearance, etc. in the event of a section. The appearance of the CrossSection control can be dynamically changed only in the ReportStart event.

CrossSectionLine Dialog Properties

You can set the CrossSectionLine properties in the CrossSectionLine dialog. To open it, with the CrossSectionLine selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the control that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

Appearance

Line style: Select a line style to use for the control. You can set it to Transparent, Solid, Dash, Dot, DashDot, DashDotDot, or Double.

Line weight: Enter the width for the line.

Line color: Select a color to use for the line.

InputFieldCheckBox

The **InputFieldCheckBox** report control provides support for a checkbox element in an exported PDF report where the control's value can be modified. The InputFieldCheckBox control inherits most of its properties from the CheckBox report control.

Note:

- The InputFieldText control is part of the **Professional Edition**. With the Standard license, the control is not editable in an exported PDF file and is rendered as fixed text.
- The report's **CompatibilityMode** property should be set to CrossPlatform for the InputFieldText control to be editable in PDF. This is the default mode for Section reports. In GDI mode, the control is non-editable.
- In export formats other than PDF, such as RTF, XLS/XLSX, TXT, HTML, and MHT, the InputFieldCheckBox control is exported as a non-editable checkmark.

Important Properties

Property	Description
CheckAlignment	Gets or sets the alignment of the check box text within the control drawing area.
Checked	Gets or sets a value indicating whether the check box is in the checked state.
CheckStyle	Gets or sets the style of the check symbol inside the CheckBox. The available options are Check, Circle, Cross, Diamond, Square, and Star.

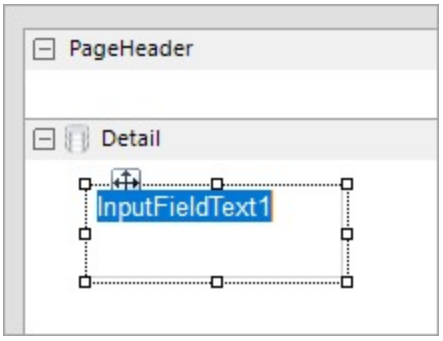
DataField	Gets or sets the field from the data source to bind to the control.
FieldName	Specifies a unique name of the field in the resulted PDF file.
ReadOnly	Prevents the user from changing the entered text content in the resulted PDF file.
Required	Forces the user to fill in the selected field of the resulted PDF file. If a user attempts to submit the form where the required field is blank, the error message appears and the empty required field is highlighted.
TabIndex	Sets the tab order of editable fields in the resulted PDF file. A field with the lowest TabIndex value is selected first.

InputFieldText

The InputFieldText report control in section reports provides support for editable text fields in an exported PDF report where the control's value can be edited. The InputFieldText control inherits most of its properties from the TextBox report control.

Note:

- The InputFieldText control is part of the **Professional Edition**. With the Standard license, the control is not editable in an exported PDF file and is rendered as fixed text.
- The report's **CompatibilityMode** property should be set to CrossPlatform for the InputFieldText control to be editable in PDF. This is the default mode for Section reports. In **GDI** mode, the control is non-editable.
- In export formats other than PDF, such as RTF, XLS/XLSX, TXT, HTML, and MHT, the InputFieldText control is exported as non-editable plain text independent of the Compatibility mode of the report.



Important Properties

Property	Description
FieldName	Specifies a unique name of the field in the resulted PDF file.
MaxLenth	Specifies the maximum length of the entered text in the resulted PDF file. When set to null, the text is not restricted to any specified length.
MultiLine	Gets or sets a value indicating whether to allow text to break to multiple lines within the control in the resulted PDF file.

Password	Displays the user-entered text as a series of asterisks (*).
ReadOnly	Prevents the user from changing the entered text content in the resulted PDF file.
Required	Forces the user to fill in the selected field of the resulted PDF file. If a user attempts to submit the form where the required field is blank, the error message appears and the empty required field is highlighted.
SpellCheck	Indicates whether to spell check the text during its input or not in the resulted PDF file.
TabIndex	Sets the tab order of editable fields in the resulted PDF file. A field with the lowest TabIndex value is selected first.

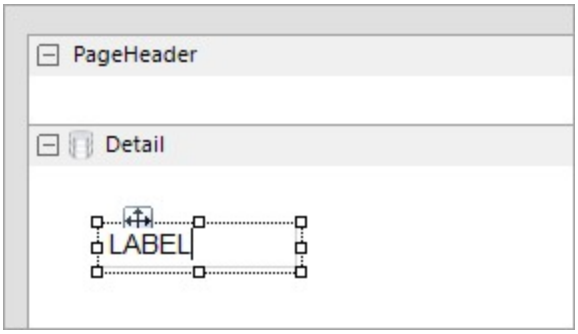
Label

The Label control is available in Section reports and is very similar to the Section report's TextBox control, just that the Label control takes only the static text. The main difference between the two controls is the **Angle** property of the Label control, and the following properties of the TextBox control: CanGrow, CanShrink, CountNullValues, Culture, DistinctField, OutputFormat, SummaryFunc, SummaryGroup, SummaryRunning, and SummaryType.

> **Note:** The CrossPlatform and the legacy GDI modes have different typography, so some text in migrated reports (for example, text without spacing, text with non-ASCII characters, etc.) may not appear correctly in the report preview. For the correct text rendering, you must manually update the report layout (for example, change the control's size).

Edit Mode

Double-clicking the Label control changes its mode to edit mode. In the edit mode, you can enter text directly in the control, or you can enter text in the **Text** property in the Properties window.



You can format text in the Label control in edit mode using the designer toolbar, or you can modify properties in the Properties window. Formats apply to all of the text in the control. Text formatting changes from the Properties window immediately appear in the control, and changes made in the toolbar are immediately reflected in the Properties window.

> **Note**: The TextBox and Label controls, unlike ReportInfo control, do not support page-related expressions.

Important Properties

Clicking the four-way arrow selects the control and reveals its properties.

Property	Description
Angle	Gets or sets the angle (slope) of the text within the control area. Set the **Angle** property to 900 to display text vertically.
CharacterSpacing	Gets or sets the space between characters in points.
DataField	Gets or sets the field name from the data source to bind to the control.
HyperLink	Gets or sets a URL to which the viewer navigates when the user clicks the label at run time. The URL becomes an anchor tag or a hyperlink in HTML and PDF exports.
LineSpacing	Gets or sets the space between lines in points.
MinCondenseRate	Specifies the minimal rate of the text horizontal scaling in percentages. Should be between 10 and 100.
MultiLine	Gets or sets a value indicating whether to allow text to break to multiple lines within the control.
ShrinkToFit	Gets or sets a value indicating whether to decrease the font size so that all of the text shows within the boundaries of the control.
Style	Gets or sets a style string for the label.
Text	Gets or sets the text to show on the report.
TextJustify	Specifies how to distribute text when the Alignment property is set to Justify. With any other Alignment setting, this property is ignored.
VerticalAlignment	Gets or sets the vertical position of the label's text within the bounds of the control.
VerticalText	Indicates whether to render the label's text vertically.
WrapMode	Indicates whether a multi-line label control wraps words or characters to the beginning of the next line when necessary.

Label Dialog Properties

General

Name: Enter a name for the label that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

DataField: Select a field from the data source to bind to the control.

Text: Enter static text to show in the label.

HyperLink: Enter a URL to use in the Viewer HyperLink event. The URL automatically converts to an anchor tag or hyperlink in PDF and HTML exports.

Appearance

Background Color: Select a color from the list to use for the background of the label.

Angle: Use the slider to set the degree of slope for the text within the control area.

Font

Name: Select a font family name or a theme font.

Size: Choose the size in points for the font.

Style: Choose **Normal** or **Italic**.

Weight: Choose from **Normal** or **Bold**.

Color: Choose a color to use for the text.

Decoration: Select check boxes for **Underline** and **Strikeout**.

GDI Charset: Enter a value to indicate the GDI character set to use. For a list of valid values, see MSDN Font.GDICharSet Property.

GDI Vertical: Select this checkbox to indicate that the font is derived from a GDI vertical font.

Format

Line spacing: Enter a value in points to use for the amount of space between lines.

Character spacing: Enter a value in points to use for the amount of space between characters.

Multiline: Select this check box to allow text to render on multiple lines within the control.

Textbox height

Can shrink text to fit fixed size control: Specify whether to decrease the text font to fit the control size.

Minimal rate of text horizontal shrinking (in %): Specify the percentage to which the text should be shrunk horizontally.

Text direction

RightToLeft: Select this check box to reverse the text direction.

Vertical text: Select this check box for top to bottom text.

Alignment

Vertical alignment: Choose **Top**, **Middle**, or **Bottom**.

Horizontal alignment: Choose **Left**, **Center**, **Right**, or **Justify**.

Justify method: Choose **Auto**, **Distribute**, or **DistributeAllLines**.

> **Note:** Setting the **Horizontal alignment** property to **Justify** enables the **Justify method** property options.

Wrap mode: Choose **NoWrap**, **WordWrap**, or **CharWrap** to determine whether and how text breaks to the next line.

Padding

Enter values in points to set the amount of space to leave around the label.

- **Top**
- **Left**
- **Right**
- **Bottom**

Keyboard Shortcuts

In edit mode, you can use the following keyboard shortcuts.

Key Combination	Action
Enter	New line.
Alt + Enter	Saves modifications and exits edit mode.

Esc	Cancels modifications and exits edit mode.

Line

The Line control allows you to draw vertical, horizontal or diagonal lines that visually separate or highlight areas within a section on a report.

> **Note**: If you need lines to span across report sections, please see the CrossSectionLine control.

You can use your mouse to visually move and resize the Line, or you can use the Properties window to change its **X1**, **X2**, **Y1**, and **Y2** properties to specify the coordinates for its starting and ending points.

Important Properties

Clicking the four-way arrow selects the control and reveals its properties.

Property	Description
LineColor	Gets or sets the color of the line.
LineStyle	Gets or sets the pen style used to draw the line. The line styles include Transparent, Solid, Dash, Dot, DashDot, DashDotDot, or Double.
LineWeight	Gets or sets the pen width of the line in points.
X1	Gets or sets the horizontal coordinate of the line's starting point.
X2	Gets or sets the horizontal coordinate of the line's end point.
Y1	Gets or sets the vertical coordinate of the line's starting point.
Y2	Gets or sets the vertical coordinate of the line's end point.

Line Dialog Properties

You can set the Line properties in the Line dialog. To open it, with the Line selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the line that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

Appearance

Line style: Select a line style to use for the line. You can set it to Transparent, Solid, Dash, Dot, DashDot, DashDotDot, or Double.

Line weight: Enter the width for the line.

Line color: Select a color to use for the line.

Anchor at bottom: Select this check box to automatically change the Y2 value to the value of the bottom edge of the containing section after it has grown to accommodate data at run time.

OleObject

Note: OleObject is not supported in:

- Section reports in the CrossPlatform compatibility mode
- WebDesigner
- .NET Core 3.1 and .NET 5+”

The OleObject control is hidden from the toolbox by default, and is only retained for backward compatibility. You can enable the OleObject control in the Visual Studio toolbox only.

To enable the control in the Visual Studio toolbox, you must change the **EnableOleObject** property to **true**. This property can be found here: `C:\Program Files (x86)\Common Files\MESCIUS\ActiveReports 18\ActiveReports.config`

Once enabled, you can add the OleObject control to reports. When you drop the control onto your report, the Insert Object dialog appears. This dialog allows you to create a new object or select one from an existing file.

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

Caution: The WPF Viewer does not support the OLE object. If you preview a report containing the OLE object in the WPF Viewer, the OLE object will not be displayed.

Important Properties

Property	Description
PictureAlignment	Gets or sets the position of the object's content within the control area.
Class	Specifies the class name of the Ole object.
SizeMode	Gets or sets a value that determines how the object is sized to fit the OleObject control area.

Insert Object Dialog

The **Insert Object** dialog provides the following two options:

- **Create New** lets you select from a list of object types that you can insert into your report.

Object Types

○ Adobe Acrobat Document ○ Microsoft Equation 3.0 ○ Microsoft Excel 97-2003 Worksheet ○ Microsoft excel Binary Worksheet ○ Microsoft Excel Chart ○ Microsoft Excel Macro-Enabled Worksheet ○ Microsoft Excel Worksheet ○ Microsoft Graph Chart ○ Microsoft PowerPoint 97-2003 Presentation ○ Microsoft PowerPoint 97-2003 Slide ○ Microsoft PowerPoint Macro-Enabled Presentation ○ Microsoft PowerPoint Macro-Enabled Slide	○ Microsoft PowerPoint Presentation ○ Microsoft PowerPoint Slide ○ Microsoft Word 97-2003 Document ○ Microsoft Word Document ○ Microsoft Macro-Enabled Document ○ OpenDocument Presentation ○ OpenDocument Spreadsheet ○ OpenDocument Text ○ Package ○ Paintbrush Picture ○ Wordpad Document

- **Create from File** allows you to insert the contents of the file as an object into your document so that you can display it while printing.

Picture

The Picture control in section reports is used to print an image on the report. In the **Image** property of the Picture control, you can select any image file to display on your report. You should use the **PictureAlignment** and **SizeMode** properties to control cropping and alignment.

Picture control supports Base64 string, Byte[], BMP, JPG, JPEG, JPE, GIF, PNG, EMF, WMF, and SVG formats in both GDI and CrossPlatform compatibility modes.

SVG Support

In CrossPlatform compatibility mode, SVG images are supported as vector images (original images) in all viewers and in PDF export; in other export formats, the images are exported as raster images. The GDI compatibility mode is recommended for use only as a fallback since SVG images are converted to raster images in all viewers and export formats.

Picture Dialog Properties

You can set the Picture properties in the Picture dialog. To open it, with the Picture selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the picture control that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

DataField: Select a field from the data source to bind to the control.

Choose image: Click this button to open a dialog where you can navigate to a folder from which to select an image file to display.

Hyperlink: Enter a URL to use in the Viewer HyperLink event. The URL automatically converts to an anchor tag or hyperlink in PDF and HTML exports.

Title: Gets or sets the title of the control. The entered text appears as a tag when an exported pdf file is opened in the Text Editor. The entered text also appears on hovering the control in an exported PDF File and is audible in the **Read Out Loud** mode of the Acrobat Reader DC.

Description: Enter text to describe the image for those who cannot see it. This is used in the HTML export for the "alt" attribute of the img tag.

Appearance

Line style: Select a line style to use for the borderline. You can set it to Transparent, Solid, Dash, Dot, DashDot, DashDotDot, or Double.

Line weight: Enter the width for the borderline.

Line color: Select a color to use for the borderline.

Background color: Select a color to use for the background of the picture control.

Picture alignment: Select how to align the image within the control. You can select from TopLeft, TopRight, Center, BottomLeft, or BottomRight.

Size mode: Select how to size the image within the control. You can select from Clip, Stretch, or Zoom - Clip uses the original image size and clips off any excess, Stretch fits the image to the size and shape of the control, and Zoom fits the image into the control while maintaining the aspect ratio of the original image.

ReportInfo

You can use the ReportInfo control in section reports to quickly display page numbers, page counts, and report dates. The ReportInfo control is a text box with a selection of preset FormatString options. You can set page counts to count the pages for the entire report, or for a specified group.

You can customize the preset values by editing the string after you select it. For example, if you want to display the total number of pages in the ReportHeader section, you can enter a value like: `Total of {PageCount} pages`

See Date, Time, and Number Formatting for information on creating formatting strings

⚠ **Caution**: 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 Section Reports.

ReportInfo Dialog Properties

You can set the ReportInfo properties in the ReportInfo dialog. To open it, with the ReportInfo selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the control that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

Appearance

Background Color: Select a color to use for the background of the control.

Font

Name: Select a font family name or a theme font.

Size: Choose the size in points for the font.

Style: Choose **Normal** or **Italic**.

Weight: Choose from **Normal** or **Bold**.

Color: Choose a color to use for the text.

Decoration: Select check boxes for **Underline** and **Strikeout**.

GDI Charset: Enter a value to indicate the GDI character set to use. For a list of valid values, see MSDN Font.GDICharSet Property.

GDI Vertical: Select this checkbox to indicate that the font is derived from a GDI vertical font.

Format

Format string: Select a formatted page numbering or report date and time value to display in the control. You may also type in this box to change or add text to display along with the formatted date and page number values.

Multiline: Select this check box to allow text to render on multiple lines within the control.

ReportInfo height

Can increase to accommodate contents: Select this check box to set CanGrow to True.

Can decrease to accommodate contents: Select this check box to set CanShrink to True.

Text direction

RightToLeft: Select this check box to reverse the text direction.

Alignment

Vertical alignment: Choose **Top**, **Middle**, or **Bottom**.

Horizontal alignment: Choose **Left**, **Center**, **Right**, or **Justify**.

Wrap mode: Choose **NoWrap**, **WordWrap**, or **CharWrap** to select whether to wrap words or characters to the next line.

Summary

SummaryGroup: Select a GroupHeader section in the report to display the number of pages in each group when using the PageCount.

SummaryRunning: Select None, Group, or All to display a summarized value.

Displaying 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. This property provides the following predefined options for page numbering and date and time formatting.

Numbering Format	Description
Page {PageNumber} of {PageCount} on {RunDateTime}	Display the page numbers along with Date and Time in the following format : `Page 1 of 100 on 1/31/2020 2:45:50 PM`
Page {PageNumber} of {PageCount}	Display the only the page numbers in the following format : `Page 1 of 100`
{RunDateTime:}	Display the Date and Time in the following format : `1/31/2020 2:45:50 PM`

{RunDateTime: M/d}	Display the Date in the following format : `1/31`
{RunDateTime: M/d/yy}	Display the Date in the following format : `1/31/20`
{RunDateTime: M/d/yyyy}	Display the Date in the following format : `1/31/2020`
{RunDateTime: MM/dd/yy}	Display the Date in the following format : `01/31/20`
{RunDateTime: MM/dd/yyyy}	Display the Date in the following format : `01/31/2020`
{RunDateTime: d-MMM}	Display the Date in the following format : `31-Jan`
{RunDateTime: d-MMM-yy}	Display the Date in the following format : `31-Jan-20`
{RunDateTime: d-MMM-yyyy}	Display the Date in the following format : `31-Jan-2020`
{RunDateTime: dd-MMM-yy}	Display the Date in the following format : `31-Jan-20`
{RunDateTime: dd-MMM-yyyy}	Display the Date in the following format : `31-Jan-2020`
{RunDateTime: MMM-yy}	Display the Date in the following format : `Jan-20`
{RunDateTime: MMM-yyyy}	Display the Date in the following format : `Jan-2020`
{RunDateTime: MMMM-yy}	Display the Date in the following format : `January-20`
{RunDateTime: MMMM-yyyy}	Display the Date in the following format : `January-2020`
{RunDateTime: MMMM d,yyyy}	Display the Date in the following format : `January 31, 2020`
{RunDateTime: M/d/yy h:mm tt}	Display the Date and Time in the following format : `1/31/20 2:45 PM`
{RunDateTime: M/d/yyyy h:mm tt}	Display the Date and Time in the following format : `1/31/2020 2:45 PM`
{RunDateTime: M/d/yy h:mm}	Display the Date and Time in the following format : `1/31/20 2:45`
{RunDateTime: M/d/yyyy h:mm}	Display the Date and Time in the following format : `1/31/2020 2:45`

1. From the Toolbox, drag the **ReportInfo** control to the desired location on the report.
2. With the ReportInfo control selected in the Properties window, drop down the **FormatString** property and select the preset value that best suits your needs.

Displaying Group Level Page Counts

1. From the toolbox, add the ReportInfo control to the GroupHeader or GroupFooter section of a report and set the **FormatString** property to a value that includes PageCount.
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**.

RichTextBox

The RichTextBox control is used to display, insert, and manipulate formatted text. It is different from the TextBox control in several ways. The most obvious is that it allows you to apply different formatting to different parts of its content.

You can add fields to the text you enter directly in the RichTextBox control by right-clicking and choosing **Insert Fields** and providing a field name or you can load an RTF, an HTML, or a Text file into the control.

> **Note**: The RichTextBox control supports loading an RTF file at design time with both **CrossPlatform** and **GDI** compatibility modes of a report. However, the RichTextBox control does not support the RTF file at run-time when the report's **CompatibilityMode** is set to **CrossPlatform**.

Supported Formats

ActiveReports supports most of the XHTML 1.1 standard; please see Wikipedia for more information. The limited support for reading XHTML5 and HTML5 documents is also available. The RichTextBox control supports formats such as:

- Simple RTF content (without complicated tables, styles) with data binding support.
- Complicated RTF content (if **RtfRenderingType** property is set to any mode, other than **RTF**); uses the latest available WordPad features.
- Plain text content (is converted to simple HTML).
- HTML content (almost all XHTML 1.1 specification with some HTML5 extensions are supported)

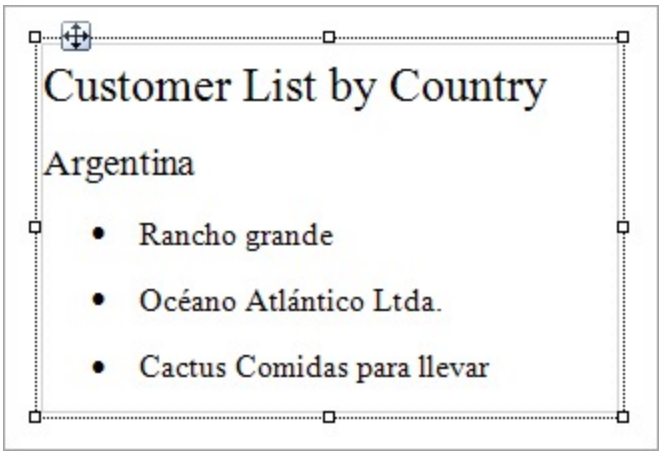
Load File Dialog

With the control selected on the report, in the Commands section at the bottom of the Properties panel, you can click **Load file** to open the dialog. This allows you to select a file to load into the control at design time. Supported file types are as follows.

- Text (*.txt)
- RichText (*.rtf)
- HTML (*.htm, *.html)

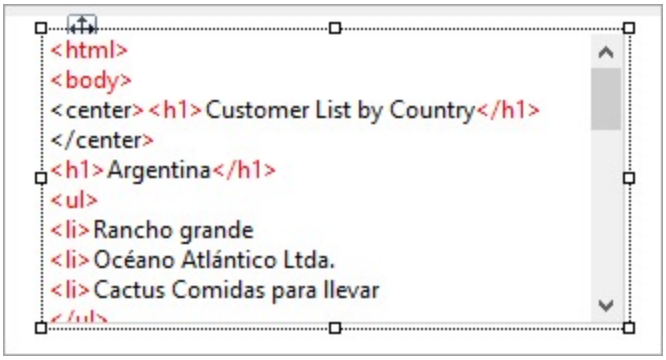
RichText file

When you load a RichText file, the control will display the file in edit mode.



Text/HTML file

When you load a Text or an HTML file, the control will display the HTML markup in the edit mode.



⚠ **Important**: The HTML text in an HTML file, loaded into the RichTextBox control, must be enclosed in the **<body> </body>** tags. Otherwise, the HTML data is converted into RTF.

Important Properties

Property	Description
CanGrow	Determines whether ActiveReports should increase the height of the control based on its content.
CanShrink	Determines whether ActiveReports should decrease the height of the field based on its value.
RenderRtfAsContinuousImage	If True (default), the property renders the content as one continuous image split across the pages. This property is actual if **RtfRenderingType** property is set to either Metafile or PNG. **Note**: This property should be set to 'False' if the RtfRenderingType property is set to a Metafile or PNG, to avoid any trimmed text at the end of the pages.
RtfRenderingType	Renders the RTF control with a legacy renderer or with the WordPad as a metafile or

	raster image. Only the legacy RTF renderer supports fields.
DataField	Gets or sets the field name from the data source to bind to the control.

RichTextBox Dialog Properties

You can set the RichTextBox properties in the RichTextBox dialog. To open it, with the RichTextBox selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the RichTextBox that is unique within the report. This name is displayed in the Document Outline. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

DataField: Select a field from the data source to bind to the control.

Max length: Enter the maximum number of characters to display in the control. If you do not specify a value, it displays an unlimited number of characters.

Appearance

Background Color: Select a color to use for the background of the control.

Format

RichTextBox height

Can increase to accommodate contents: Select this check box to set CanGrow to True.

Can decrease to accommodate contents: Select this check box to set CanShrink to True.

Multiline: Select this check box to allow the control to display multiple lines of text.

Supported HTML Tags and Attributes

The RichTextBox control supports HTML tags and attributes same as the FormattedText control. For more information, please see the topic on FormattedText.

Write an RTF file to load into a RichTextBox control

1. Open WordPad, and paste a formatted text into it, for example:

   [Paste the following section 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

- Familia Arquibaldo
- Wellington Improtadora
- Que Delícia
- Tradição Hipermercados
- Ricardo Adocicados
- Hanari Carnes
- Queen Cozinha
- Comércio Mineiro
- Gourmet Lanchonetes

- -

2. Save the file as **sample.rtf**.

Note: The RichTextBox control is limited in its support for advanced RTF features such as the ones supported by Microsoft Word. In general, the features supported by WordPad are supported in this control.

Load an RTF file into a RichTextBox control

1. On the report design surface, add a RichTextBox control.
2. With the RichTextBox control selected, at the bottom of the Properties Panel, click **Load file**.
3. In the **Open** dialog that appears, browse to an *.RTF file (For example, sample.rtf) and click the **Open** button to load the file in the RichTextBox control.

Write an HTML file to load into a RichTextBox control

1. Open a Notepad, and paste an HTML code into it, for example:

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>Brazi1>
<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>
</body>
</html>
```

2. Save the file as **sample.html**.

Load an HTML file into the RichTextBox control

1. On the report design surface, add a RichTextBox control.
2. With the RichTextBox control selected, at the bottom of the Properties window, click **Load file**.
3. In the **Open** dialog that appears, browse to a *.html file (For example, sample.html) and click the **Open** button to load the file in the RichTextBox control.

ActiveReports also provides limited support for reading XHTML5 and HTML5 documents.

Keyboard Shortcuts

In the edit mode, you can use the following keyboard shortcuts.

Key Combination	Action
Enter	New line.

Alt + Enter	Saves modifications and exits edit mode.
Esc	Cancels modifications and exits edit mode.

Shape

The Shape report control is used to display one of the available shape types on a report. You can add a shape report control to a report by dragging it from the toolbox and dropping it onto the report design surface.

In the **Style** property of the Shape report control, you can select **Rectangle**, **RoundRect** or **Ellipse**, or you can use an expression to assign fields, datasets, parameters, constants, operations or common values to it.

You can highlight different sections or parts of a report using a shape report control. For example, you can use a Rectangle as border around different report controls or the entire page or you can use an Ellipse to highlight a note on your report.

Important Properties

Clicking the Shape control reveals its properties in the Properties window.

Property	Description
LineColor	Gets or sets the color of the shape lines.
LineStyle	Gets or sets the pen style used to draw the line.
LineWeight	Gets or sets the pen width used to draw the shape.
Style	Gets or sets the shape type to draw. You can select from Rectangle, Ellipse and a RoundedRect.
RoundingRadius	Sets the radius of each corner for the RoundRect shape type. You can select Default, TopLeft, TopRight, BottomLeft or BottomRight. Selecting Default sets the radius of all the corners of the Shape control to a specified percentage. Default value = 10 (percent).

Shape Dialog Properties

You can set the Shape properties in the Shape dialog. To open it, with the Shape selected on the report, under the Properties window, click the **Property dialog** link.

General

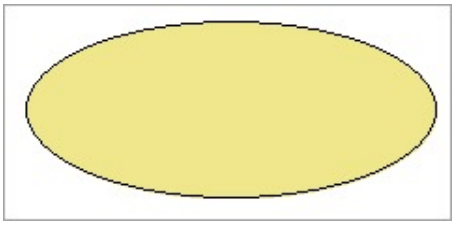
Name: Enter a name for the shape that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), backslash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

Appearance

Shape type: Select the type of shape to display. You can choose from **Rectangle**, **Ellipse**, or **RoundRect.** For a circle, set the control width and height properties to the same value, and choose Ellipse, or choose RoundRect and set the Rounding radius to 100%.

Rectangle	RoundRect	Ellipse
		

Rounded Rectangle: When the Shape type is set to **RoundRect**, you can specify the radius for each corner of the shape independently. Drag the handlers ◇ available at each corner of the shape to set the value of the radius at each corner.

> **Note:** To enable specific corners, select the checkbox available near each corner of the Shape control.

- **Use the same radius on specified corners**: Select this option to apply the same radius to all selected corners of the shape.
- **Use different radius on specified corners:** Select this option to apply a different radius to each selected corner of the shape.

Line style: Select a line style to use for the shape line. You can set it to Transparent, Solid, Dash, Dot, DashDot, DashDotDot, or Double.

Line weight: Enter the width for the shape line.

Line color: Select a color to use for the shape line.

Background style: Select a background style for the shape from Solid, Gradient, and Pattern. Depending on the style selected, other properties are available.

Background color: Select a color to use for the background of the shape.

Background color2: Select a color2 to use for the background of the shape. This property becomes available if the **Background style** is set to Gradient or Pattern.

Gradient style: Select a gradient style. This property becomes available if the **Background style** is set to Gradient.

Background pattern: Select a background pattern to hatch style. This property becomes available if the **Background style** is set to Pattern.

SubReport

In section reports, you can use the SubReport control to embed a report into another report. Once you place the SubReport control on a report, use code to create an instance of the report you want to load in it, and attach the report object to the SubReport.

You can also pass parameters to the subreport from the main report so that data related to the main report displays in each instance of the subreport.

Note: You cannot use a Page/RDLX report as the target of a SubReport in a Section report.

Remove page-dependent features from reports to be used as subreports

Subreports are disconnected from any concept of a printed page because they render inside the main report. 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

Coding best practices

Use the **ReportStart** event of the main report to create an instance of the report for your SubReport control, and then dispose of it in the **ReportEnd** event. This way, you are creating only one subreport instance when you run the main report.

In the **Format** event of the containing section, use the **Report** property of the SubReport control to attach a report object to the SubReport control.

Caution: It is not a recommended practice to initialize the subreport in the **Format** event. Doing so 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.

Important Properties

Clicking the SubReport control reveals its properties in the Properties window.

Property	Description
CanGrow	Determines whether ActiveReports increases the height of the control based on its content.
CanShrink	Determines whether ActiveReports decreases the height of the control based on its value.
CloseBorder	By default, the bottom border of the control does not render until the end of the subreport. Set this property to True to have it render at the bottom of each page. (Only available in code)
Report	Attaches a report object to the control. (Only available in code)

SubReport Dialog Properties

With the control selected on the report, in the Commands section at the bottom of the Properties window, you can click the **Property dialog** command to open the dialog.

General

Name: Enter a name for the SubReport that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), back slash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

ReportName: Enter the name of the report.

Format

SubReport Height

Can increase to accommodate contents: Clear this check box to set CanGrow to False.

Can decrease to accommodate contents: Clear this check box to set CanShrink to False.

Using SubReport Control in Reports

To use the SubReport control in your report, you must create two reports - one parent and one child report. Then, after dragging the SubReport control onto the parent report, you need to add code to create an instance of the child report in the parent report, and another code to display the child report in the SubReport control on a parent report.

For details on working with SubReport control in section reports, see Work with Subreports.

TextBox

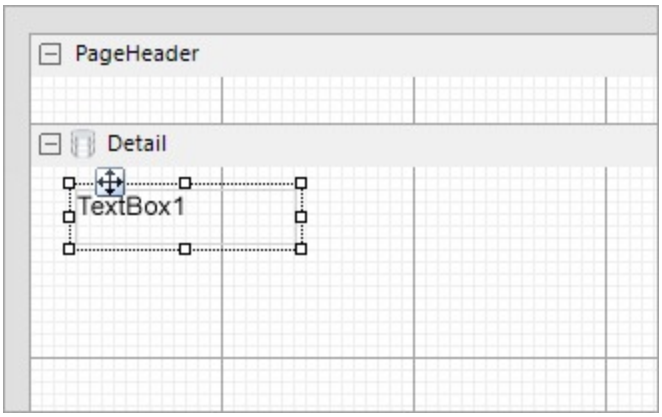
The TextBox control is the basis of reporting as it is used to display textual data in section reports. You can add multiple lines, wrap text to the control size, shrink text to fit within the bounds of control, and perform basic formatting.

The TextBox control can be bound to any data or set at run time. Also, it is the same control that forms when you drag a field onto a report from the Report Explorer. In the **Text** property of the TextBox, you can enter static text or an expression. To enter a text directly into the TextBox, just double-click inside the control on the design surface of the report. An expression in TextBox can display fields from a database, calculate a value, or visually display data.

> **Note**: There is a difference in how the text is rendered in the CrossPlatform and in the GDI compatibility modes. Some text in the CrossPlatform mode (for example, text without spacing, text with non-ASCII characters, etc.) may not appear correctly in the report preview. For the correct text rendering, you must manually update the report layout (for example, change the control's size).

Edit Mode

You can double-click in the TextBox control to enter edit mode and enter text directly in the control, or you can enter text in the Properties window or code through the **Text** property.



You can format text in the TextBox control in edit mode using the ActiveReports toolbar, or you can modify properties in the Properties window. Formats apply to all of the text in the control. Text formatting changes in the Properties window immediately appear in the control, and changes made in the toolbar are immediately reflected in the Properties window.

Note: The TextBox and Label controls, unlike ReportInfo control, do not support page-related expressions.

Important Properties

Clicking the TextBox control reveals its properties in the Properties window.

Property	Description
CanGrow	Determines whether ActiveReports should increase the height of the control based on its content.
CanShrink	Determines whether ActiveReports should decrease the height of the control based on its value.
CharacterSpacing	Gets or sets a character spacing in points.
CountNullValues	Boolean which determines whether DBNull values should be included as zeroes in summary fields.
Culture	Gets or sets CultureInfo used for value output formatting.
DataField	Gets or sets the field name from the data source to bind to the control.
DistinctField	Gets or sets the name of the data field used in a distinct summary function.
HyperLink	Gets or sets the hyperlink for the text control.
LineSpacing	Gets or sets a line spacing in points.
MinCondenseRate	Specifies the minimal rate of the text horizontal scaling in percentages. Should be between 10 and 100.
MultiLine	Gets or sets a value indicating whether this is a multi-line textbox control.
OutputFormat	Gets or sets the mask string used to format the Value property before placing it in the Text property.
ShrinkToFit	Determines whether ActiveReports decreases the font size when text values exceed available space.

Style	Gets or sets a style string for the textbox.
SummaryFunc	Gets or sets the summary function type used to process the DataField Values.
SummaryGroup	Gets or sets the name of the group header section that is used to reset the summary value when calculating subtotals.
SummaryRunning	Gets or sets a value that determines whether that data field summary value will be accumulated or reset for each level (detail, group, or page).
SummaryType	Gets or sets a value that determines the summary type to be performed.
Text	Gets or sets the formatted text value to be rendered in the control.
TextJustify	Specifies text justification with TextAlign set to Justify.
VerticalAlignment	Gets or sets the position of the textbox's text vertically within the bounds of the control.
VerticalText	Gets or sets whether to render text according to vertical layout rules.
WrapMode	Indicates whether a multi-line textbox control automatically wraps words or characters to the beginning of the next line when necessary.

TextBox Dialog Properties

You can set the TextBox properties in the TextBox dialog. To open it, with the TextBox selected on the report, under the Properties window, click the **Property dialog** link.

General

Name: Enter a name for the textbox that is unique within the report. This name is displayed in the Document Outline and in XML exports. You can only use underscore (_) as a special character in the Name field. Other special characters such as period (.), space ( ), forward slash (/), back slash (\), exclamation (!), and hyphen (-) are not supported.

Tag: Enter a string that you want to persist with the control. If you access this property in code, it is an object, but in the Properties window or Property dialog, it is a string.

Visible: Clear this check box to hide the control.

DataField: Select a field from the data source to bind to the control.

Text: Enter static text to show in the textbox. If you specify a DataField value, this property is ignored.

HyperLink: Enter a URL to use in the Viewer HyperLink event. The URL automatically converts to an anchor tag or hyperlink in PDF and HTML exports.

Appearance

Background Color: Select a color to use for the background of the textbox.

Font

Name: Select a font family name or a theme font.

Size: Choose the size in points for the font.

Style: Choose **Normal** or **Italic**.

Weight: Choose from **Normal** or **Bold**.

Color: Choose a color to use for the text.

Decoration: Select check boxes for **Underline** and **Strikeout**.

GDI Charset: Enter a value to indicate the GDI character set to use. For a list of valid values, see MSDN Font.GDICharSet Property.

GDI Vertical: Select this checkbox to indicate that the font is derived from a GDI vertical font.

Format

Line Spacing: This property sets the space between lines of text.

Character Spacing: This property sets the space between characters.

Multiline: Select this check box to allow text to render on multiple lines within the control.

Textbox height

Can increase to accommodate contents: Select this check box to set CanGrow to True.

Can decrease to accommodate contents: Select this check box to set CanShrink to True.

Can shrink text to fit fixed size control: Select this check box to set ShrinkToFit to True.

Minimal rate of text horizontal shrinking (in %): Specify the percentage to which the text should be shrunk horizontally.

Text direction

RightToLeft: Select this check box to reverse the text direction.

Vertical text: Select this check box for top to bottom text.

Alignment

Vertical alignment: Choose **Top**, **Middle**, or **Bottom**.

Horizontal alignment: Choose **Left**, **Center**, **Right**, or **Justify**.

Justify method: Choose **Auto**, **Distribute**, or **DistributeAllLines**.

Wrap mode: Choose NoWrap, WordWrap, or CharWrap to select whether to wrap words or characters to the next line.

> **Note:** You must select **Justify** in the **Horizontal alignment** property to enable the **Justification method** property options.

Padding

- **Top**: Set the top padding in points.
- **Left**: Set the left padding in points.
- **Right**: Set the right padding in points.
- **Bottom**: Set the bottom padding in points.

Summary

SummaryFunc: Select a type of summary function to use if you set the SummaryRunning and SummaryType properties to a value other than None. For descriptions of the available functions, see the **SummaryFunc ('SummaryFunc Enumeration' in the on-line documentation)**enumeration.

SummaryGroup: Select a section group that you have added to the report. If you also set the SummaryRunning property to Group, the textbox summarizes only the values for that group.

SummaryRunning: Select **None**, **Group**, or **All**. If None, the textbox shows the value for each record. If Group, the textbox summarizes the value for the selected SummaryGroup. If All, the textbox summarizes the value for the entire report.

SummaryType: Select the type of summary to use. For descriptions of the available types, see the **SummaryType ('SummaryType Enumeration' in the on-line documentation)** enumeration.

Distinct Field: Select a field to use with one of the SummaryFunc distinct enumerated values.

Count null values: Select this check box to include null values as zeroes in summary fields.

Text Justification

The **TextJustify** property of a Textbox control provides you justification options for aligning your text within a control. It is important to note that the **Alignment** property must be set to Justify for **TextJustify** property to affect the text layout.

> **Note**: The TextJustify property is also available in the Label control.

You can choose from the following values of the TextJustify property:

Auto

Results in Standard MSWord like justification where space at the end of a line is spread across other words in that line. This is the default value.

Distribute

Spaces individual characters within a word, except for the last line.

DistributeAllLines

Spaces individual characters within words and also sets the justification on the last line according to the length of other lines.

To set text justification,

1. Select the **TextBox** control to view its properties in the Properties window.
2. In the Properties window, set the **Alignment** to **Justify**.
3. Go to the **TextJustify** property and from the drop down list select any one option.

Text justification is supported when you preview a report in the Viewer, print a report, or export a report in PDF, and TIFF formats.

Shrink Text to Fit in a Control

When working with the TextBox or Label control in a Section report, you can use the **ShrinkToFit** property to reduce the size of the text so that it fits within the bounds of the control. The text shrinks at run time, so you can see the reduced font size when you preview, print or export the report.

You can use other text formatting properties in combination with the **ShrinkToFit** property.

> ⚠ **Caution**:
>
> - When both **CanGrow** and **ShrinkToFit** are set to True, CanGrow setting is ignored and only ShrinkToFit is applied.
> - When **ShrinkToFit** is set to True and **Angle** is set to a value other than 0, the ShrinkToFit property is ignored.
> - **ShrinkToFit** property does not work in the following conditions:
>   - VerticalText property is set for a control.
>   - Multiline property is set to False.

On exporting a report, various file formats handle **ShrinkToFit** differently. ShrinkToFit gets exported in all formats except **Text**.

Multiline in TextBox

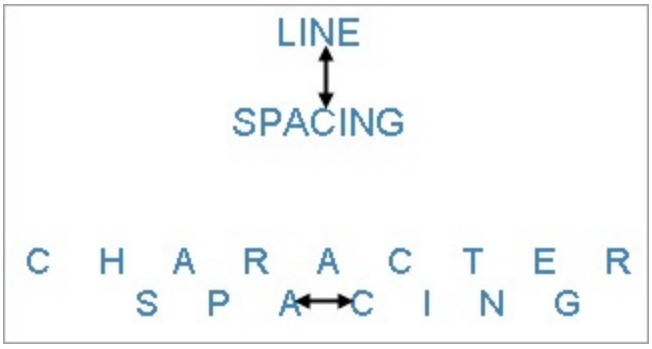
You can display multiline text in TextBox and some other controls such as RichTextBox and Label for section reports.

In a Section report, you can enable multiline display in your report control by setting the **Multiline** property to True. Then, with your control in edit mode, insert line breaks at the desired location using the **Enter** key or **Ctrl** + **Enter** keys to create multiline text. However, when the MultiLine property is set to False, the text entered into the control is displayed on a single line.

> **Note**: In edit mode, scrollbars appear automatically to fit multiline content within a control. However, these are not displayed at preview, so you may need to adjust the **Size** property of the control to display all of the text.

Line Spacing and Character Spacing

In edit mode, scrollbars appear automatically to fit multiline content within a control. However, these are not displayed at preview, so you may need to adjust the **Size** property of the control to display all of the text.



To set Line or Character spacing,

1. On the design surface, click the TextBox control to display it in the Properties window.
2. In the Properties window, click the **Property dialog** command at the bottom to open the control dialog.
3. In the TextBox dialog, go to the **Format** page and set the Line Spacing or Character Spacing values in points.

Line and character spacing are supported when you preview a report, print a report, or export a Section report in HTML , PDF, and TIFF formats.

Keyboard Shortcuts

In the edit mode, you can use the following keyboard shortcuts.

Key Combination	Action
Enter	New line.
Alt + Enter	Saves modifications and exits edit mode.
Esc	Cancels modifications and exits edit mode.

In Visual Studio Integrated Designer, you can disable this feature in the **EditModeEntering Event (on-line documentation)** and **EditModeExit Event (on-line documentation)**.

Data Binding

This section discusses the data binding using the ActiveReports Designer and is divided into two main parts.

- Data Binding in Page/RDLX Reports
- Data Binding in Section Reports

Data Binding in Page/RDLX Reports

The data binding process involves setting up the data connection and associating the data with the report.

On creating a new report, you see the **New Report** wizard, where you first select a report type (a Page report or an

RDLX), and then select the data source type from where the data binding process starts. The data binding requires you to set up a connection to the data source by configuring the data source connection properties, and then set up queries to add a dataset to fetch the data you want to show in the report.

In an existing report where you want to bind the report to data, you should navigate to the Report Explorer, right-click the Data Sources node, and select the Add Data Source option, followed by right-clicking an existing data source and selecting Add Data Set.

This section discusses the data binding using the ActiveReports Designer. For information on run-time binding, please see Bind a Page/RDLX report to a Data Source at Run Time topic.

Supported Data Sources

The supported data sources for designing Page/RDLX reports are:

- Microsoft SQL Server
- MySQL
- PostgreSQL
- SQLite
- JSON
- XML
- CSV
- Excel
- JSON API
- Dataset Provider
- Microsoft ODBC Provider
- Microsoft OLEDB Provider

Connect to a Data Source

This topic discusses about connecting to a data source at design time through the Report Explorer in a Page report or an RDLX report. You can also connect to a shared data source if you want to use the same data source for many reports.

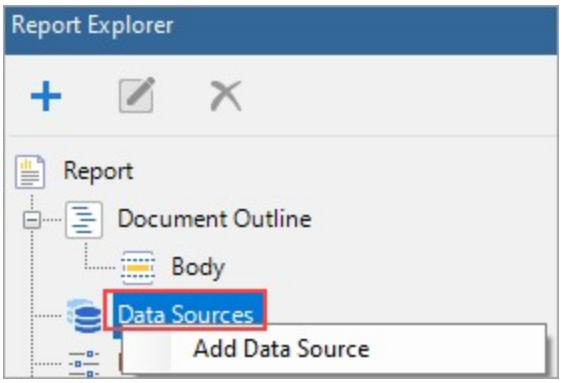
Checkout data binding while creating a new report using the wizard that guides you through the whole process in the topics that follow.

In ActiveReports Designer, you can setup the data source connection in the Report Data Source dialog. The Report Data Source dialog is where you select the type of data source to use, provide a connection string, and choose other options for your data source. You can also decide to use a shared data source, use a single transaction, and select a method for handling credentials. Once you add a data source, it appears in the Report Explorer under the Data Sources node. You can also add multiple data sources in a single report.

Add a data source

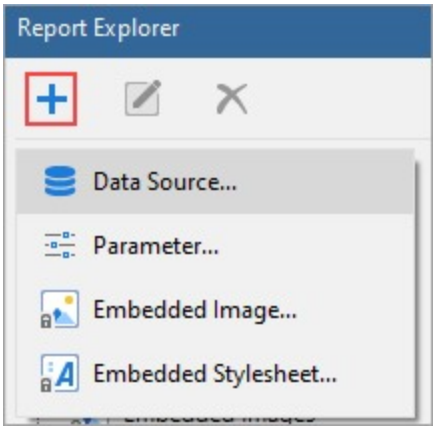
Use the following steps to add a new data source in your report.

1. In the designer, go to the **Report Explorer**-
    - Right-click the **Data Sources** node and select the **Add Data Source** option.



Or,

- Click the **Add** button and then select the **Data Source** option.



2. In the **Report Data Source** dialog that appears, select the **General** page and enter the name of the data source. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under **Type**, select the type of data source you want to use.
   - Microsoft SQL Client Provider
   - CSV
   - Dataset and Object Providers
   - Excel
   - JSON Provider
   - Microsoft ODBC Provider
   - Microsoft OLEDB Provider
   - MySQL Provider
   - PostgreSQL Provider
   - XML Provider
4. Depending on the data source selected, different options/tabs are displayed to configure the connection.
   - For SQL or OleDb data sources, **Connection Properties**, **Connection String**, and **Advanced Settings** tabs appear.
   - For XML data source, the **Connection Properties** and **Connection String** tabs appear.
   - For JSON data source, the **Content**, **Schema** and **Connection String** tabs appear.
   - For other data source types, only the **Connection String** tab appears.

> **Note:** You can create dynamic connection strings by using the runtime variables, such as report parameters.

5. Enter the configuration details for the chosen data source type. See **Report Data Source dialog properties** for more information.
6. Click the **OK** button on the lower right corner to close the dialog. You have successfully connected the report to a data source.

Custom Data Providers

You can also implement some custom data providers such following, by manually setting up the dependencies and configuration file:

- SQLite (see Configure ActiveReports)

  For information on writing an SQL query that you can use for SQLite, see the SQLite Tutorial. You will note that there are some differences in writing a standard SQL query and an SQL query for SQLite.

- Oracle (see Oracle Data Provider sample)
- OData (see OData Data Source sample)

Also, see the Custom Data Provider sample.

You can also create a shared data source if you want to use the same data source in different reports.

Edit a data source

These steps assume that you have already connected your report to a data source.

1. To open the Report Data Source dialog, do one of the following:
   - In the Report Explorer, right-click a data source node and in the context menu that appears, select **Edit** or Edit Shared Data Source option.
   - In the Report Explorer toolbar, click the **Edit** button.

   
2. In the **Report Data Source** dialog that appears, edit the data connection information.

Report Data Source dialog properties

The Report Data Source dialog provides the following pages where you can set data source properties.

GENERAL

The General page of the Report Data Source dialog is where you can set the Name, Type, and Connection string of a new data source, or choose to use a shared data source reference.

- **Name**: Enter a name for the data source. This name must be unique within the report. By default, the name is set to DataSource1.
- **Shared Reference**: Select this check box to connect to a shared data source reference. See **Add a Shared Data Source** for further information. Once you have chosen the **Shared Reference** option, the **Reference** field to select a Shared Data Source becomes available. In the **Reference** field, select **From File** and select a shared data source file on your machine.

  If the **Shared Reference** checkbox is clear, you can select a data source type from the **Type** dropdown field.
- **Use Single Transaction**: Check this option to execute the datasets using this data source in a single transaction.
- **Type**: Choose the type of data source provider you want to use. The available options are Microsoft SQL Client Provider, Csv Provider, DataSet Provider, Object Provider, JSON Provider, Microsoft ODBC Provider, Microsoft OleDb Provider, and XML Provider.

  If you select SQL or OleDb as the data source **Type** value, the **Connection Properties**, **Connection String** and **Advanced Settings** pages appear under the **Connection** section. For XML data source, **Connection Properties** and **Connection String** pages appear. For JSON data source, **Content**, **Schema**, and **Connection String** pages appear. In other data source types, only the **Connection String** page appears.

CREDENTIALS

The Credentials page gives you the following four options for the level of security you need for the data in your report. This page is available if the data source **Type** value is set to Microsoft SQL Client Provider, Microsoft ODBC Provider, and Microsoft OleDb Provider.

- **Use Windows Authentication**: Select this option when you know that any users with a valid Windows account are cleared for access to the data, and you do not want to prompt them for a user name and password.
- **Use a specific user name and password**: Select this option when you want to allow only a single user name and password to access the data in the report.
- **Prompt for credentials**: Select this option when there is a subset of users who can access the data. The Prompt string textbox allows you to customize the text requesting a user name and password from users.
- **No credentials**: Select this option only if the data in the report is for general public consumption.

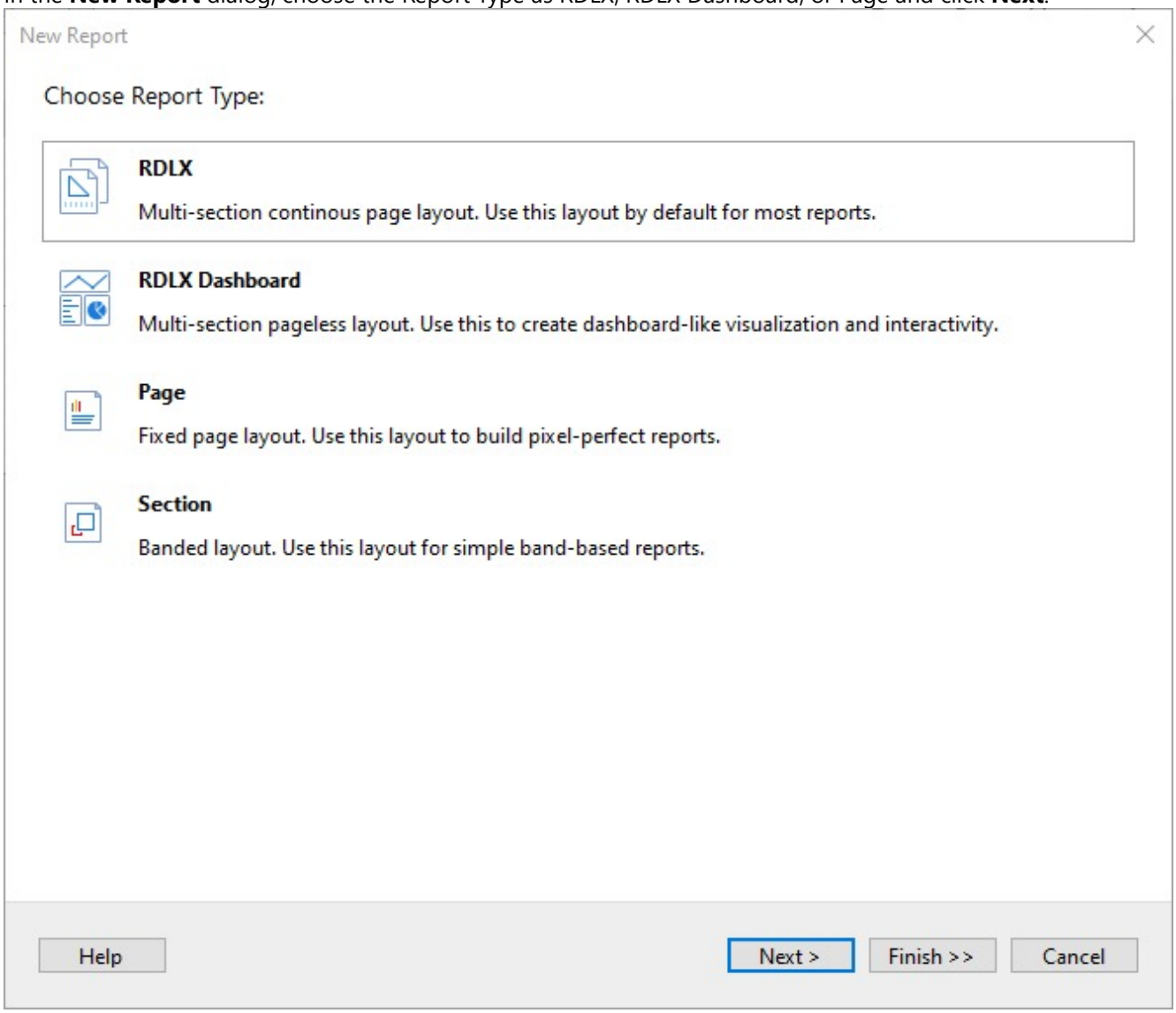
MS SQL Server

This article explains connecting a Page or an RDLX report to a MS SQL Server/ Microsoft SQL Client data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

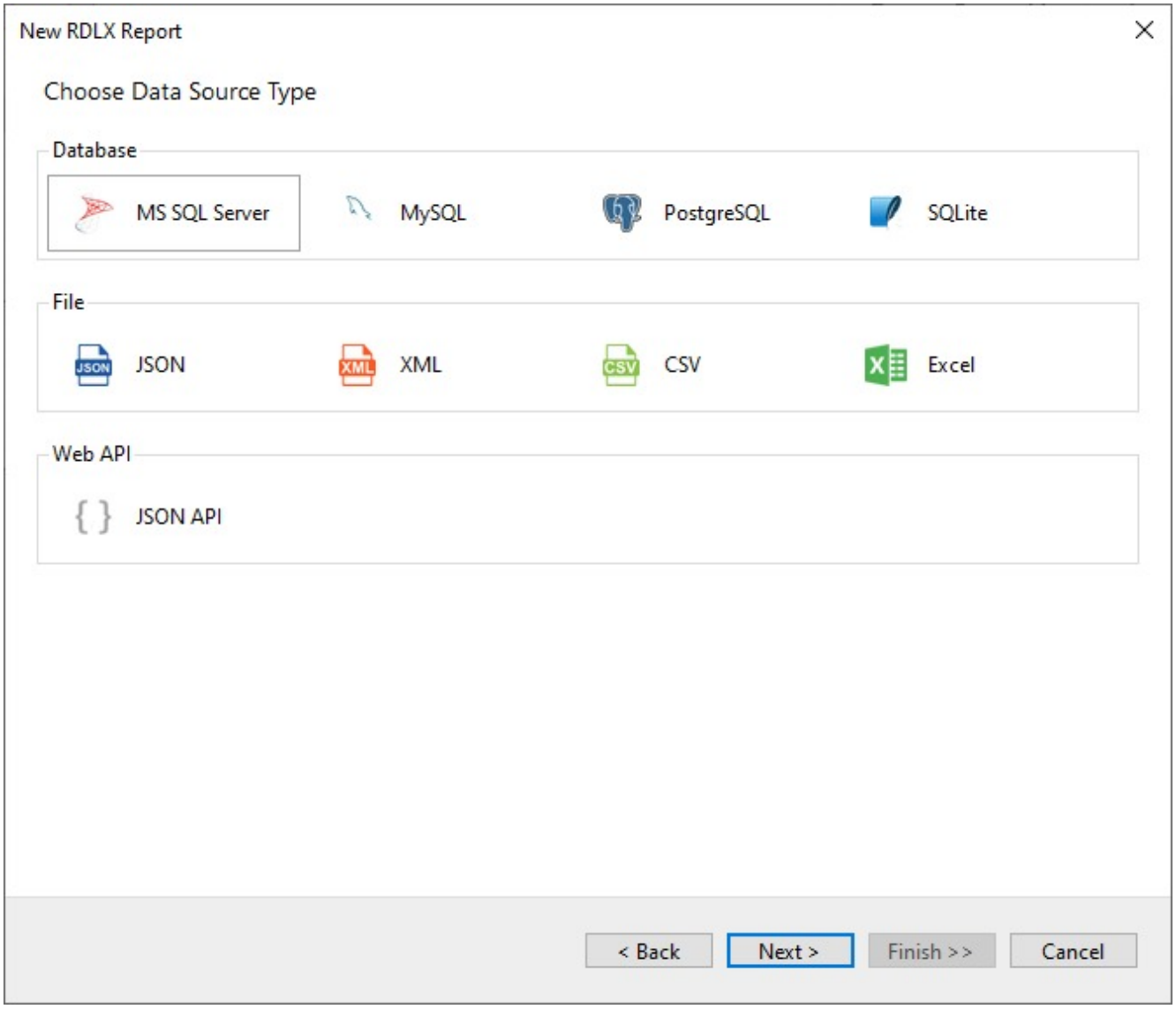
Connect to MS SQL Server Data Source using Report Wizard

The steps to connect to the MS SQL Server data source are:

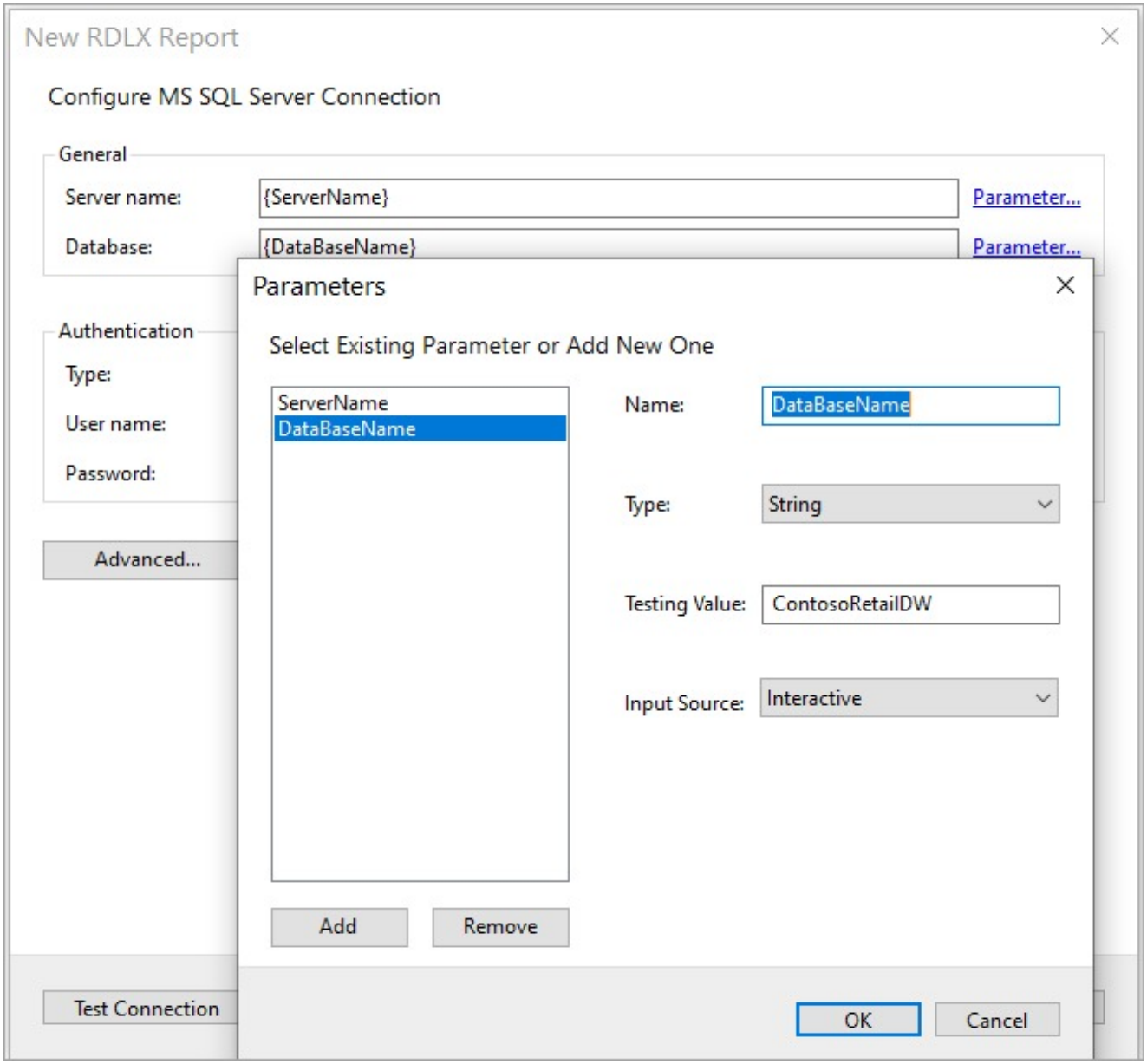
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



3. Select the Data Source Type as **MS SQL Server** and click **Next**.



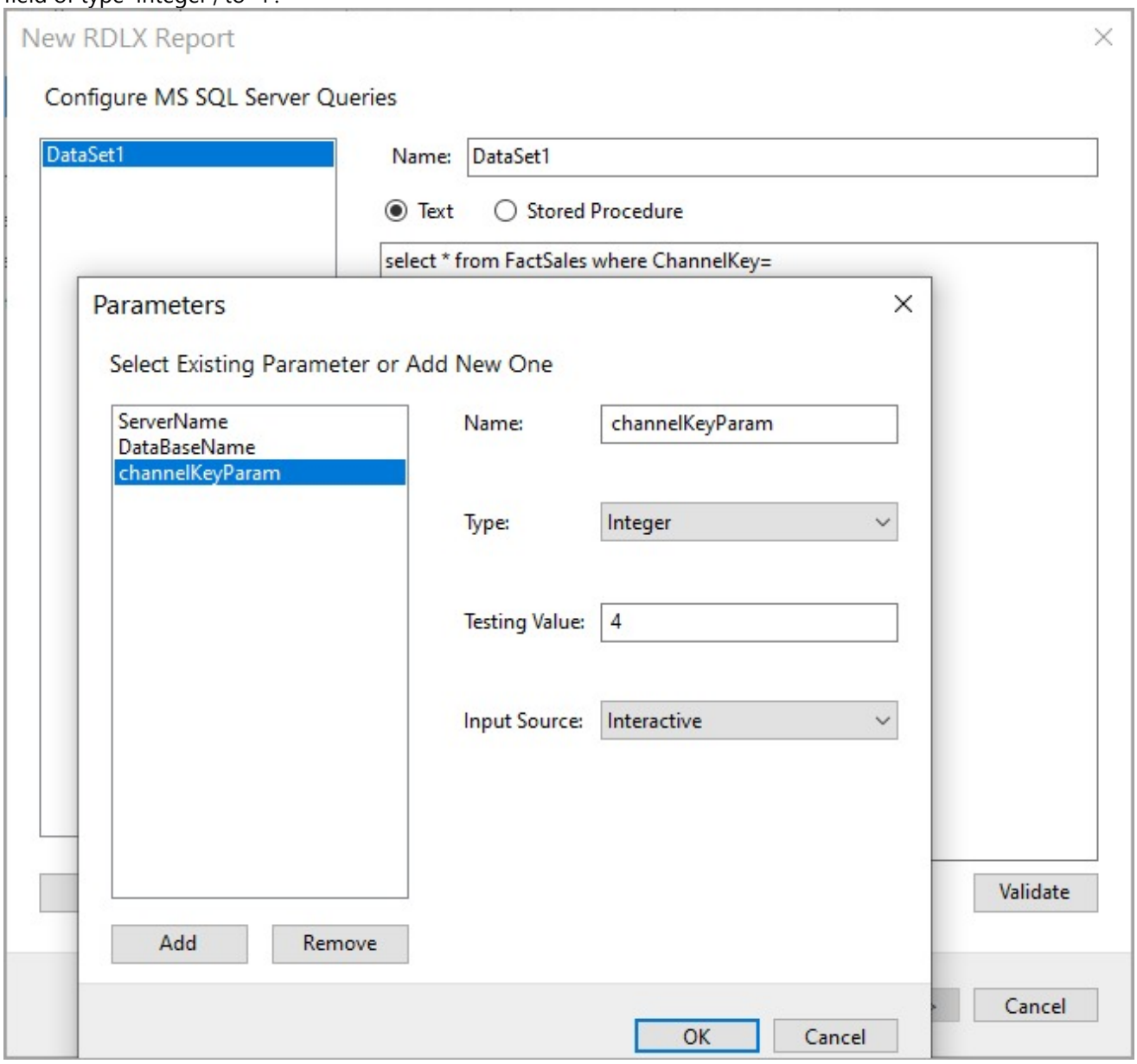
4. Enter the **Server name** and **Database** you want to connect.



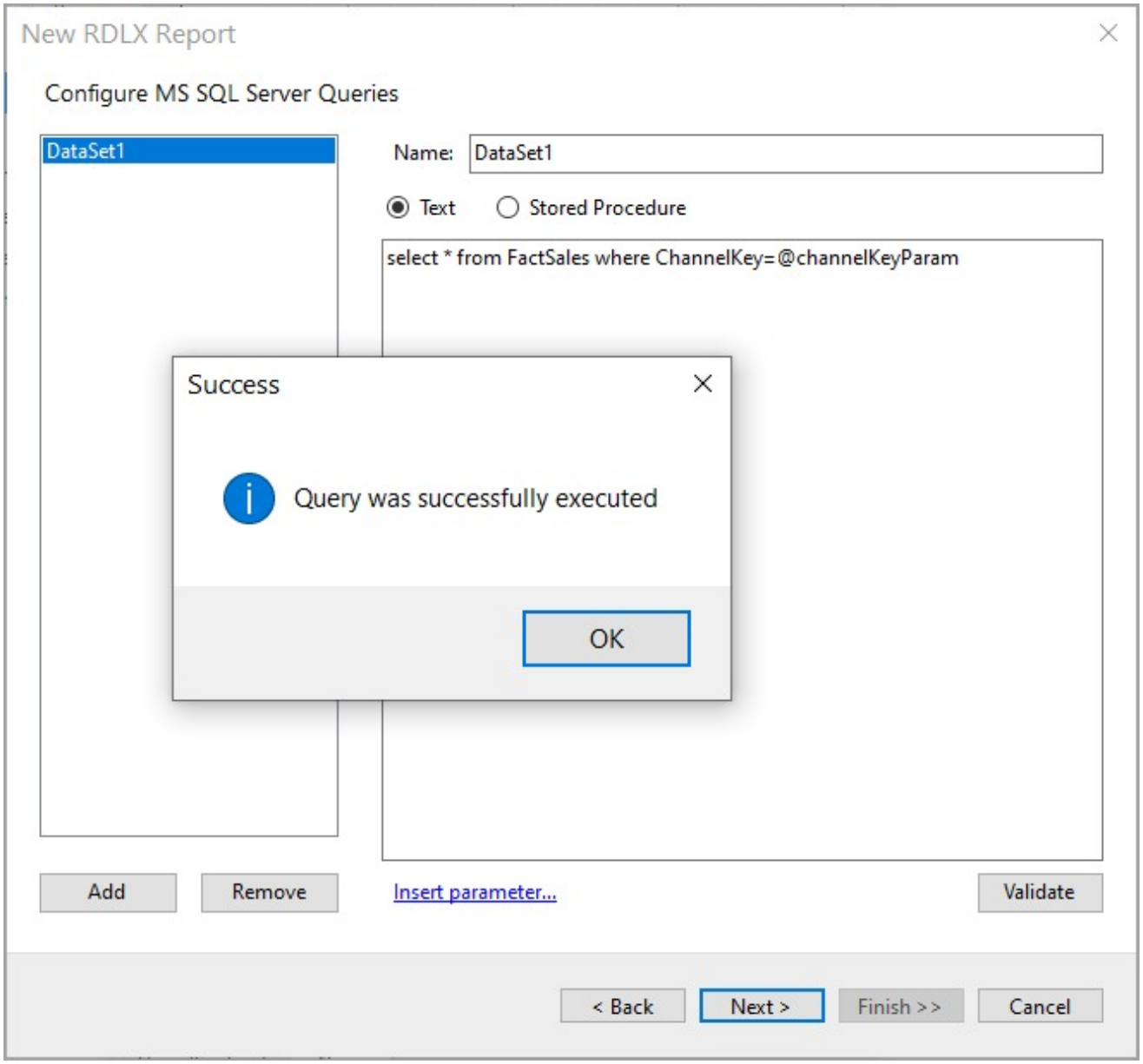
5. To specify the runtime connection values, let us add two parameters: for the Server name and for the Database, as follows:
    1. Click on the **Parameter** link to open the **Parameters** dialog, and then click the **Add** button to add a new parameter.
    2. Specify the below details:
        - **Name**: Specify the name of the parameter.
        - **Type**: Select the value type (string by default) from the drop-down list.
        - **Testing Value**: Specify the runtime value for the connection properties.
        - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.
    3. Click **OK** to finish adding the parameter.
6. To specify the authentication method for the data source connection, choose the authentication Type as **SQL Server** or **Windows**.
   If you choose the **SQL Server** authentication option, enter the **User name** and **Password** in the respective fields.

> **Note**: The **User name** and **Password** fields are disabled in case of **Windows** authentication.

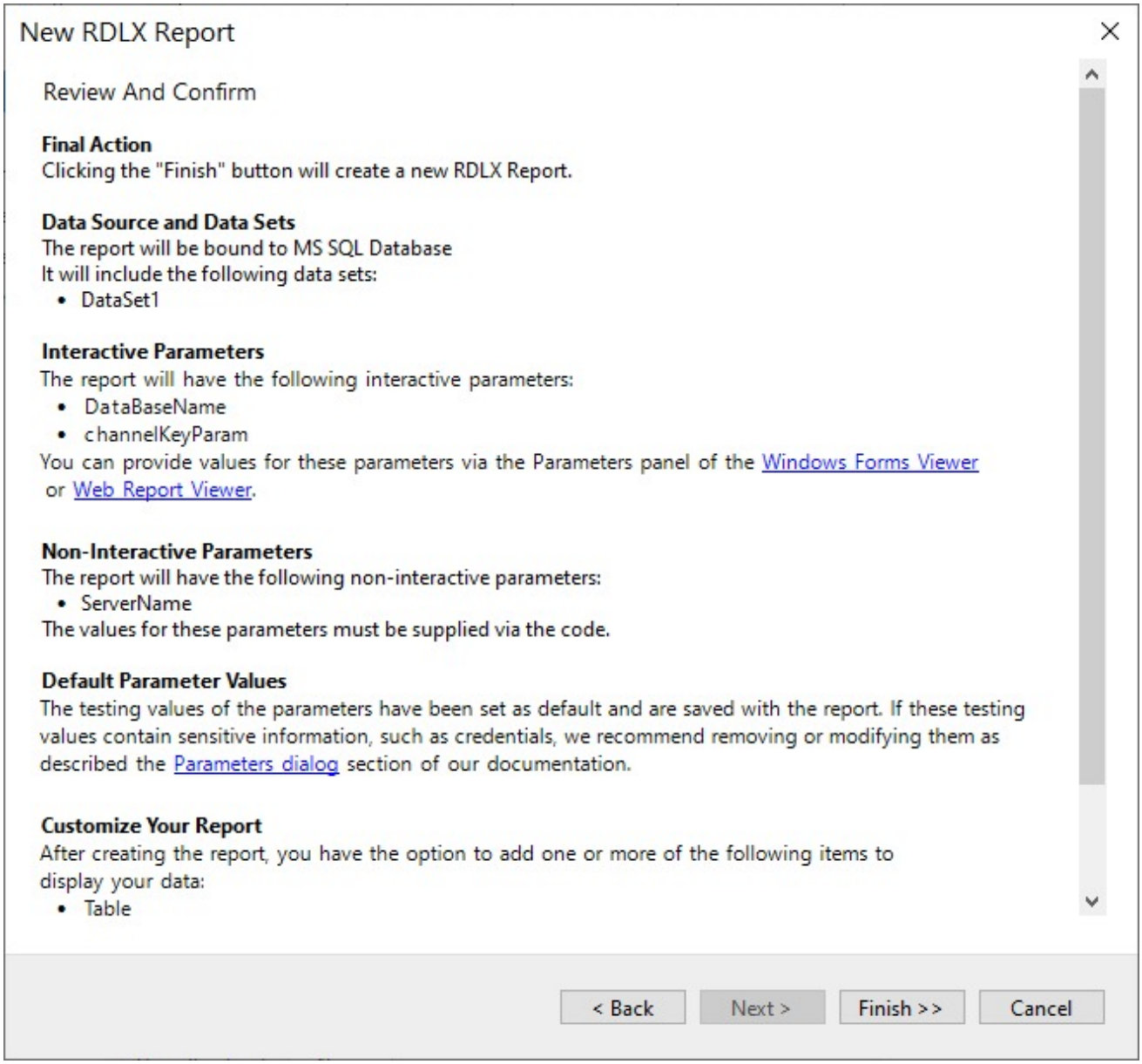
7. You can select **Advanced** properties for additional configuration, for example, setting UI timeout option to allow entering key-value pairs.
8. Click **Test Connection** to test the connection.
9. Click **Next** and configure the dataset to retrieve the fields.
10. Click **Insert parameter** to specify the interactive parameter 'channelKeyParam', to set the value of 'ChannelKey' field of type 'Integer', to '4'.



11. You should add a valid query that utilizes the parameter added. Validate the query and click **Next**.

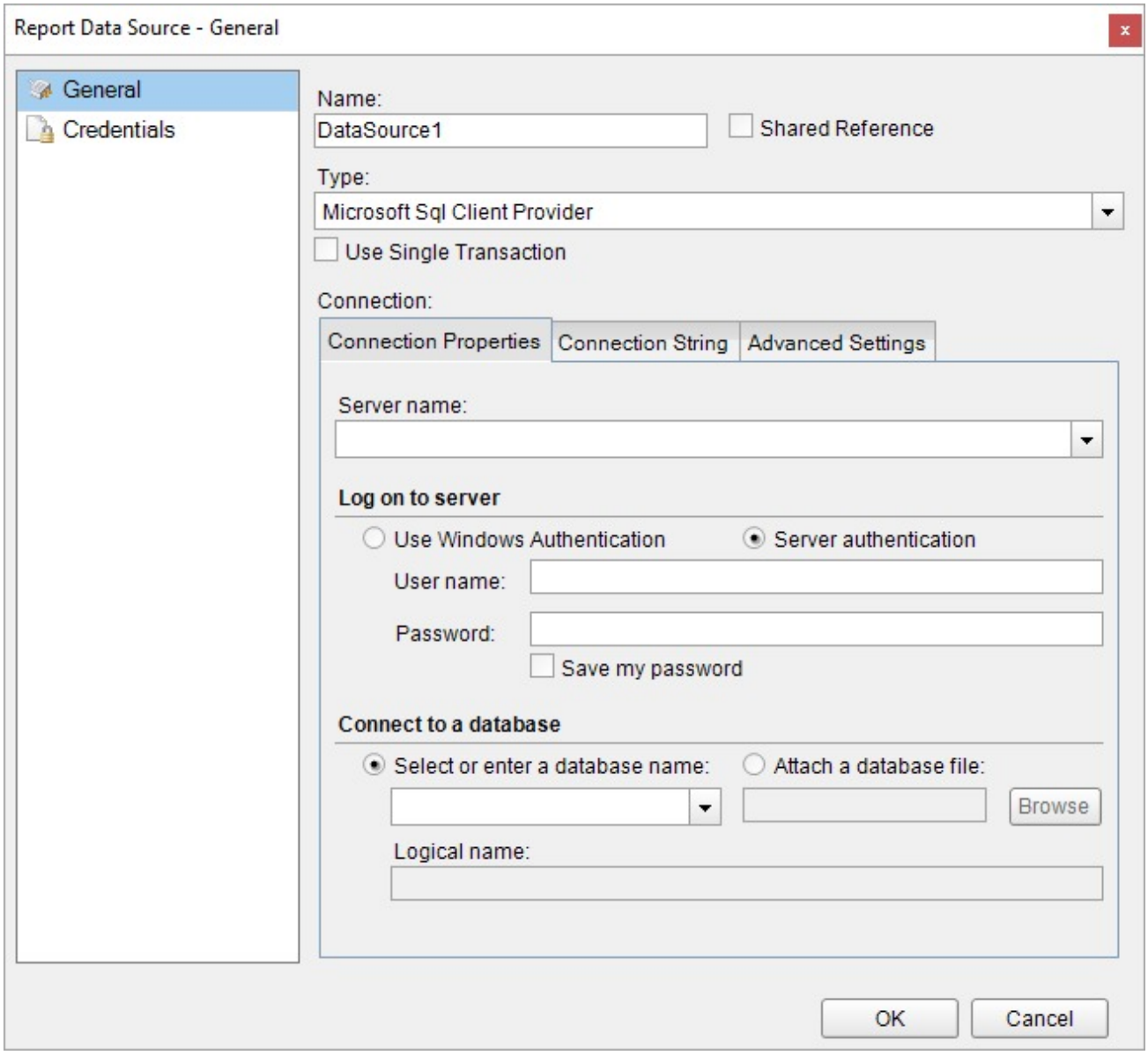


12. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully create the report with the MS SQL Server data source.



Connect to a Microsoft SQL Client Data Source using Report Data Source dialog

1. In the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the data source name in the **Name** field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select **Microsoft Sql Client Provider**.



4. In the **Connection Properties** tab, enter the server name that you want to connect.
5. To specify the authentication method for the data source connection, select the **Use Windows Authentication** or **Server authentication** option.
   If you choose the **Server authentication** option, enter the user name and password in the respective fields.

   > **Note**: The **User name** and **Password** fields are disabled in case of Windows authentication.

6. To connect to a Database, choose the **Select or enter a database name** or **Attach a database file** option. If you choose the **Select or enter a database name** option, enter a database name or select it from the drop-down list. In case the **Attach a database file** option is selected, click on **Browse** and navigate to the desired database file on your system. You can also give a **Logical Name** to the selected database.

   > **Note**: The **Logical Name** field is disabled in case of **Select or enter a database name** option.

   The **Connection String** tab displays the generated connection string as shown below.

```
data source=20.186.17.78;initial catalog=northwind;user id=qatester;password=*********;
```

For more information, see the **Configuration Settings for Microsoft SQL Client Data Source** section.

7. Verify the generated connection string by clicking the **Validate DataSource** icon  .
8. Click the **OK** button to save the changes.

Configuration Settings

The Microsoft SQL Client Data Provider provides the following configuration settings under the Connection section in the **Report Data Source** dialog. Based on the defined configuration settings, the connection string is generated in the **Connection String** tab.

The **Connection Properties** tab describes the configuration settings for the Microsoft SQL Client data provider. It includes details related to server name, type of authentication method, and database name.

Setting	Description
Server name	Enter a server name
Log on to server	Select whether to use Windows authentication or server authentication which requires a user name and password. Below this field you can also check the **Save my password** option for future reference.
Connect to a database	Select whether to enter a database name or attach a database file.

The **Advanced Settings** tab gives access to the following properties.

Setting	Description
Application Name	Indicates the client application name.
Current Language	Indicates the SQL Server language name. It also identifies the language used for system message selection and formatting. The language must be installed on the SQL Server, otherwise opening the connection will fail.
Network Address	Indicates the network address of the SQL Server, specified by the Location property.
Network Library	Indicates the name of the network library (DLL) used to communicate with the SQL Server. The name should not contain the path or the .dll file name extension. The default name is provided by the SQL Server client configuration.
Packet Size	Indicates a network packet size in bytes. The Packet Size property value must be between 512 and 32767. By default, the SQLOLEDB network packet size is 4096.
Trusted Connection	Indicates the user authentication mode. You can set this property to Yes or No. By default, the property value is set to No. If Yes, the SQLOLEDB uses the Microsoft Windows NT Authentication Mode to authorize user access to the SQL Server database, specified by the Location and Datasource property values. If this property is set to No, then the SQLOLEDB uses the Mixed mode to authorize user access to the SQL Server database. The SQL Server login and password are specified in the User Id and Password properties.
Workstation ID	Denotes a string that identifies the workstation.

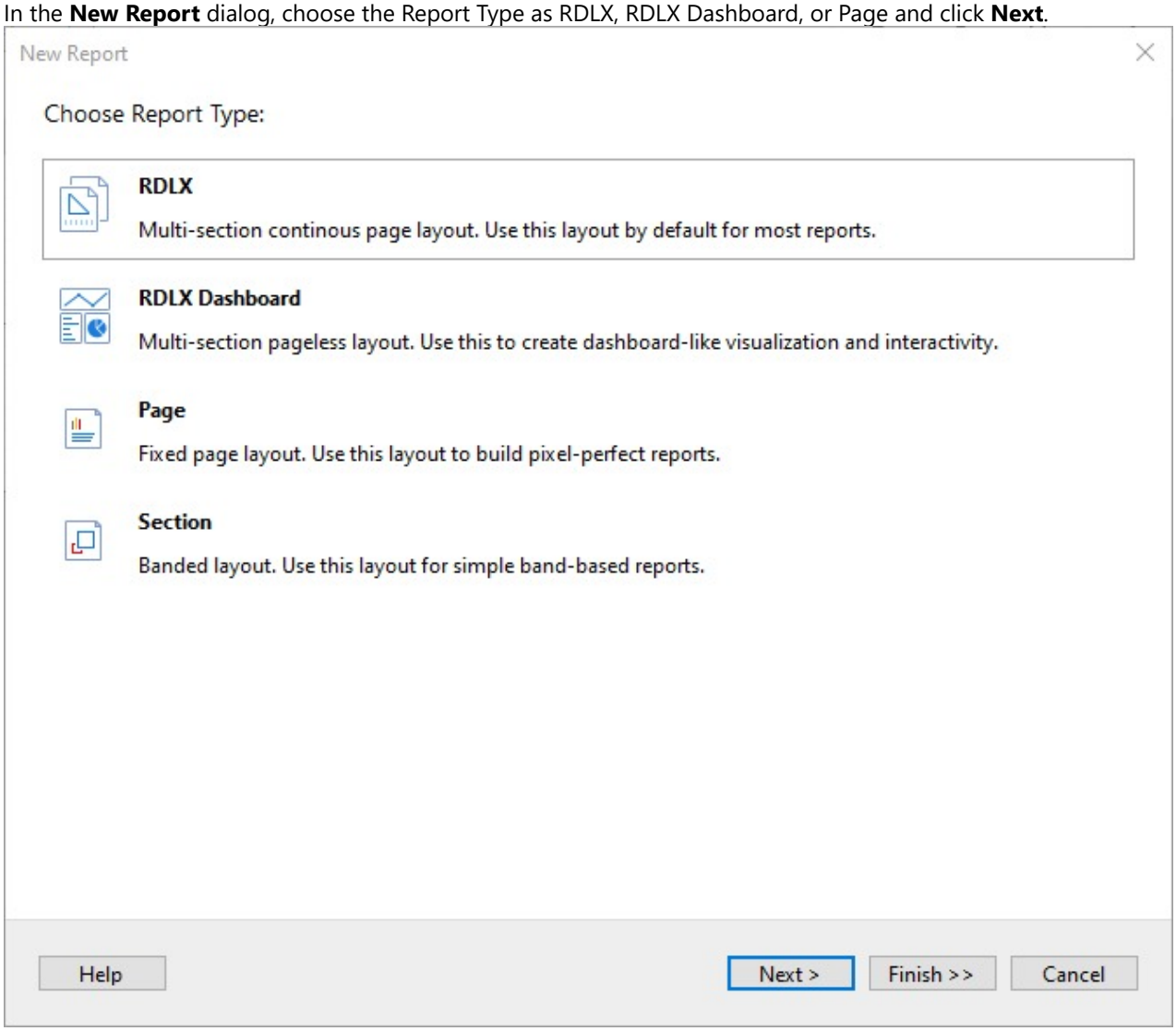
MySQL

This article explains connecting a Page or an RDLX report to the MySQL data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

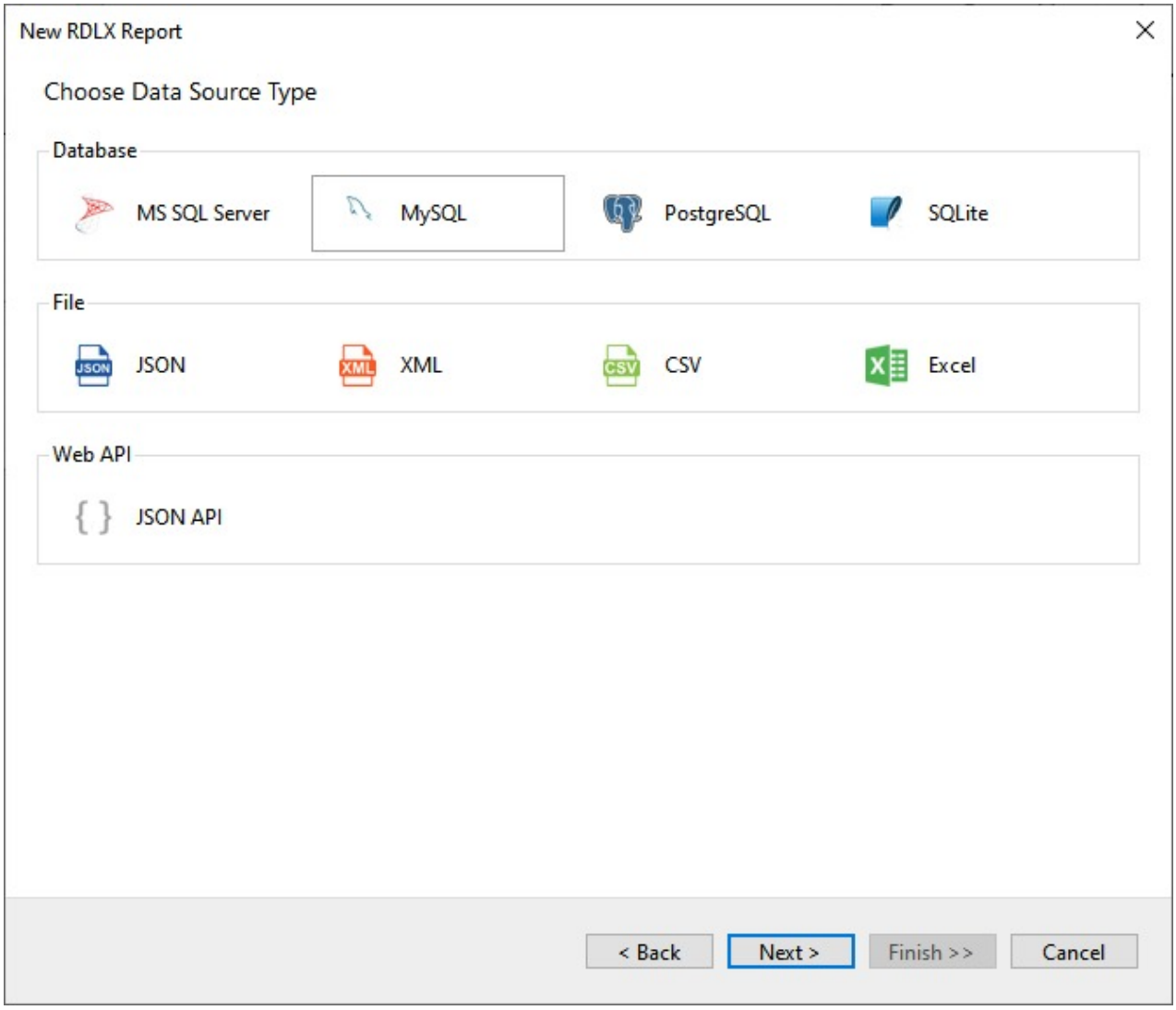
Connect to MySQL Data Source using Report Wizard

The steps to connect to the MySQL data source are:

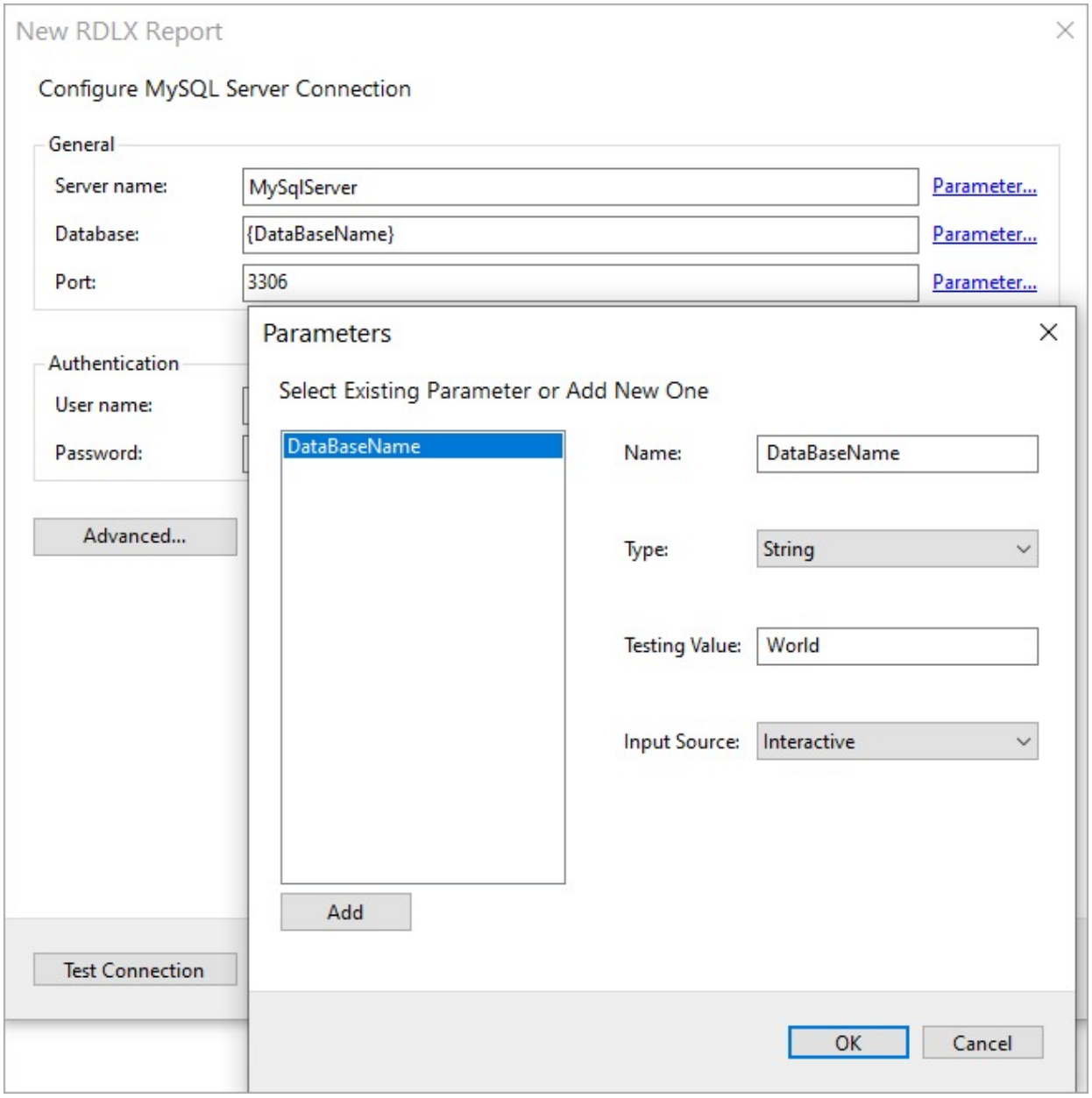
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



3. Select the Data Source Type as **MySQL** and click **Next**.

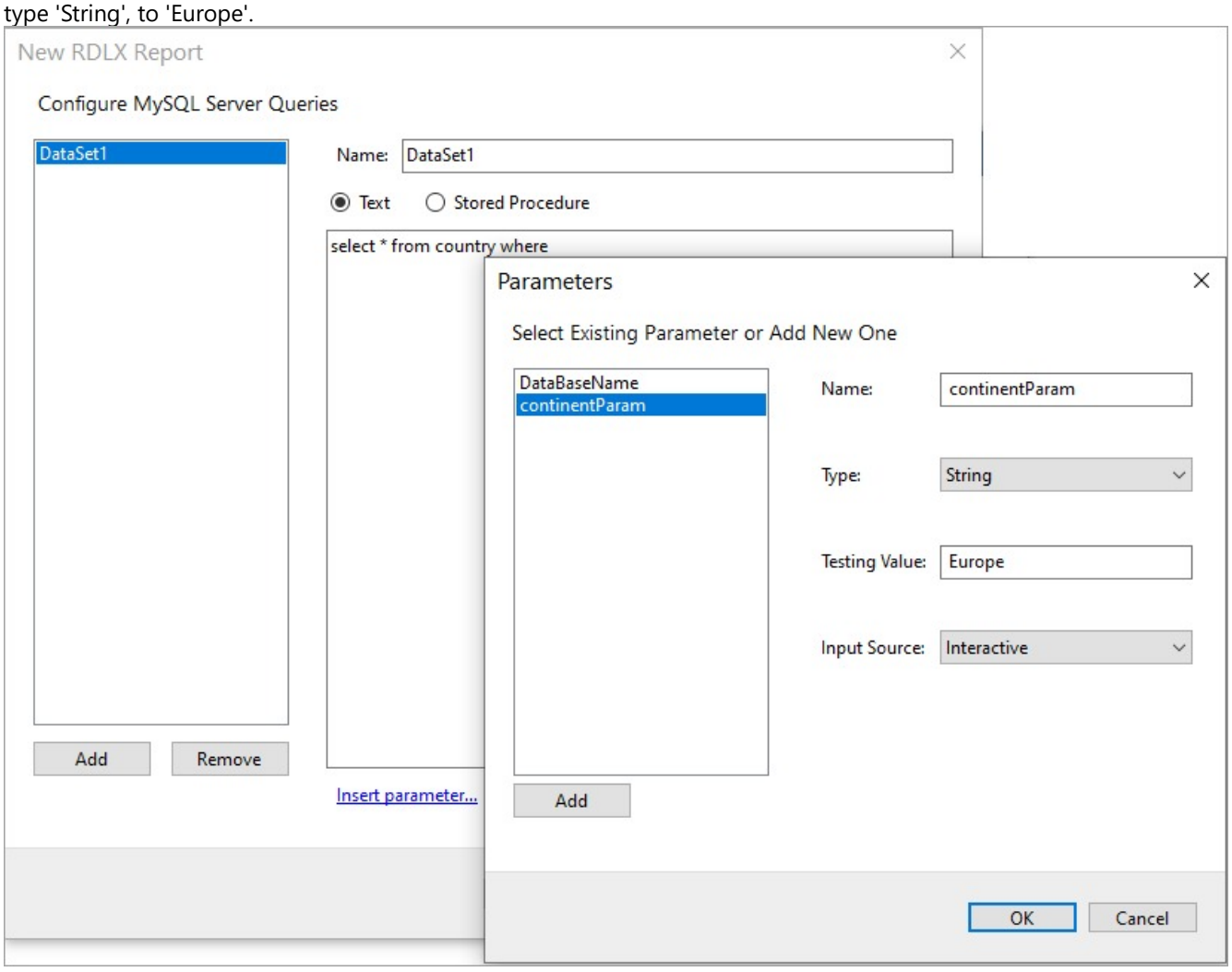


4. Enter the server connection details, including **Server name, Database, Port, User name**, and **Password**.

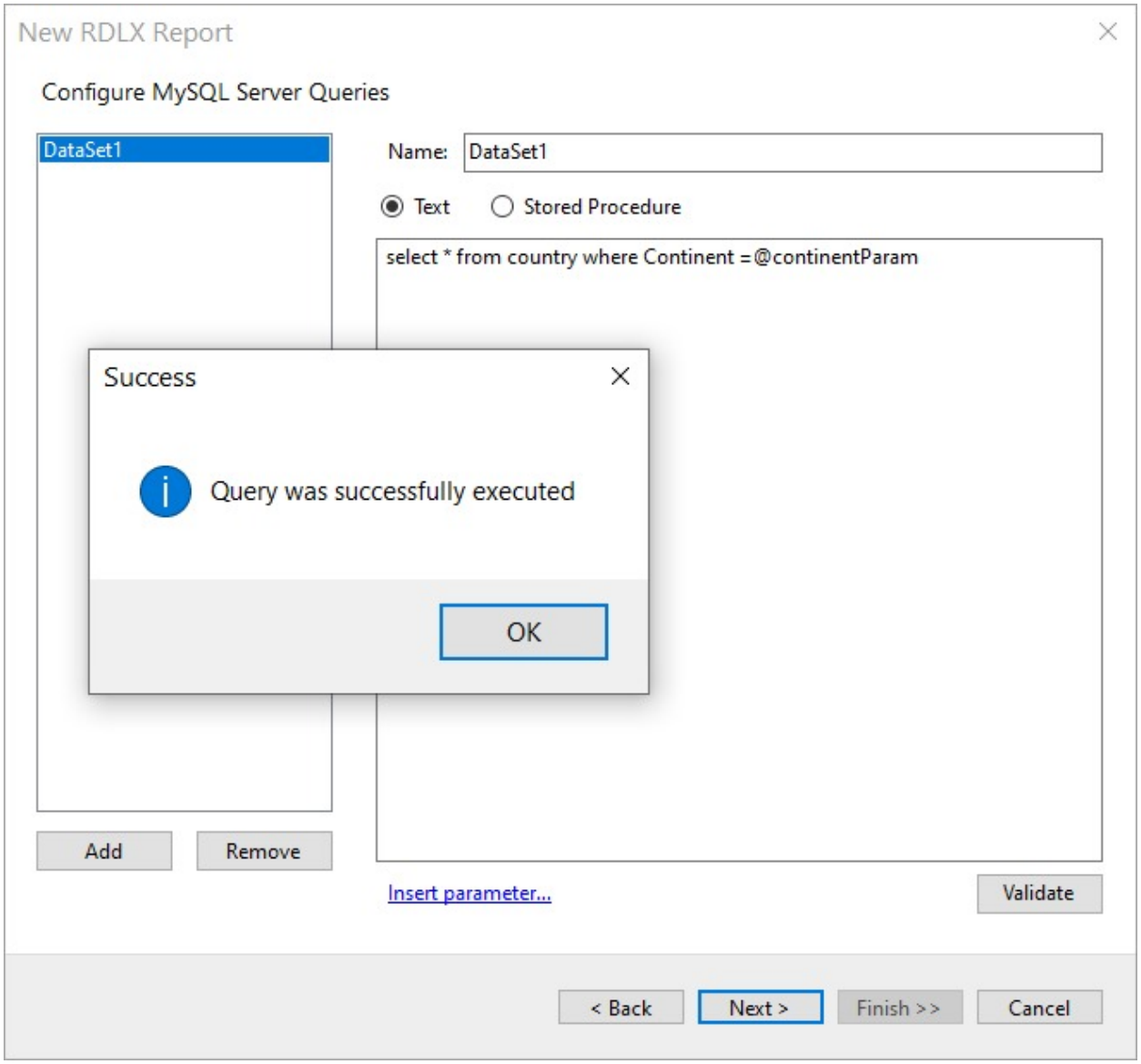


5. To specify the runtime connection values, let us add a parameter for the Database, as follows:
    1. Click on the **Parameter** link to open the **Parameters** dialog, and then click the **Add** button to add a new parameter.
    2. Specify the below details:
        - **Name**: Specify the name of the parameter.
        - **Type**: Select the value type (string by default) from the drop-down list.
        - **Testing Value**: Specify the runtime value for the connection properties.
        - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.
    3. Click **OK** to finish adding the parameter.
6. For additional configuration, for example, setting UI timeout option to allow entering key-value pairs, select **Advanced** properties.
7. Click **Test Connection** to test the connection.
8. Click the **Next** option and configure the dataset to retrieve the fields.
9. Click **Insert parameter** to specify the interactive parameter 'continentParam', to set the value of 'Continent' field of

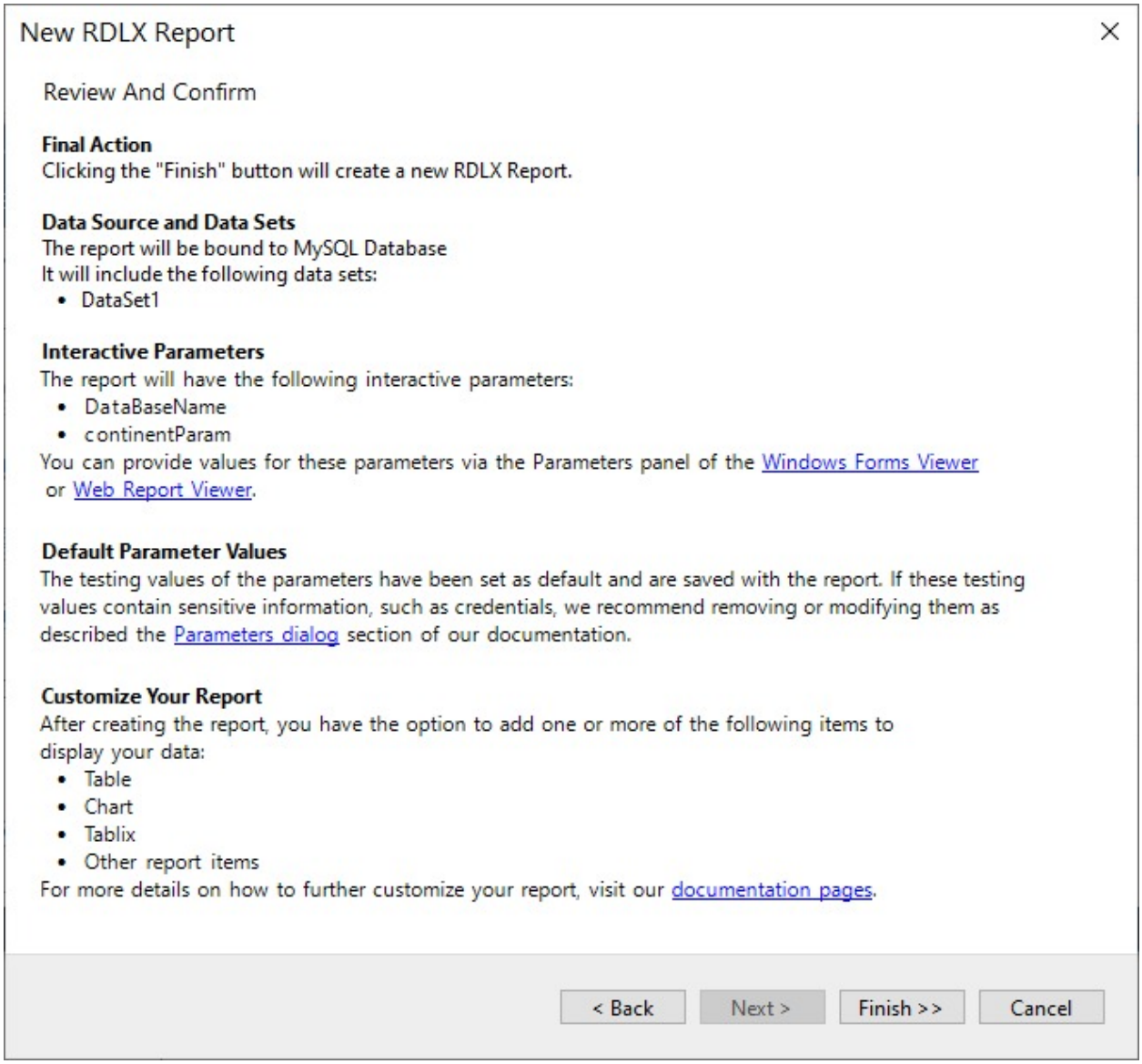
type 'String', to 'Europe'.



10. You should add a valid query that utilizes the parameter added. Validate the query and click **Next**.

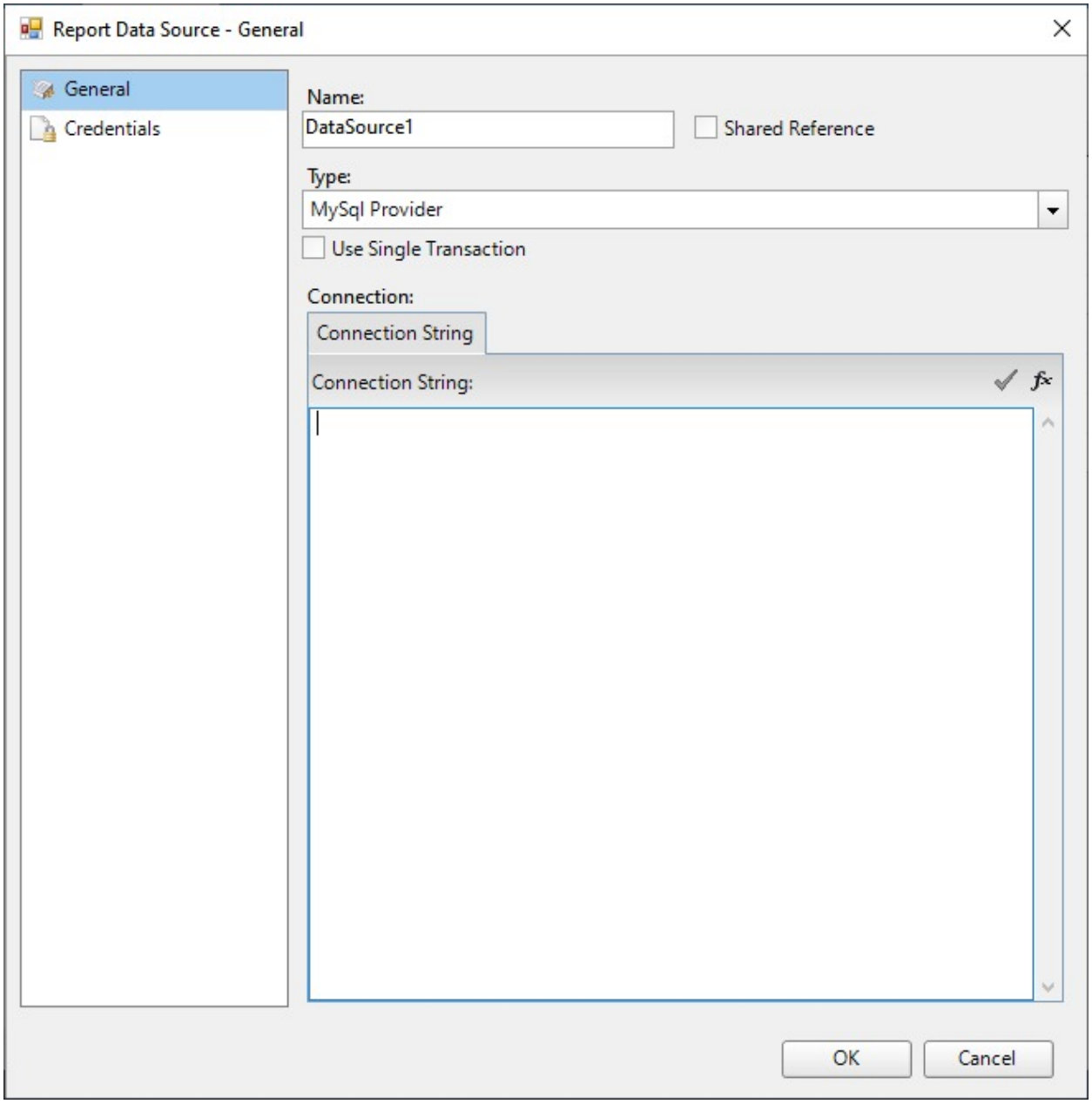


11. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the MySQL data source.



Connect to a MySQL Data Source using Report Data Source dialog

1. In the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the data source name in the **Name** field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select **MySql Provider**.



4. On the same page under the **Connection** section, enter the connection string to connect to the **MySql** data source. For example:

   ```
   host=10.64.1.240;port=3306;database=mysql;username=root;password=******;
   ```

5. Verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and close the **Report Data Source** dialog box.

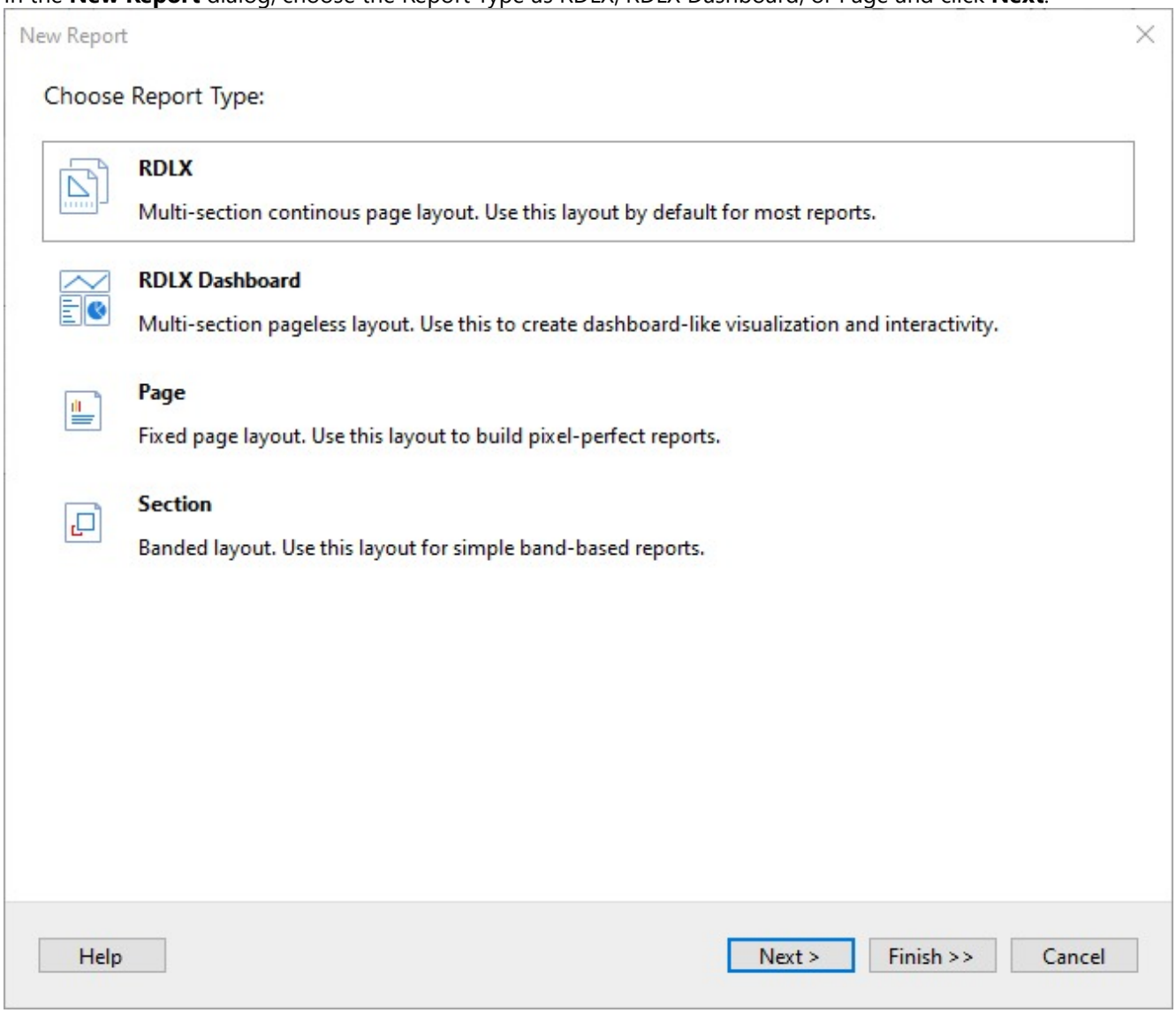
PostgreSQL

This article explains connecting a Page or an RDLX report to the PostgreSQL data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

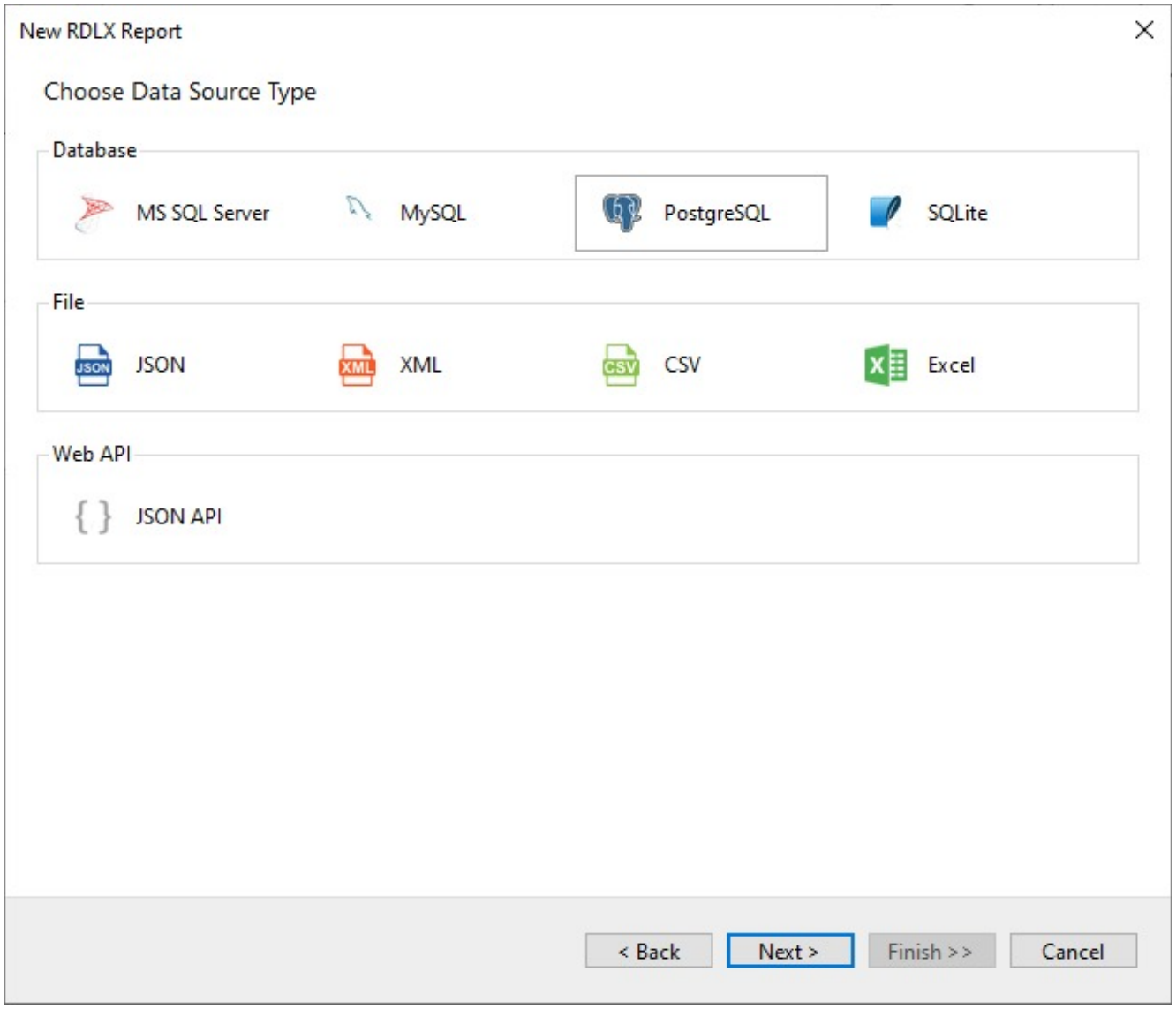
Connect to PostgreSQL Data Source using Report Wizard

The steps to connect to the PostgreSQL data source are:

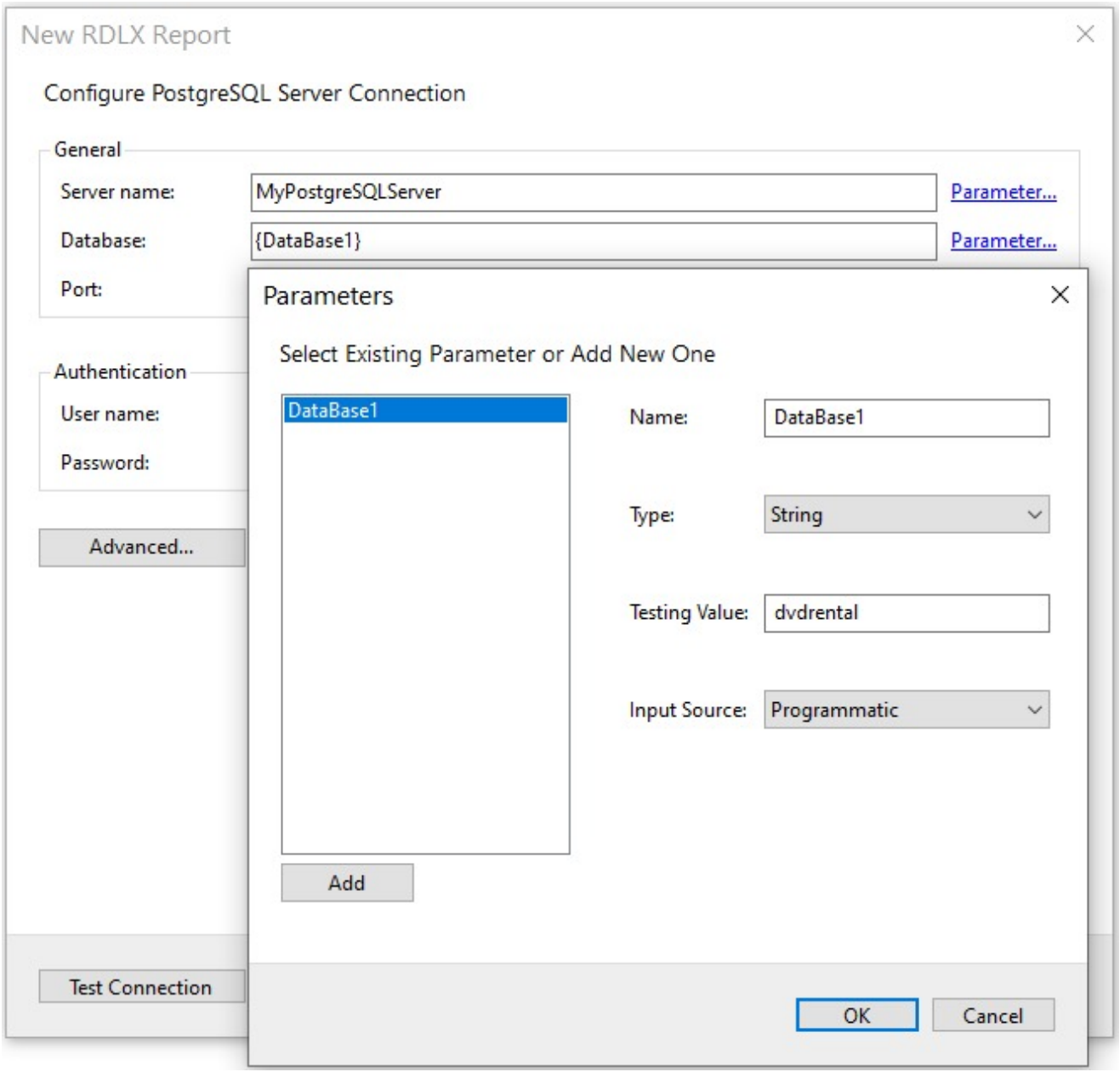
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



3. Select the Data Source Type as **PostgreSQL** and click **Next**.

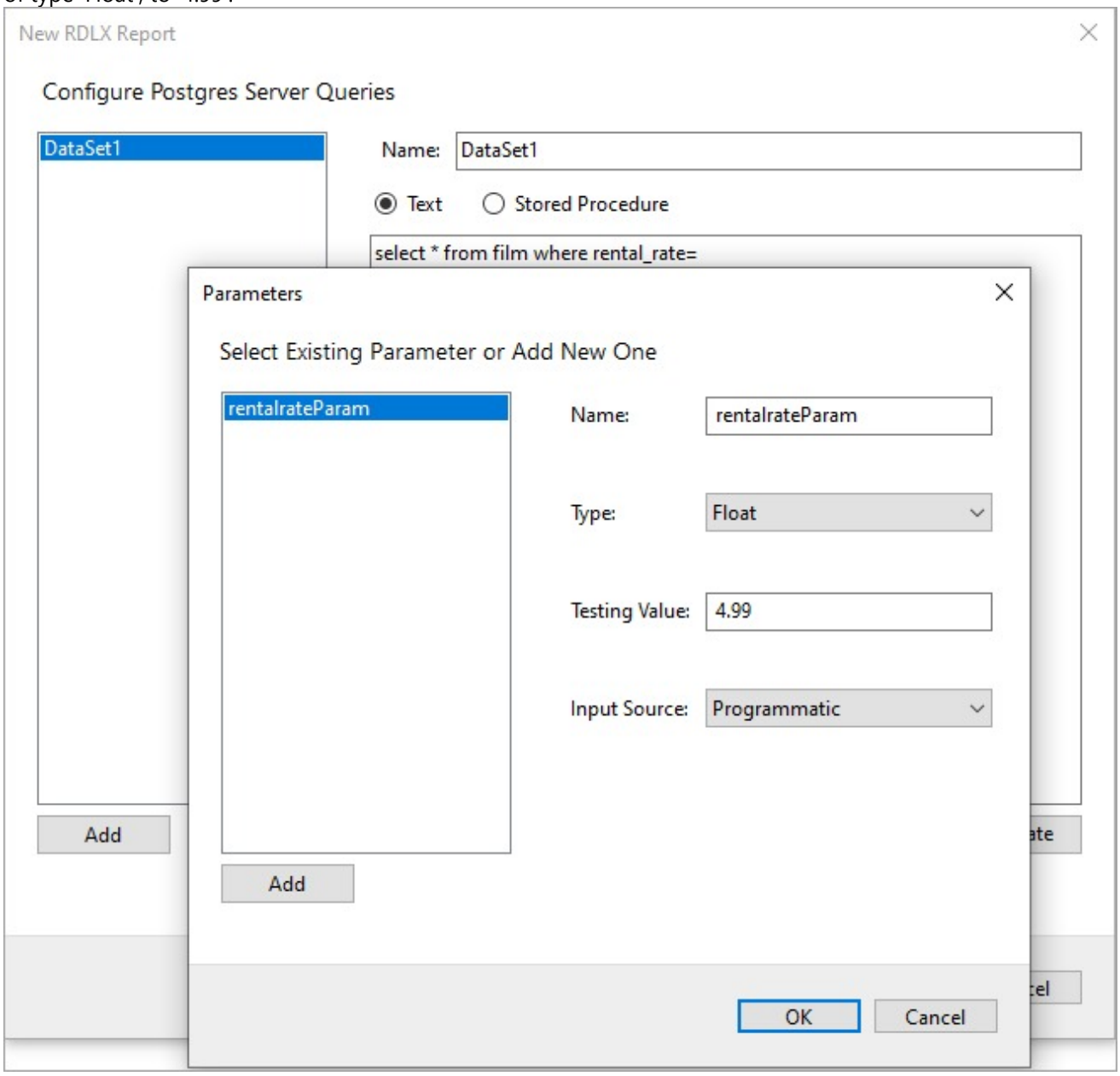


4. Enter the server connection details, including **Server name, Database, Port, User name**, and **Password**.

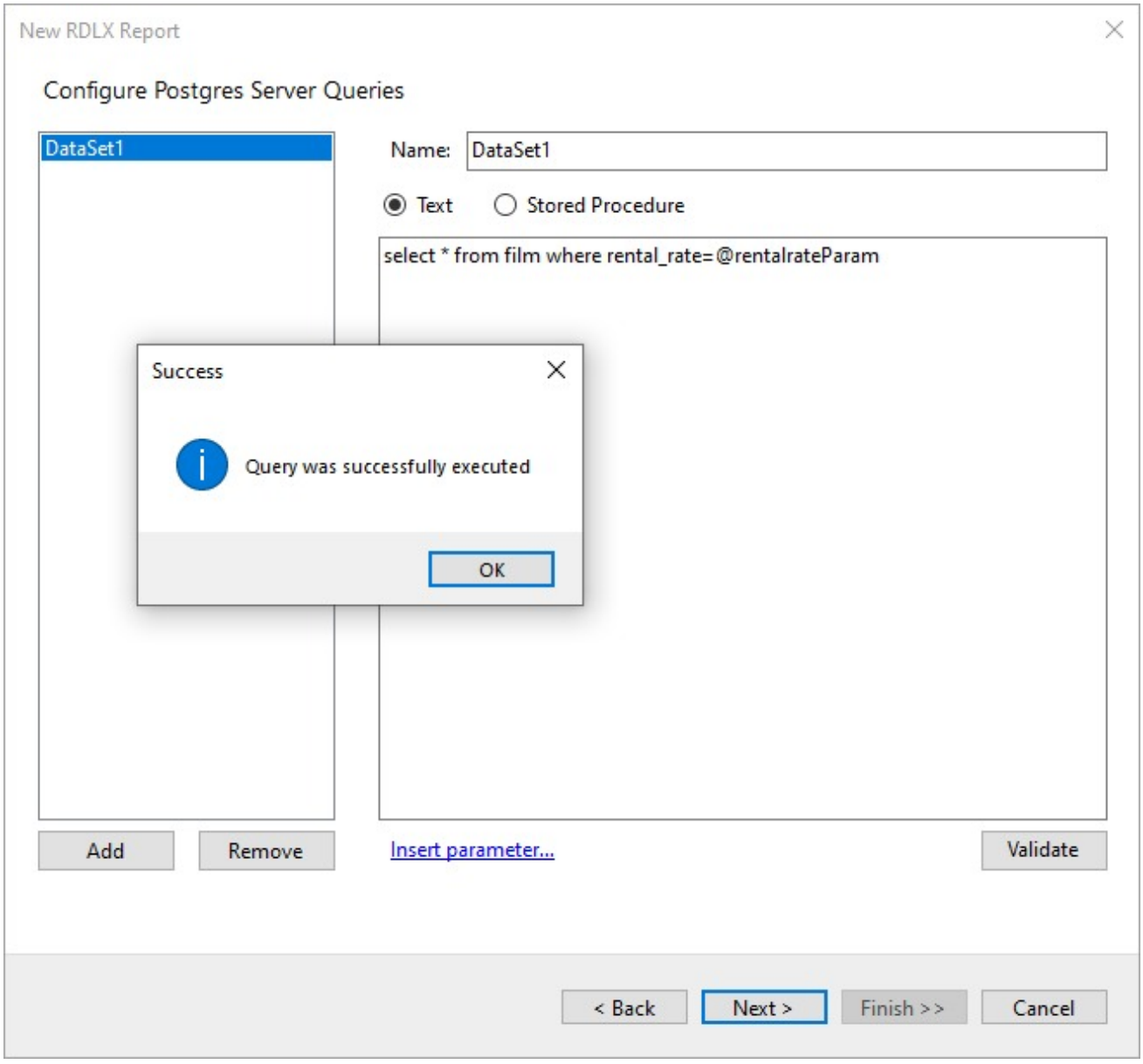


5. To specify the runtime connection values, let us add the parameter for Database.
    1. Click **Parameter** to open the **Parameters** dialog. and then click the **Add** button to add a new parameter.
    2. Specify the below details:
        - **Name**: Specify the name of the parameter.
        - **Type**: Select the value type (string by default) from the drop-down list.
        - **Testing Value**: Specify the runtime value for the connection properties.
        - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.
    3. Click OK to finish adding the parameter.
6. For additional configuration, for example, setting UI timeout option to allow entering key-value pairs, select **Advanced** properties.
7. Click **Test Connection** to test the connection.
8. Click **Next** and configure the dataset to retrieve the fields.
9. Click **Insert parameter** to specify the interactive parameter 'rentalrateParam', to set the value of 'rental_rate' field

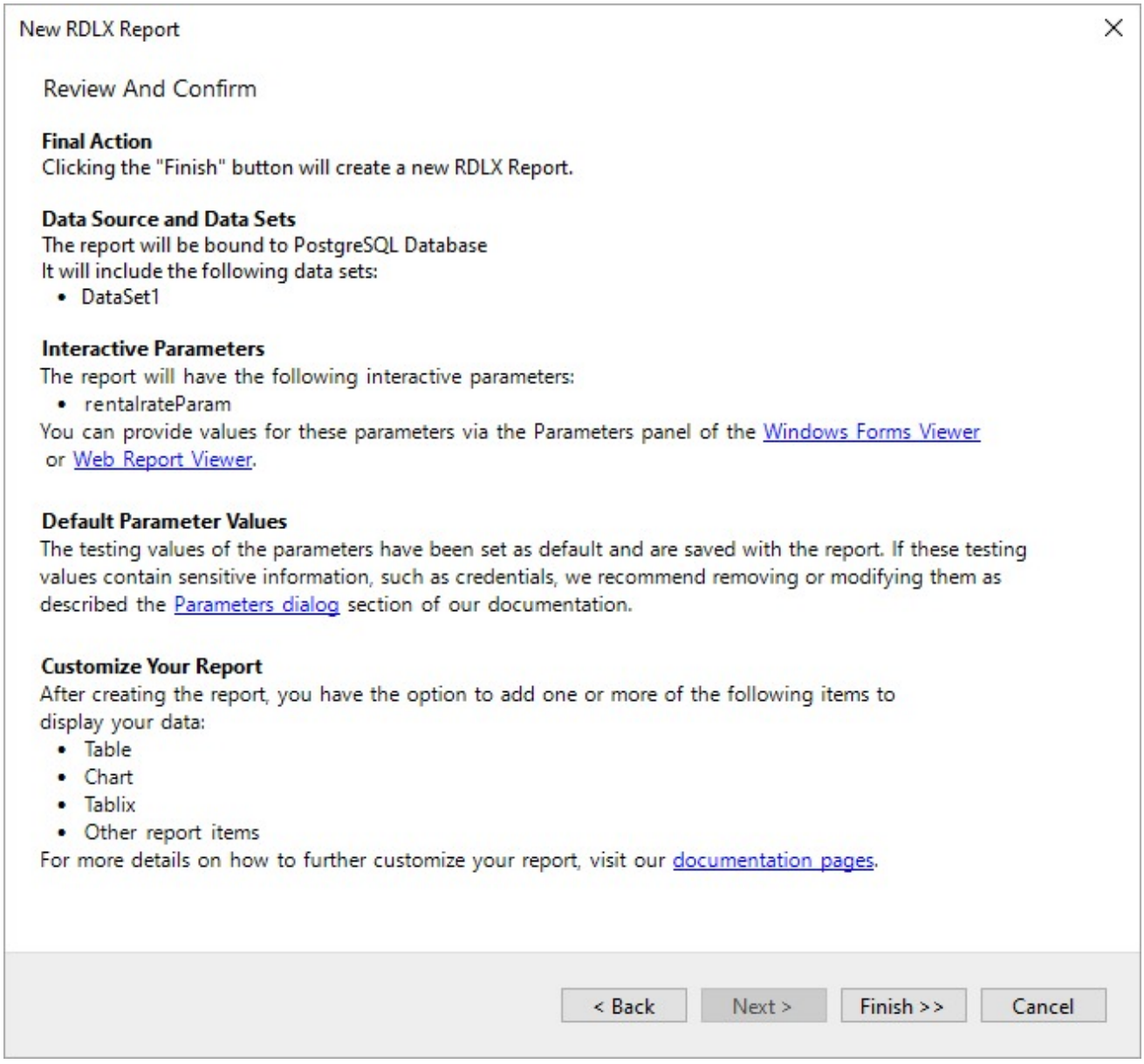
of type 'Float', to '4.99'.



10. You should add a valid query that utilizes the parameter added. Validate the query and click **Next**.

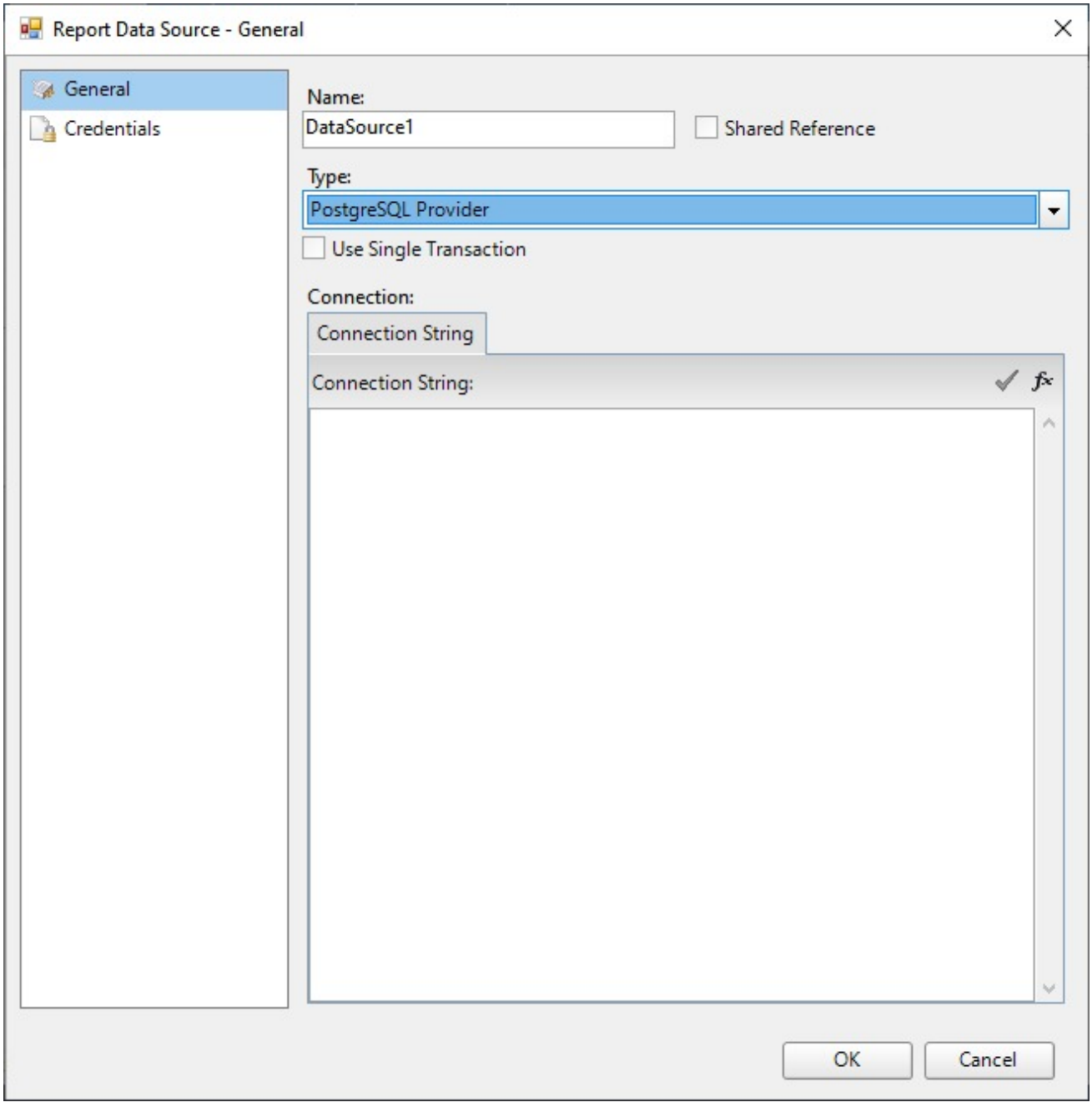


11. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the PostgreSQL data source.



Connect to a PostgreSQL Data Source using Report Data Source dialog

1. In the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the data source name in the **Name** field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select **PostgreSQL Provider**.



4. On the same page under the **Connection** section, enter the connection string to connect to the **PostgreSQL** data source. For example:

   ```
   host=10.64.1.240;port=5432;database=postgres;username=postgres;password=********;
   ```

5. Verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and close the Report Data Source dialog box.

SQLite

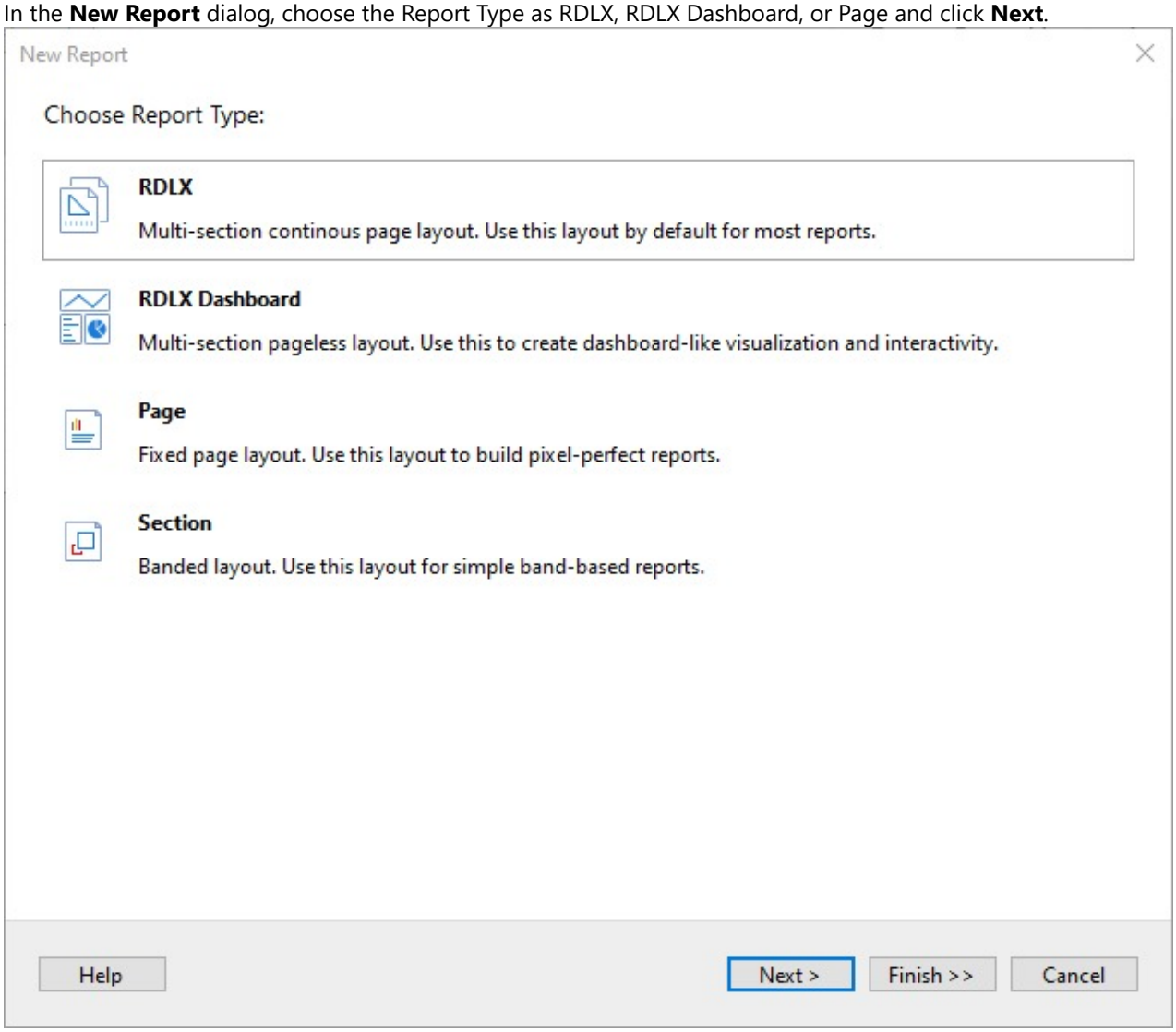
This article explains connecting a Page or an RDLX report to a SQLite data source. You can connect to this data source while

creating a new report (via report wizard) or using report explorer (via report data source dialog). SQLite data source is a custom data provider, so the data provider needs to be configured. See section **Custom Data Provider** in Configure ActiveReports topic before you begin connecting to the SQLite Provider.

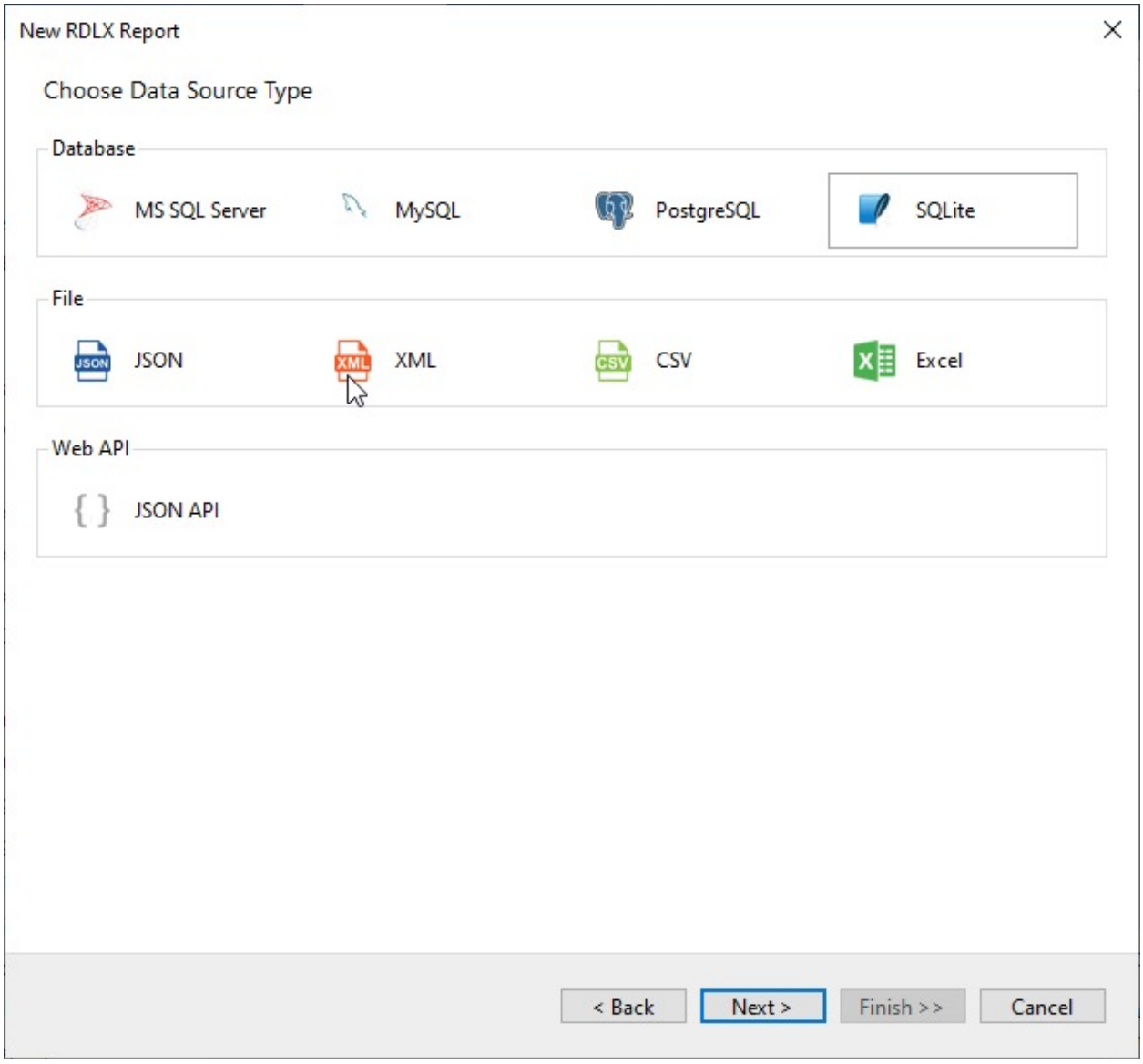
Connect to SQLite Data Source using Report Wizard

The steps to connect to the SQLite data source are:

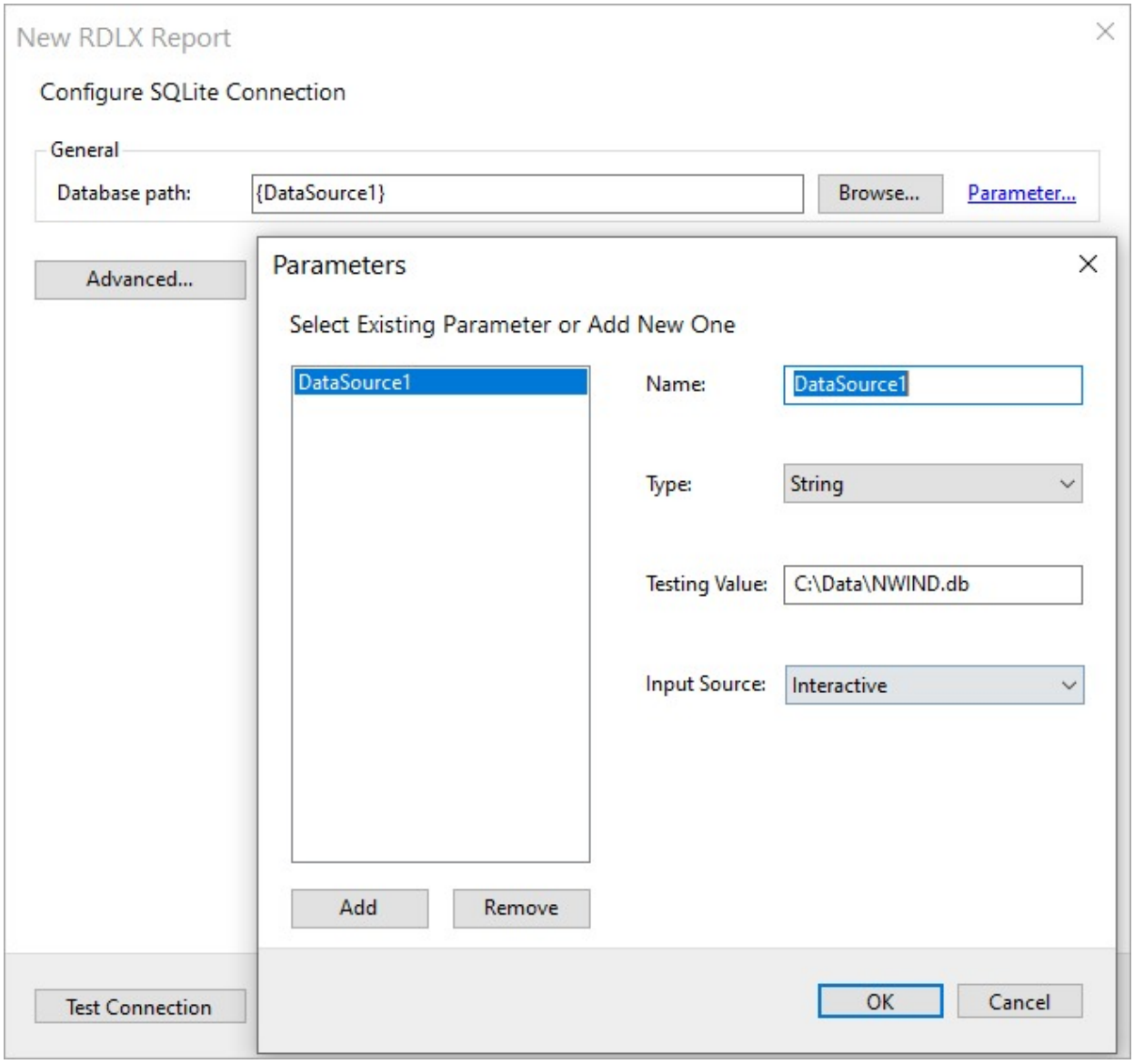
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



3. Select the Data Source Type as **SQLite** and click **Next**.

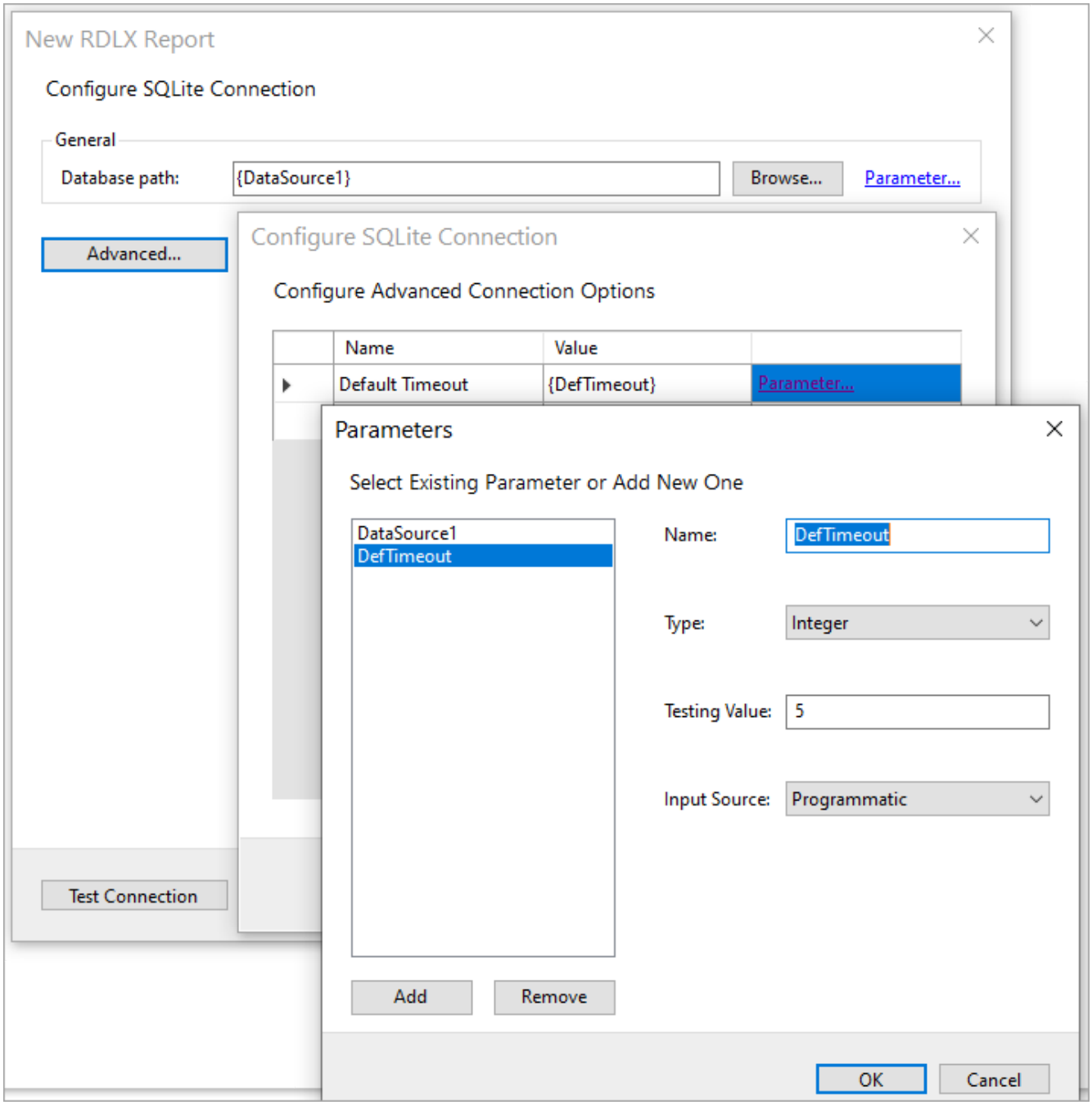


4. To specify the **Database Path**, click **Browse** and navigate to the desired file on your system.

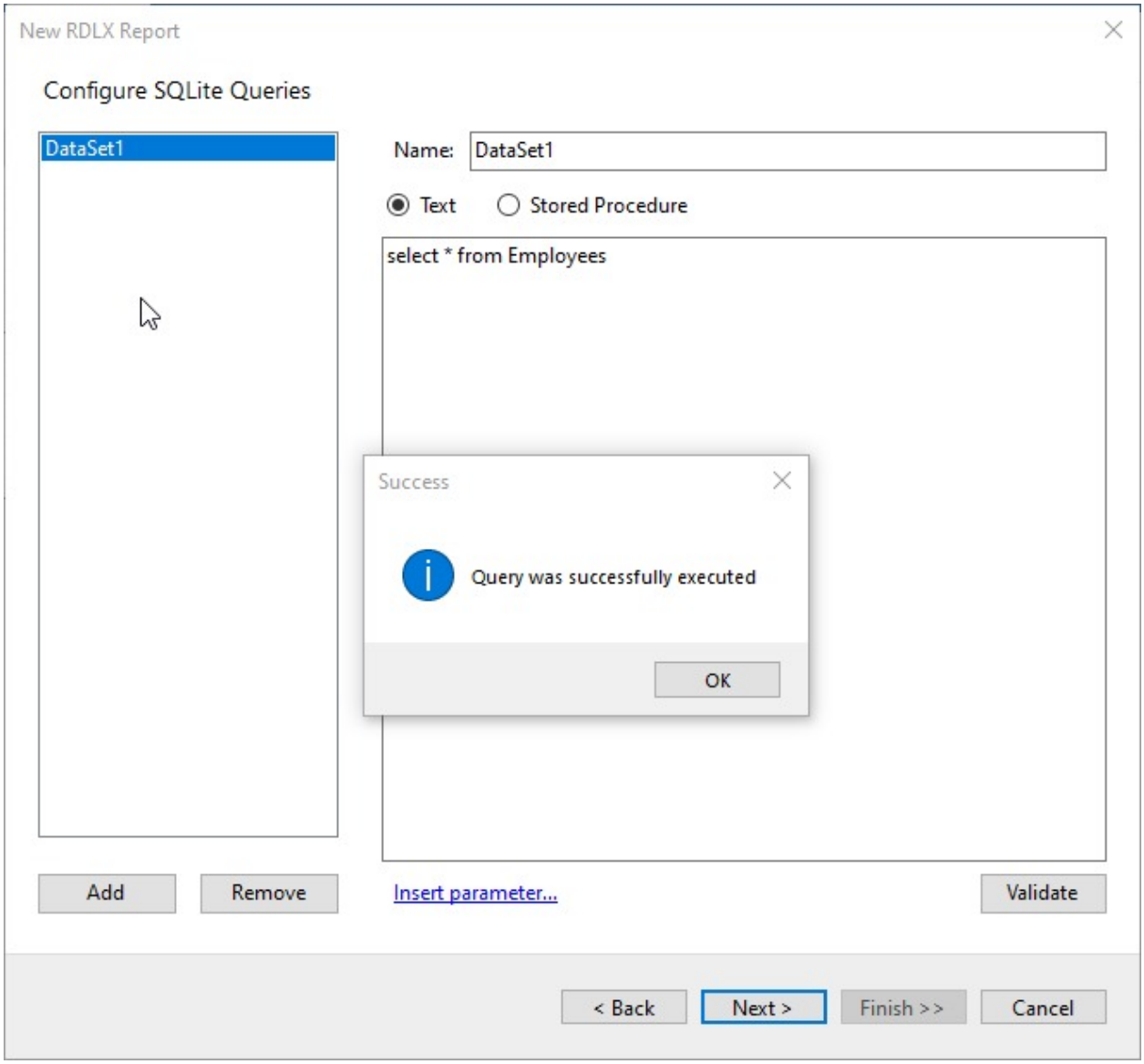


5. To specify the runtime connection values for database path, let us add a parameter as follows:
    1. Click on the **Parameter** link to open the **Parameters** dialog, and then click the **Add** button to add a new parameter.
    2. Specify the below details:
        - **Name**: Specify the name of the parameter.
        - **Type**: Select the value type (string by default) from the drop-down list.
        - **Testing Value**: Specify the runtime value for the connection properties, eg 'C:\Data\NWIND.db'.
        - **Input Source**: Select **Interactive** (in our case) for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.
    3. Click **OK** to finish adding the parameter.
6. Select **Advanced** to specify key-value pairs for additional configuration, and add a new parameter with details as in the previous step.

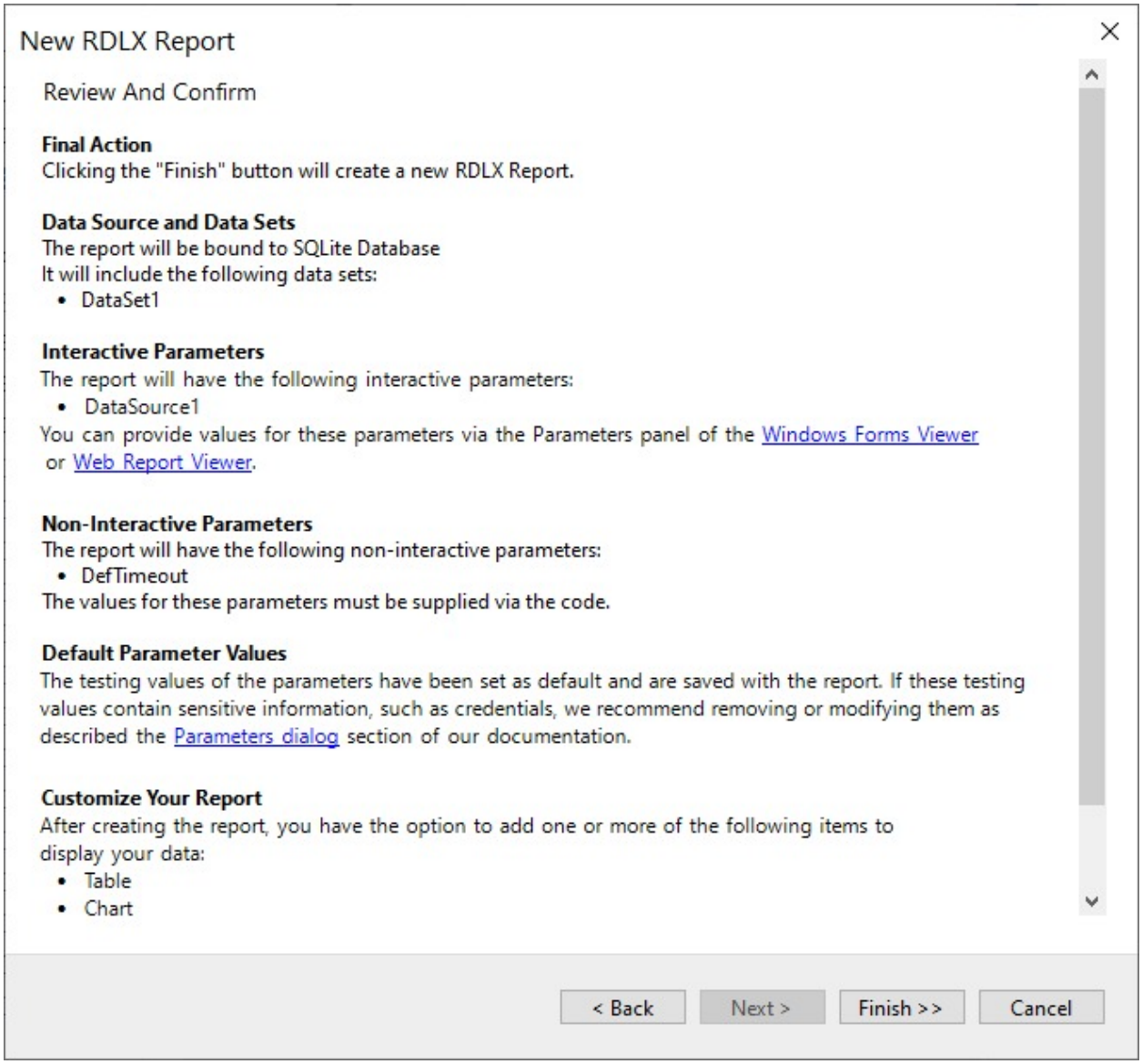
    Let us add the 'Default Timeout' option to specify the hidden parameter 'DefTimeout' of type 'Integer' to '5'.



7. Click **Test Connection** to test the connection.
8. Click **Next**, and configure the dataset to retrieve the fields by adding a valid query.
   You can also provide a parameterized query by first using the 'Insert parameter' option to create a parameter and then utilizing the parameter in the query.

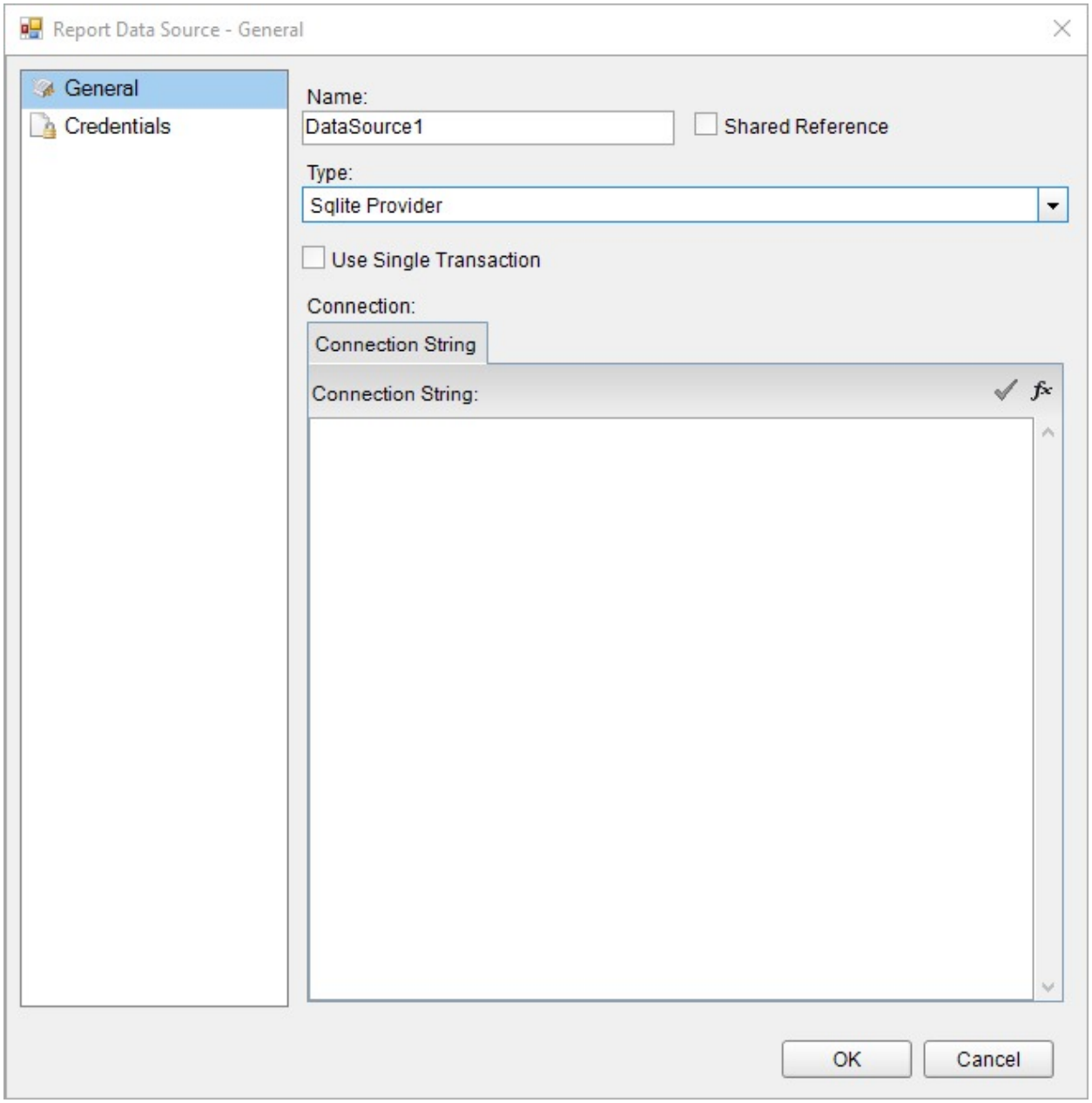


9. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.



Connect to a SQLite Data Source using Report Data Source dialog

1. In the designer, go to the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and then select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the name of the data source in the **Name** field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select SQLite Provider.



4. In the **Connection Properties** tab, enter the connection string to connect to the data source. For example, you can connect to the 'NWIND.db' sample data source that can be downloaded from GitHub and write the connection string as shown below:

```
data source=C:\NWIND.db;
```

5. Click the **Validate DataSource** icon to verify the connection string.
6. Click **OK** to close the **Report Data Source** dialog. Your report is now connected to the SQLite data source successfully.

To specify the authentication method for the data source connection, go to the **Credentials** page and select the **Use Windows Authentication** or **Use a specific user name and password** option.

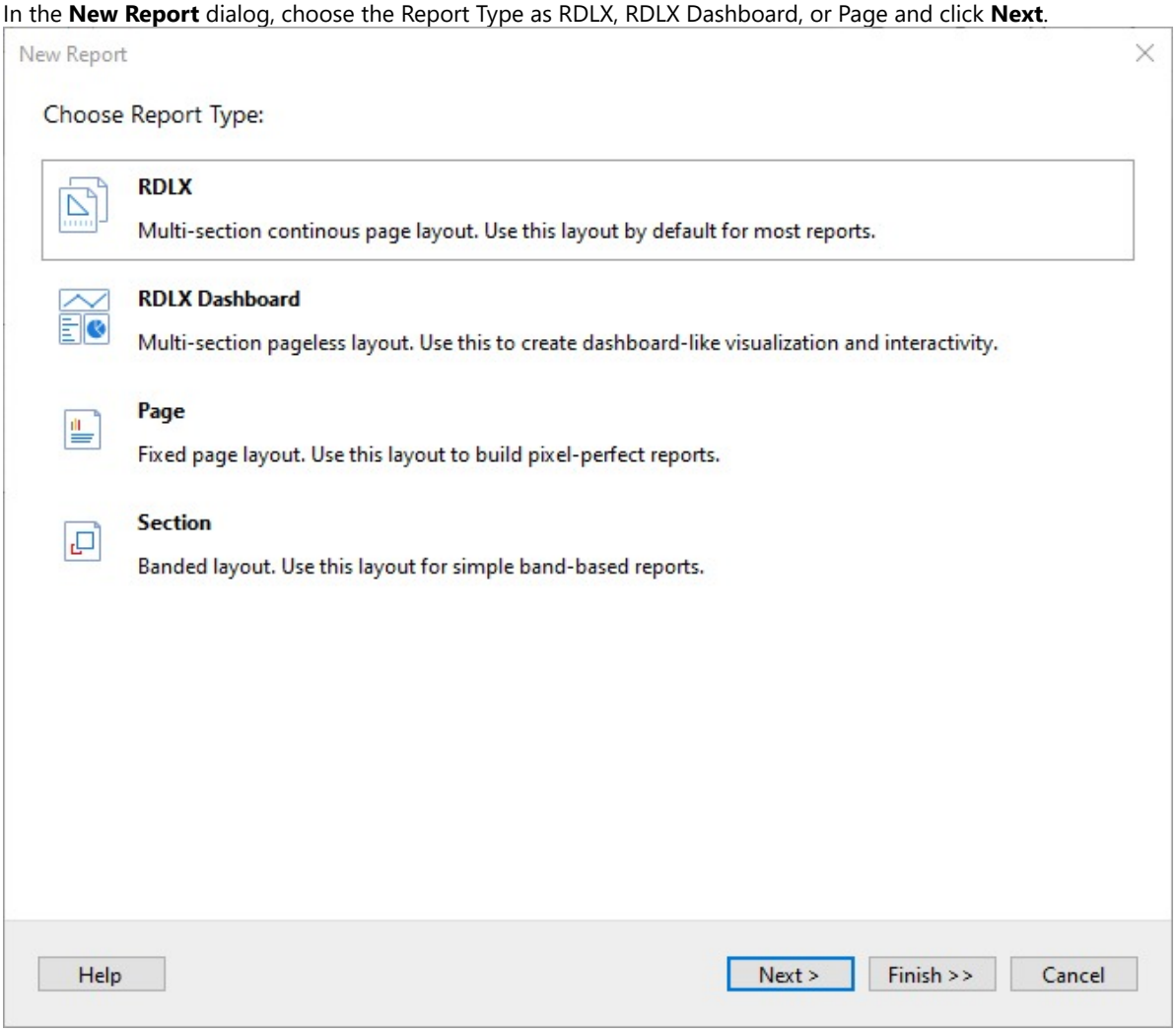
JSON

This article explains connecting a Page or an RDLX report to a JSON data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

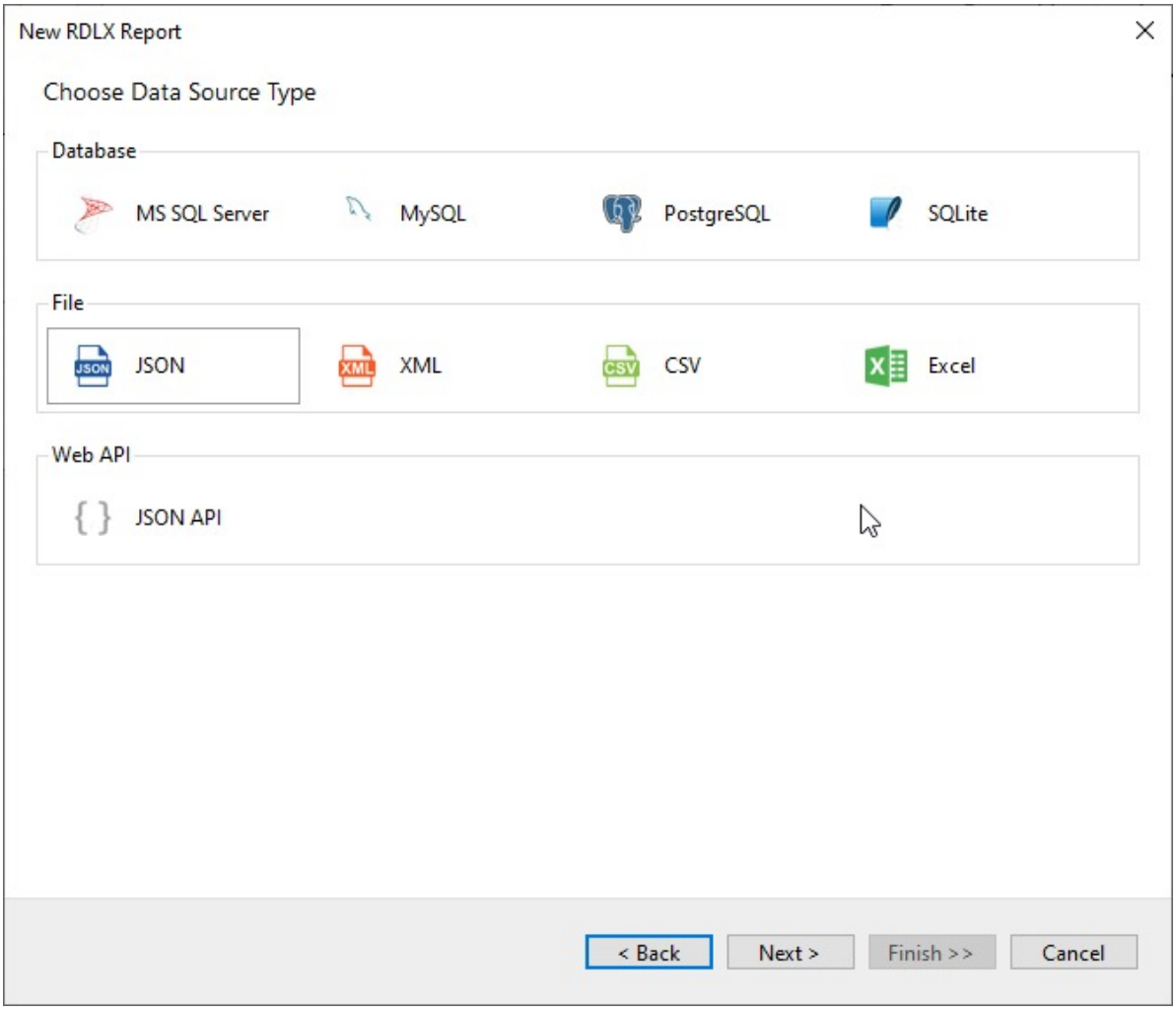
Connect to JSON Data Source using Report Wizard

The steps to connect to the JSON data source are:

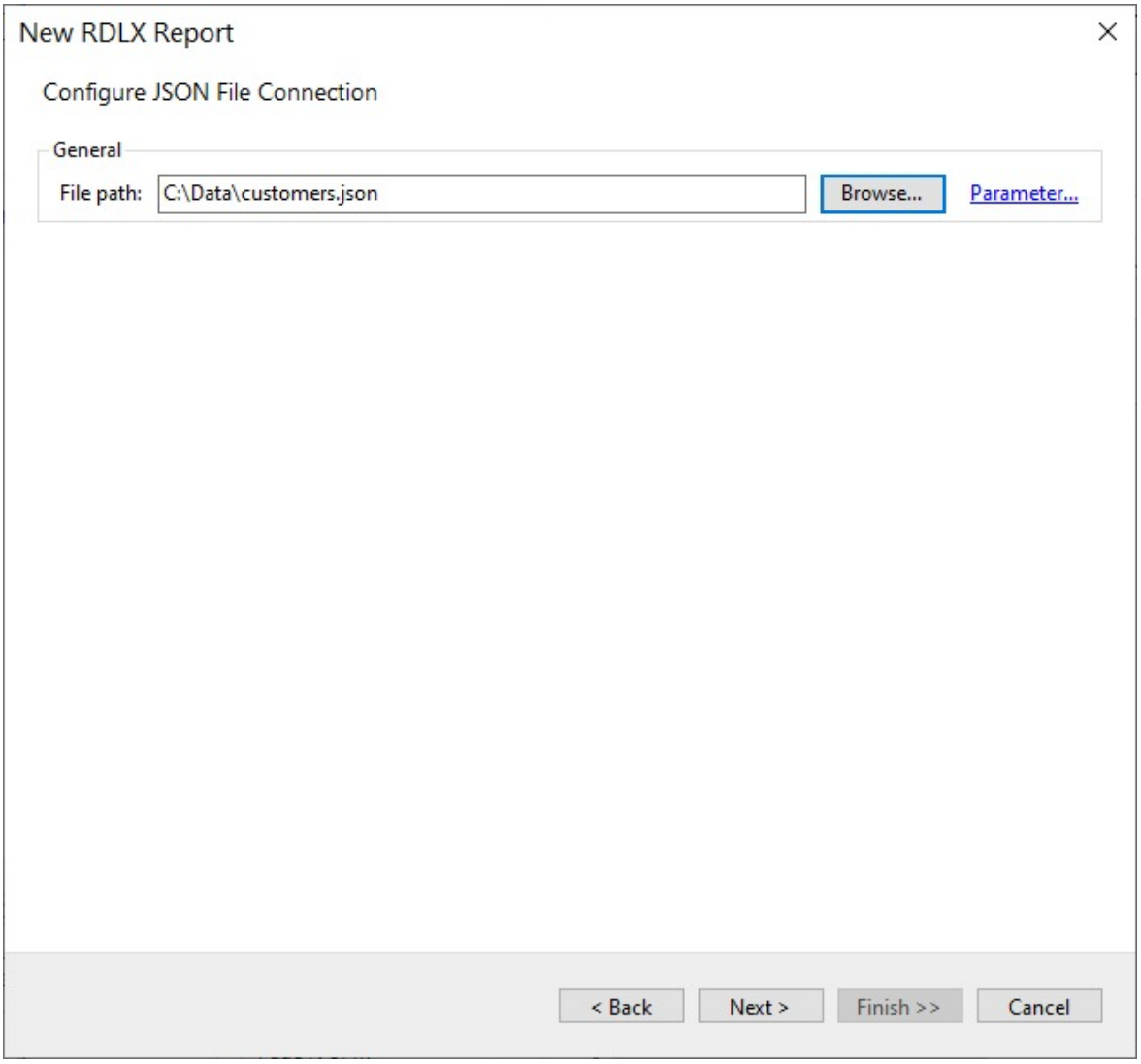
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



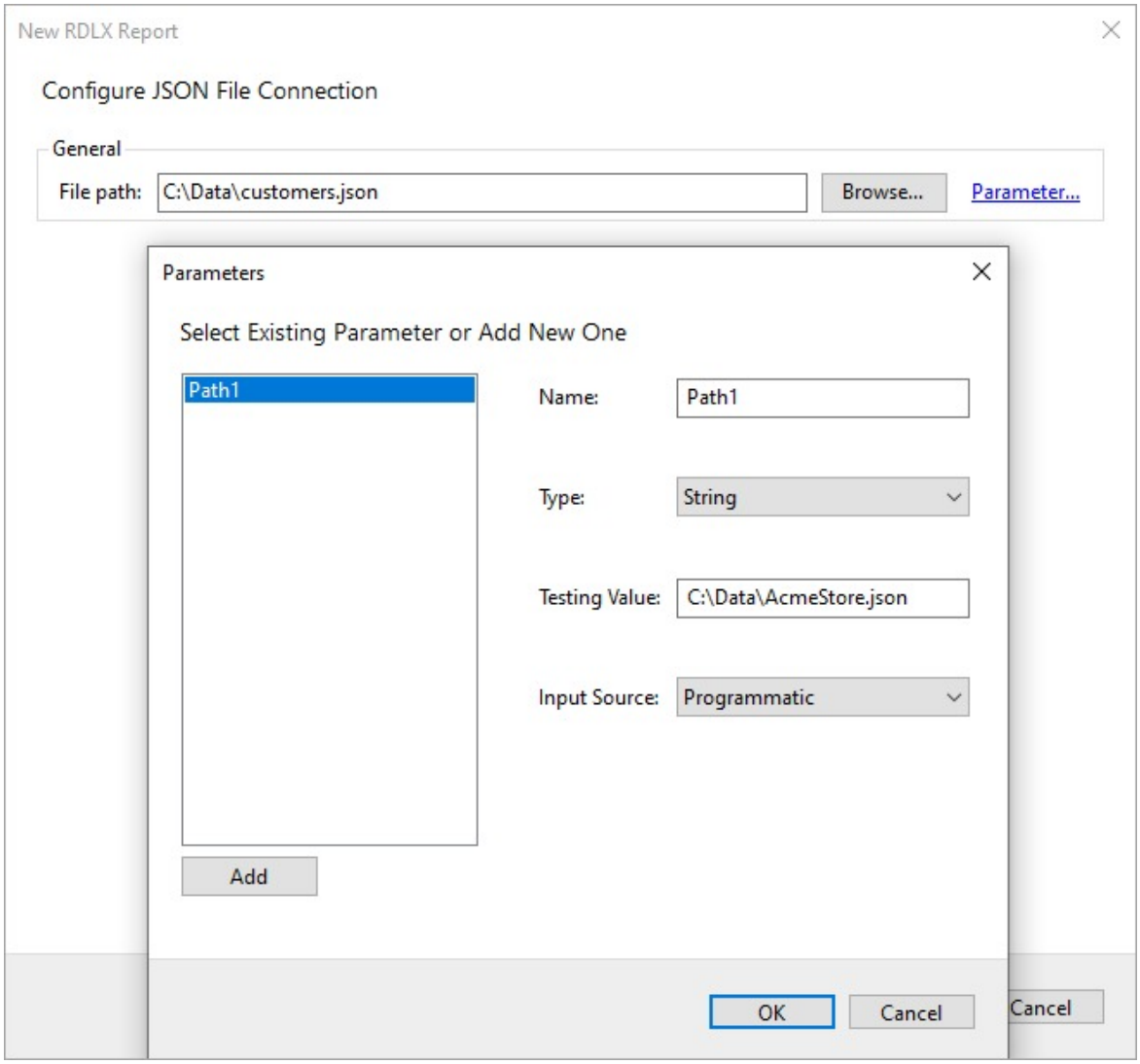
3. Select the Data Source Type as **JSON** and click Next.



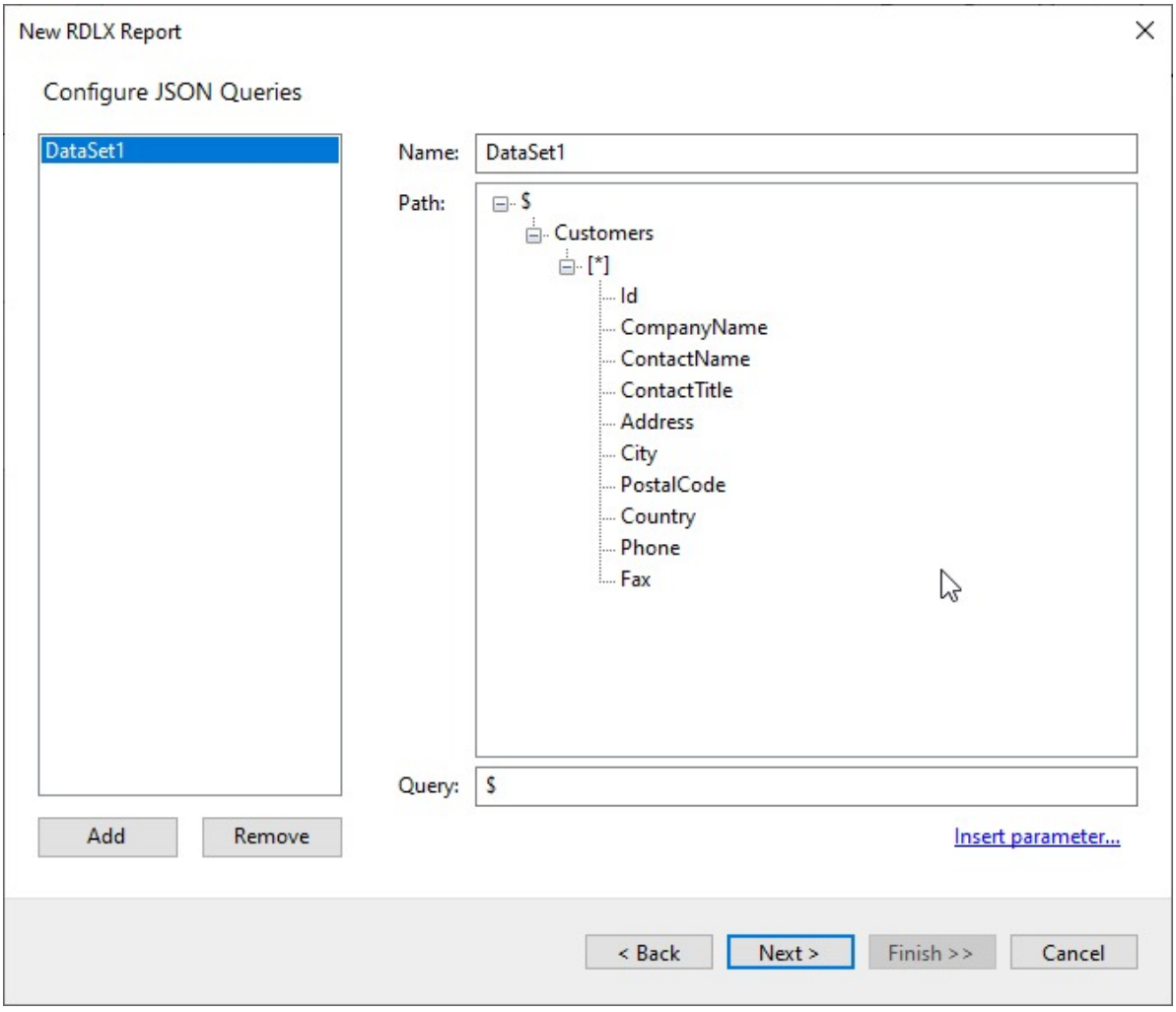
4. To specify JSON **File Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the 'customer.json' sample data source that can be downloaded from GitHub.



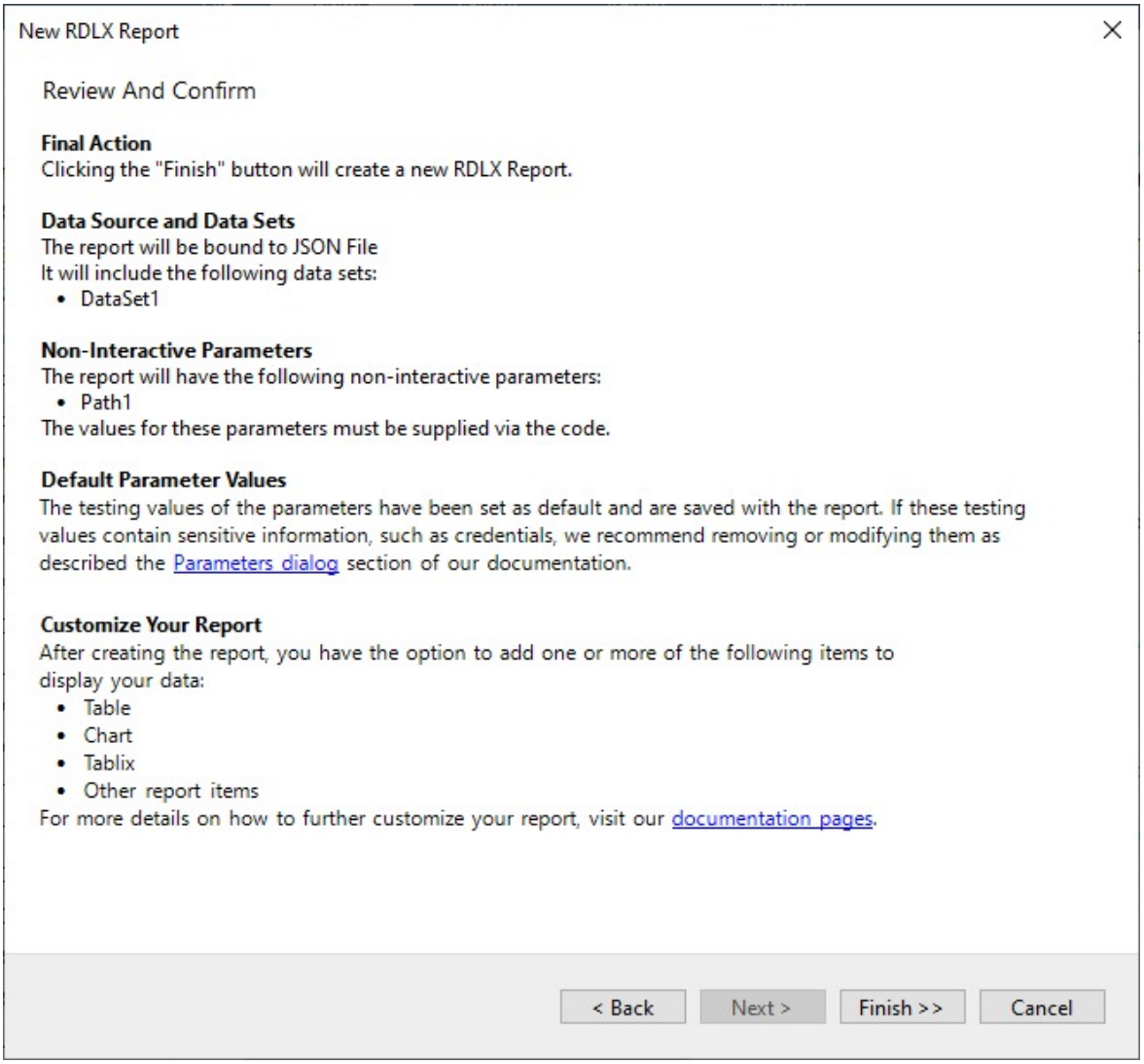
5. To specify the runtime connection values, click **Parameter** to open the **Parameters** dialog. Then click the **Add** button to add a new parameter, or select the existing parameter and specify the below details:
    - **Name**: Specify the name of the parameter.
    - **Type**: Select the value type (string by default) from the drop-down list.
    - **Testing Value**: Specify the runtime value for the connection properties.
    - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.



6. Click the **Next** option and configure the dataset by adding a valid query. Select the node from the data tree in the **Path** section to generate the path. The resulting query is displayed in the **Query** field.

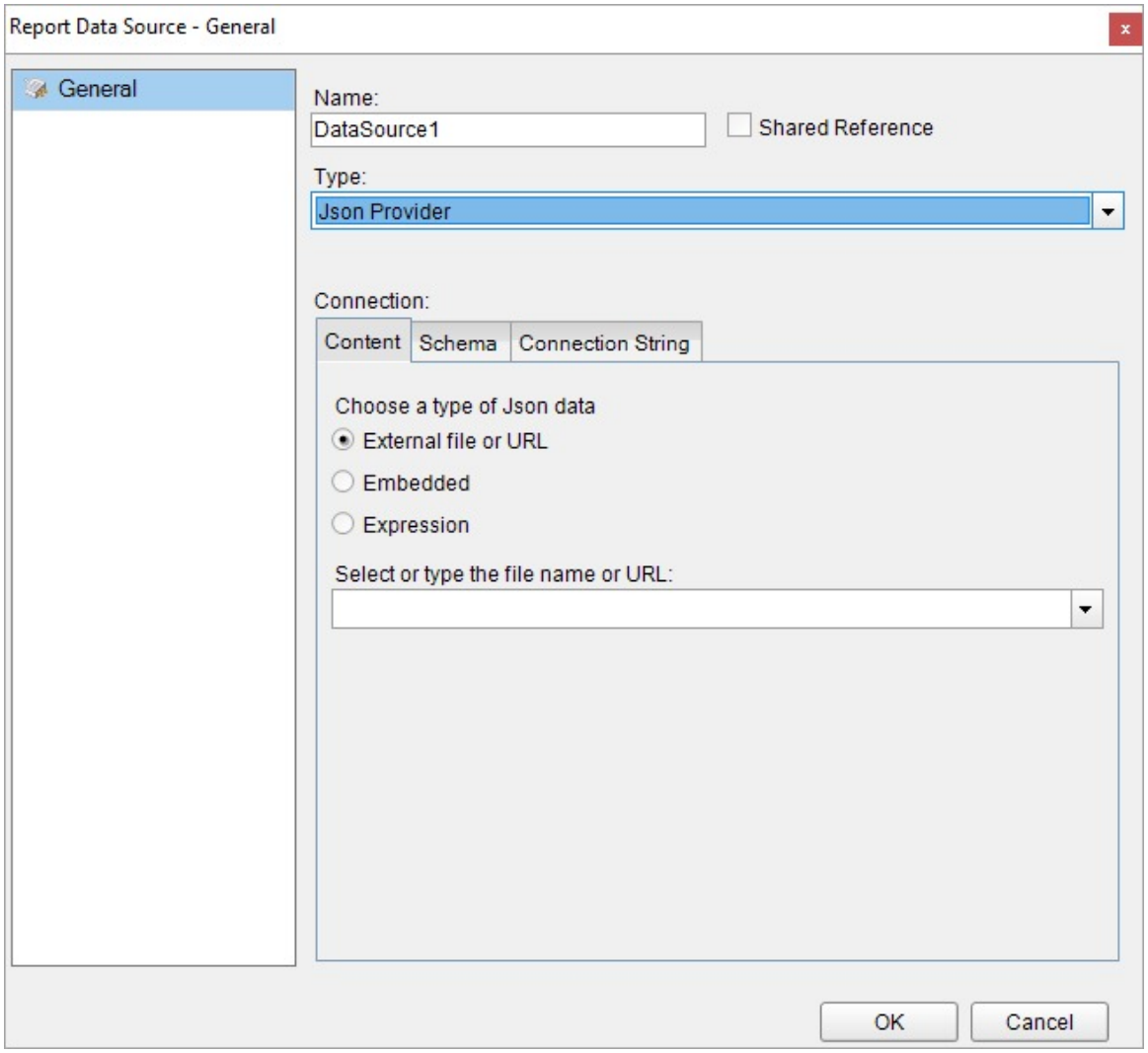


7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the JSON data source.



Connect to a JSON Data Source using Report Data Source dialog

1. In the designer, go to the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and then select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the name of the data source in the Name field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select Json Provider.



4. In the **Content** tab, set the type of Json data to 'External file or URL'.
5. Click the drop-down icon next to the **Select or type the file name or URL** field and select the **<Browse...>** option to specify the json file path, or enter the URL. For example, connect to the 'https://demodata.mescius.io/northwind/odata/v1/Orders' data source.

   The **Connection String** tab displays the generated connection string as shown below:

   ```
   jsondoc=https://demodata.mescius.io/northwind/odata/v1/Orders
   ```

   The **Configuration Settings** are explained in the next section.
6. Verify the generated connection string by clicking the **Validate DataSource** icon .
7. Click **OK** to save the changes and close the **Report Data Source** dialog.

Configuration Settings

The JSON Data Provider provides the following configuration settings under the Connection section in the **Report Data Source** dialog.

The **Content** tab describes the type of JSON data you want to use for connecting to a data source. The options available for specifying the JSON data are as follows:

- **External file or URL:** Enter the path or URL of an external JSON data file or select the file from the drop-down which

displays the JSON files available in the same folder as the report. The connection string generated using this option starts with the keyword **jsondoc**.
- **Embedded:** Enter the path of the JSON data file to embed in the report. You can enter the data manually or edit the data in the selected JSON file. The connection string generated using this option starts with the keyword **jsondata**.
- **Expression:** Enter an expression to bind to the JSON data at runtime.

The **Schema** tab describes the options available for specifying the JSON schema in ActiveReports.

- **Auto:** This is the default option that auto-generates the JSON schema.
- **External file or URL:** Enter the path or URL of an external JSON schema file or select the file from the drop-down which displays the JSON files available in your system. The connection string generated using this option starts with the keyword **schemadoc**.
- **Embedded:** Enter the path of the JSON schema file to embed in the report. You can enter the schema manually or edit the schema in the selected JSON file. The connection string generated using this option starts with the keyword **schemadata**.
  For generating JSON schema, use the JSON schema generator available at https://jsonschema.net.

The JSON schema describes the structure of the JSON data. In ActiveReports, the JSON data provider uses the JSON schema to obtain fields. For more information on JSON schema, please see https://json-schema.org/understanding-json-schema/.
JSON data provider does not support the following schema keywords, such as:

- **type**: Indicates the type of JSON schema element. See here for more information on the type keyword.
- **properties**: Indicates the properties collection for JSON schema elements with the object type. See here for more information on properties keyword.
- **items**: Indicates the definition of items for JSON schema elements with array type. Only single values are supported. For example, `"items" : [ {...}, {...}, {...} ]` is not supported because it contains multiple values. See here for more information on items keyword.
- **definitions**: Indicates the independent definitions which can be used by other JSON schema elements using `$ref` keyword. See here for more information on definitions keyword.
- **$ref**: Indicates the reference to a definition for JSON schema elements with the object type. Only "definitions" (`{ $ref : #/definitions/... }`) references are supported.

The **Connection String** tab displays the JSON connection string based on the defined configuration settings in the **Content** and **Schema** tab.

- If you choose the **External file or URL** option, the connection string will be as follows -
  `jsondoc=C:\customers.json`
  or
  `jsondoc=https://demodata.mescius.io/northwind/odata/v1/Orders`
- If you choose the **Embedded** option, the connection string will be generated as follows -

Connection String
`jsondata={"Customers":[{"Id":"ALFKI", "CompanyName":"Alfreds Futterkiste", "Country":"Germany"}, {"Id":"ANATR", "CompanyName":"Ana Trujillo Emparedados y helados", "Country":"Mexico"}], "ResponseStatus":{}}`

- If you choose the **Expression** option, the connection string will start with an "equals to" as shown -

Connection String
`="jsondata={ 'Name': 'Name'};schemadata={ '$schema': 'http://json-schema.org/draft-04/schema#', 'definitions': {}, 'id': 'http://example.com/example.json', 'properties': { 'Name': { 'id': '/properties/Name', 'type': 'string' } }, 'type': 'object'}"`

> ⚠ **Caution:** If you include an expression in the connection string, use single quotes (') in jsondoc or jsondata and schemadoc or schemadata instead of the double quotes (") .
>
> For example, the following connection string is invalid:

Connection String (invalid)

```
="jsondata={ "Name": "Name"};schemadata={ "$schema": "http://json-schema.org/draft-04/schema#", "definitions": {}, "id": "http://example.com/example.json", "properties": { "Name": { "id": "/properties/Name", "type": "string" } }, "type": "object"}"
```

Build Connection String

It is possible to configure the JSON data source connection string by using the **Build** button in the **Connection String** tab of the **Report Data Source - General** dialog. This allows fetching a JSON from an external URL using the specified method (GET, POST) and other request options. Clicking the **Build** button opens the **Configure JSON Data Source** dialog where you can create a JSON connection string with parameters such as data path, method (GET, POST), headers, and body.

To connect to a external URL containing JSON data, some settings are required to be configured as elaborated.

Data Path:

The URL or the endpoint of the request consists of service root URI, resource path, and query options. The fields from json provided in the data source's endpoint are extracted depending on the dataset's settings. A URL with IP address (eg. *http://10.64.2.17:51980/admin*) or with a machine name (eg. *http://in-esxi-w10v17:8080/*) is also valid.

> **Note**: The Data Path must be accurate with a valid URL, else it can not be validated by the builder.

HTTP Headers:

Contains additional info related to the request or the response, for eg, provide expected content type necessary for establishing connection with endpoint which must return json. For example, providing credentials to login to a service/URI to retrieve data. Collection editor to add header/value pair for headers is available. This parameter has to be set to send the request body in JSON format.

Example of Header,Value pairs:

- *Content-Type, application-json* (to send the request body in JSON format)
- *Accept, application-json* (to read the response in the desired format)

HTTP Method:

Request method can be **GET** (included in URL) or **POST** (included in 'body').

- **GET** requests data from a specified resource. GET method examples are provided in **Sample connection strings and dataset queries** section.
- **POST** requests allow defining a body as an expression to obtain data.

Request Body (POST):

Text area, applicable only in case of POST HTTP request.

You should provide a valid combination of configuration settings for the successful processing of the connection string with the POST method. If a combination of a valid URL, a valid header name, a valid header value, and a valid body is provided, then the connection string can be saved and validated. Only in this case, it is possible to create a dataset based on this data source.

If an invalid URL, an invalid header name or header value, invalid body, or a combination of these settings is entered, error message is displayed while validating the connection string.

Sample connection strings and dataset queries

1. Simple

Connection String
`jsondoc=https://demodata.mescius.io/northwind/api/v1/Categories`
Dataset Query
`$.[*]`

To convert the format of data in the requested URI to JSON, use either *$format* query or Http Header as shown in the following connection strings:

- *$format* query:

Connection String
`jsondoc=https://services.odata.org/v3/northwind/northwind.svc/Invoices?$format=json`

- Http Header:

Connection String
`headers={"Accept":"application/json"};jsondoc=https://services.odata.org/v3/northwind/northwind.svc/Invoices`
Dataset Query
`$.value[*]`

2. Using Parameters to query specific fields

Connection String
`headers={"Accept":"application/json"};jsondoc=https://services.odata.org/v3/northwind/northwind.svc/Invoices?$select=ShipName,ShipRegion`
Dataset Query
`$.value[*]`

3. Using Parameters to filter database fields

Connection String
`headers={"Accept":"application/json"};jsondoc=https://services.odata.org/v3/northwind/northwind.svc/Invoices?$filter=ShipName eq 'Alfreds%20Futterkiste'`
Dataset Query
`$.value[*]`

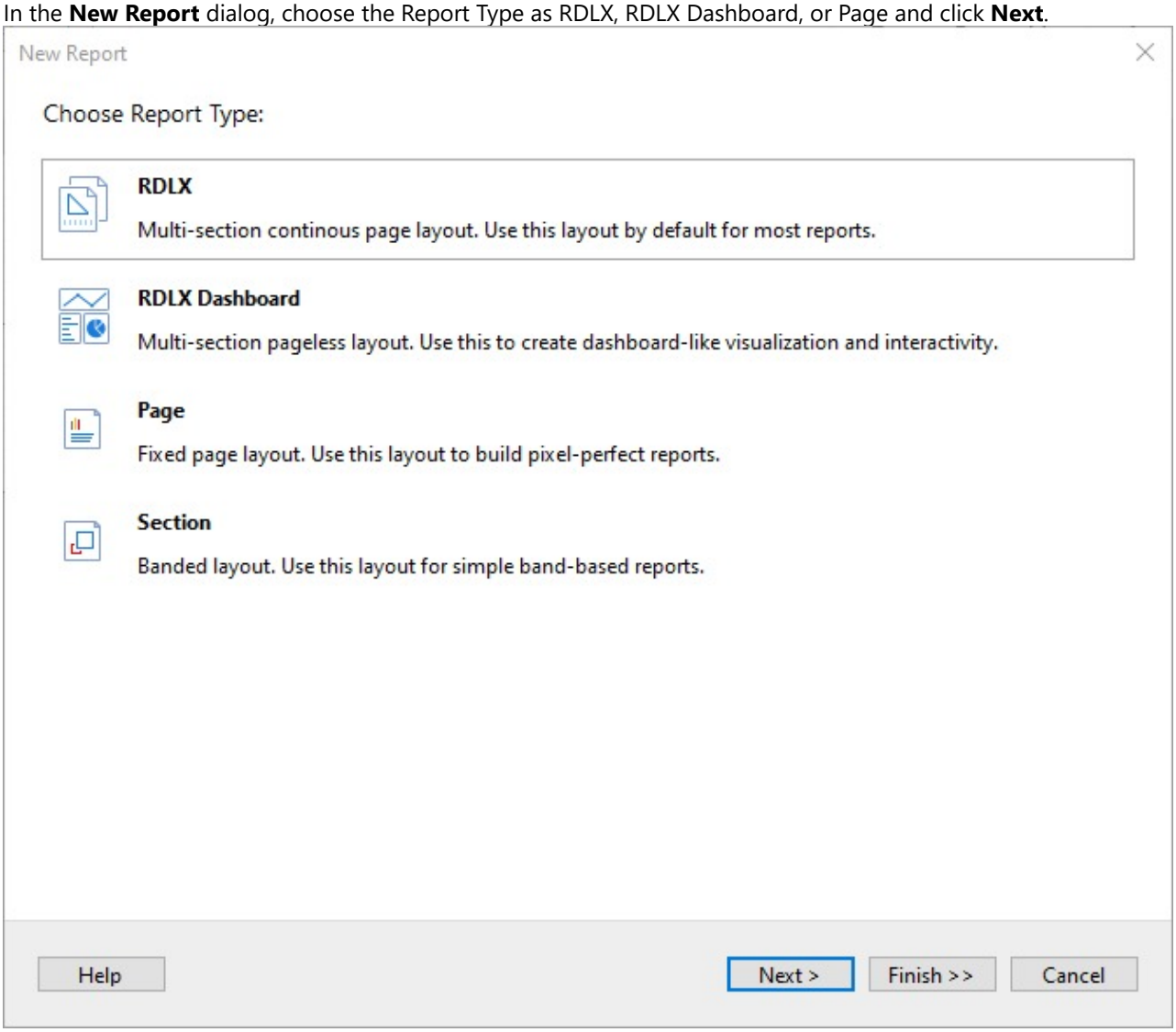
XML

This article explains connecting a Page or an RDLX report to a XML data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

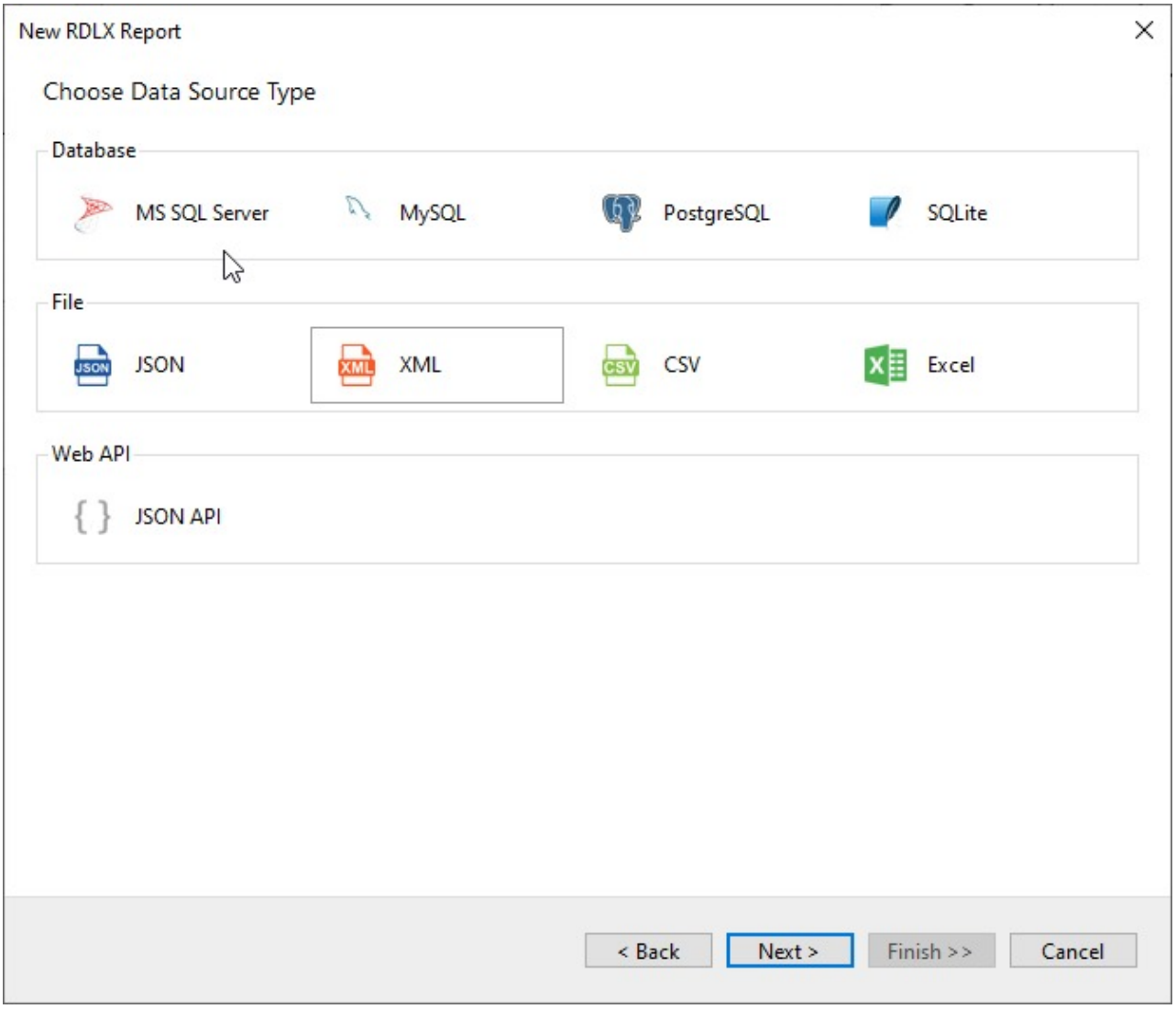
Connect to XML Data Source using Report Wizard

The steps to connect to the XML data source are:

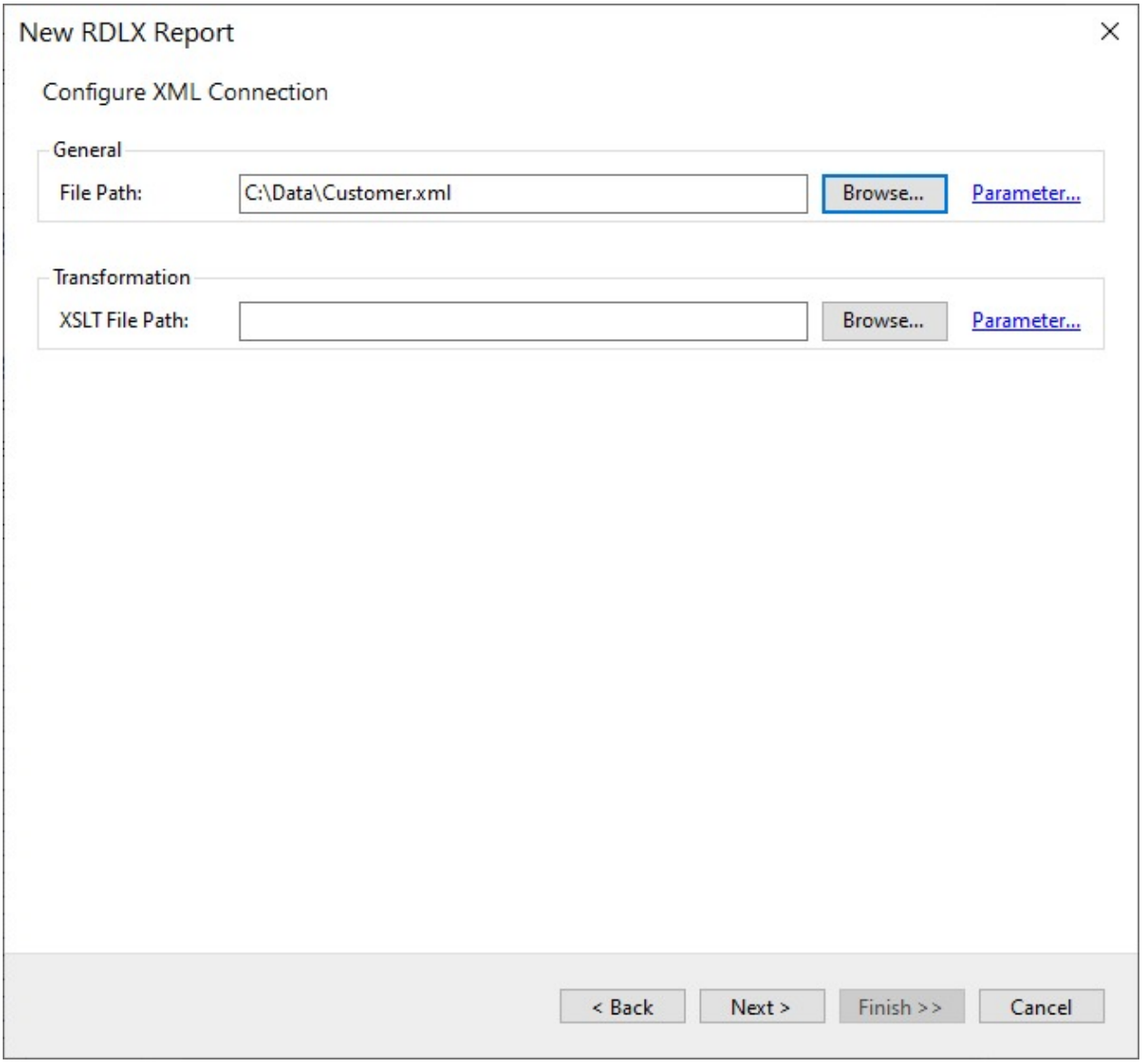
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



3. Select the Data Source Type as **XML** and click Next.

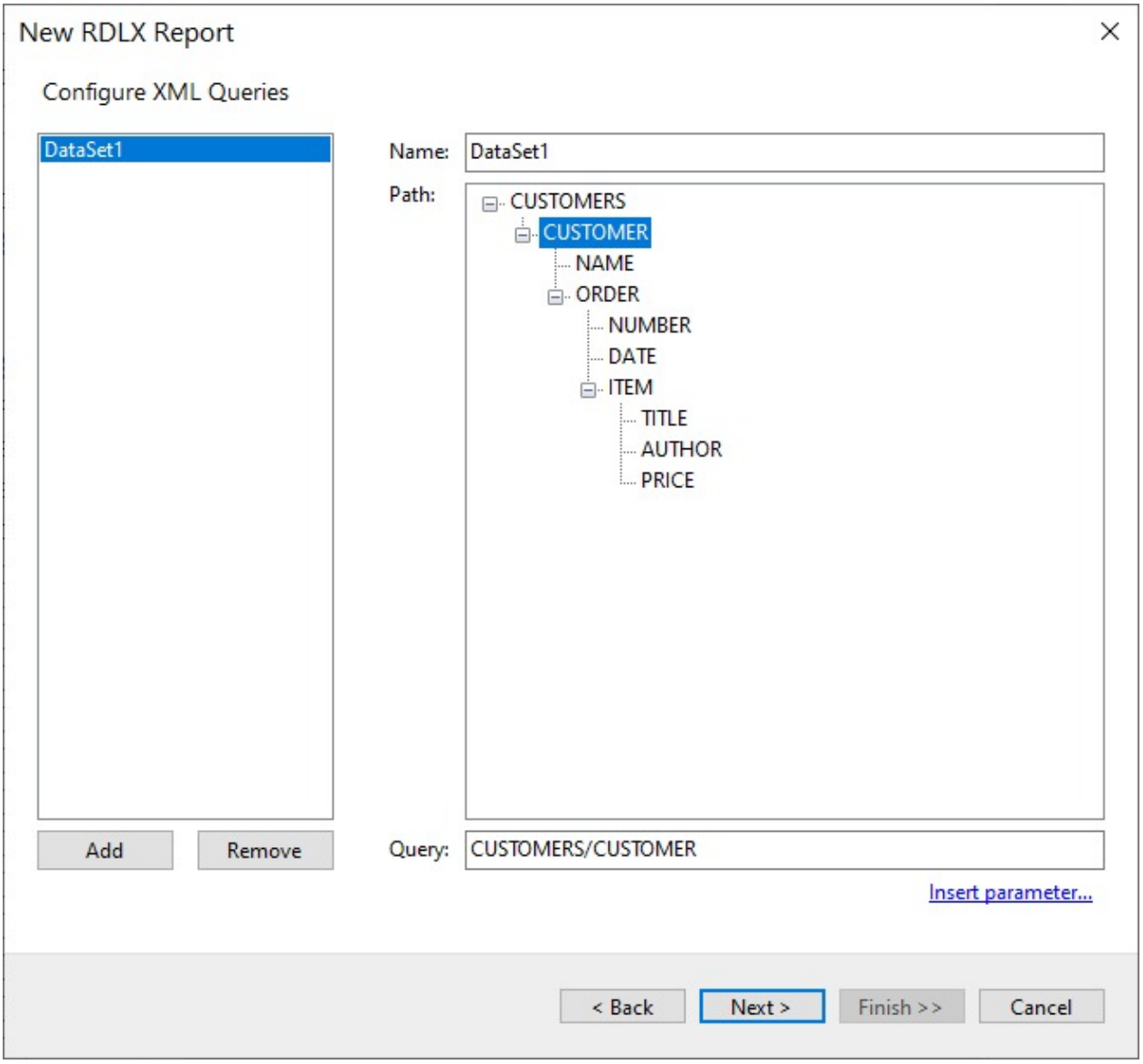


4. To specify the xml **File Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the 'Factbook.xml' sample data source that can be downloaded from GitHub.

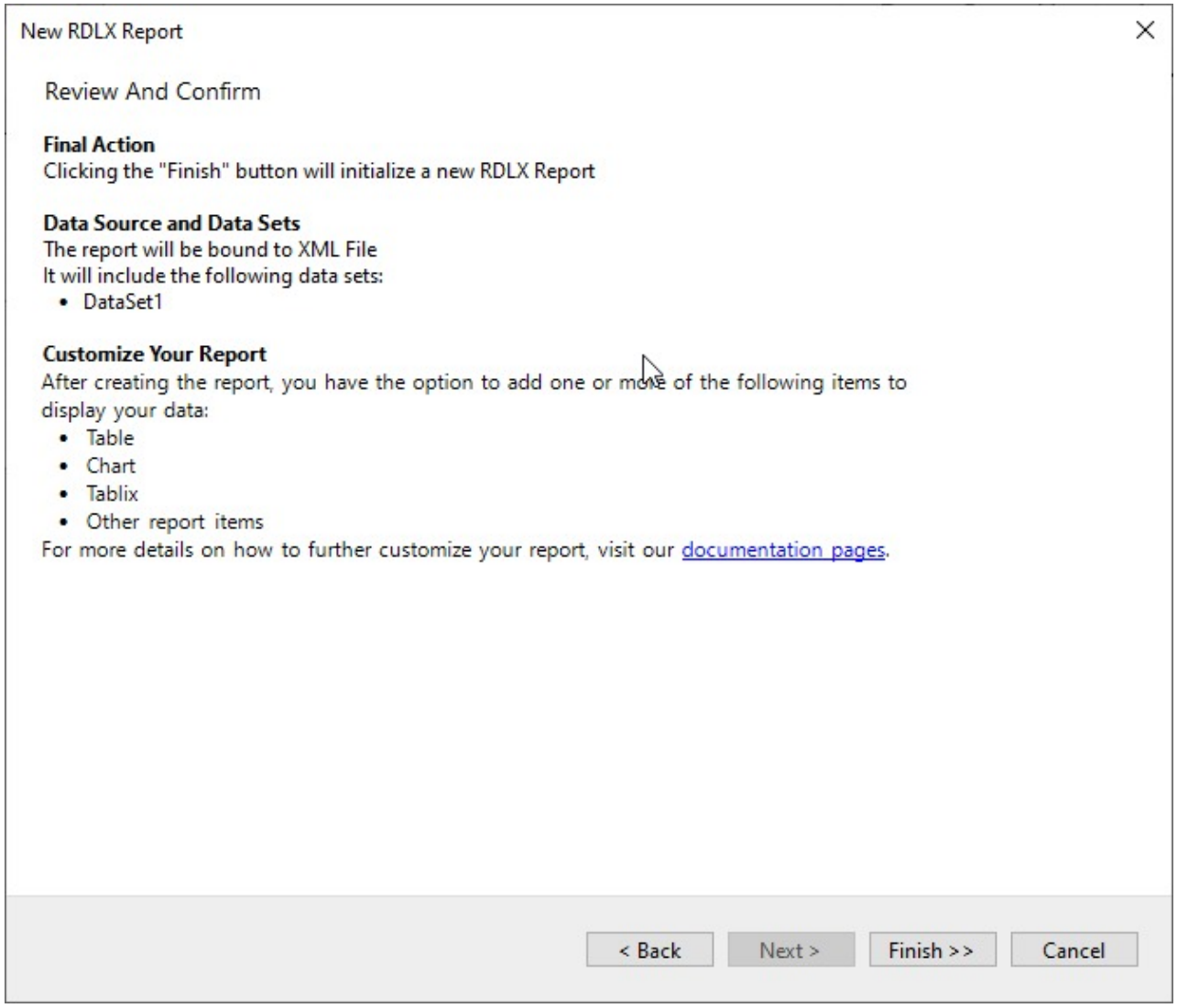


5. To specify the runtime connection values, click **Parameter** (or **Insert Parameter** for queries) to open the **Parameters** dialog. Then click the **Add** button to add a new parameter, or select the existing parameter and specify the below details:
    - **Name**: Specify the name of the parameter.
    - **Type**: Select the value type (string by default) from the drop-down list.
    - **Testing Value**: Specify the runtime value for the connection properties.
    - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.

6. Click the **Next** option and configure the dataset by adding a valid query. Select the node from the data tree in the **Path** section to generate the query. The resulting query is displayed in the **Query** field.

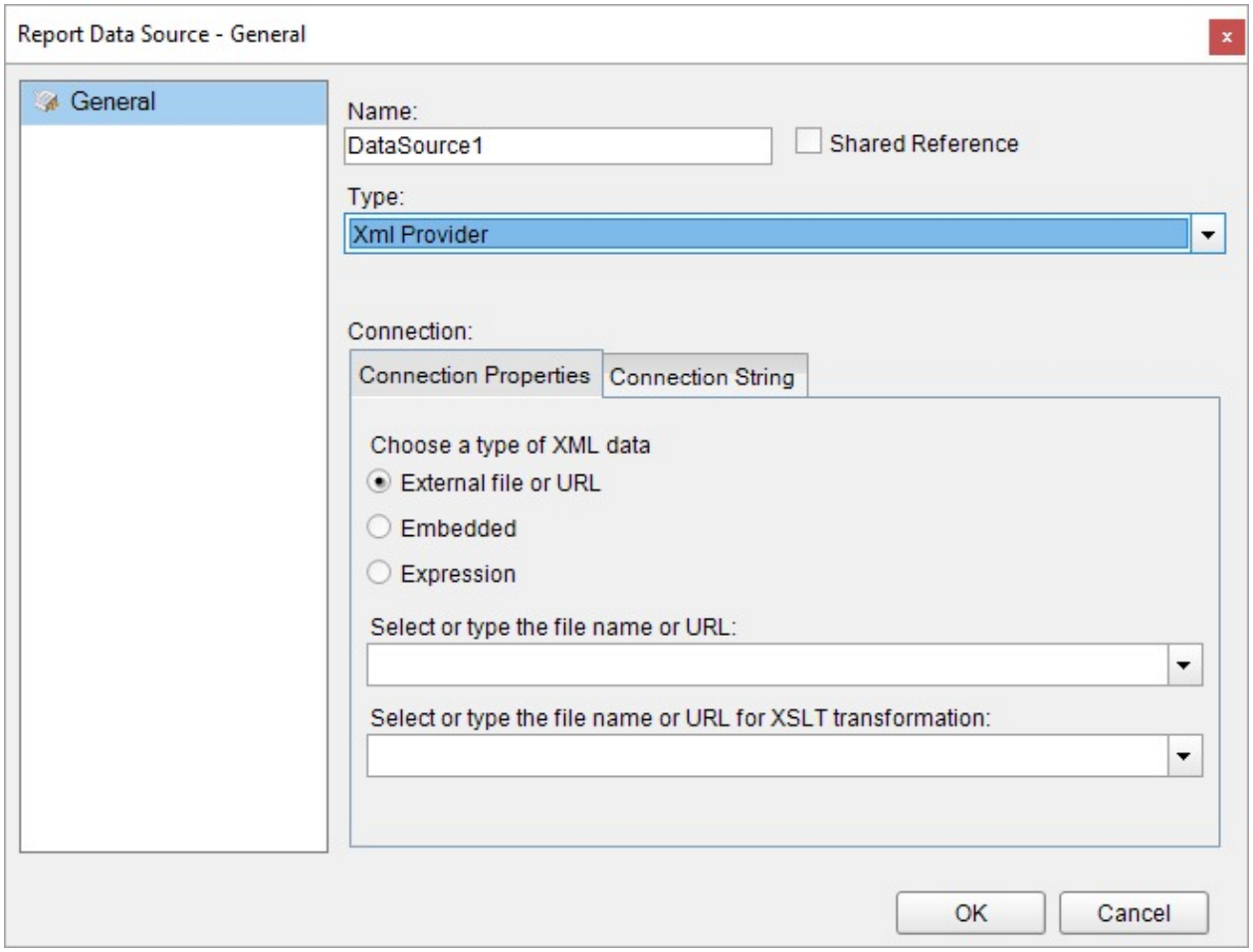


7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the XML data source.



Connect to an XML Data Source using Report Data Source dialog

1. In the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the data source name in the Name field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select Xml Provider.



4. In the **Connection Properties** tab, select the type of XML data as an **External file or URL**.
5. Click the dropdown next to the **Select or type the file name or URL** field and select the **<Browse...>** option to specify the xml file path. For example, you can connect to the Factbook.xml sample data source which can be downloaded from GitHub.

   The **Connection String** tab displays the generated connection string as shown below:
   `xmldoc=C:\Factbook.xml`
   For more information, see the **Configuration Settings for XML Data Source** section.
6. Verify the generated connection string by clicking the **Validate DataSource** icon .
7. Click the **OK** button to save the changes.

Configuration Settings for XML Data Source

The XML Data Provider provides the following configuration settings in the **Report Data Source** dialog.

The **Connection Properties** tab describes the type of xml data you want to use for connecting to a data source. You can also specify an XSLT file to apply to the XML data.

- **External file or URL**: This field requires you to enter the path of an external XML source such as a local file or the http location of a file.
- **Embedded**: This field requires you to enter the path of the XML file to embed in the report. You can also enter the data manually or edit the data in the selected XML file.
- **Expression**: This field requires you to enter the path expression. Users can also enter the path expression in the connection string.

The **Connection String** tab displays the XML connection string based on the defined configuration settings in the **Connection Properties** tab.

- If you choose the **External file or URL** option, the connection string will be as follows -
  `xmldoc=C:\MyXmlFile.xml;`
  where the **xmldoc** refers to a specific XML file located on either the file system or at a web-accessible location.
- If you choose the **Embedded** option, the connection string will be as follows -

```
xmldata=<people>
  <person>
    <name>
      <given>John</given>
      <family>Doe</family>
    </name>
  </person>
  <person>
    <name>
      <given>Jane</given>
      <family>Smith</family>
    </name>
  </person>
</people>
```

  where the **xmldata** provides specific XML data in the connection string itself.
- If you choose the **Expression** option, the connection string will start with an "equals to" as shown -
  `="xmldoc=" & [@Parameter]`

Note that elements in the connection string must be terminated with a semicolon (;) character.

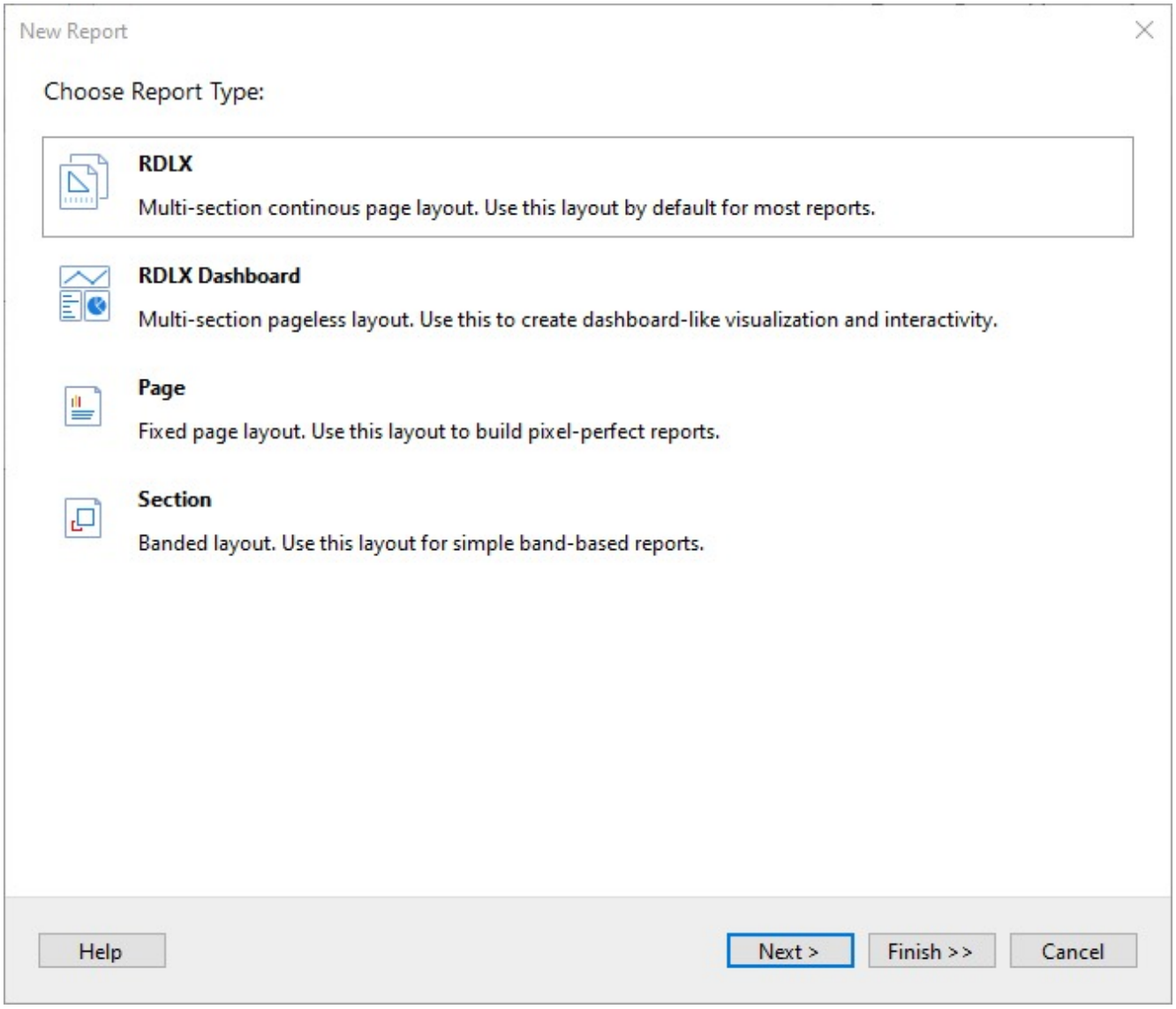
CSV

This article explains connecting a Page or an RDLX report to a CSV data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

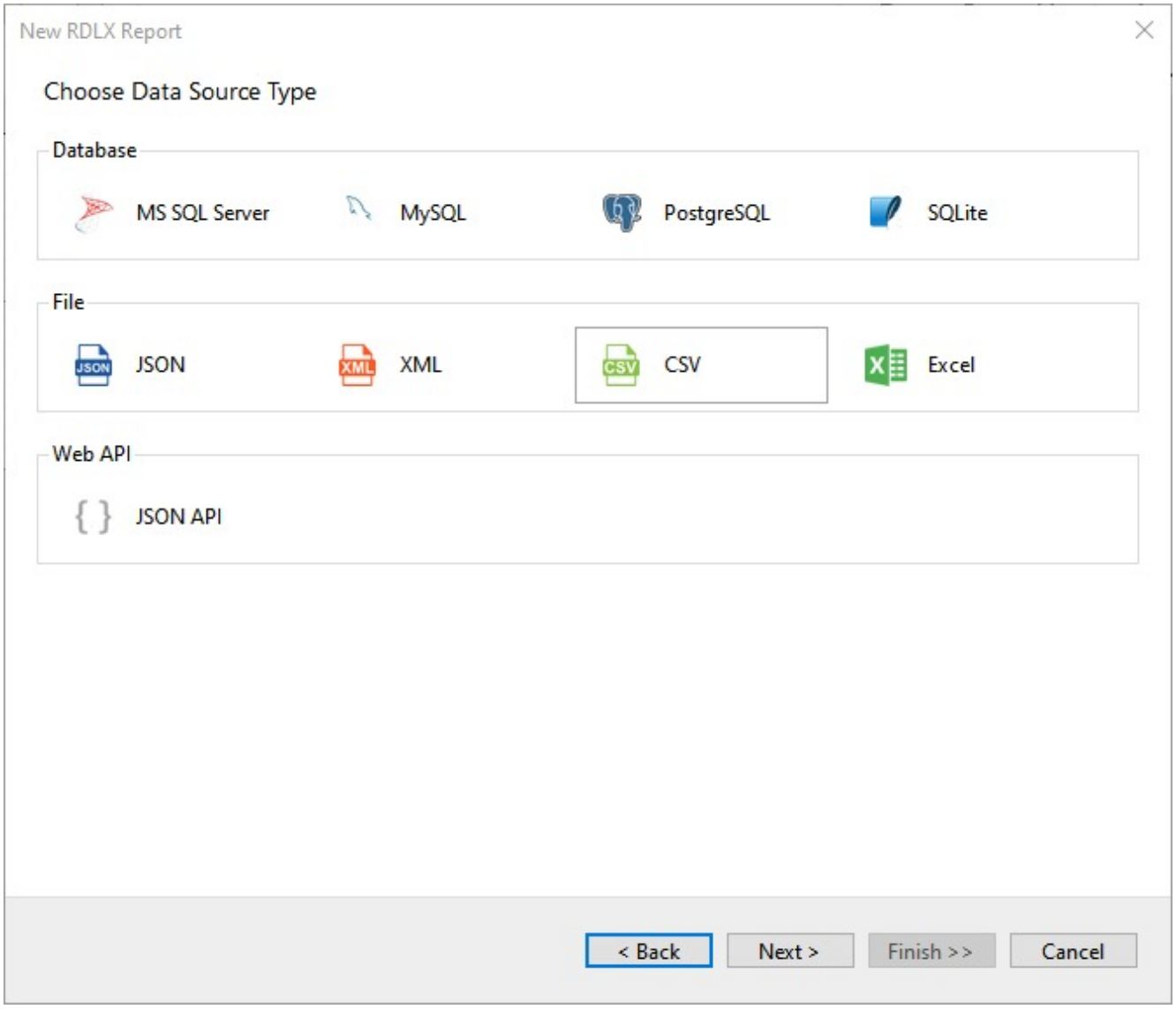
Connect to CSV Data Source using Report Wizard

The steps to connect to the CSV data source are:

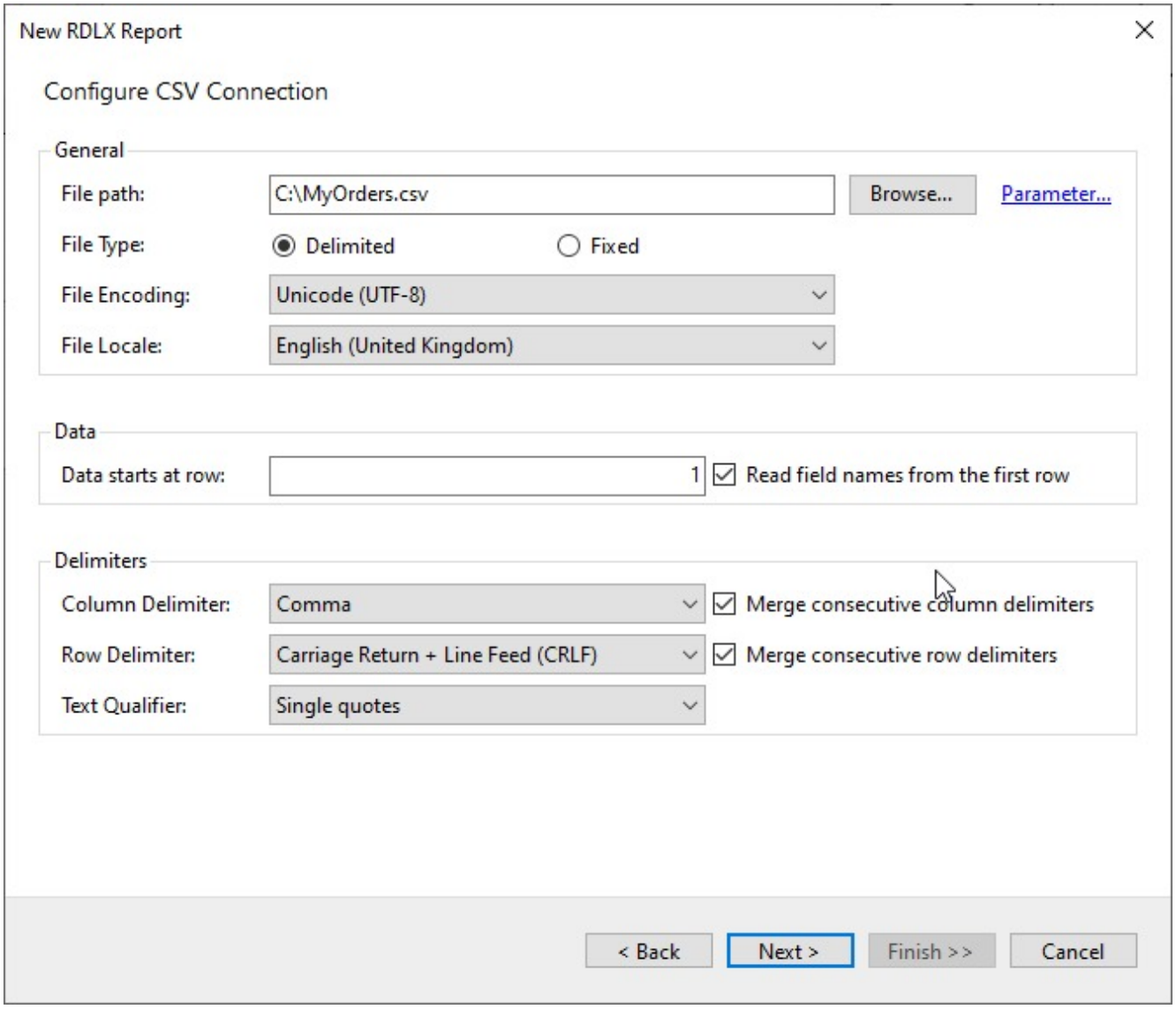
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



3. Select the Data Source Type as **CSV** and click **Next**.

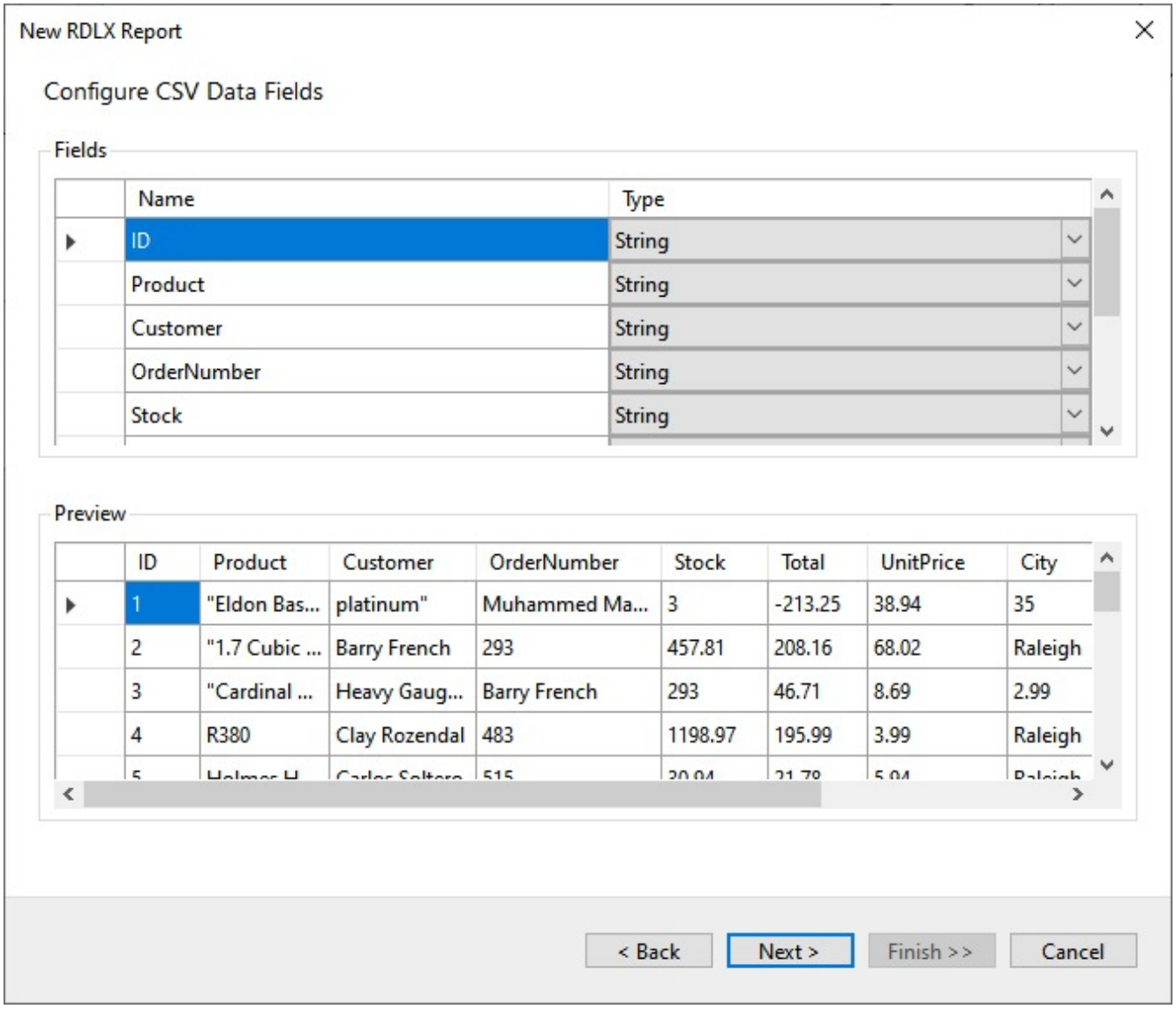


4. To specify the **File Path**, click the **Browse** button and navigate to the desired folder on your system. For example, you can connect to the **MyOrders.csv** sample data source which can be downloaded from GitHub.

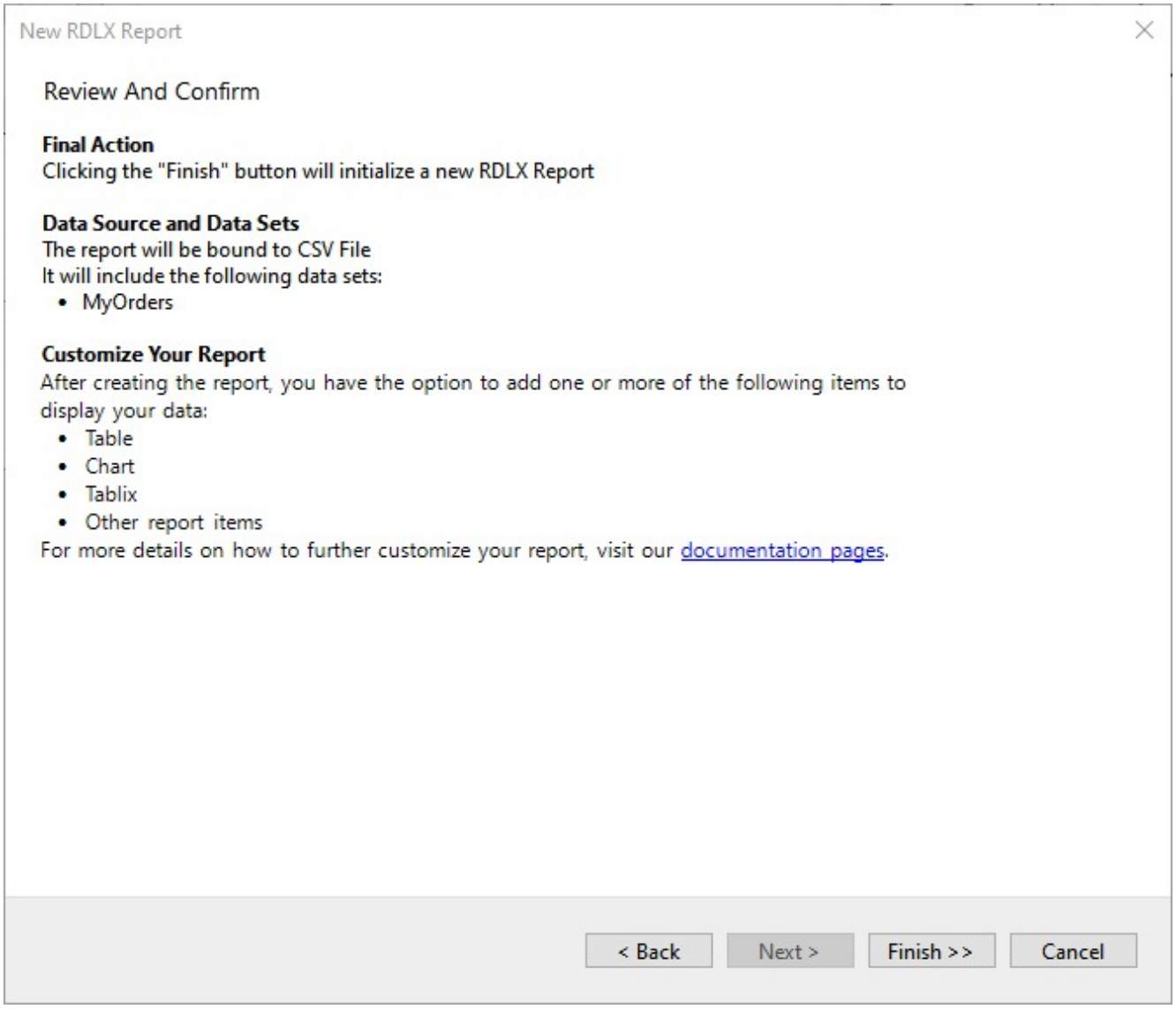


5. To specify the runtime connection values, click **Parameter** (or **Insert Parameter** for queries) to open the **Parameters** dialog. Then click the **Add** button to add a new parameter, or select the existing parameter and specify the below details:
    - **Name**: Specify the name of the parameter.
    - **Type**: Select the value type (string by default) from the drop-down list.
    - **Testing Value**: Specify the runtime value for the connection properties.
    - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.

6. Select the **File Type** as **Delimited** or **Fixed**.
7. Then click **Next** to check the **Fields** section, which includes field names with their corresponding data types present in the CSV file. This allows you to modify the field names and their data type (like String, Boolean, DateTime, Integer, Float, Decimal, Double, or Long) for the columns. You can check the changes in the **Preview** section.

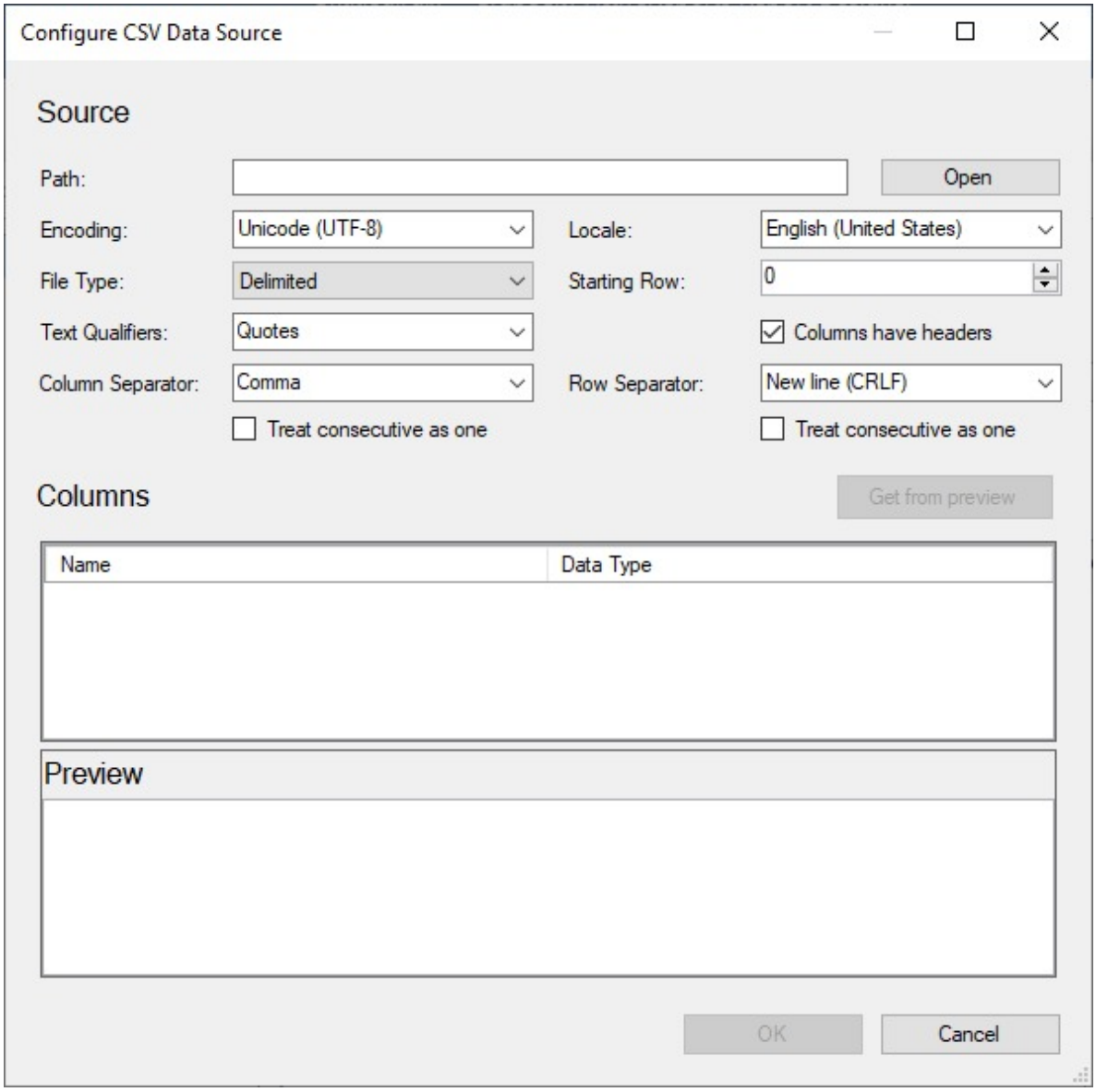


8. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the CSV data source.

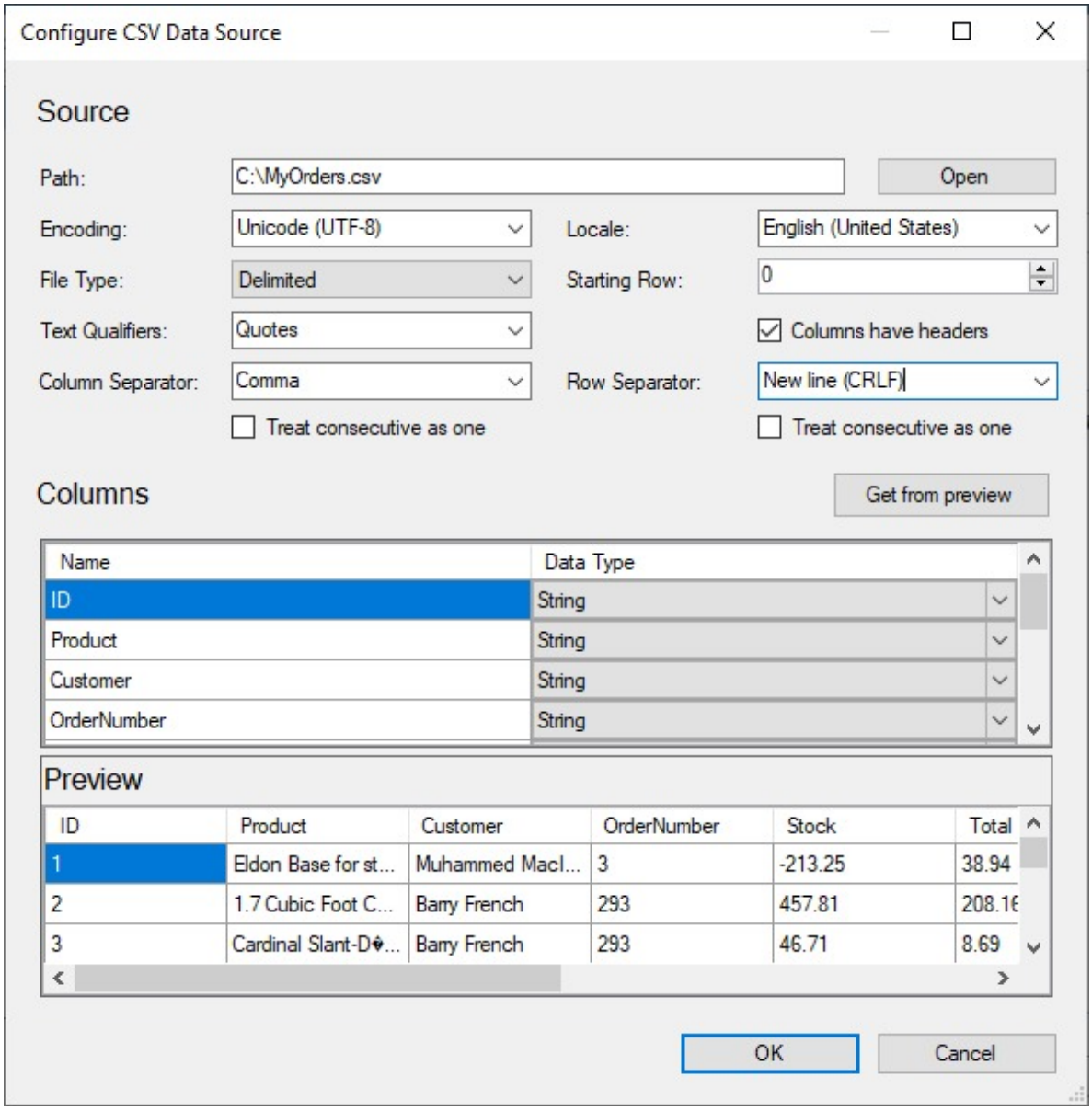


Connect to a CSV Data Source using Report Data Source dialog

1. In the designer, go to the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and then select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the name of the data source in the Name field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select Csv Provider.
4. In the **Connection String** tab, click the **Build** icon to open the **Configure CSV Data Source** dialog.



5. To specify the **Path** of the file, click the **Open** button and navigate to the desired folder on your system. For example, you can connect to the **MyOrders.csv** sample data source which can be downloaded from GitHub.
6. Select the **Column Separator** as **Comma** from the drop-down menu.
7. Click the **Get from Preview** button to fill the Columns area with the column names and their corresponding data types (string by default) present in the CSV file. This allows you to modify the name and data type (like String, Boolean, DateTime, Integer, Float, Decimal, Double, or Long) for the columns.
   For more information, see the **Configuration Settings for CSV Data Source** section.
8. Click **OK** to save the changes and close the **Configure CSV Data Source** dialog.



The **Connection String** tab displays the generated connection string as shown below:

```
    Path=C:\\MyOrders.csv;Locale=en-
IN;TextQualifier=";ColumnsSeparator=,;RowsSeparator=\r\n;Columns=ID,Product,Customer,OrderNu
```

You can validate the connection string by clicking the**Validate DataSource** icon.

9. Click **OK** to close the **Report Data Source** dialog. Your report is now connected to the CSV data source successfully.

Configuration Settings for CSV Data Source

The CSV Data Provider provides the following configuration settings in the **Configure CSV Data Source** dialog. Based on the defined configuration settings, the CSV connection string is generated.

Setting	Description
File Path/Path	Path to the CSV file - both local and relative; or a URL for centrally located CSV data sources.
File Type/Type	Define the type of CSV file. You can choose from Fixed and Delimited options.
File Encoding/Encoding	Specify the character encoding used in the CSV file.
File Locale/Locale	Specify the locale used in the CSV file.
Data Starts at row/Starting Row	Row number to start fetching data.
Use starting row as column headers	Select this check box to use the first row as column header.
Column Delimiter/Column Separator[1]	Specify the symbol used to separate the columns in the CSV file. You can choose from Comma, Semicolon, Tab, and Space options.
Merge consecutive column delimiters	Select this check box to treat consecutive column delimiters as one.
Row Delimiter/Row Separator[1]	Symbol used to separate the rows in the CSV file. You can choose from CRLF (carriage return and line feed), CR (carriage return), and LF (line feed) new line formats.
Merge consecutive row delimiters	Select this check box to treat consecutive row delimiters as one.
Text Qualifier[1]	Character to specify where the text begins and ends, that is, the character that encloses values in the CSV file. You can choose from Quotes and Single quotes options.
Columns have headers	Specify whether the CSV file has columns with headers or not.
Treat consecutive as one[1]	Specify whether to join the column separators or row separators as one.
Get from preview	Fills the Columns area with names and data types (string by default) for columns present in the CSV file. This allows you to modify the name and data type (like String, Boolean, DateTime, Integer, Float, Decimal, Double, or Long) for the columns.

[1] These options are not available for **Fixed** file type.

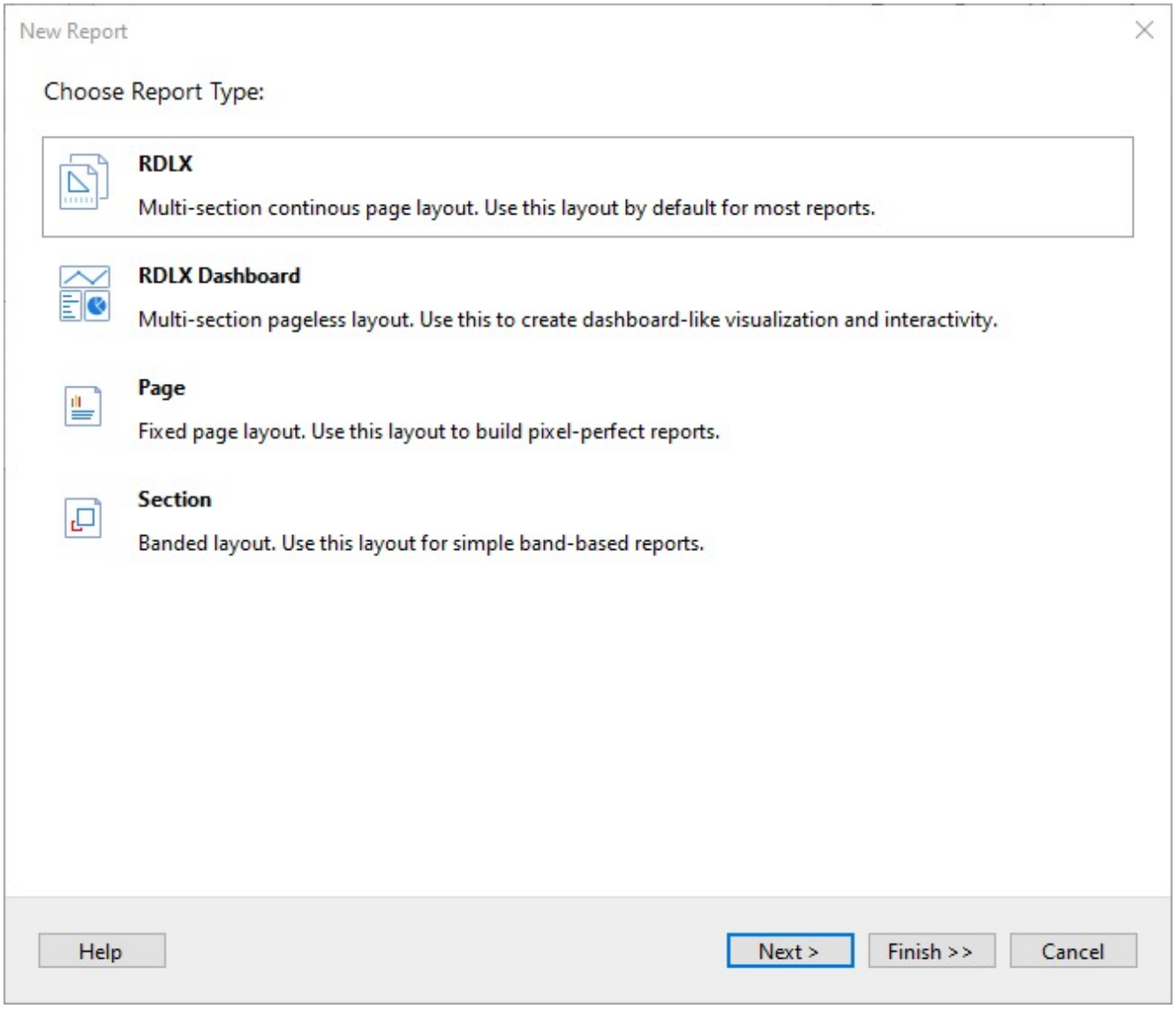
Excel

This article explains connecting a Page or an RDLX report to a Excel data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

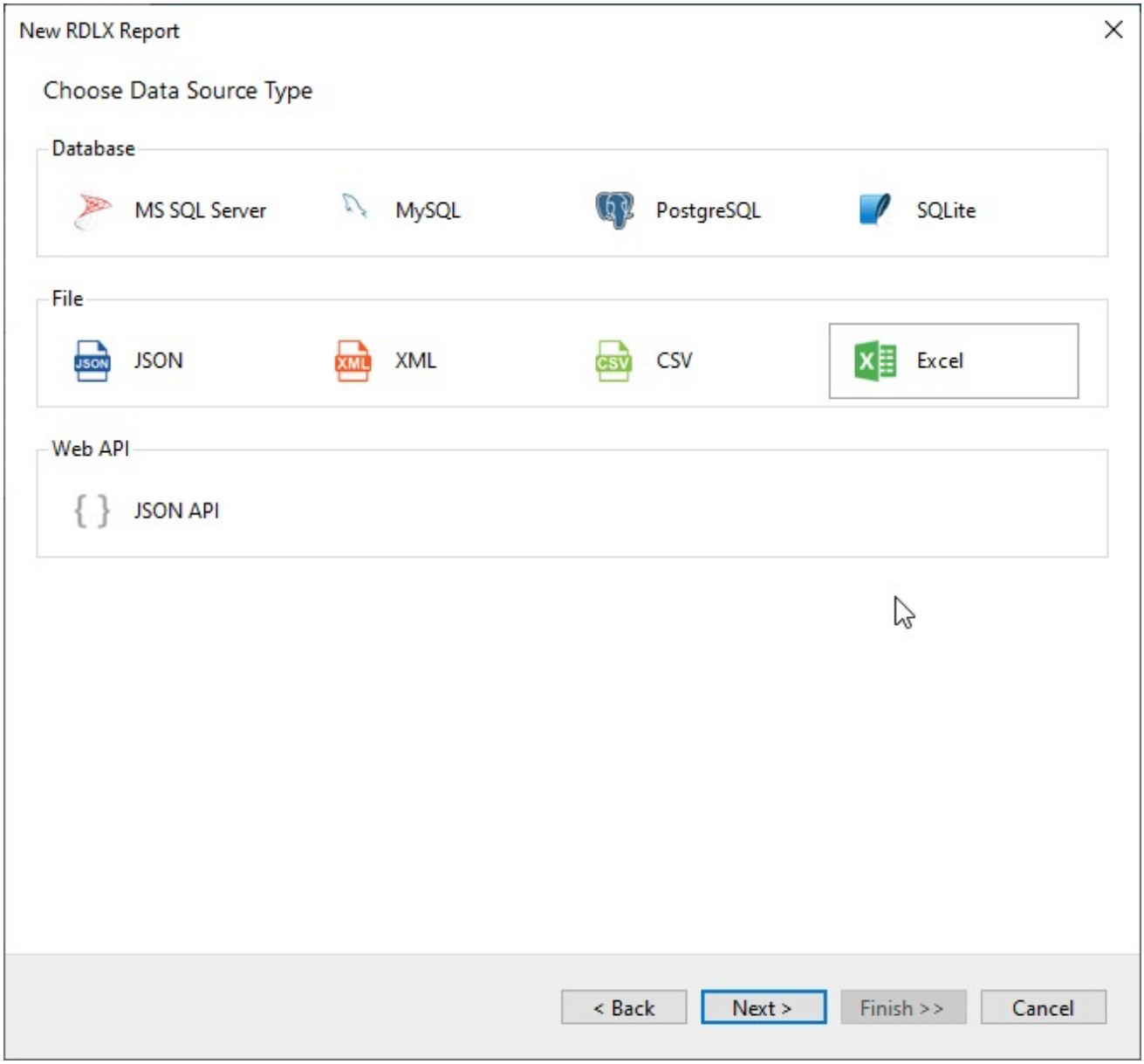
Connect to Excel Data Source using Report Wizard

The steps to connect to the Excel data source are:

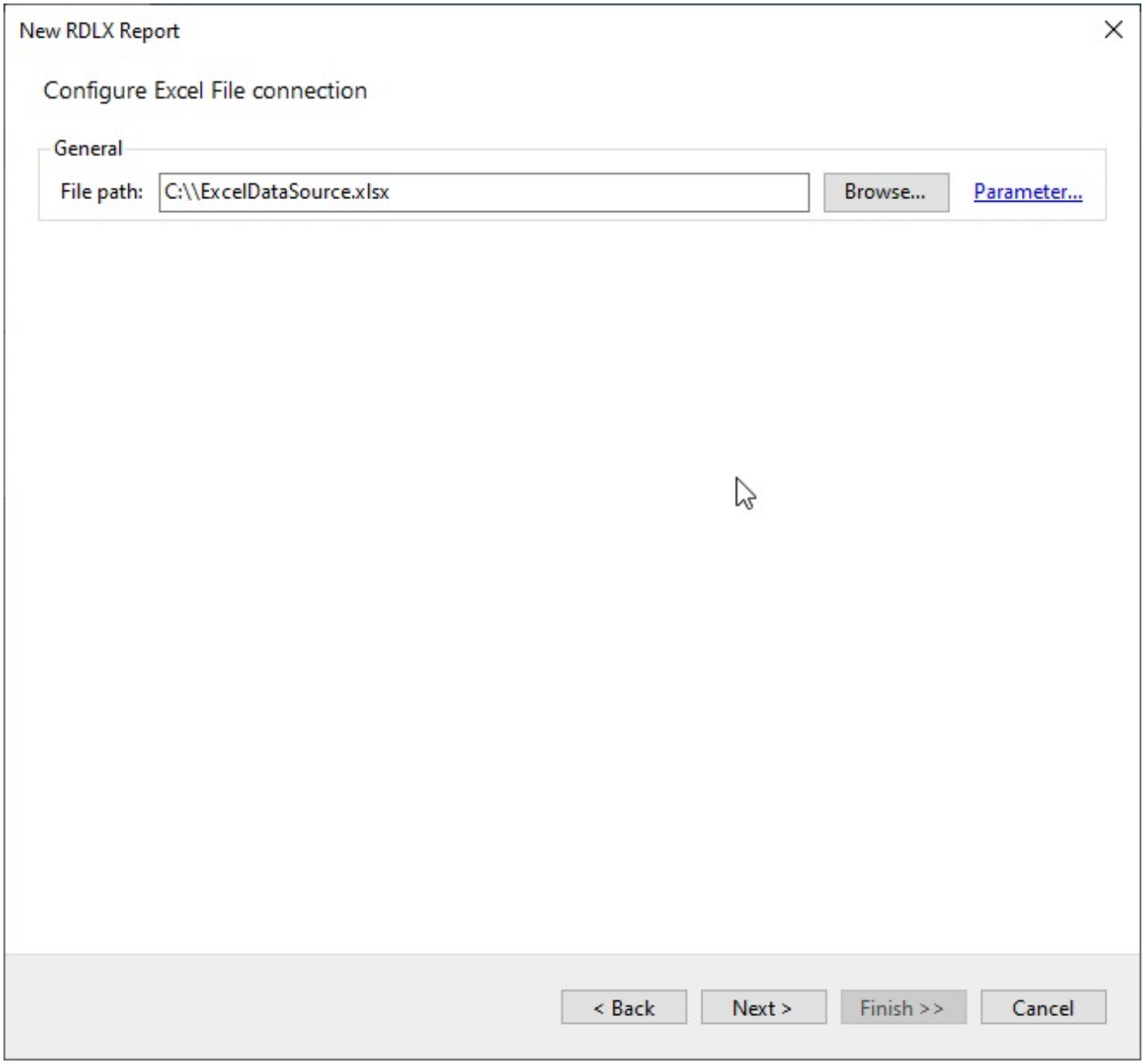
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page and click **Next**.



3. Select the Data Source Type as **Excel** and click **Next**.

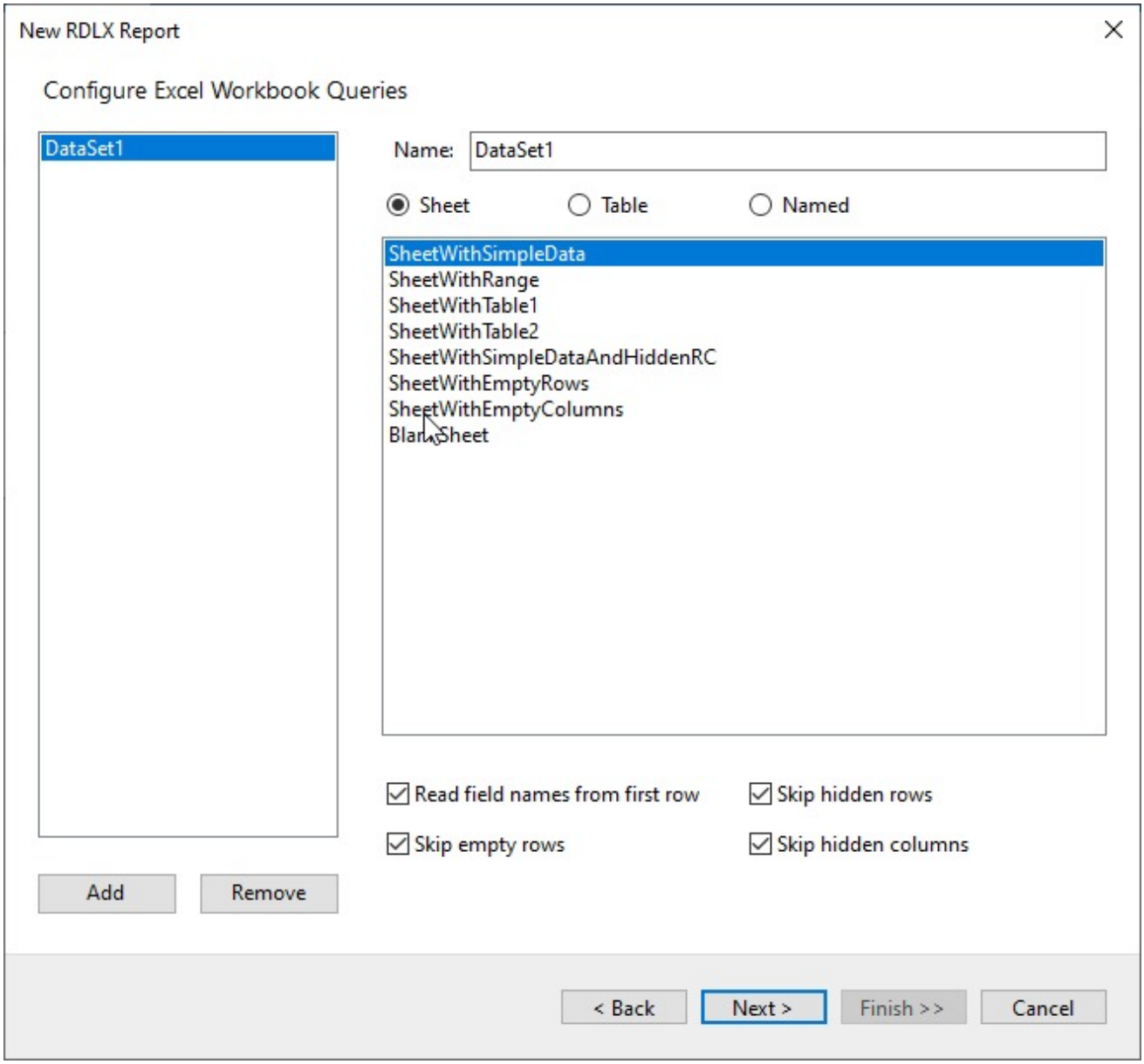


4. To specify the **File Path**, click the **Browse** button and navigate to the desired file on your system.

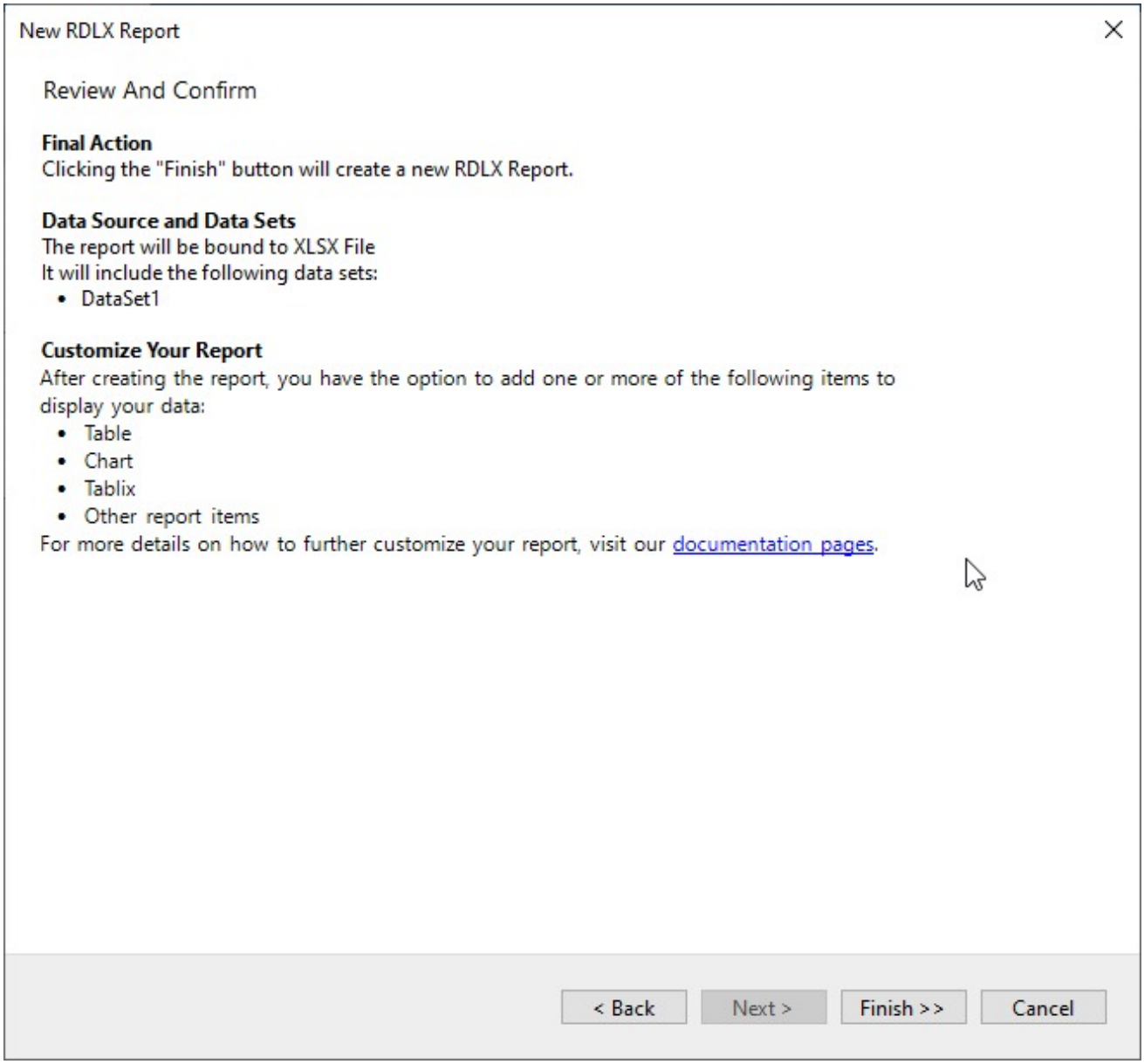


5. To specify the runtime connection values, click **Parameter** (or **Insert Parameter** for queries) to open the **Parameters** dialog. Then click the **Add** button to add a new parameter, or select the existing parameter and specify the below details:
    - **Name**: Specify the name of the parameter.
    - **Type**: Select the value type (string by default) from the drop-down list.
    - **Testing Value**: Specify the runtime value for the connection properties.
    - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.

6. To configure the dataset, select the desired **Sheet** , **Table**, and **Named** (range) of the Excel file, and click **Next**.

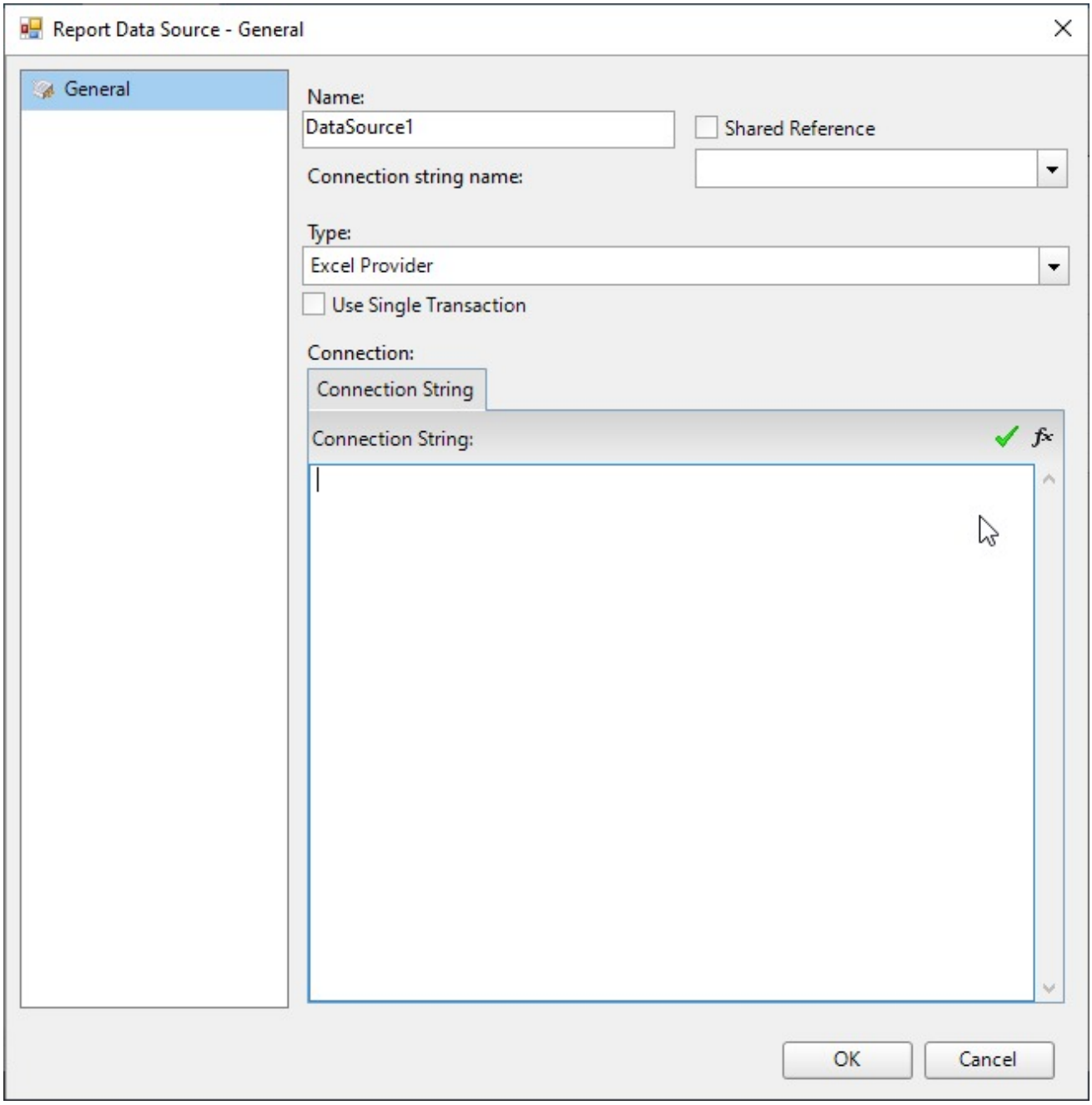


7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the Excel data source.



Connect to a Excel Data Source using Report Data Source dialog

1. In the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the data source name in the **Name** field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select **Excel Provider**.



4. On the same page under the **Connection** section, enter the connection string to connect to the **Excel** data source. For example:
   `Path=C:\\ExcelDataSource.xlsx;`
5. Verify the generated connection string by clicking the **Validate DataSource** icon.
6. Click **OK** to save the changes and close the **Report Data Source** dialog box.

Configuration Settings for Excel Data Source

The Excel Data Provider provides the following configuration settings in the Report Wizard dialog.

Setting	Description

Sheet	Select the sheet of the Excel file that you want to use.
Table	Select the table that you want to use.
Named	Select the desired named range of the Excel file.
Read field names from the first row	Select the check box to use the first row from the Excel table as field names.
Skip hidden rows	Select the check box to skip the hidden rows present in the Excel table.
Skip empty rows	Select the check box to skip the empty rows present in the Excel table.
Skip hidden columns	Select the check box to skip the hidden columns present in the Excel table.

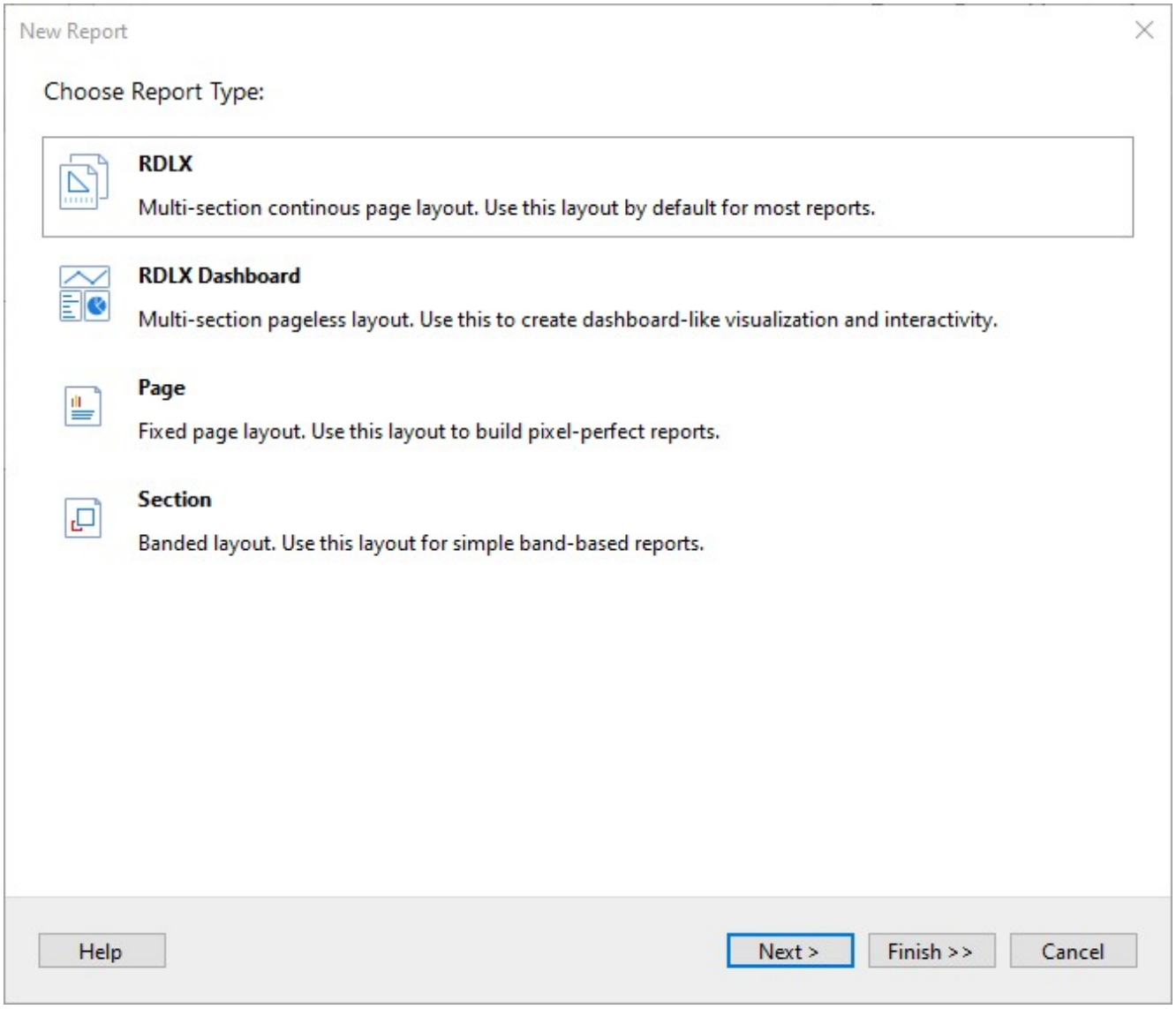
JSON API

This article explains connecting a Page or an RDLX report to a JSON API/Web API data source. You can connect to this data source while creating a new report (via report wizard) or using report explorer (via report data source dialog).

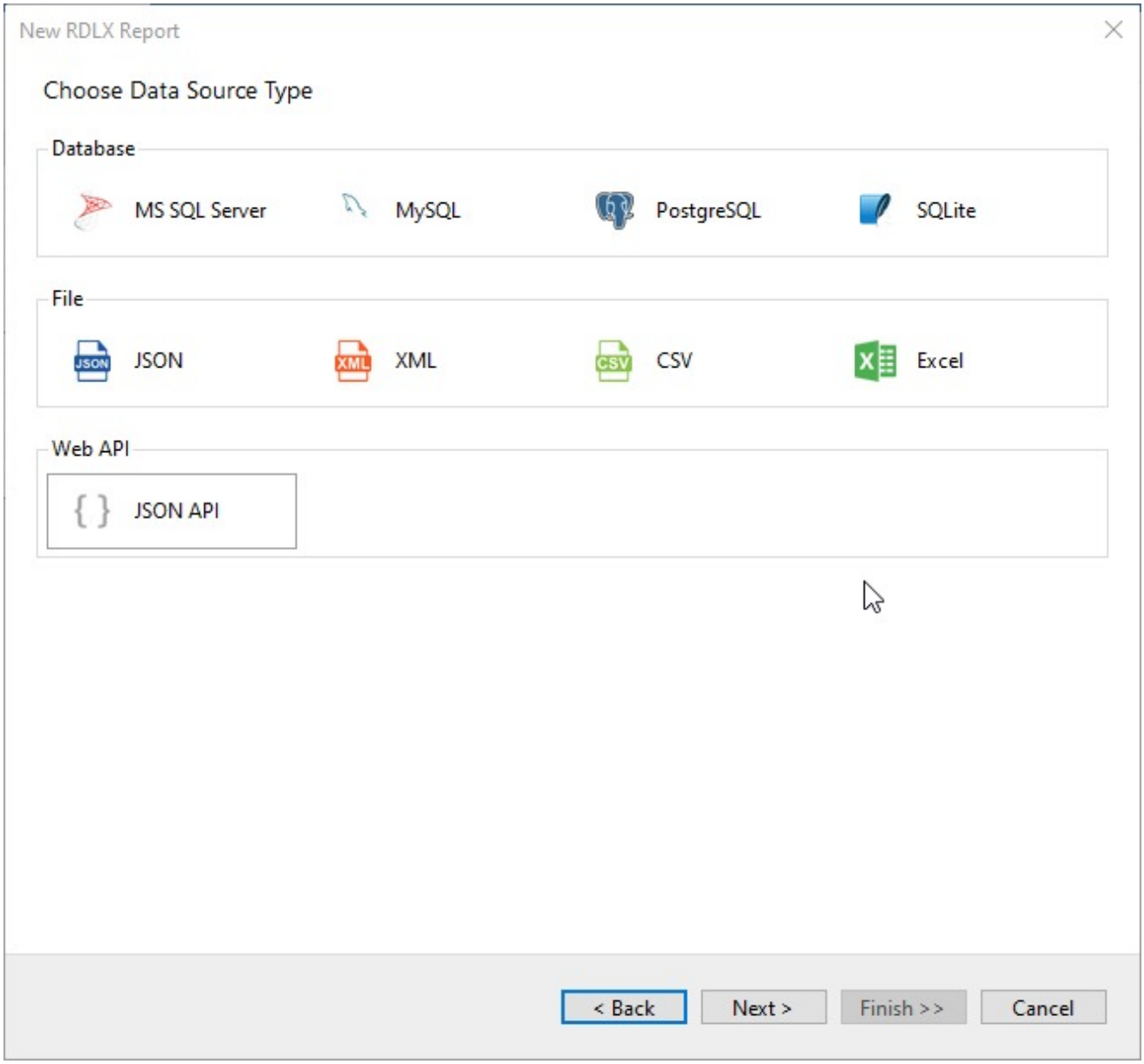
Connect to JSON API Data Source using Report Wizard

The steps to connect to the JSON API data source are:

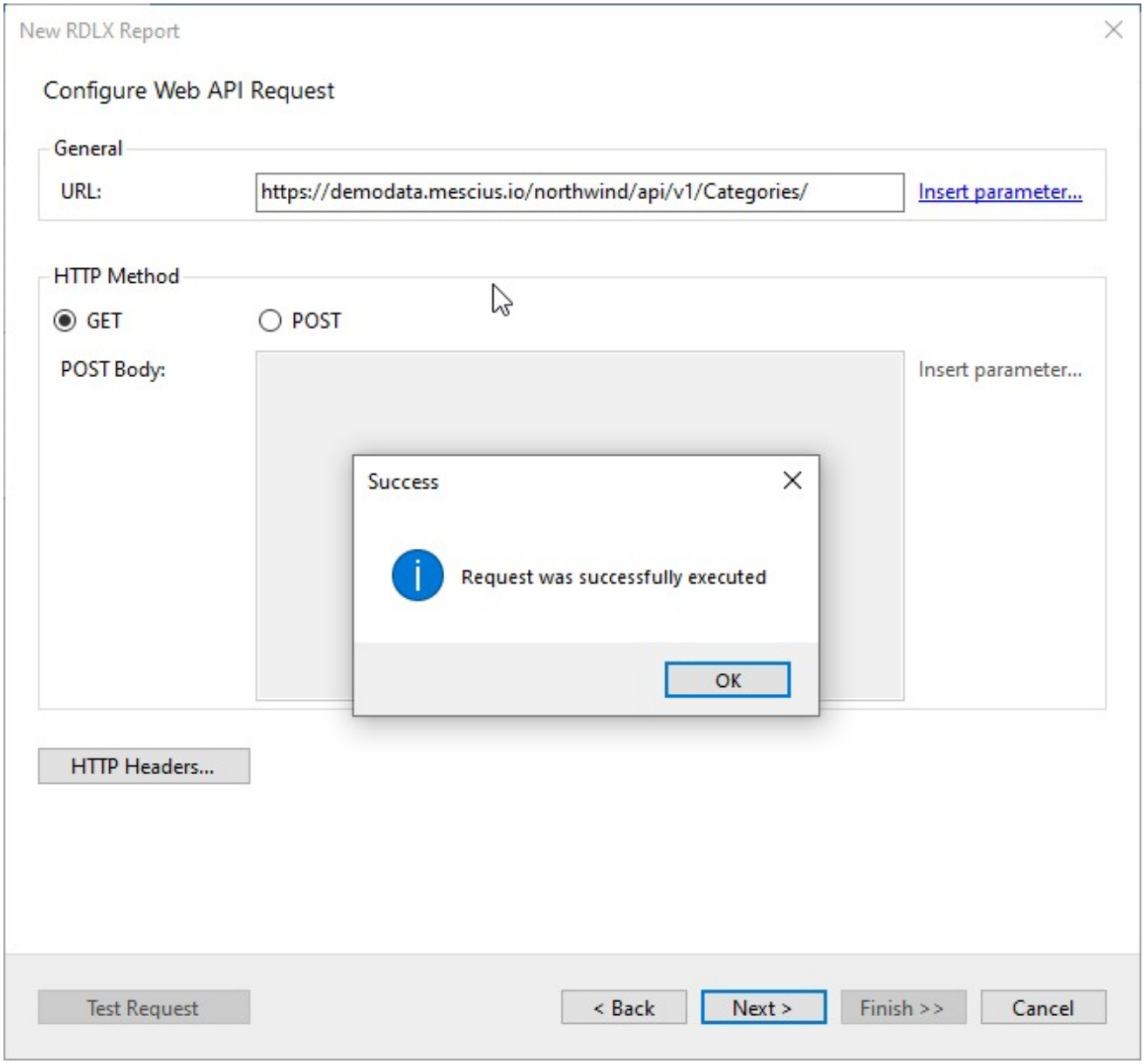
1. Create a New Report.
2. In the **New Report** dialog, choose the Report Type as RDLX, RDLX Dashboard, or Page report and click **Next**.



3. Select the Data Source Type as **JSON API** and click Next.

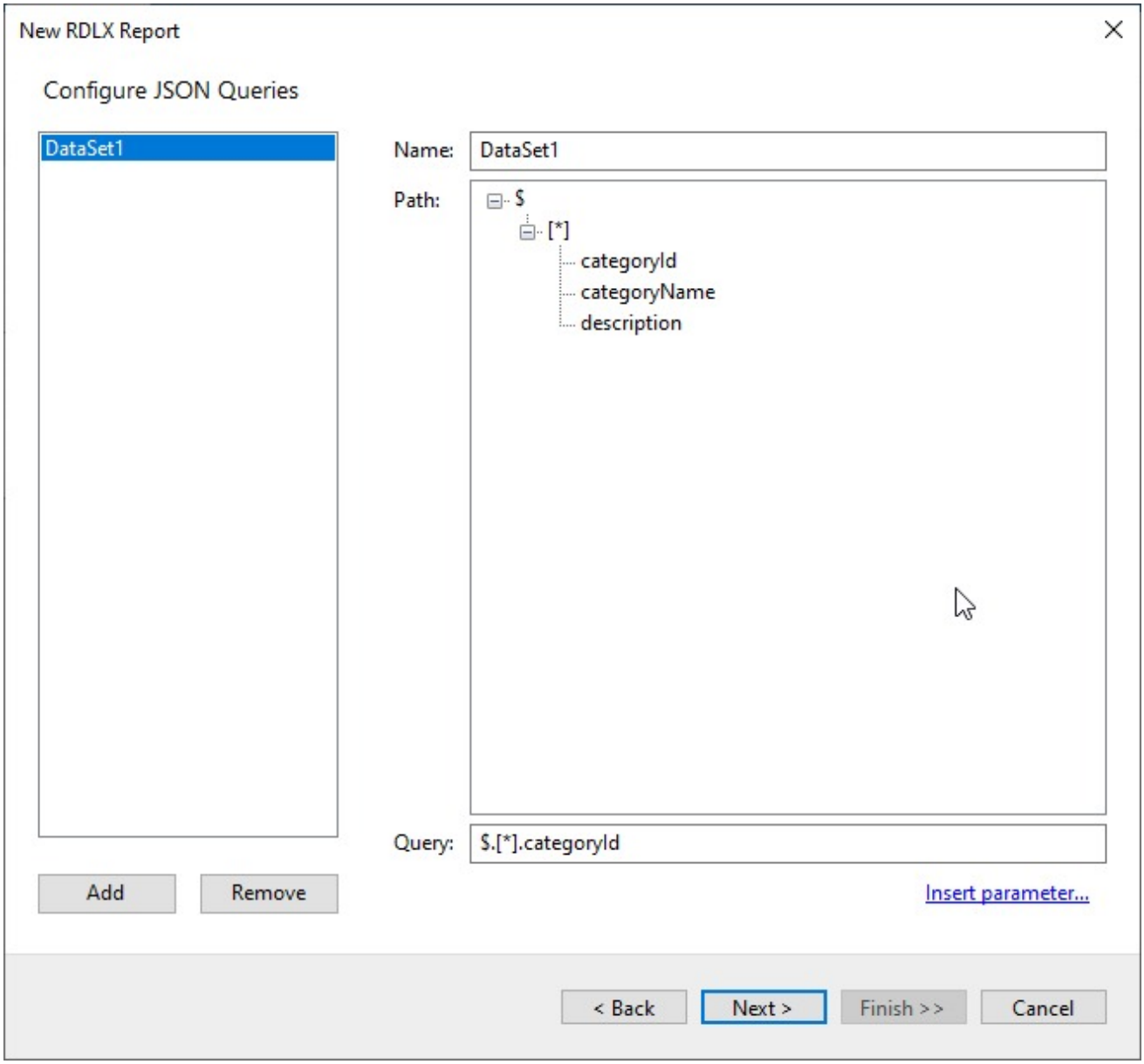


4. To configure Web API request, select **GET** from the **HTTP Method** section and enter the URL in the **General** section or you use the **POST** HTTP method to connect to the server.

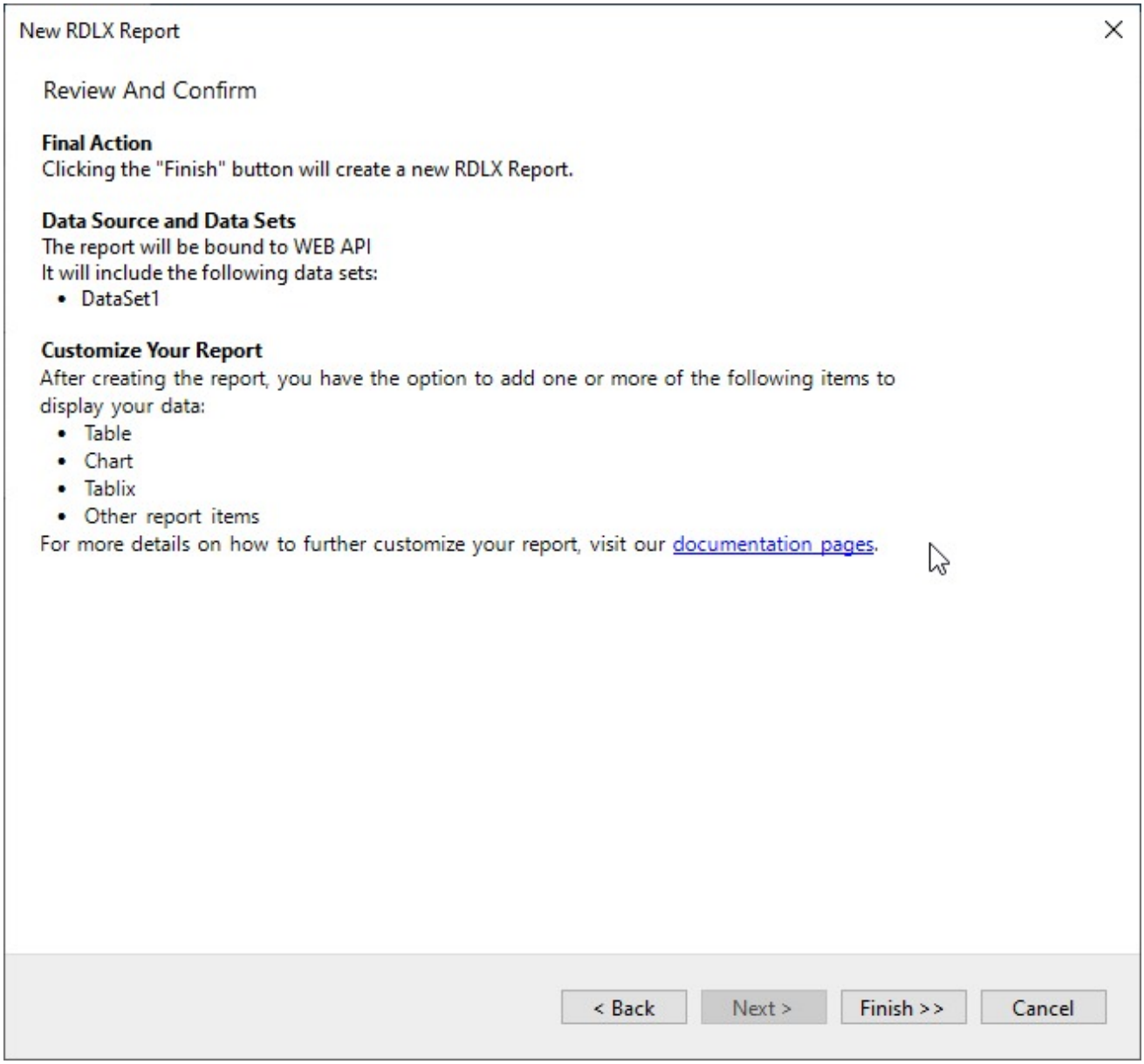


5. To specify the runtime connection values, click **Insert Parameter** to open the **Parameters** dialog. Then click the **Add** button to add a new parameter, or select the existing parameter and specify the below details:
    - **Name**: Specify the name of the parameter.
    - **Type**: Select the value type (string by default) from the drop-down list.
    - **Testing Value**: Specify the runtime value for the connection properties.
    - **Input Source**: Select **Interactive** for non-hidden parameters and **Programmatic** for hidden parameters from the drop-down list.

6. Then click the **Next** option and configure the dataset by adding a valid query. Select the node from the data tree in the **Path** section to generate the query. The resulting query is displayed in the **Query** field.



7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the JSON API/Web API data source.



Dataset Provider

The data source and data set for DataSet Provider and Object Provider data types can be set at run time. For more information, see Bind a Page/RDLX Report to Data.

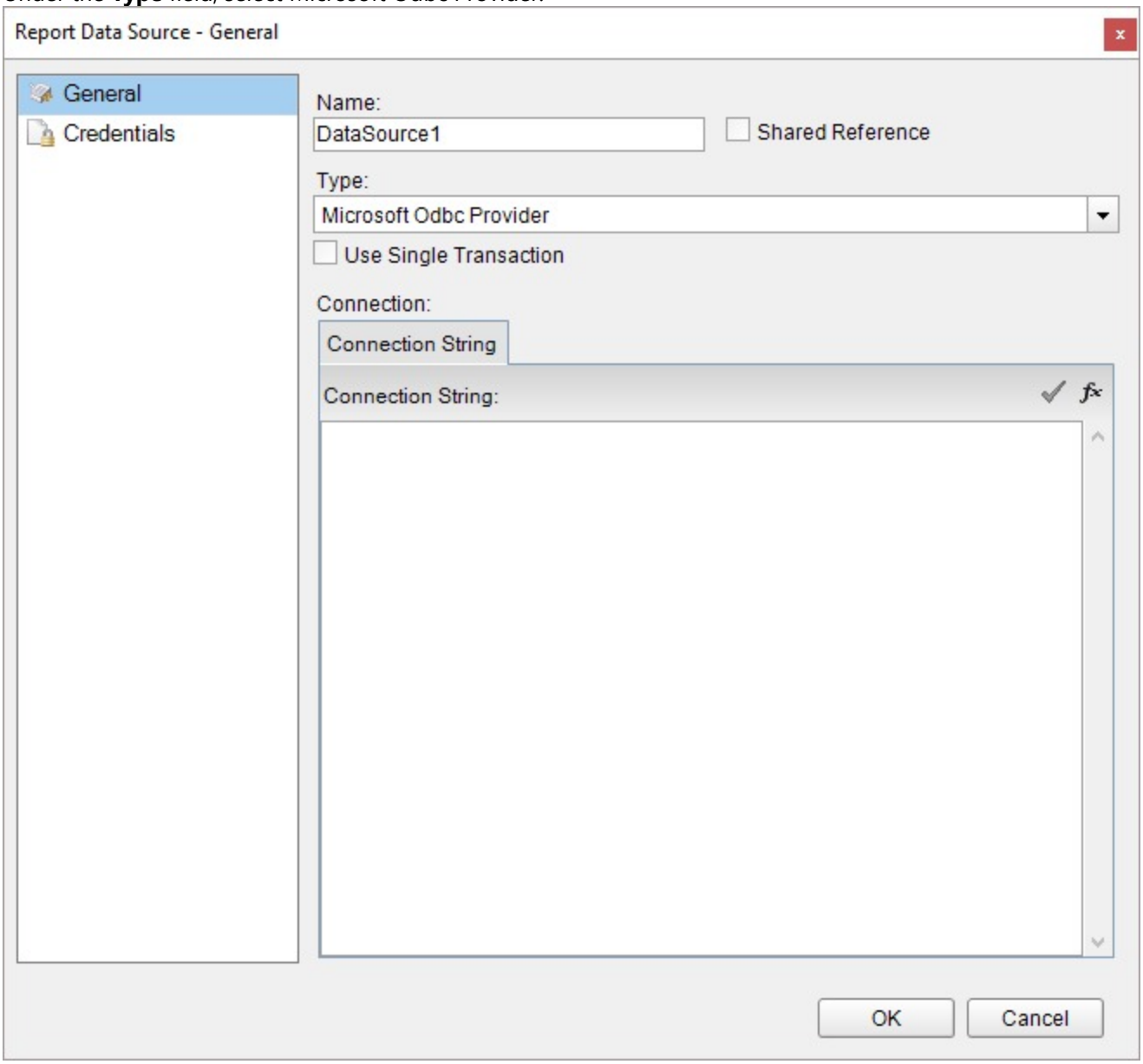
Microsoft ODBC Provider

This article explains connecting a Page or an RDLX report to an ODBC data source.

Note: The ODBC model depends on the installed drivers.

Connect to an ODBC Data Source

1. In the designer, go to the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and then select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the name of the data source in the Name field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select Microsoft Odbc Provider.



4. On the same page under the Connection section, enter the connection string to connect to an ODBC data source. The following sample connection string specifies the type of the ODBC Driver along with location of the file required for an ODBC data source connection.
   For example, you can connect to the NWIND.mdb sample data source which can be downloaded from GitHub.

   ```
   Driver=Microsoft Access Driver (*.mdb);Dbq=C:\NWIND.mdb;
   ```

   For more information about the **Credentials** page in the Report Data Source dialog, see Report Data Source

dialog properties.
5. Click **OK** to save the changes and close the **Report Data Source** dialog box.

Connect to PostgreSQL Database

This article demonstrates how to connect to a PostgreSQL database with the ODBC driver. The following steps assume that you have the required configuration details to connect to a PostgreSQL database.

You will need to install the PostgreSQL ODBC driver that will allow you to connect with PostgreSQL data.

Install the PostgreSQL ODBC driver

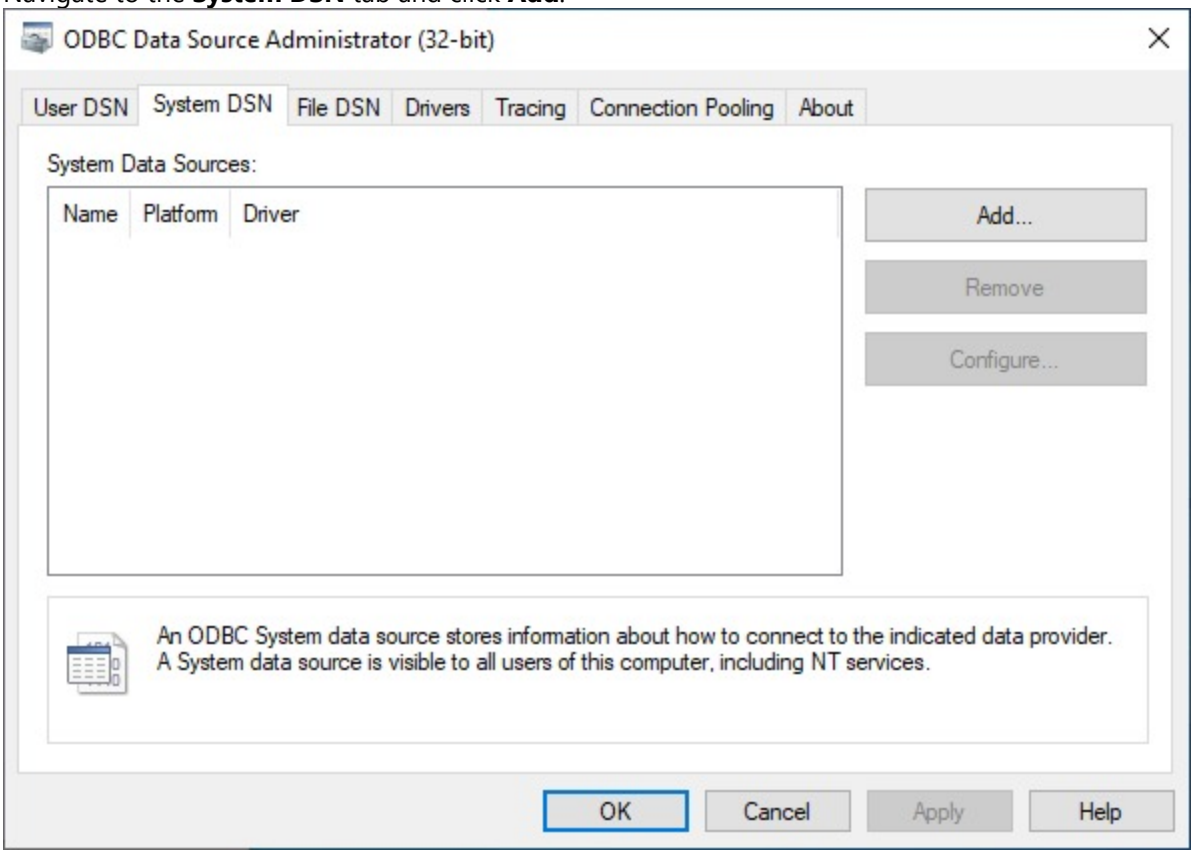
1. Download the latest 32-bit version of Postgres ODBC driver.

> **Note:** Since the ActiveReports Designer application is a 32-bit application, you should use the 32-bit PostgreSQL ODBC driver. In case you are using the ActiveReports Visual Studio Integrated Designer in your .NET application, depending on the target platform x86 or x64, you need to accordingly download a 32-bit or a 64-bit PostgreSQL ODBC driver.

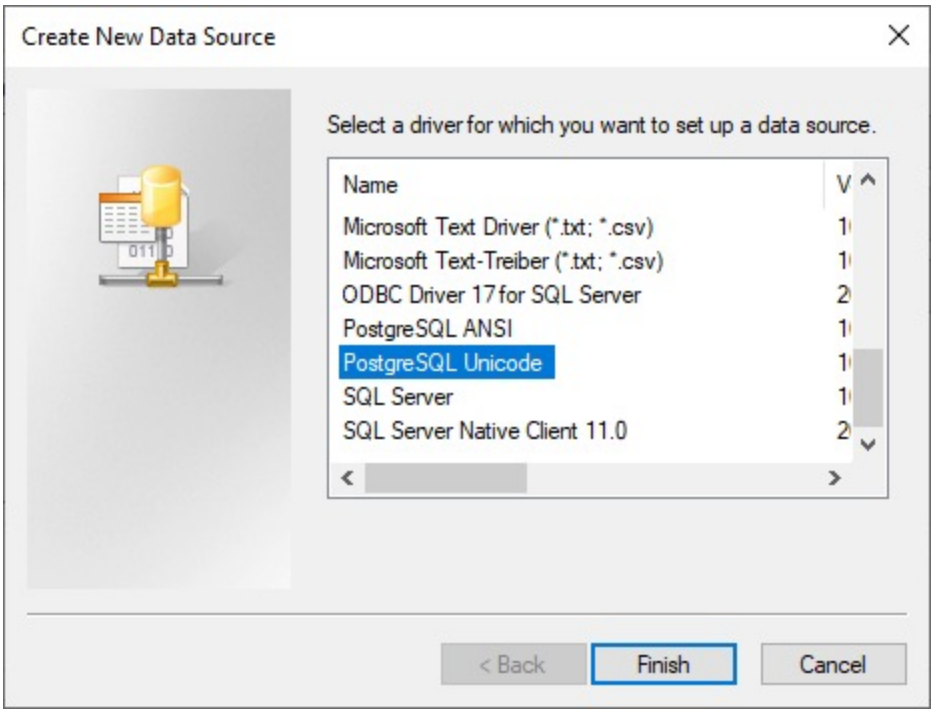
2. Extract the files and run the .msi file.

Set up the ODBC data source for PostgreSQL

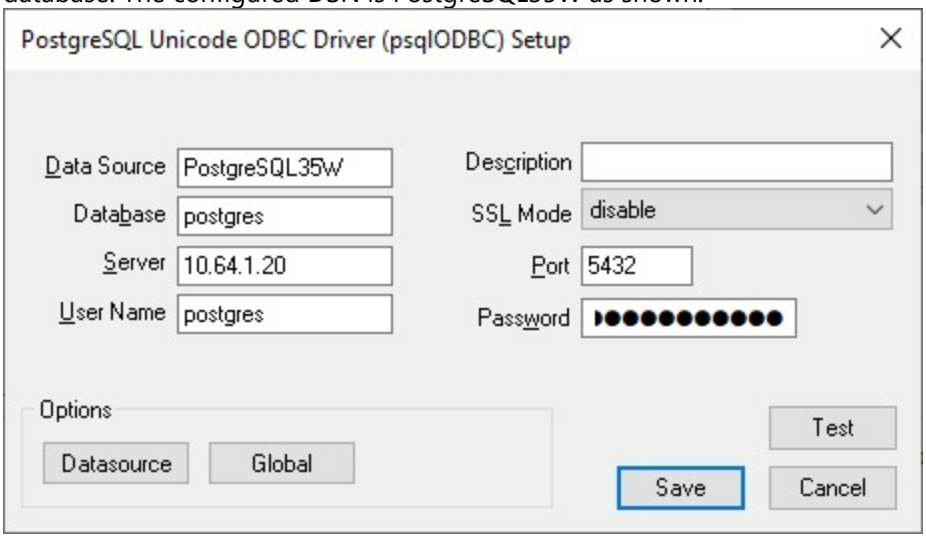
3. Open the **ODBC Data Source** application. An **ODBC Data Source Administrator** dialog box appears.
4. Navigate to the **System DSN** tab and click **Add**.



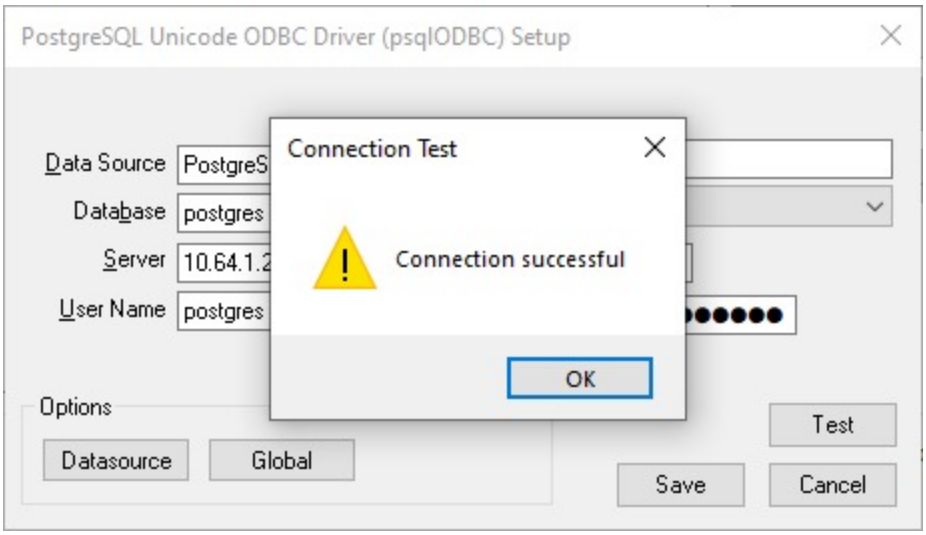
5. In the **Create New Data Source** dialog box that displays, select the **PostgreSQL Unicode** driver from the list, and click **Finish**.



6. Fill in the configuration details required for the PostgreSQL including the server, port, username, password, and database. The configured DSN is PostgreSQL35W as shown.



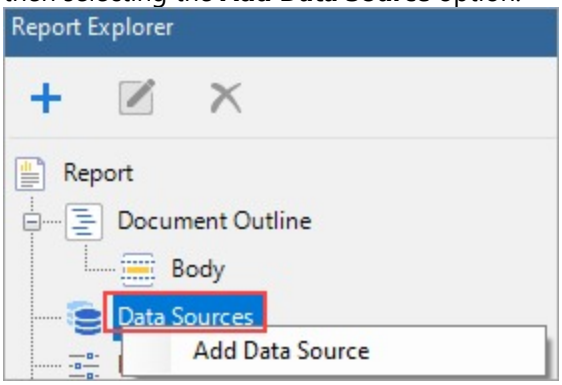
7. Click **Test** to verify the data source connection. The following dialog box will appear on a successful connection.



8. Click **OK** to save the changes.

Connect to PostgreSQL with the ODBC connection string

8. Open the ActiveReports Designer application. By default, an RDLX report is created.
   As you create a new report, **Report Data Source dialog** appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the **Report Explorer** and then selecting the **Add Data Source** option.



9. In the **Report Data Source** dialog, select the **General** page and enter the name of the data source in the **Name** field.
10. Under the **Type** field, select Microsoft Odbc Provider.
11. On the same page, under the **Connection** section, enter the connection string to connect to the data source.
    The format of the connection string is as follows -

Connection String
`DSN=PostgreSQL35W;Server=<server>;Port=<port>;User Id=<user id>;Password=<password>;Database=<database>`

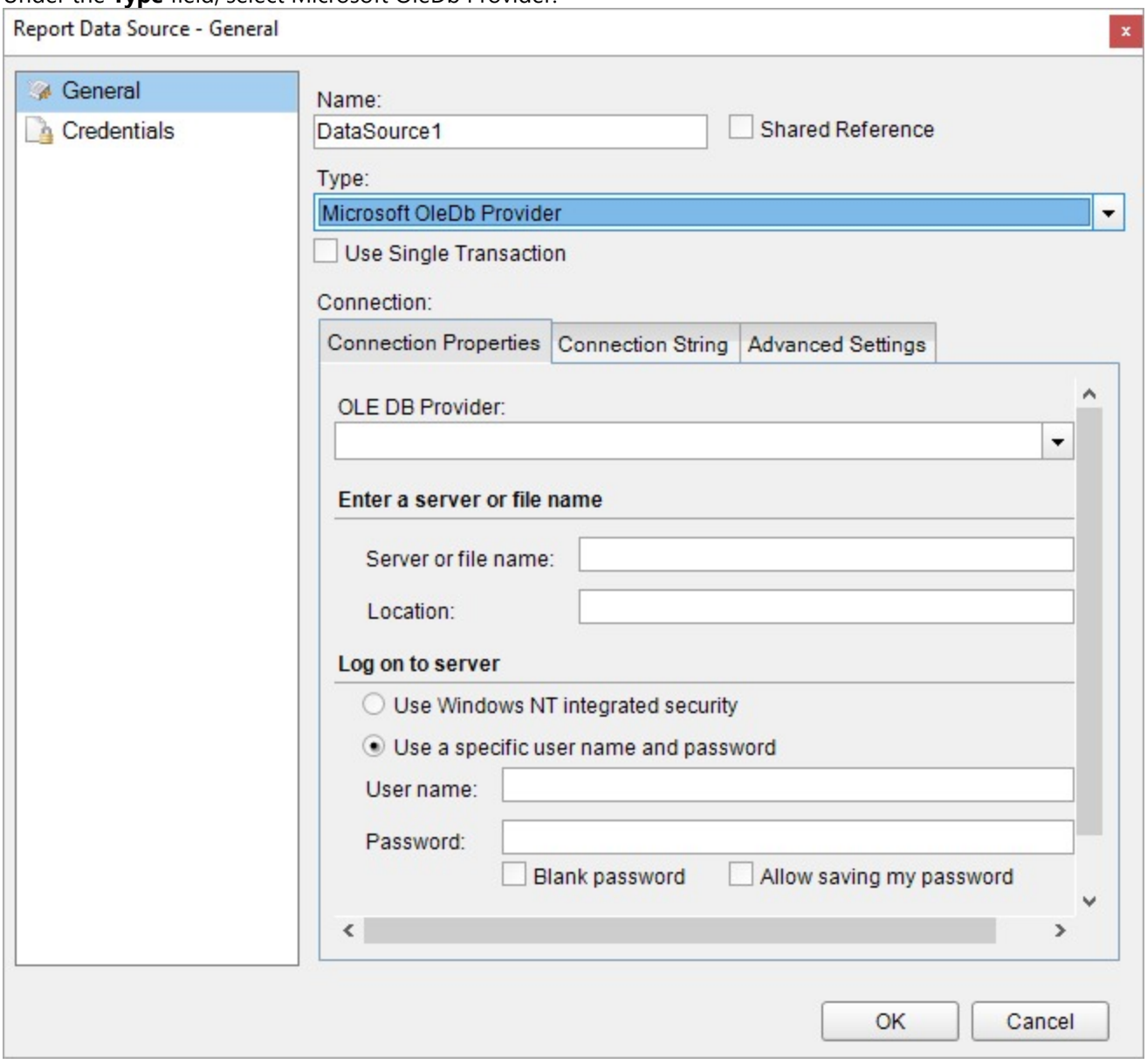
Microsoft OLEDB Provider

This article explains connecting a Page or an RDLX report to an OLEDB data source.

> **Note**: The OLEDB model depends on the installed drivers.

Connect to an OLEDB Data Source

1. In the designer, go to the **Report Explorer**, right-click the **Data Sources** node and select the **Add Data Source** option or click the **Add** button and then select the **Data Source** option.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the name of the data source in the Name field. By default, the data source name is set to DataSource1. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Under the **Type** field, select Microsoft OleDb Provider.



4. In the **Connection Properties** tab, specify the OLE DB Provider you want to use to connect to the data source.

5. To specify the **Path** of the file, click the **Browse** button and navigate to the desired folder on your system. For example, you can connect to the 'NWIND.mdb' sample data source which can be downloaded from GitHub.

   The **Connection String** tab displays the generated connection string as shown below:

```
provider=Microsoft.Jet.OLEDB.4.0;data source=C:\NWIND.mdb;
```

For more information, see the next section on 'Configuration Settings for OLEDB Data Source'.

6. Click the **Validate DataSource** icon to verify the connection string. See Report Data Source dialog properties to learn about the properties available in the **Credentials** page of the dialog.
7. Click **OK** to close the **Report Data Source** dialog. Your report is now connected to the OLEDB data source successfully.

Configuration Settings for OLEDB Data Source

The OLEDB Data Provider provides the following configuration settings under the Connection section in the **Report Data Source** dialog. Based on the defined configuration settings, the OLEDB connection string is generated in the **Connection String** tab.

The **Connection Properties** tab gives access to properties specific to the following data types.

Setting	Description	Example
OLE DB Provider	Choose which OLE DB Provider you want to use for the data source connection. You need to be selective in the choice of the provider. For example, Microsoft.Jet.OLEDB.4.0 is not supported in 64 bit OS while Microsoft.ACE.OLEDB.12.0 is supported. See Troubleshooting article if an exception occurs on previewing reports connecting Microsoft Access OLE DB provider in a 64-bit system.	Microsoft.Jet.OLEDB.4.0
Enter a server or file name	Enter a server or a file name along with its location.	C://NWIND.mdb
Log on to server	Select whether to use Windows NT integrated security or a specific user name and password.	Check the Use Windows NT integrated security option

Advanced Settings for Provider-Specific Connection Parameters

The **Advanced Settings** tab gives access to the Microsoft Jet OLEDB provider-specific connection parameters.

Setting	Description
Jet OLEDB: Compact Reclaimed Space Amount	Indicates an estimate of the amount of space, in bytes, that can be reclaimed by compacting the database. This value is only valid after a database connection has been established.
Jet OLEDB: Connection Control	Indicates whether users can connect to the database.
Jet OLEDB: Create System Database	Indicates whether to create a system database when creating a new data source.
Jet OLEDB:	Indicates the locking mode for this database. The first user to open the database determines

Database Locking Mode	what mode to use when the database is open.
Jet OLEDB: Database Password	Indicates the database password.
Jet OLEDB: Don't Copy Locale on Compact	Indicates whether the Jet should copy locale information when compacting a database.
Jet OLEDB: Encrypt Database	Indicates whether a compacted database should be encrypted. If this property is not set, the compacted database will be encrypted if the original database was encrypted.
Jet OLEDB: Engine Type	Indicates the storage engine to access the current data store.
Jet OLEDB: Exclusive Async Delay	Indicates the maximum length of time, in milliseconds, that the Jet can delay asynchronous writes to disk when the database is opened exclusively. This property is ignored unless Jet OLEDB: Flush Transaction Timeout is set to 0.
Jet OLEDB: Flush Transaction Timeout	Indicates the amount of time before data stored in a cache for asynchronous writing is actually written to disk. This setting overrides the values for Jet OLEDB:Shared Async Delay and jet OLEDB: Exclusive Async Delay.
Jet OLEDB: Global Bulk Transactions	Indicates whether the SQL bulk transactions are transacted.
Jet OLEDB: Global Partial Bulk Ops	Indicates the password to open the database.
Jet OLEDB: Implicit Commit Sync	Indicates whether the changes made in internal implicit transactions are written in synchronous or asynchronous mode.
Jet OLEDB: Lock Delay	Indicates the number of milliseconds before attempting to acquire a lock after a previous attempt has failed.
Jet OLEDB: Lock Retry	Indicates the frequency of attempts to access a locked page.
Jet OLEDB: Max Buffer Size	Indicates the maximum amount of memory, in kilobytes, the Jet can use before it starts flushing changes to disk.
Jet OLEDB: MaxLocksPerFile	Indicates the maximum number of locks the Jet can place on a database. The default value is 9500.
Jet OLEDB: New Database Password	Indicates the new password for this database. The old password is stored in Jet OLEDB: Database Password.
Jet OLEDB: ODBC Command Time Out	Indicates the number of milliseconds before a remote ODBC query from the Jet will timeout.
Jet OLEDB: Page Locks to Table	Indicates how many pages to lock within a transaction before the Jet attempts to promote the lock to a table lock. If this value is 0, then the lock is never promoted.

Lock	
Jet OLEDB: Page Timeout	Indicates the number of milliseconds before the Jet will check if its cache is out of date with the database file.
Jet OLEDB: Recycle Long-Valued Pages	Indicates whether the Jet should aggressively try to reclaim BLOB pages when they are freed.
Jet OLEDB: Registry Path	Indicates the Windows registry key that contains values for the Jet database engine.
Jet OLEDB: Reset ISAM Stats	Indicates whether the schema Recordset DBSCHEMA_JETOLEDB_ISAMSTATS should reset its performance counters after returning performance information.
Jet OLEDB: Shared Async Delay	Indicates the maximum amount of time, in milliseconds, the Jet can delay asynchronous writes to disk when the database is opened in the multi-user mode.
Jet OLEDB: System Database	Indicates the path and file name for the workgroup information file (system database).
Jet OLEDB: Transaction Commit Mode	Indicates whether the Jet writes data to disk synchronously or asynchronously when a transaction is committed.
Jet OLEDB: User Commit Sync	Indicates whether changes made in transactions are written in the synchronous or the asynchronous mode.

Add Dataset

After connecting to a data source in a Page/RDLX report, you need to add a dataset. It is the dataset that contains the data to which a report connects. This article demonstrates the steps to add a dataset in a Page/RDLX report, and the various configuration settings available in the **Dataset** dialog box.

A dataset fetches data from the data source to display in a report. The **DataSet** dialog is where you provide a command type and query string and choose other options for your dataset.

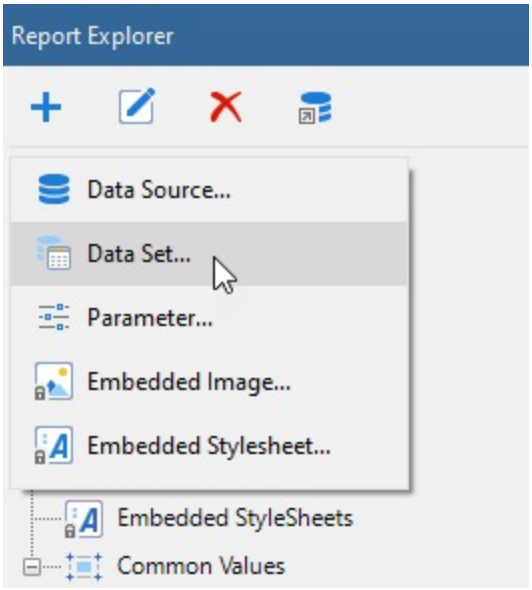
You can also control the timeout period and other data options, and add fields, parameters, and filters to fetch the data you need. Once you have added a dataset, the dataset fields appear under the Data Source node in the Report Explorer. You can add multiple datasets for a data source. For example, you can use multiple data sets in a report when nesting data regions such as Tablix, List, Chart, BandedList, Table, and Sparkline.

Furthermore, ActiveReports provides a Visual Query Designer for Microsoft SQL Client and Microsoft OLEDB providers and Query Designers for JSON and XML providers, which allow the users to build the dataset queries easily.

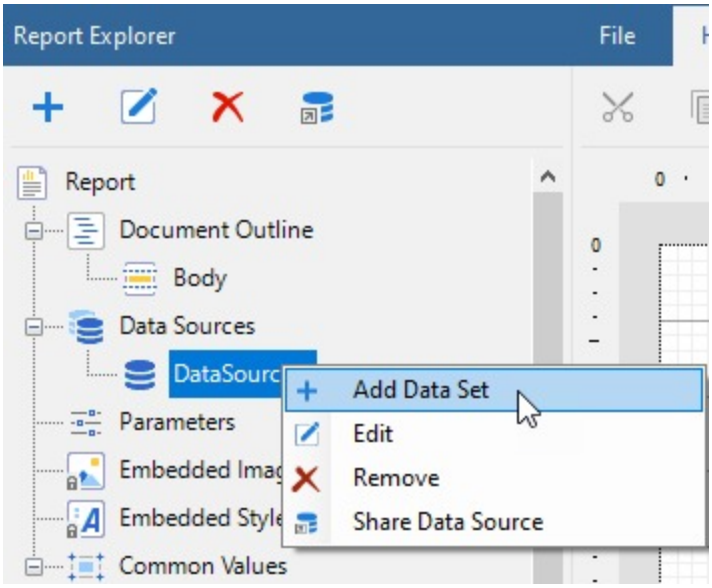
> **Note**: The dataset for a CSV data source is automatically added. The default name of the data set is the name of the CSV file selected. You can later modify the dataset through the **Edit** option in the context menu.

Add a dataset

1. In ActiveReports designer, go to the **Report Explorer**.
    - With the Data Source node (like DataSource1) selected, click the **Add** icon on the top left and select **Data Set**.



- Right-click an existing data source and select **Add Data Set**.



2. In the **DataSet** dialog box that appears, select the **General** page and enter the name of the dataset. By default, the dataset name is set to DataSet1. This name appears as a child node to the Data Source node in the Report Explorer.
3. Navigate to the **Query** tab. Based on the chosen command type, enter the query to retrieve the data from a database. You can verify the query by clicking the **Validate DataSet** icon . See **DataSet dialog properties** for setting up the dataset.

   > The availability of a specific query designer depends on the report's data source connection. For more information on query designers, see articles Query Builder in JSON and XML Providers and Query Builder in Microsoft SQL Client and OLEDB Providers.

4. Click **OK** to save the changes.

DataSet dialog properties

The DataSet dialog provides the following pages where you can set the dataset properties.

General

The **General** page of the DataSet dialog is where you can set the name of the dataset.

Name: Enter a name for the dataset. By default, the name is set to DataSet1. The name of the dataset appears in the tree view of the Report Explorer. It is also used to call the dataset in code so it should be unique within the report.

Query

The **Query** page of the DataSet dialog is where you set the SQL query, stored procedure, or table to define the data you want to fetch in the dataset of your report.

> **Note:** The **Query** page is unavailable for the CSV data provider.

- **Command type:** You can choose from the three enumerated command types.
    - Select **Text** if you want to write an SQL query to retrieve data.
    - Select **StoredProcedure** if you want to use a stored procedure. This option is not available in SQLite, JSON, and XML data providers. See Stored Procedure as Dataset topic for more information.
    - Select **TableDirect** if you want to return all rows and columns from one or more tables. This option is available only in OLEDB data provider.
- **Query:** Based on the command type you select above, you can set the query string in this field.

    > **Note:**
    >
    > - If you select the **TableDirect** command type, you may need to use escape characters or qualifying characters in case any of the table names include special characters.
    > - Specify the calculated index for arrays in a JSONPath expression in the following ways:
    >   - To obtain the last entry in an array, use `-1:` square brackets. For example, use $..book[-1:].
    >   - To obtain evaluated expressions correctly, the field names in square brackets should be in single quotes. For example, use $..book[0]['category','author'].

    To create multiple datasets based on the JSON data provider, check the **Select multiple nodes** option in the JSON Query Builder.

    For information on writing an SQL query that you can use for SQLite (Custom Data Provider), see the SQLite Tutorial. You will note that there are some differences in writing a standard SQL query and an SQL query for SQLite.
- **Timeout:** Specifies the number of seconds you want the report server to wait for the query to return the data before it stops trying.

Options

The **Options** page is where you select the following options available for the dataset.

> **Note**: The **Options** page is unavailable in case of CSV, JSON, and XML data providers.

- **CaseSensitivity:** Set this value to Auto, True, or False to indicate whether to make distinctions between upper and lower case letters. Auto, the default value, causes the report server to get the value from the data provider. If the data provider does not set the value, the report runs without case sensitivity.
- **Collation:** Choose from Default or a country from the list to indicate which collation sequence to use to sort data. The Default value causes the report server to get the value from the data provider. If the data provider does not set the value, the report uses the server locale. This is important with international data, as the sort order for different languages can be different from the machine sort.
- **KanaTypeSensitivity:** Set this value to Auto, True, or False with Japanese data to indicate whether distinctions are made between Hiragana and Katakana kana types. Auto, the default value, causes the report server to get the value from the data provider. If the data provider does not set the value, the report runs without kana type sensitivity.
- **WidthSensitivity:** Set this value to Auto, True, or False with Japanese data to indicate whether distinctions are made between single-byte (half-width) characters and double-byte (full-width) characters. Auto, the default value, causes the report server to get the value from the data provider. If the data provider does not set the value, the report runs without width sensitivity.
- **AccentSensitivity:** Set this value to Auto, True, or False to indicate whether distinctions are made between accented and unaccented letters. Auto, the default value, causes the report server to get the value from the data provider. If the data provider does not set the value, the report runs without accent sensitivity.

Fields

The **Fields** page of the DataSet dialog populates automatically for OleDb, ODBC, SQL, JSON, CSV, and XML data providers. To see a list of fields in the **Name** and **Value** columns of the **Fields** page, enter a valid query, table name, or stored procedure on the **Query** page.

You can edit the populated fields, delete them by using the **Remove (X)** icon, or add new ones by using the **Add (+)** icon above the Fields list. Any fields you add in this list show up in the **Report Explorer** tree view and you can drag and drop them onto the design surface. The field name must be unique within the dataset.

When working with **Fields**, the meaning of the value varies depending on the data source type. In most cases, this is simply the name of the field. The following list describes the meaning of the field value and gives some examples of how to use the value.

- In **SQL** and **OleDb** data sources, the field value is the name of a field returned by the query. For example,
  `OrderQuantity`
  `FirstName`
- In **Dataset** data source, the field value can be the name of a field in the DataTable specified by the query. You can also use DataRelations in a DataSet, specify the name of the relation followed by a period and then the name of a field in the related DataTable. For example,
  `Quantity`
  `OrdersToOrderDetails.CustomerID`
- In **XML** data source, the field value is an XPath expression that returns a value when evaluated with the query. For example,
  `Statistics/Game/TeamName`
- In **JSON** data source, the field value is a JSONPath expression that returns a value when evaluated with the query. For example,
  `$.Statistics.Game[*].TeamName`
- In **Object** data source, the field value can be the name of a property of the object contained in the collection returned by the data provider. You may also use properties available for the object returned from a property. For example,
  `Quantity`
  `Order.Customer.FirstName`
- In **CSV data source**, the field value is the name of a field returned by each column specified in the connection string. For example,
  `Path=C:\\FixedWidth.csv;Locale=en-US;TextQualifier=";ColumnsSeparator=,;RowsSeparator=\r\n;HasHeaders=True`

Parameters

The Parameters page of the Dataset dialog is where you can pass a Report Parameter into the parameter you enter in the **Query** page. You can edit the populated fields, delete them by using the **Remove (X)** icon, or add new ones by using the **Add (+)** icon above the Parameters list. For each parameter in this list, there is a **Name** and a **Value**.

> **Note**: This **Parameters** page is unavailable for the CSV, JSON, and XML data providers.

Enter a **Name** that matches the name of the Report Parameter and a **Value** for each parameter in this page. The **Name** of the parameter must be unique within the dataset.

The **Value** of parameter can be a static value or an expression referring to an object within the report. It cannot refer to a report control or field. For more information, see Adding Parameter for different data sources.

Filters

The **Filters** page of the Dataset dialog allows you to filter data after it is returned from the data source. This is useful when you have a data source (such as XML) that does not support query parameters.

You need to provide three values to add a new filter to the collection: Expression, Operator, and Value.

Expression: Enter the expression to use for evaluating whether data should be included in the group.

Operator: Select from the following operators to decide how to compare the expression to the left with the value to the right.

- **Equal** Only choose data for which the value on the left is equal to the value on the right.
- **Like** Only choose data for which the value on the left is similar to the value on the right. For more information on using the **Like** operator, see the MSDN Web site.
- **NotEqual** Only choose data for which the value on the left is not equal to the value on the right.
- **GreaterThan** Only choose data for which the value on the left is greater than the value on the right.
- **GreaterThanOrEqual** Only choose data for which the value on the left is greater than or equal to the value on the right.
- **LessThan** Only choose data for which the value on the left is less than the value on the right.
- **LessThanOrEqual** Only choose data for which the value on the left is less than or equal to the value on the right.
- **TopN** Only choose items from the value on the left which are the top number specified in the value on the right.
- **BottomN** Only choose items from the value on the left which are the bottom number specified in the value on the right.
- **TopPercent** Only choose items from the value on the left which are the top percent specified in the value on the right.
- **BottomPercent** Only choose items from the value on the left which are the bottom percent specified in the value on the right.
- **In** Only choose items from the value on the left which are in the array of values specified on the right. Selecting this operator enables the Values list at the bottom.
- **Between** Only choose items from the value on the left which fall between the pair of values you specify on the right. Selecting this operator enables two Value boxes instead of one.

Value: Enter a value to compare with the expression on the left based on the selected operator. For multiple values used with the **Between** operator, the lower two value boxes are enabled.

Values: When you choose the **In** operator, you can enter as many values as you need in this list.

Query Builder in JSON and XML Providers

ActiveReports provides query builders for some providers that assist in generating complex queries conveniently. This article demonstrates using the query builders for designing queries for JSON and XML providers.

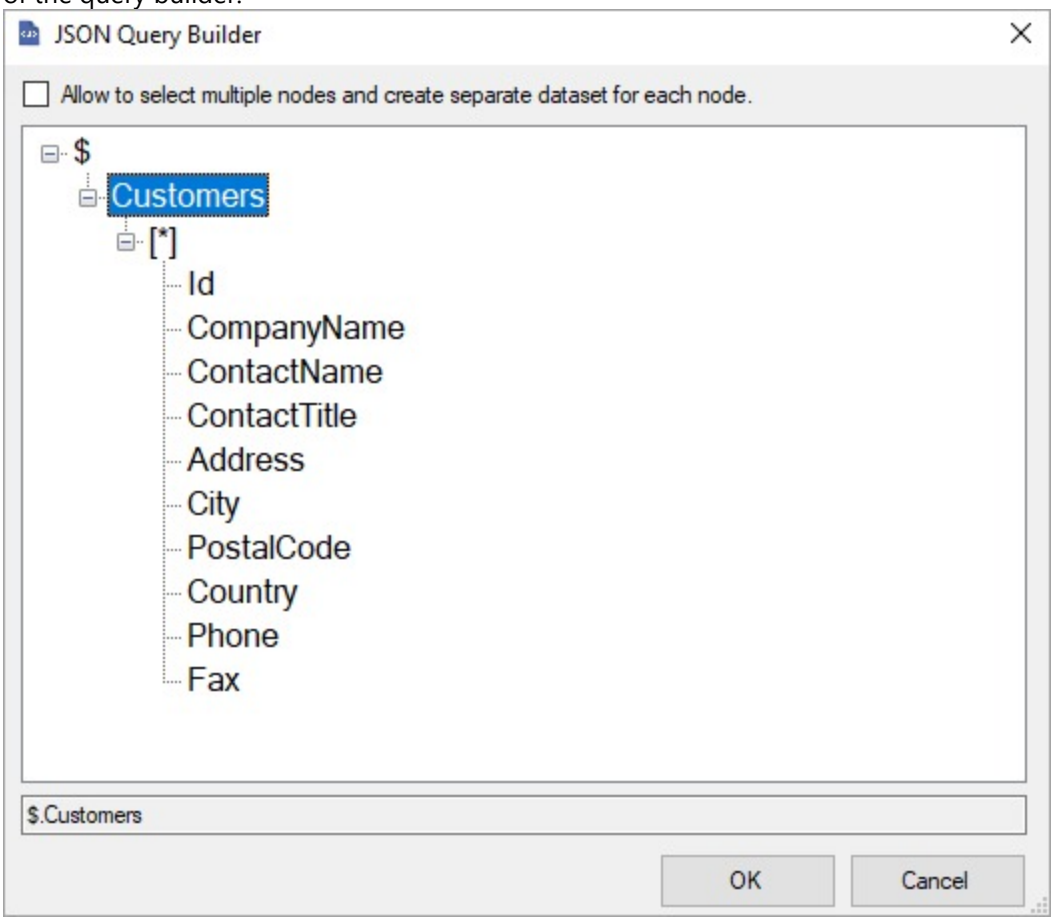
Designing queries in JSON Query Designer

Follow the below steps to build a query using the JSON Query Designer. Our report already connects to the 'customers.json' data source available on GitHub.

1. In the **Report Explorer**, right-click an existing data source and select the **Add Data Set** option.
2. In the **DataSet** dialog box that appears, select the **General** page and enter the name of the dataset. By default,

the dataset name is set to DataSet1.

3. Navigate to the **Query** page and click the **Edit with JSON Query Designer** icon to open the **JSON Query Builder**.
4. Select a node from the data tree to generate the JSON path. The resulting JSON path is displayed at the bottom of the query builder.



5. If you want to select multiple nodes and create separate datasets for each node, check the **Allow to select multiple nodes and create separate dataset for each node** option.
6. Click the **OK** button to save the query.
7. Verify the query by clicking the **Validate DataSet** icon. You can go to the **Fields** page to view the dataset fields fetched from the generated query.
8. Click the **OK** button to close the **DataSet** dialog box and complete adding a dataset.

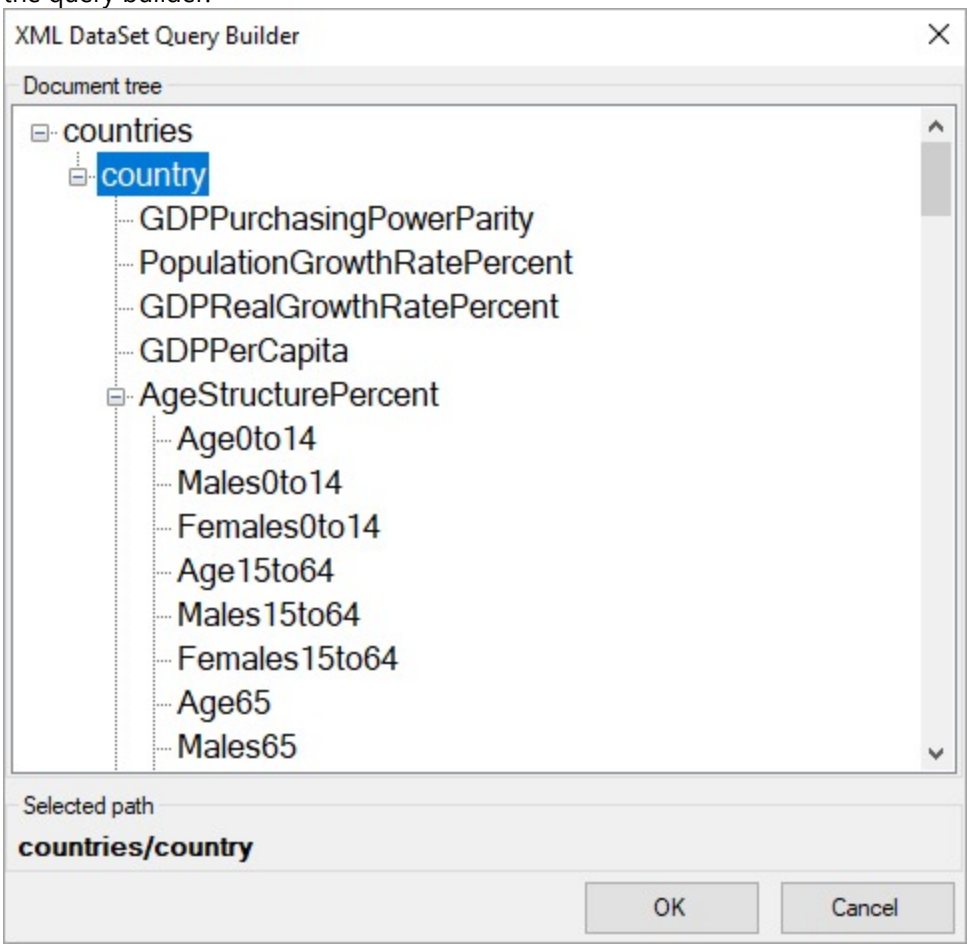
Designing queries in XML Query Designer

Follow the below steps to build a query using the XML Query Designer. Our report already connects to the 'factbook.xml' data source available on GitHub.

1. In the **Report Explorer**, right-click an existing data source and select the **Add Data Set** option.
2. In the **DataSet** dialog box that appears, select the **General** page and enter the name of the dataset. By default, the dataset name is set to DataSet1. This name appears as a child node to the Data Source node in the Report Explorer.
3. Navigate to the **Query** tab and click the **Edit with XML Query Designer** icon to open the **XML DataSet**

Query Builder.

4. Select a node from the data tree to generate the XML path. The resulting XML path is displayed at the bottom of the query builder.

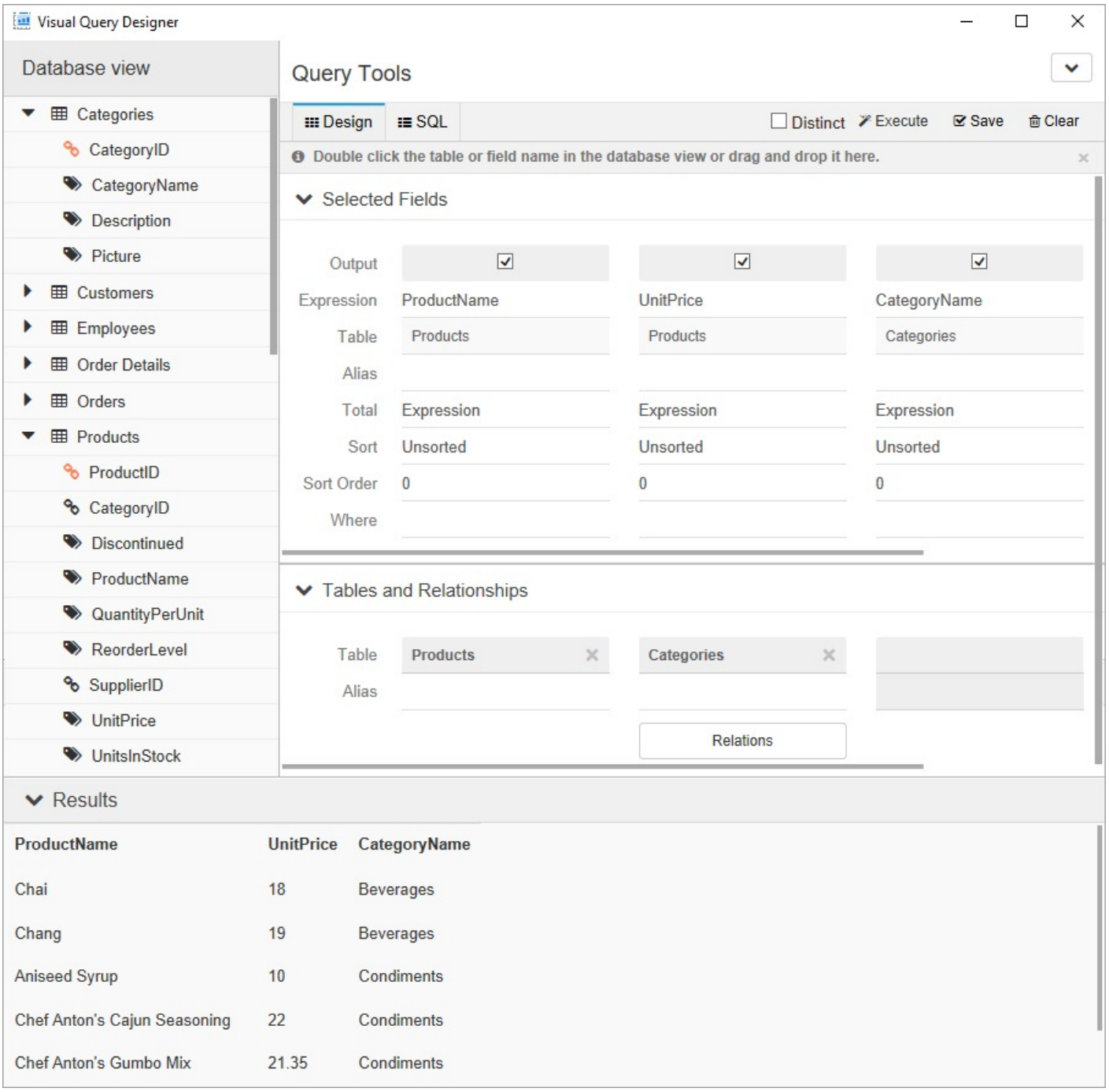


5. Click the **OK** button to save the changes.
6. Verify the query by clicking the **Validate DataSet** icon in the **DataSet** dialog box. You can go to the **Fields** page to view the dataset fields fetched from the generated query.
7. Click the **OK** button to close the **DataSet** dialog box and complete adding a dataset.

Query Builder in Microsoft SQL Client and OLEDB Providers

In a Page report or an RDLX report, you can use the **Visual Query Designer** for creating queries in Microsoft SQL Client and OLEDB data providers. The Visual Query Designer is a graphical interface that simplifies data binding by allowing users to interactively build queries and view the results. With its interactive interface, users who are unfamiliar with SQL can easily design, edit, preview, and save queries.

Visual Query Designer supports the following SQL capabilities, such as custom expressions, select fields and tables, sort and filter data, join tables, apply group and aggregate functions to fields, and set aliases for the selected fields and tables.



Interface Elements

Database view

The **Database view** appears on the left of the query builder. It displays the structure of the database including namespaces, tables, views, and columns based on the data source configuration details specified earlier. You can drag and drop or double click the elements in the Database view to add them to the **Design** tab. Alternatively, you can

double-click the crossed arrow icon on the right-hand side of each element in the Database view to add it to the Design tab. This is the first step in query building through the Visual Query Designer. A SQL query is generated as you add the database elements to the **Design** tab.

Query Tools

The Visual Query Designer provides several tools to generate a query. The **Query Tools** appear on the right of the query builder and is broadly classified into the following three major areas:

- **Design tab**
- **SQL tab**
- **Toolbar buttons**

Design tab

The **Design** tab is the area of the Visual Query Designer where you set up queries. It provides a visual interface for the SQL query you want to generate.

The Design tab consists of two panels: **Selected Fields** and **Table and Relationships**.

The **Selected Fields** panel displays the fields, tables, or any other element selected from the Database view. Each field in this panel has its own set of editable options.

Option	Description
Output	Determines whether the field will be included in the result set. By default, this checkbox is selected when a field is added to the **Selected Fields** panel. You can clear this checkbox if you do not want the field to be displayed in the **Results** panel.
Table	Displays the name of the table the selected field belongs to.
Alias	Allows the user to provide an alternative name for the field.
Total	Applies grouping or aggregates on a field. Total (expression) is used to perform a calculation, retrieve the value of a control, define regulations, create calculated fields, and define a group level for a report. • Expression - Allows the selection of fields from a table. Custom expressions can also be specified here. • GroupBy - Groups data based on the selected field. • Count - Returns the number of items in a group. Implements the SQL COUNT aggregate. • Avg - Returns the average of values in a group. Implements the SQL AVG aggregate. • Sum - Returns the sum of all the values in a group. Implements the SQL SUM aggregate. • Min - Returns the minimum value in a group. Implements the SQL MIN aggregate. • Max - Returns the maximum value in a group. Implements the SQL MAX aggregate. • StDev - Returns the statistical standard deviation of all the values in a group. Implements the SQL STDEV aggregate. • Var - Returns the statistical variance of all values in the group. Implements the SQL VAR aggregate.
Sort	Arranges data in a prescribed sequence i.e. in Ascending or Descending order. By default, this

Option	Description
	option is set to Unsorted.
Sort Order	Allows the user to set the order of sorting in case multiple fields are to be sorted.
Where	Allows the user to set a filtering condition for the column data. The WHERE clause can be used when you want to fetch any specific data from a table omitting other unrelated data.

Note: If you want to delete a field in the **Selected Fields** panel, hover the mouse cursor over the field and click the **Delete** button.

The **Tables and Relationships** panel displays a list of all the tables with fields in the Selected Fields panel. The **Relations** button at the bottom of the related table's name shows the relationship between two or more tables when associating rows of one table with the rows of another.

You can set up these relationships between tables using SQL Joins like Inner Join, Left Join, and Right Join in the Visual Query Designer. The relationship you set up between the data in these tables determines how the data appears in the result set.

- **Inner Join (simple join)**: It matches rows from table1 with rows in table2, and allows the user to retrieve rows that show a relationship in both the tables. Inner join produces a set of data that matches both table1 and table2.

  Query Syntax

  ```
  SELECT table1_name.column_name FROM table1_name INNER JOIN table2_name ON
  table1_name.column_name = table2_name.column_name
  ```

- **Left Outer Join (left join)** - It selects rows that match from both the left and right tables, plus all the rows from the left table (table1). This means only those rows from table2 that intersect with table1 appear in the result set.

  Query Syntax

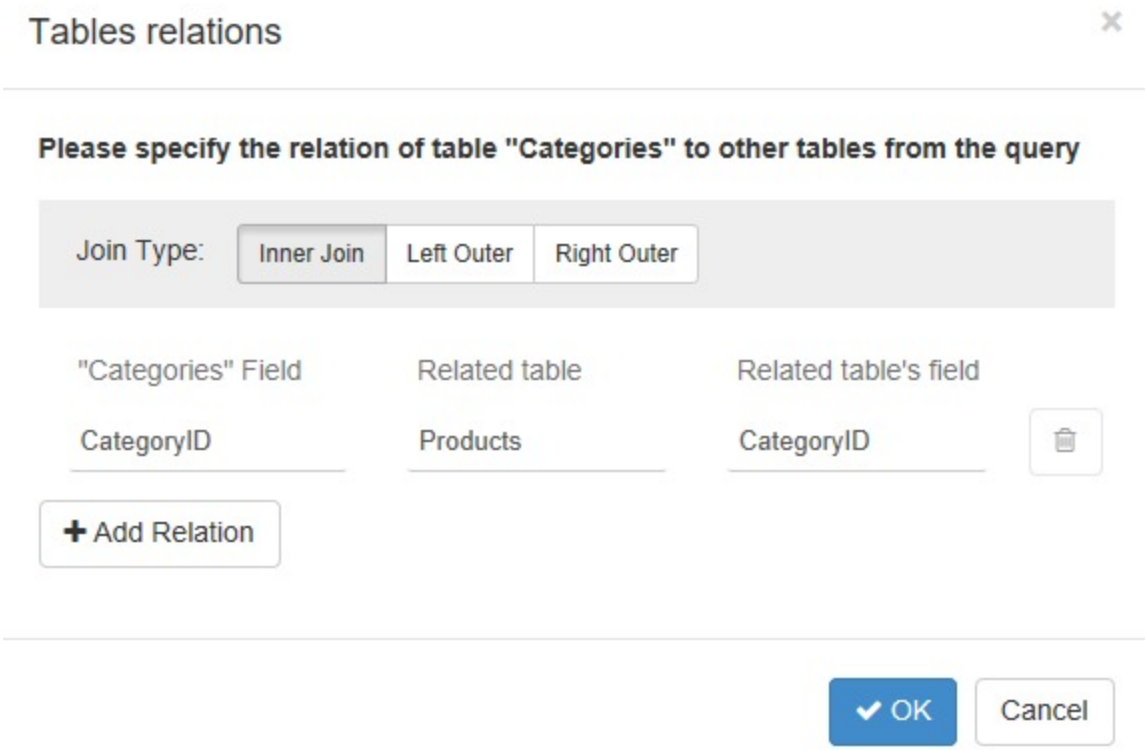
  ```
  SELECT table1_name.column_name FROM table1_name LEFT [OUTER] JOIN table2_name ON
  table1_name.column_name = table2_name.column_name
  ```

- **Right Outer Join (right join)** - Right outer join allows users to select rows that match from both the left and right tables, plus all the rows from the right table (table2). This means that only those rows from table1 that intersect with table2 appear in the result set.

  Query Syntax

  ```
  SELECT table1_name.column_name FROM table1_name RIGHT [OUTER] JOIN table2_name ON
  table1_name.column_name = table2_name.column_name
  ```

Use the **Tables relations** dialog box to set up a relationship between two different tables with at least one common field (or column). This dialog box automatically appears when you add another field from a different table in the **Database view** tab to the **Selected Fields** panel. Make sure that at least one field between these two tables matches. In other words, the second table should contain a foreign key corresponding to which the first table contains rows with a same primary key.



The **Tables relations** dialog box provides the following options.

Option	Description
Join Type	Enables users to specify the join type for joining two tables. You can set the join type to Inner Join, Left Outer Join, and Right Outer Join.
<Table Name> Field	Displays the name of the field that is common between tables i.e. the foreign key name in the second table. Example: "Products" Field contains the 'Category ID' field in the image above.
Related table	Displays the name of the table to which the relationship has been set up.
Related table's field	Displays the name of the field from the table to which the relationship has been set up.
Delete	Deletes the currently added relation in the dialog.
Add Relation	Allows users to add another relationship to the table.

The **Query Tools** section also has a drop-down on the top right corner with the following two options.

- **Toggle Panels**: Allows users to expand or collapse the **Selected Fields** and the **Tables and Relationships** panel.
- **Show Hints**: Enables users to show or hide hints on how to use the Visual Query Designer effectively. For example, a hint is displayed on top of the **Selected Fields** panel saying - *Double click the table or field name in the database view or drag and drop it here*.

SQL tab

The **SQL tab** automatically generates an SQL query based on the options specified in the **Designer** tab. Here, you can

also manually edit an existing query or create a new query.

Toolbar buttons

The options available through the toolbar buttons are elaborated in this table.

Option	Description
Distinct	Allows users to remove duplicate values from the result set of a SELECT statement. You can check this option to display only distinct values.
Execute	Allows users to execute their query and view the result in **Results** panel.
Save	Allows users to save the query to a **DataSet** dialog.
Clear	Allows users to clear all the panels in the Visual Query Designer and the SQL tab along with it.

Results panel

Displays the result of the query set in the Visual Query Designer.

This panel is populated when you click the **Execute** button on the Visual Query Designer toolbar after adding the required fields or tables in the Selected Fields panel.

Limitations

Following are few limitations of Visual Query Designer

- Unions, nested queries, and stored procedures are not supported in the **Design** tab.
- Crosses, full joins, provider-specific joins, and other SQL-specific implementation capabilities are not supported in the **Design** tab.

Designing queries in Visual Query Designer

Follow the below steps to build a query in the Visual Query Designer.

1. In the **Report Explorer**, right-click an existing data source and select the **Add Data Set** option.
2. In the **DataSet** dialog box that appears, select the **General** page and enter the name of the dataset. By default, the dataset name is set to DataSet1. This name appears as a child node to the Data Source node in the Report Explorer.
3. Navigate to the **Query** tab and click the **Edit with the Visual Query Designer** icon  in the **DataSet** dialog box.

   The following sections describe the operations that you can perform in a Visual Query Designer.

   #### Add a field or table

   This is the primary step while designing queries in the query designer. There are two ways to add fields in the query.

   - Double-click the field name or table name in the **Database view**.
   - Drag the desired field or table from the **Database view** to the **Selected Fields** panel.

Note: If you add a table to the **Selected Fields** panel, all the fields in that table are added to the query.

Resultant Query

```
//Query for adding fields
SELECT table_name.field_name FROM table_name

//Query for adding a table
SELECT * FROM table_name
```

Specify an alias for a field

Alias is a temporary name given to a field for the duration of a query. You can specify an alias to make the field name more readable.

1. Select the desired field in the **Selected Fields** panel.
2. Go to the **Alias** option and enter the alias for the field in the space provided.

Resultant Query

```
SELECT table_name.field_name AS alias_name FROM table_name
```

Specify an alias for a table

Alias is a temporary name given to a table for the duration of a query. You can specify an alias to make the table name more readable.

1. Select the desired table in the **Tables and Relationships** panel.
2. Go to the **Alias** option and enter the alias for the table in the space provided.

Resultant Query

```
SELECT * from table_name AS alias_name
```

Join tables

If you want to combine rows from two or more tables, you need to join the tables based on a logical relationship between them. A **Tables relation** dialog box automatically appears when you add a field belonging to a different table. Use this dialog box to specify the relationships between the two tables.

1. Drag and drop a field from a table in the **Database view** to the **Selection Fields** panel.
2. Add another field from a different table in the **Database view** to the **Selected Fields** panel. The **Tables relations** dialog automatically pops up on the screen.
   Make sure that at least one field between these two tables matches, i.e., the second table should contain a foreign key corresponding to which the first table contains rows with a same primary key.
3. Specify the **Join Type** you want to set for combining the two tables by clicking the **Inner Join**, **Left Outer Join**, or **Right Outer Join** button.
   The dialog box automatically picks the common field name present in both the tables, related table name, and related field name based on the fields selection. However, you can modify these details by clicking the associated drop-down arrows.
4. Click **OK** to save the changes.

Resultant Query

```
//Query for inner join
SELECT table1_name.column_name FROM table1_name INNER JOIN table2_name ON
table1_name.column_name = table2_name.column_name

//Query for left outer join
SELECT table1_name.column_name FROM table1_name LEFT [OUTER] JOIN table2_name ON
table1_name.column_name = table2_name.column_name

//Query for right outer join
SELECT table1_name.column_name FROM table1_name RIGHT [OUTER] JOIN table2_name ON
table1_name.column_name = table2_name.column_name
```

Delete a field

When you delete a field from a query, the field remains in the database but is no longer used in the query.

1. In the **Selected Fields** panel, hover your mouse over the field to display the **Delete** icon.
2. Click the **Delete** icon to delete the field.

Delete a table

When you delete a table from a query, the table remains in the database but is no longer used in the query.

1. In the **Tables and Relationships** panel, select the table you want to delete.
2. Click the **Close** icon. This will delete the selected table from the query.

Sort data

You can sort the fetched data in either ascending or descending order based on one or more fields in the table(s).

1. Select the desired field in the **Selected Fields** panel.
2. Go to the **Sort** option and choose one of the following options - Ascending or Descending. This will sort the field values in the chosen order.

Resultant Query
`SELECT * FROM table_name ORDER BY field_name ASC|DSC`

Filter data

Applying filters allows you to fetch a specific data from a table that matches a certain criteria

1. Select the desired field in the **Selected Fields** panel.
2. Go to the **Where** option and enter the filter criteria with an required operator (like '=', '>', '<', '<>', etc). This will filter the rows based on the specified filter condition.
   If you want to filter the data based on a parameter value instead of a constant value, set the **Where** field to '`=@<field name>`'. This will generate a parameterized query that generally prompts the user to enter a value before the query is executed, to determine the type of data to be displayed in the result set.

Resultant Query

```
SELECT * FROM table_name where [CONDITION]
```

Apply aggregate functions and grouping

You can group data on a field and create an aggregate query that involves a function such as Sum or Avg in the Visual Query Designer.

1. Select the desired field in the **Selected Fields** panel on which you want to apply grouping or aggregate function.
2. Go to the **Total** option and choose one of the following options from the drop-down - Expression, GroupBy, Count, Avg, Sum, Min, Max, StDev, and Var.

Resultant Query

```
//Query for grouping data
SELECT * FROM table_name WHERE [CONDITION] GROUP BY table_name.field_name

//Query for aggregate functions
SELECT aggregate_function(table_name.field_name) FROM table_name WHERE [CONDITION]
```

4. Preview the query using the **Execute** button in the Toolbar.
   You can save the query in the **DataSet** dialog box by clicking the **Save** button next to the **Execute** button.
5. Click the **Validate DataSet** icon to verify the query in the **DataSet** dialog box. Go to the **Fields** page to view the dataset fields fetched from the generated query.
6. Click the **OK** button to close the **DataSet** dialog box.

Stored Procedure as Dataset

A stored procedure is a group of SQL statements that encapsulate a set of operations or queries to execute on a database.

Follow the below steps to add a parameterized stored procedure as a data set.

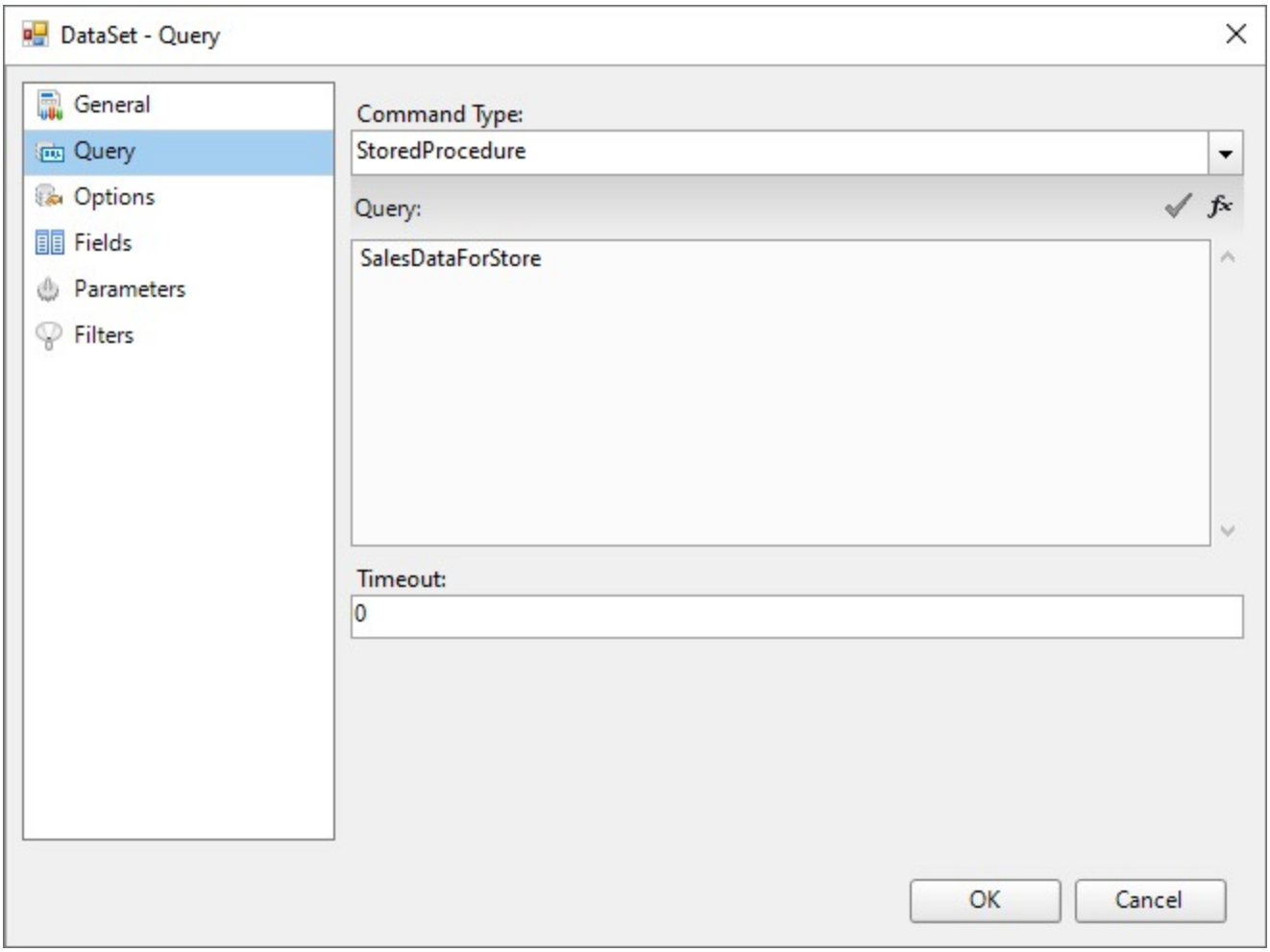
1. First, Connect to OLEDB Data Source (here we will connect to Reels.mdb).

2. In the Report Explorer, right-click the data source node and select the **Add Data Set** option or select **Data Set** from the Add button.

1. In the Add Dataset that appears, select the **General** page and enter the name of the dataset as 'SalesDataForStore'. This name appears as a child node to the data source icon in the Report Explorer.
2. On the **Query** page of this dialog, set the **Command Type** to 'StoredProcedure'.
3. In the **Query** field, enter the stored procedure name (e.g. SalesDataForStore).
4. Click the **Validate** icon to validate the query.

   Note: You may get an error at this point since the required parameters have not yet been added.

5. Go to the **Parameters** page and add a Parameter using the Add(+) button.
6. On the same page, enter **Name** as StoreID and **Value** as 1002.
7. Select the **Query** page of **DataSet** dialog, and click the **Validate** icon to validate the query and load the fields.



8. Click **OK** to close the dialog. Your data set and queried fields appear as nodes in the Report Explorer.

Create Dynamic JSON Connection String

You can enter a connection string for the JSON data as an expression and pass values using parameters to set up the data sources dynamically.

Let us create a tabular report where data in the table is fetched from a dynamically built JSON data source.

Create a Report

In the ActiveReports Designer, create a new RDLX report.

Add First Data Source

Connect to a Data Source

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option.
2. In the dialog, select the **General** page and enter the name of the data source, 'Categories'.
3. Under **Type**, select 'Json Provider'.

4. In the **Content** tab, set the type of Json data to 'External file or URL'.
5. In the **Select or type the file name or URL** field, enter the following URL: https://demodata.mescius.io/northwind/api/v1/Categories
   The **Connection String** tab displays the generated connection string as shown below:

Connection String
`jsondoc=https://demodata.mescius.io/northwind/api/v1/Categories`

   For more information, see the JSON Provider topic.
6. Verify the generated connection string by clicking the **Validate DataSource**  icon.
7. Click **OK** to save the changes and close the **Report Data Source** dialog.

Add Dataset

1. Right-click the added data source and select **Add Dataset**.
2. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'dsCategories'.
3. Go to the **Query** tab and enter the following query to fetch the required fields:

Dataset Query
`$.[*]`

Add a Parameter

1. In the Report Explorer, right-click the Parameters node and select the **Add Parameters** option.
2. In the **Report - Parameters** dialog box that appears, set the following:

General tab	
Property	**Value**
Name	paramCategories
Data Type	String
Text for prompting users for value	Select a Category
Available Values tab (>From query)	
Property	**Value**
Dataset	dsCategories
Value field	categoryId
Label field	categoryName
Order By: Condition	Label
Order By: Direction	Ascending

3. Click OK.

Add Dynamically Built Data

Connect to a Data Source

1. Right-click the Data Sources node in the Report Explorer and then select the **Add Data Source** option.
2. In the dialog, select the **General** page and enter the name of the data source, 'Products'.

3. Under **Type**, select 'Json Provider'.

   Let us first fetch the fields from data source using an external URL without expression, to fill the dataset fields collection. We will then replace the URL with an expression to create dynamic JSON connection string.

4. In the **Content** tab, set the type of JSON data to 'External file or URL'.
5. In the **Select or type the file name or URL** field, enter the following URL: https://demodata.mescius.io/northwind/api/v1/Categories/1/Products
6. Click OK to close the dialog.

Add Dataset

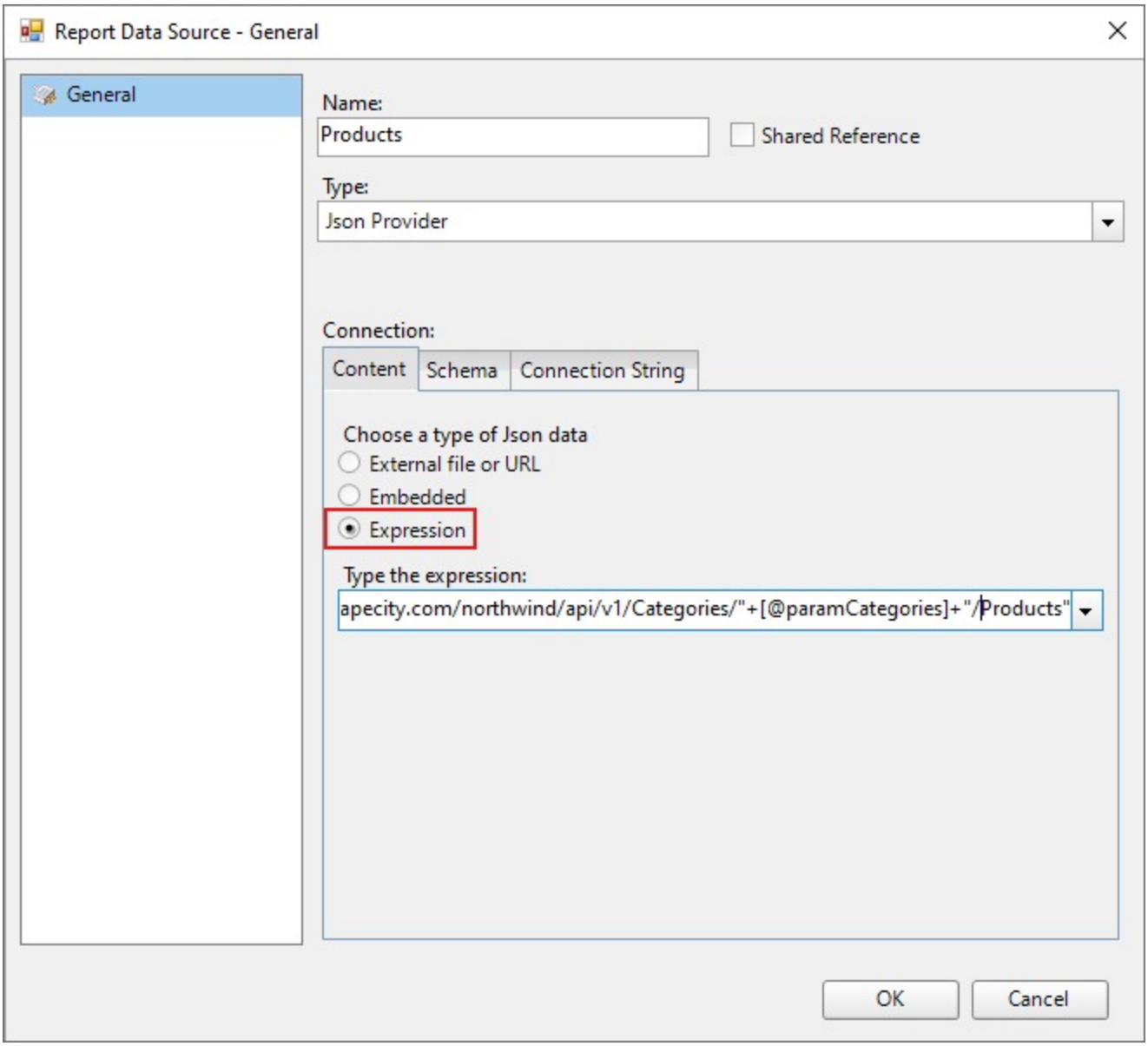
1. Right-click the added data source and select **Add Dataset**.
2. In the Dataset dialog, select the **General** page and enter the name of the dataset, 'dsProducts'.
3. Go to the **Query** tab and enter the following query to fetch the required fields:

Dataset Query
`$.[*]`

4. Click OK.

Update Data Source Connection

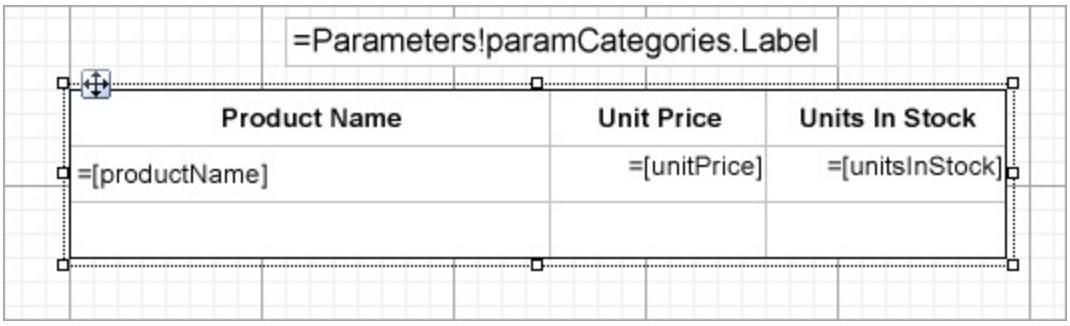
1. Edit the 'Products' data source.
2. Go to the **Content** tab, select **Expression**, and enter the expression in the editor like the following:



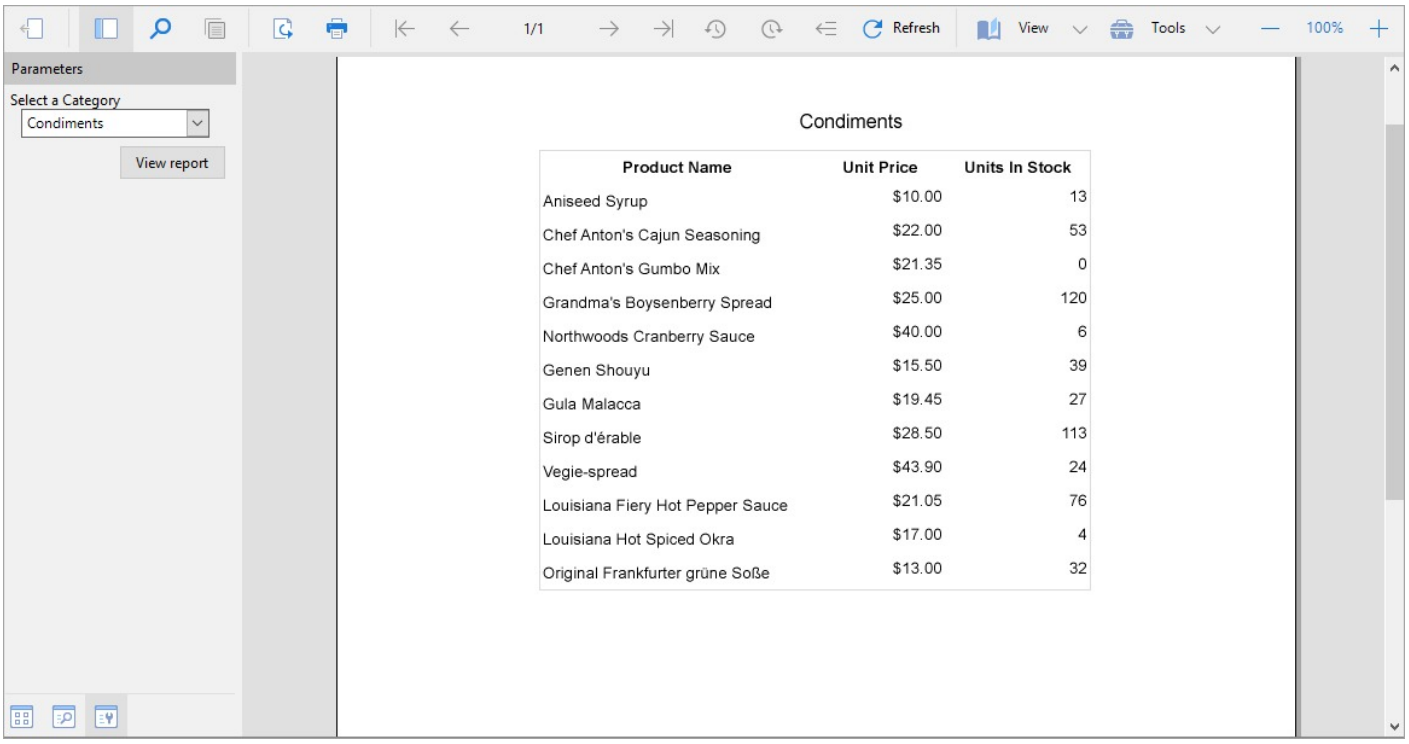
Connection String
="jsondoc=https://demodata.mescius.io/northwind/api/v1/Categories/"+ [@paramCategories]+"/Products"

3. Click OK.

Design Report Layout

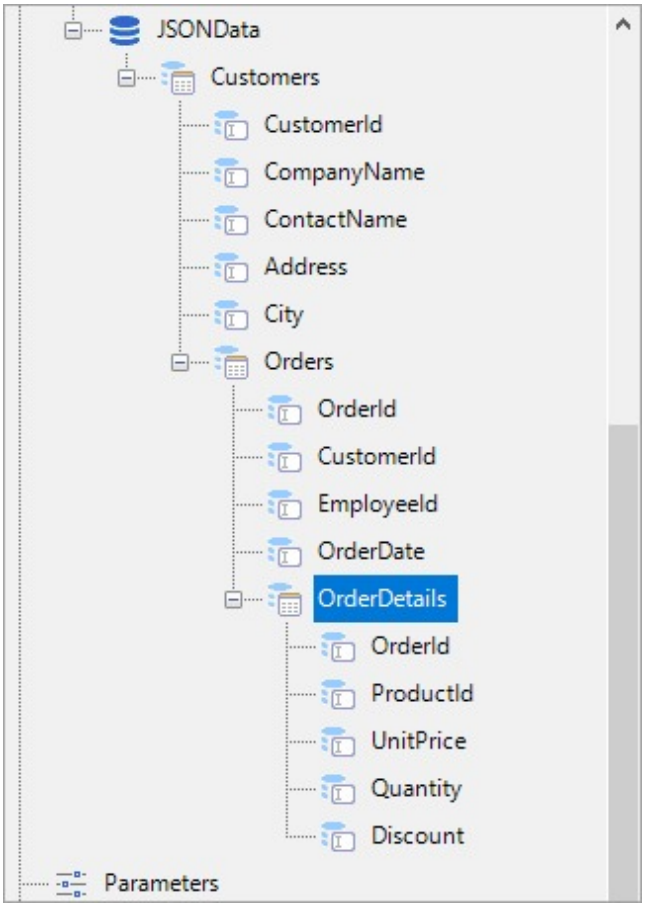


1. Drag and drop the **Table** data region onto the report's designer and set the **DataSetName** property to 'dsProducts'.
2. Fill the details row of the table with the following fields:
    - `=Fields!productName.Value`
    - `=Fields!unitPrice.Value`
    - `=Fields!UnitsInStock.Value`
3. To display the selected parameter, drag and drop a **TextBox** control above the table and enter the **Value** as `=Parameters!paramCategories.Label`.
4. Press **F5** to preview the report.
5. Select a parameter from the drop-down and click **View report**.



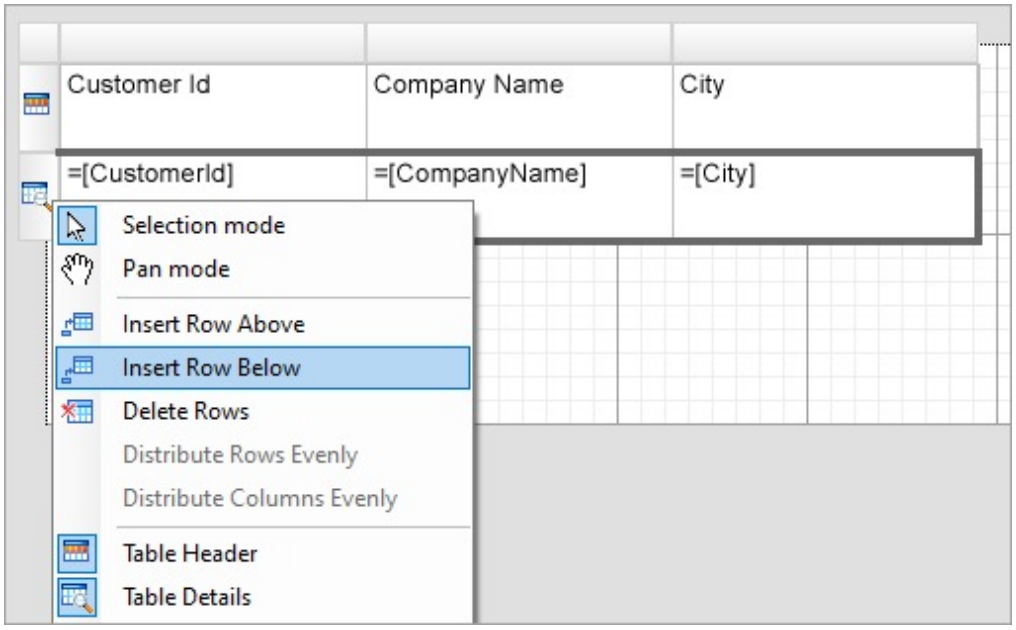
Nested Datasets (JSON and XML)

A nested dataset represents JSON or XML data as an hierarchical structure. A nested JSON or XML dataset is most commonly used in a bound data region, where you can use the dataset nodes partially. For example, you can create a report with the Table data region, bound to the **Customers** dataset, and the nested List data region, bound directly to the nested dataset **OrderDetails**.

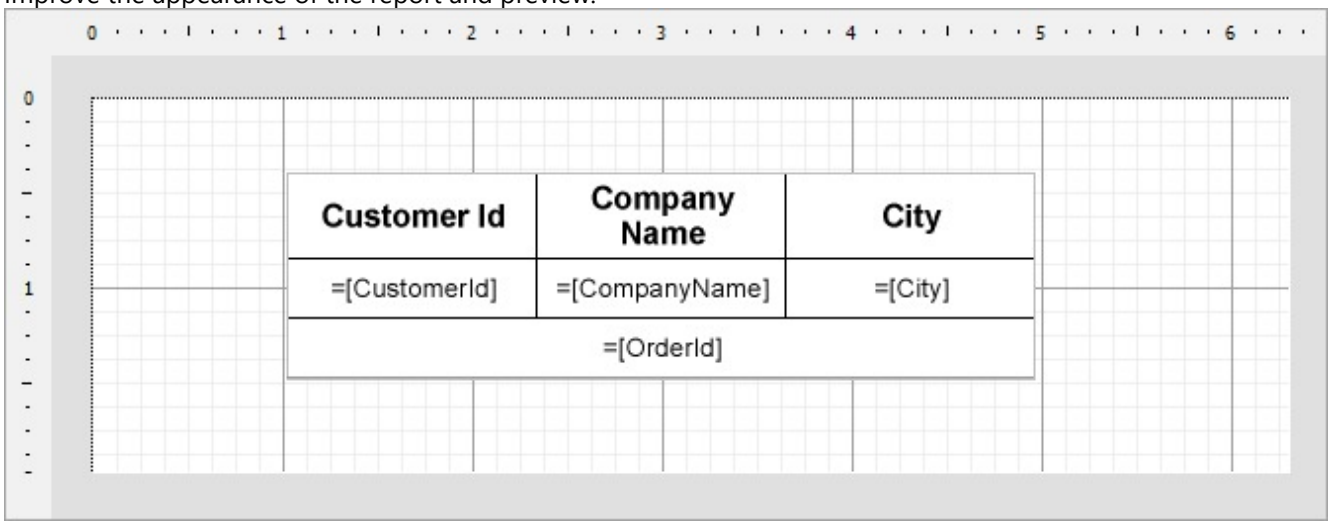


Follow these steps to create a new RDLX report with a JSON nested dataset.

1. In the ActiveReports Designer, create a new RDLX report.
2. On the **Choose Data Source Type** screen of the wizard, select **JSON** and click **Next**.
3. On the **Configure JSON Path Connection** screen, enter the following URL:
   https://demodata.mescius.io/northwind/odata/v1/Customers?$top=3&$expand=Orders($expand=OrderDetails,Shipper)
   and click **Next**.
4. On the **Configure JSON Queries** screen, enter the name of the dataset, 'Customers', and in the **Query** field, specify **$.value[*]**.
5. On the **Review and Confirm** screen, click **Finish**.
6. From the Toolbox, drag and drop the **Table** data region to the design area and set the data fields in the Table Details row to data values (**CustomerId**, **CompanyName**, and **City**) from the bound **Customers** dataset.
7. Right-click the Table Details and select **Insert Row Below**.



8. From the Toolbox, drag and drop the **List** data region to a Table cell in the new row.
9. Drag and drop the **Textbox** control to the List and set its Value to **OrderID** from the nested **Orders** dataset.
10. Improve the appearance of the report and preview.



Work with Local Shared Data Sources

In ActiveReports, a shared data source refers to a file in RDSX format that contains data connection information. RDSX (Report Data Source XML) is a proprietary file format that functions as a data source for a single report or multiple reports.

You can use this data source in the Page report and the RDLX report.

> **Note**: The WebDesigner and Blazor WebDesigner components uses the 'shared data source' approach to encapsulate the data connection on the server-side. See Configure and Use Shared Data Sources in WebDesigner and Configure and Use Shared Data Sources in Blazor Designer for more information.

Advantages of a shared data source

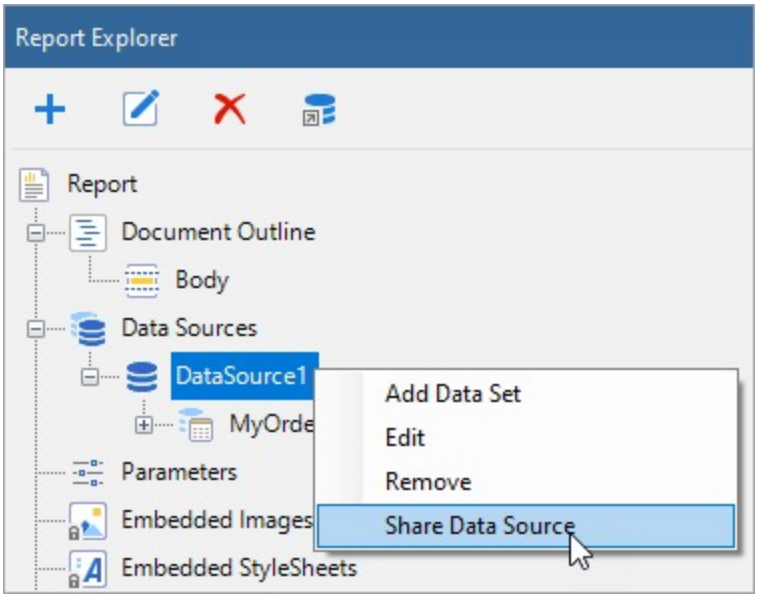
- It is a reusable data connection that you can use in a single report or multiple reports.
- It is a separate file in RDSX format that you can access from any report, move to different folders, and rename.
- It sets the data connection in a report, so you need only add a SQL query to create a data set.
- It lets you update the connection string in one place for all reports referencing the Shared Data Source.
- You can use it with any data connection type.

> **Note**: An RDSX file contains the connection string, but you cannot define a data set in it. See Add a Dataset for information on creating a data set.

You can save a data connection type such as an OleDb connection, Json connection, or an XML connection as a Shared Data Source (RDSX). This topic provides the steps to create a shared data source with a data connection. Using shared data source, multiple reports can connect to a same data source. See Connect to a Data Source on how to connect a report to a data source.

Create a shared data source

1. Right-click the data source in the Report Explorer and select **Share Data Source** option.



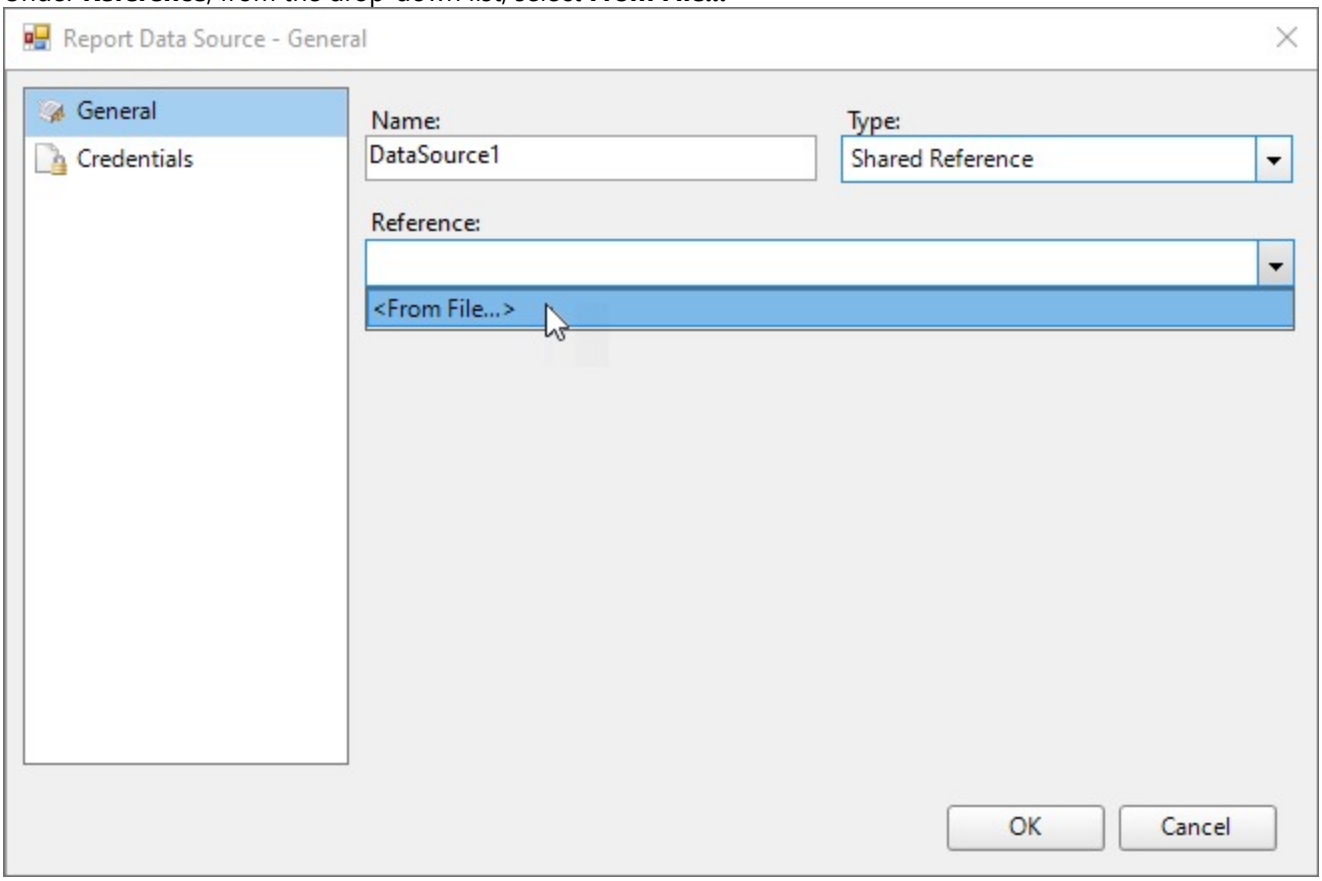
2. In the **Save Shared Data Source File** dialog that appears, enter the file name and click the **Save** button to save the file in RDSX format.
   Notice that the data source icon changes to show sharing.

Data Source Icon	Shared Data Source Icon
	

Connect to a shared data source

In ActiveReports, you can connect to most data sources using the steps in the Connect to a Data Source. However, you need to follow the below steps to connect to a Shared Data Source.

1. In the **Report Explorer**, right-click the Data Sources node and select the **Add Data Source** option or select **Data Source** from the **Add** button.
2. In the **Report Data Source** dialog that appears, select the **General** page and enter the name of the data source. This name appears as a child node to the Data Sources node in the Report Explorer.
3. Select **Shared Reference** as the **Type**.
4. Under **Reference**, from the drop-down list, select **From File...**

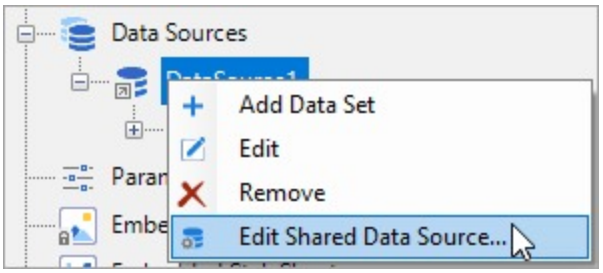


5. In the **Shared Data Source File** dialog that appears, select the file and click Open on the lower right corner to close the dialog.
   A shared data source node appears in the Report Explorer.

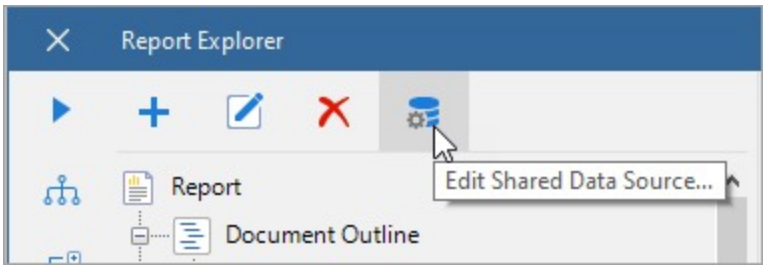
Edit a shared data source

These steps assume that you have already connected your report to a shared data source.

1. Open the Report Data Source dialog using any of the following ways:
   - In the **Report Explorer**, right-click a shared data source node and from the context menu, select **Edit Shared Data Source**.



- In the **Report Explorer**, select the shared data source node, go to the **Report Explorer** toolbar and click the **Edit Shared Data Source** button.



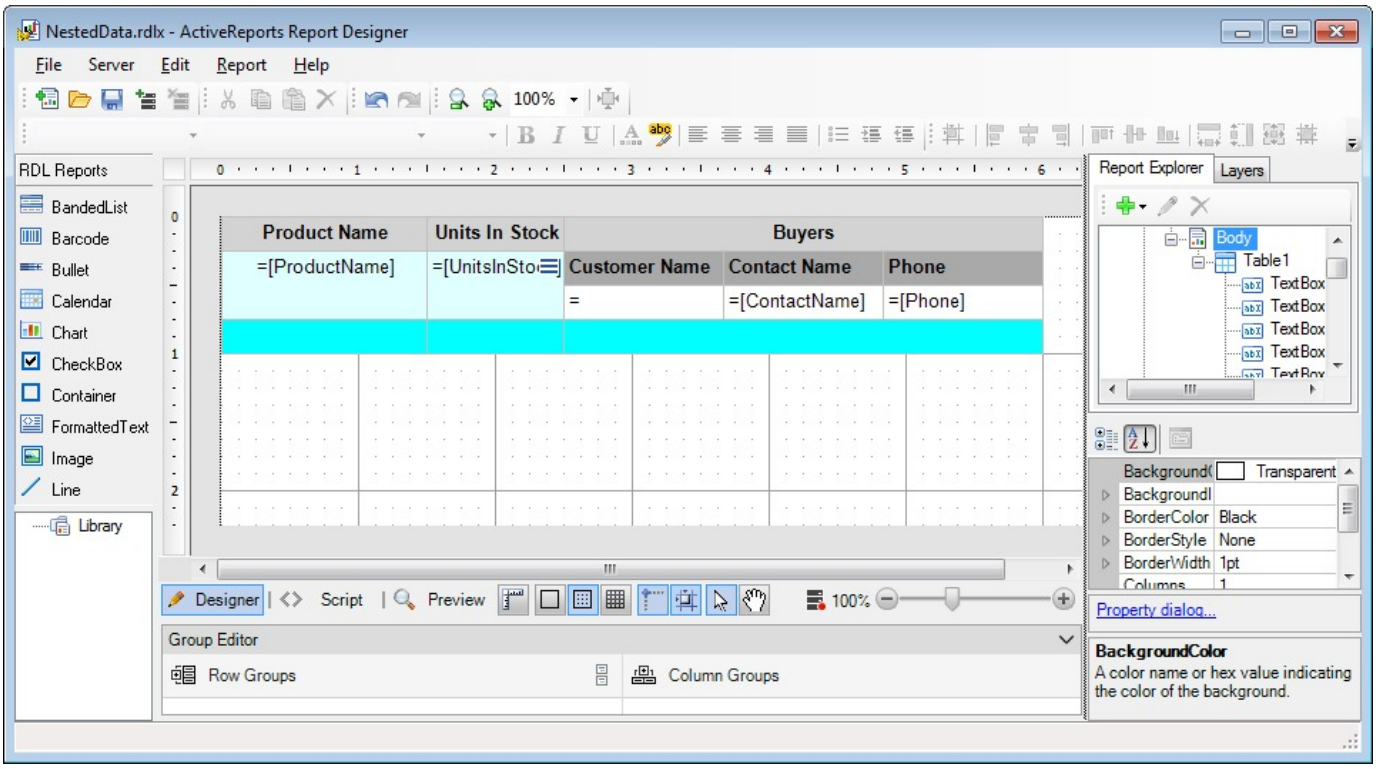
2. In the **Report Data Source** dialog that appears, edit the data connection information.
3. Click **OK** to save the edits.

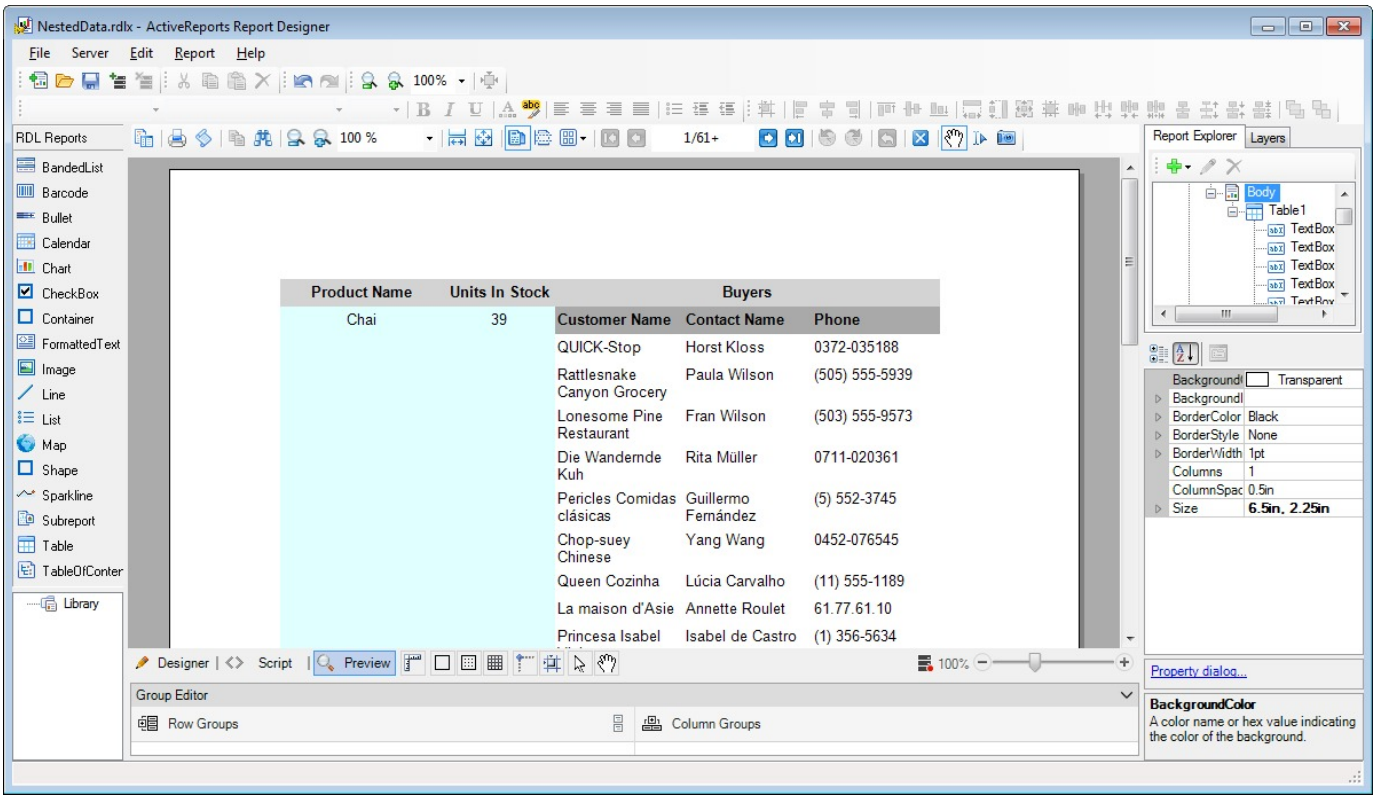
Bind Nested Data Regions to Different Datasets

In Page and RDLX reports, you can use nested data regions that are bound to different datasets. To display data, you can either use a **filter** for a nested data region or a **parameter** that is set in the **DataSetParameters ('DataSetParameters Property' in the on-line documentation)** property.

Binding data regions to different data is available for all data regions that you can use in Page and RDLX reports, that is Tablix, List, Chart, BandedList, Table, and Sparkline.

The report below shows the customer's contact name and phone information for products that are still in stock. The report layout consists of three nested Table data regions, each data region bound to a different dataset - **Products** (Table1), **Invoices** (Table2), and **Customers** (Table3).





Using a filter

To display data in nested data regions that are bound to different data, you can set a filter in the **Table - Filters** dialog.

This filter will contain the value from a nested data region in the left side and the value from a parent data region in the right side of it.

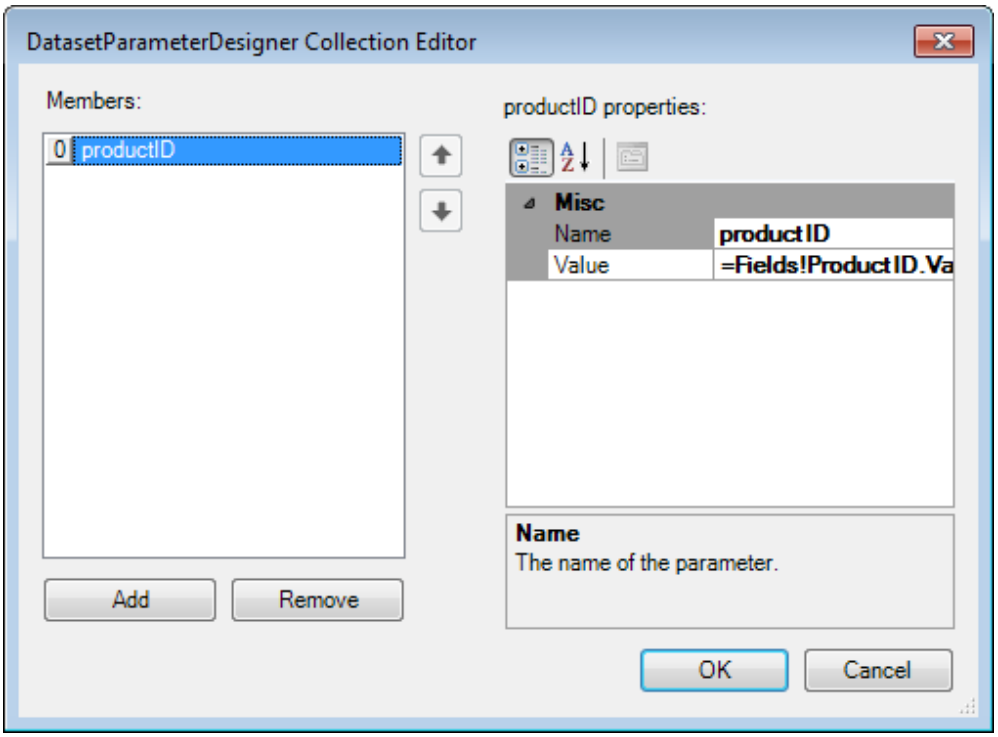
In the sample report above, two filters are created. For the Table2 data region bound to the **Invoices** dataset, we create the filter with the expression [ProductID]=[ProductID]. For the Table3 data region bound to the **Customers** dataset, we create the filter with the expression [CustomerID]=[CustomerID].

As a result, the report shows the Product Name and Units In Stock information from the **Products** dataset. For each Product Name, the report shows the Customer Name information from the **Invoices** dataset and the Contact Name and Phone information from the **Customers** dataset.

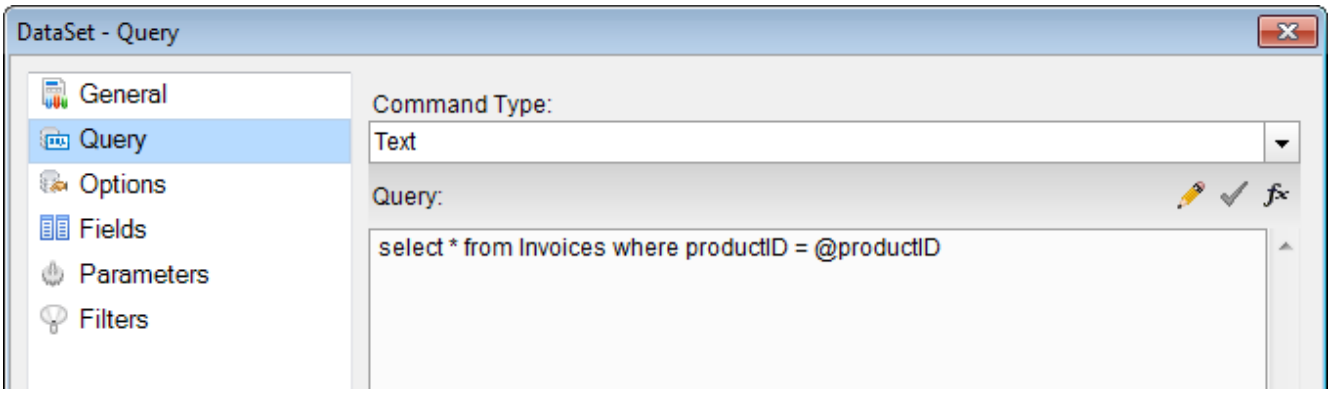
Using a parameter

Setting a parameter in the **DataSetParameters** property also allows displaying data in nested data regions that are bound to different datasets. The basic steps are as follows.

1. In the **DataSetParameters** property of a data region, add a new parameter, for example, **productID**.



2. In the data region's dataset, add a new parameter - **productID**.
3. In the data region's dataset, add a substring with the parameter to the existing dataset query, for example: **WHERE productID = @productID**



For the sample report layout above, two parameters are created. For the Table2 data region, bound to the **Invoices** dataset, we create the parameter **productID** and modify the dataset query as *select * from Invoices where productID = @productID*. For the Table3 data region, bound to the **Customers** dataset, we create the parameter **customerID** and modify the dataset query as *select * from Invoices where customerID = @customerID*.

As a result, the report shows the Product Name and Units In Stock information from the **Products** dataset. For each Product Name, the report shows the Customer Name information from the **Invoices** dataset and the Contact Name and Phone information from the **Customers** dataset.

Data Binding in Section Reports

Setting up a connection to the data source is the first step in binding data to the report. Once a successful connection is established, a query is required to get the data you want to show in the report.

Data Sources

In Section reports, you can set the data source information in the Report Data Source dialog properties.

The Report Data Source dialog is where you select the type of data to use, provide a connection string, create a query string, and choose other options for your data source. You can also control the timeout period and select a method for handling credentials. Once you add a data source, its fields appear under the Bound node in the Report Explorer. You can also add calculated fields in a section report.

To quickly build the query strings, ActiveReports provides a JSON Query Designer for JSON provider, and a Visual Query Designer for OLEDB and SQL providers.

> **Note**: The Chart data region in Section reports have their own data sources and should be bound separately, similar to how the main report is bound.

Supported Data Providers

The supported data providers for designing section reports are:

- OLEDB Provider
- ODBC Provider
- SQL Provider
- XML Provider
- CSV Provider
- JSON Provider

- Unbound Provider

Connect to a Data Source

At design time, you can connect a section report to a data source through the **Report Data Source** dialog. This topic describes the steps to connect to a data source in a section report.

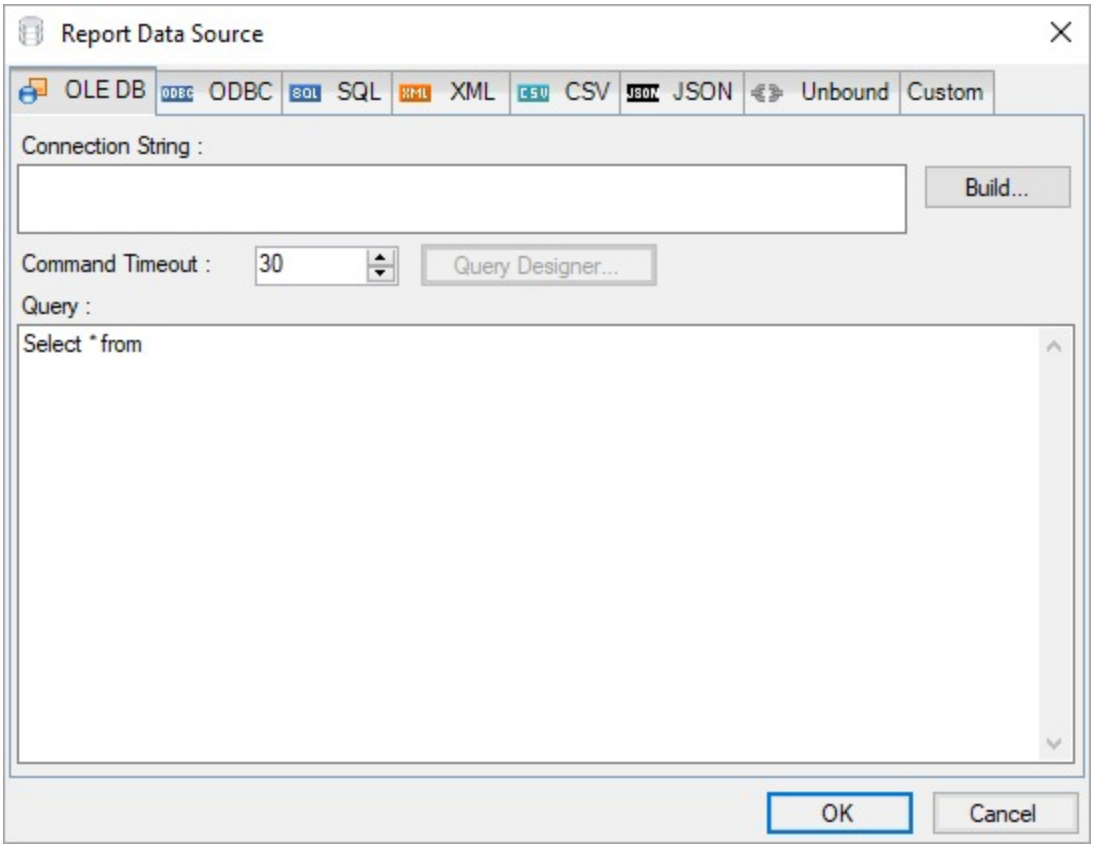
Add a Data Source

Use the following steps to add a new data source in your report.

1. In the designer, you can access the **Report Data Source** dialog by doing one of the following:
   - On the detail section band, click the **Data Source** icon.

   - Click the gray area around the design surface and select the **Edit Data Source** link in the Properties pane.
2. In the **Report Data Source** dialog box that appears, there are separate tabs for each data source. For example,
   - OLEDB
   - ODBC
   - SQL
   - XML
   - CSV
   - JSON
   - Unbound



3. Each tab displays different options/settings to configure the data source connection.
4. Enter the configuration details for the data source connection.
5. Click the **OK** button on the lower right corner to close the dialog. You have successfully connected the report to a data source.

Custom Data Providers

You can also implement some custom data providers such as SQLite, by manually setting up the dependencies and configuration file. See Bound Data sample for complete implementation and Configure ActiveReports topic for information on setting up the configuration file.

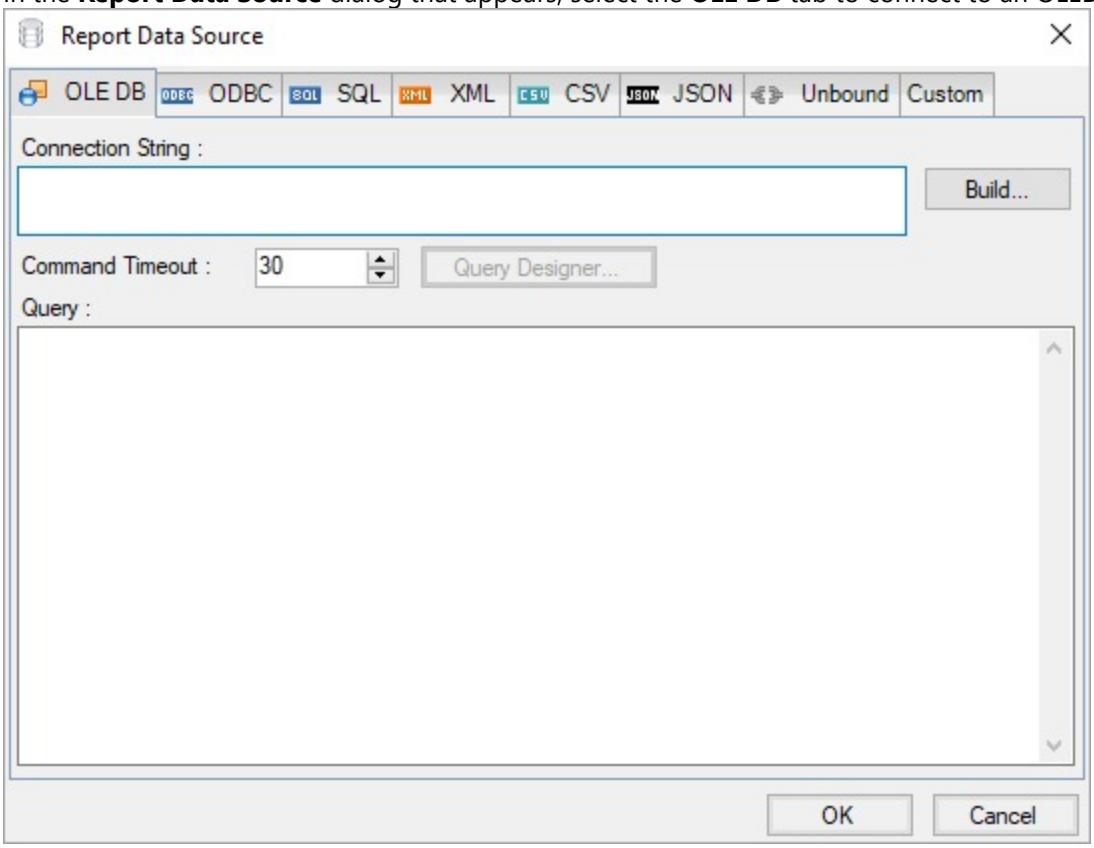
OLEDB Provider

This article explains connecting a Section report to an OLEDB data source.

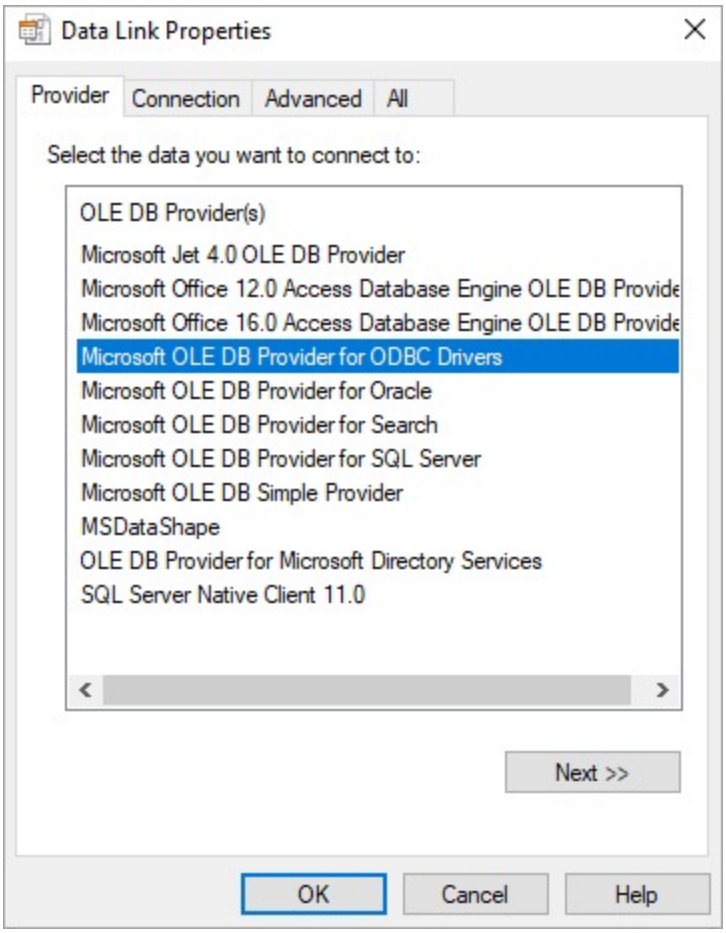
> **Note**: The OLEDB model depends on the installed drivers.

Connect to an OLEDB Data Source

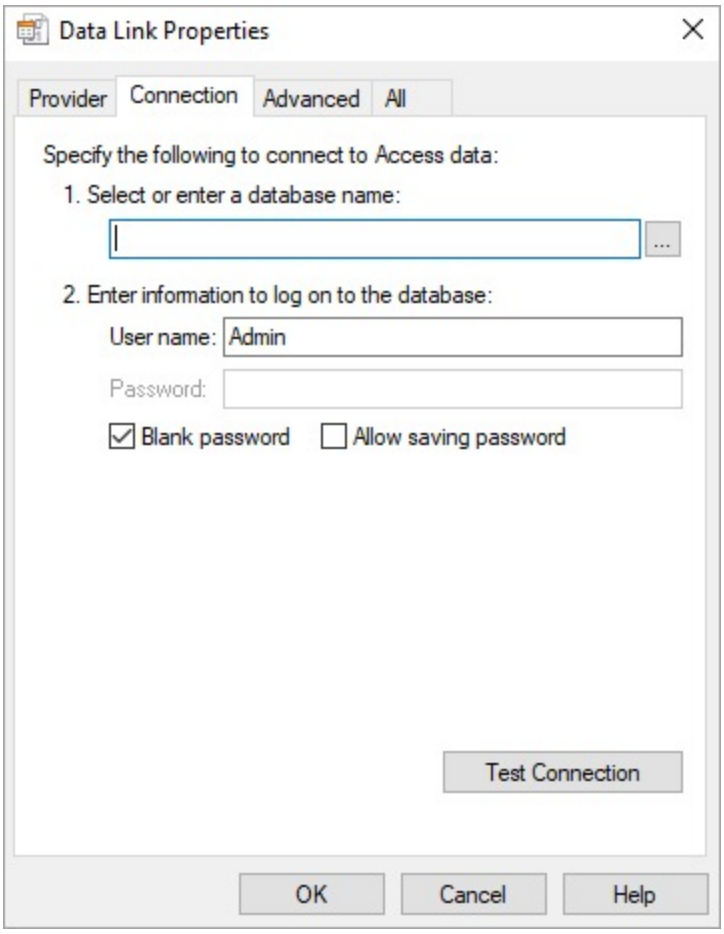
1. In the designer, select the **Data Source** icon on the Detail section of the report or click the gray area around the design surface and select the **Edit Data Source** link in the Properties pane.
2. In the **Report Data Source** dialog that appears, select the **OLE DB** tab to connect to an OLEDB data source.



3. Click the **Build** button next to the Connection String section to open the **Data Link Properties** dialog box.



4. In the **Providers** tab, specify the OLEDB provider you want to use to connect to the data source. For example, select **Microsoft JET 4.0 OLE DB Provider** and click the **Next** button to move to the **Connection** tab.
   The fields in the Connection tab depends on the chosen OLEDB provider. For more information, see the **Configuration Settings for OLEDB Data Source**.
5. To specify the path for the file, click the ellipsis (...) button and navigate to the desired folder on your system. For example, you can connect to the 'NWIND.mdb' sample data source which can be downloaded from GitHub.



6. Click the **Test Connection** button to see if you have successfully connected to the database.
7. Then, click the **OK** button to close the **Data Link Properties** dialog box.
   The **Connection String** section displays the generated connection string as shown below:
   `Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\NWIND.mdb;Persist Security Info=False`
8. In the **Command Timeout** field, specify the wait time (in seconds) for a command to execute. The default value is 30 seconds.
9. In the **Query** field on the **OLE DB** tab, enter a SQL query to fetch the required data from the connected database. For example,
   `SELECT * FROM Customers`
   You can also click the **Query Designer** button to access the Visual Query Designer for creating SQL queries. For information on how to create a query using the interactive query designer, visit the Visual Query Designer article.
10. Click the **OK** button to save the changes.

Configuration Settings for OLEDB Data Source

The OLEDB Data Provider provides the following configuration settings in the **Report Data Source** dialog.

The **Provider** tab describes the type of OLEDB provider you want to use for connecting to a data source. You can choose from the following -

- Microsoft JET 4.0 OLE DB Provider
- Microsoft Office 12.0 Access Database Engine OLE DB Provider
- Microsoft Office 16.0 Access Database Engine OLE DB Provider

- Microsoft OLE DB Provider for ODBC Drivers
- Microsoft OLE DB Provider for Oracle
- Microsoft OLE DB Provider for Search
- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Simple Provider
- MS Data Shape
- OLE DB Provider for Microsoft Directory Services
- SQL Server Native Client 11.0

The **Connection** tab specifies the properties required for accessing the data. The properties in the tab depend upon the OLEDB provider you want to use.

For Microsoft JET 4.0 OLE DB Provider and Microsoft OLE DB Provider, following properties are available in the Connection tab -

Setting	Description
Select or enter a database name	Enter a server or a file name along with its location.
Enter information to log on to the database	Specify the username and password required to access the database.

For Microsoft Office 12.0 Access Database Engine OLE DB Provider, Microsoft Office 12.0 Access Database Engine OLE DB Provider, Microsoft OLE DB Provider for ODBC Drivers, Microsoft OLE DB Simple Provider, MS Data Shape, and OLE DB Provider for Microsoft Directory Services, following properties are available in the Connection tab -

Setting	Description
Enter the data source and/or location of the data	Enter a server or a file name along with its location.
Enter information to log on to server	Select whether to use Windows authentication or server authentication which requires a user name and password. Below this field you can also check the **Save my password** option for future reference.
Enter initial catalog to use	Specify the name of the database you want to connect.

For Microsoft OLE DB Provider for SQL Server and SQL Server Native Client 11.0, following properties are available in the Connection tab -

Setting	Description
Select or enter server name	Select your server from the drop down list.
Enter information to log on to the server	Select whether to use Windows authentication or server authentication which requires a user name and password. Below this field you can also check the **Save my password** option for future reference.
Select the database on the server	Select a database from the server or attach a database file.

The **Advanced** tab gives access to the initialization properties required by the chosen OLEDB provider.

Setting	Description
Impersonation Level	Set the impersonation level to any of the following options - Anonymous, Delegate, Identify, and Impersonate.
Protection Level	From the drop down, choose Call, Connect, None, Pkt, Pkt Integrity, or Pkt Privacy.
Connect Timeout	Specify the amount of time (in seconds) for a connection to establish.
Access Permissions	Specify the access permissions on the database, such as Read, ReadWrite, Share Deny None, Share Deny Read, and Share Deny Write.

Note: In case of SQL Server Native Client 11.0 provider, only the **Connect Timeout** setting is available.

The **All** tab lets you view and edit the initialization properties available for the selected OLEDB provider. These properties can differ based on the OLEDB provider you want to use.

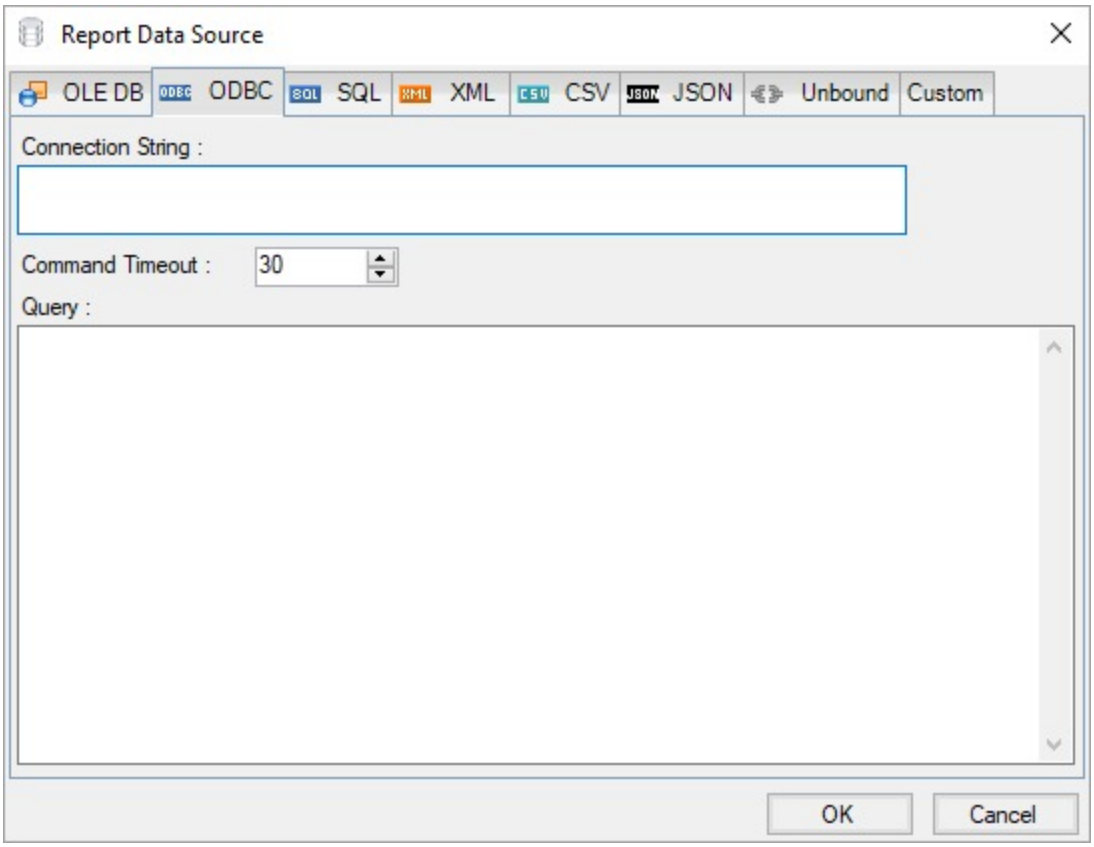
ODBC Provider

This article explains connecting a Section report to an ODBC data source.

Note: The ODBC model depends on the installed drivers.

Connect to an ODBC Data Source

1. In the designer, select the **Data Source** icon on the Detail section of the report or click the gray area around the design surface and select the **Edit Data Source** link in the Properties pane.
2. In the **Report Data Source** dialog that appears, select the **ODBC** tab to connect to an ODBC data source.



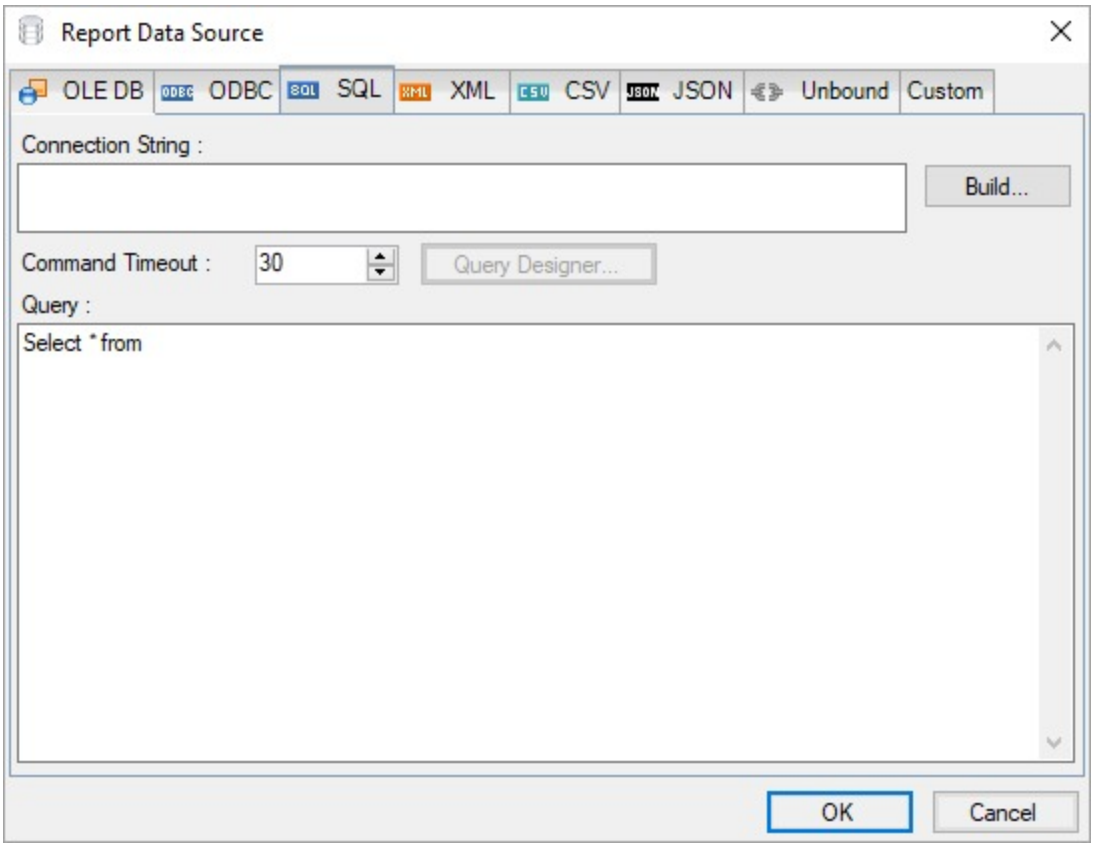
3. Under the Connection section, enter the connection string to connect to an ODBC data source. The following sample connection string specifies the type of the ODBC Driver along with location of the file required for an ODBC data source connection.
   For example, you can connect to the NWIND.mdb sample data source which can be downloaded from GitHub.
   `Driver=Microsoft Access Driver (*.mdb);Dbq=C:\NWIND.mdb;`
4. In the **Command Timeout**, specify the wait time (in seconds) for a command to execute. The default value is 30 seconds.
5. On the same page under the Query section, enter a SQL query to retrieve the data from the connected data source. For example,
   `SELECT * FROM Orders`
6. Click the **OK** button to save the changes and close the **Report Data Source** dialog box.

SQL Provider

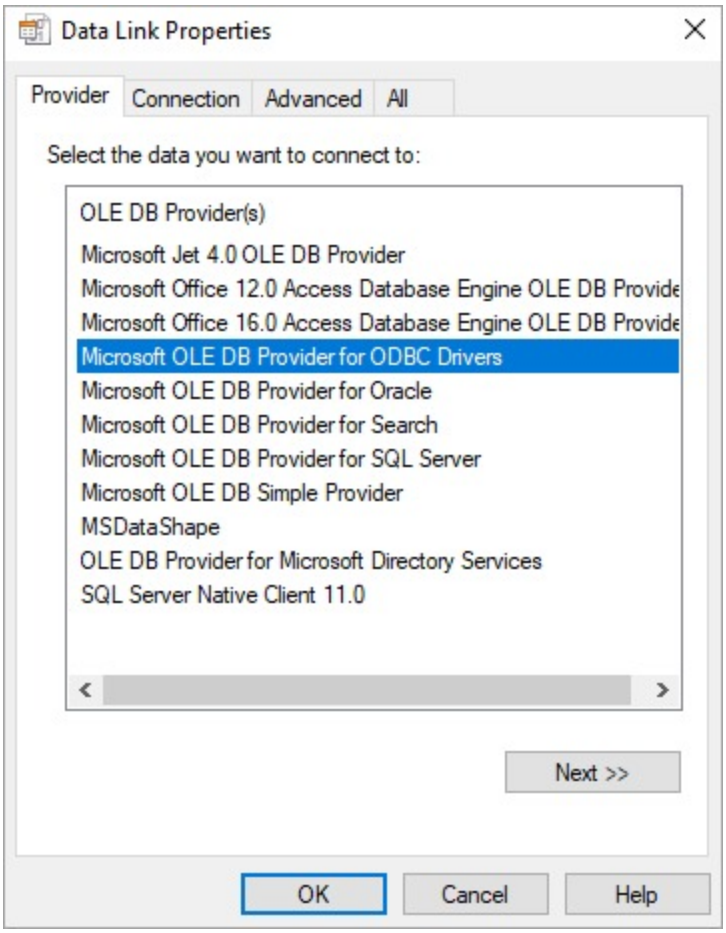
This article explains connecting a Section report to a SQL data source.

Connect to a SQL Data Source

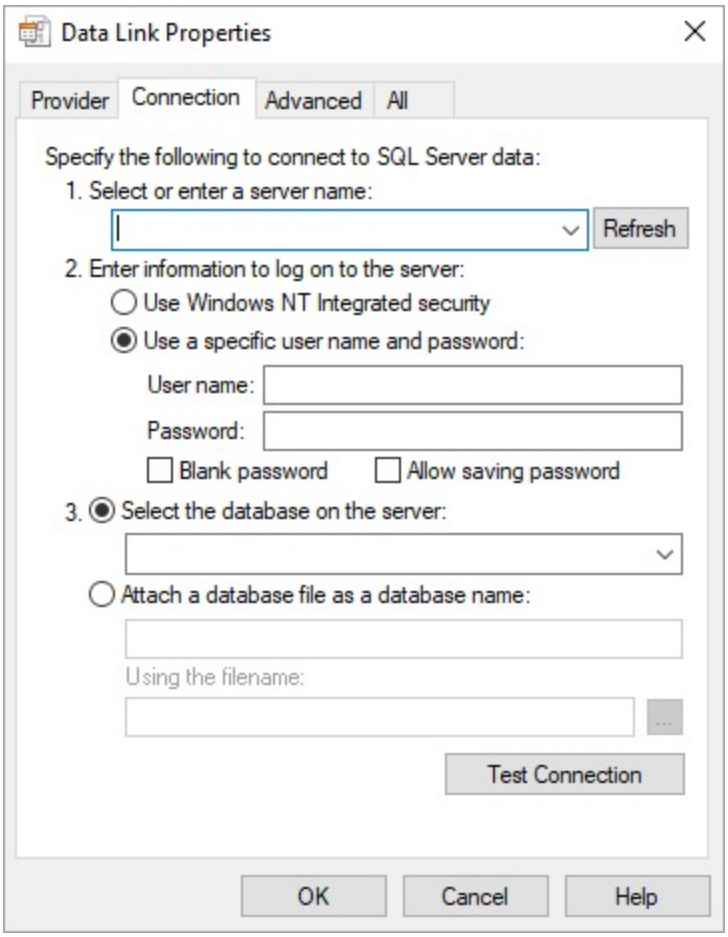
1. In the designer, select the **Data Source** icon on the Detail section of the report or click the gray area around the design surface and select the **Edit Data Source** link in the Properties pane.
2. In the **Report Data Source** dialog that appears, select the **SQL** tab to connect to a SQL data source.



3. Click the **Build** button next to the Connection String section to open the **Data Link Properties** dialog box.



4. In the **Providers** tab, specify the OLEDB provider you want to use to connect to the data source. For example, select **Microsoft OLE DB Provider for SQL Server** and click the **Next** button to move to the **Connection** tab. The fields in the Connection tab depends on the chosen OLEDB provider. For more information, see the **Configuration Settings for SQL Data Source**.
5. In the **Connection** tab, enter the server name that you want to connect.



6. To specify the authentication method for the data source connection, select the **Use Windows NT Integrated security** or **Use a specific username and password** option.
   If you choose the **Use a specific username and password** option, enter the user name and password in the respective fields. Note: The **User name** and **Password** fields are disabled in case of Windows authentication.
7. Click the dropdown next to the **Select the database on the server** field and choose the database you want to use.
8. You can verify the data source connection through the **Test Connection** button.
9. Then, click the **OK** button to close the **Data Link Properties** dialog box.
   The **Connection String** section displays the generated connection string as shown below:
   `data source=10.64.1.85\sql_2k8r2;persist security info=False;initial catalog=auditing;user id=sa;password=**********;`
10. In the **Query** field on the **SQL** tab, enter a SQL query to fetch the required data from the connected database. For example,
    `SELECT * FROM Location`
    You can also click the **Query Designer** button to access the Visual Query Designer for creating SQL queries. For information on how to create a query using the interactive query designer, visit the Visual Query Designer article.
11. Click the **OK** button to save the changes.

Configuration Settings for SQL Data Source

The SQL Data Provider provides the following configuration settings in the **Report Data Source** dialog.

The **Provider** tab describes the type of OLEDB provider you want to use for connecting to a data source. You can choose from the following -

- Microsoft JET 4.0 OLE DB Provider
- Microsoft Office 12.0 Access Database Engine OLE DB Provider
- Microsoft Office 16.0 Access Database Engine OLE DB Provider
- Microsoft OLE DB Provider for ODBC Drivers
- Microsoft OLE DB Provider for Oracle
- Microsoft OLE DB Provider for Search
- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Simple Provider
- MSDataShape
- OLE DB Provider for Microsoft Directory Services
- SQL Server Native Client 11.0

The **Connection** tab specifies the properties required for accessing the data. The properties in the tab depend upon the OLEDB provider you want to use.

For Microsoft JET 4.0 OLE DB Provider and Microsoft OLE DB Provider, following properties are available in the Connection tab -

Setting	Description
Select or enter a database name	Enter a server or a file name along with its location.
Enter information to log on to the database	Specify the username and password required to access the database.

For Microsoft Office 12.0 Access Database Engine OLE DB Provider, Microsoft Office 12.0 Access Database Engine OLE DB Provider, Microsoft OLE DB Provider for ODBC Drivers, Microsoft OLE DB Simple Provider, MSDataShape, and OLE DB Provider for Microsoft Directory Services, following properties are available in the Connection tab -

Setting	Description
Enter the data source and/or location of the data	Enter a server or a file name along with its location.
Enter information to log on to server	Select whether to use Windows authentication or server authentication which requires a user name and password. Below this field you can also check the **Save my password** option for future reference.
Enter initial catalog to use	Specify the name of the database you want to connect.

For Microsoft OLE DB Provider for SQL Server and SQL Server Native Client 11.0, following properties are available in the Connection tab -

Setting	Description
Select or enter server name	Select your server from the drop down list.
Enter information to log on to the	Select whether to use Windows authentication or server authentication which requires a user name and password. Below this field you can also check the **Save my password** option for future

server	reference.
Select the database on the server	Select a database from the server or attach a database file.

The **Advanced** tab gives access to the initialization properties required by the chosen OLEDB provider.

Setting	Description
Impersonation Level	Set the impersonation level to any of the following options - Anonymous, Delegate, Identify, and Impersonate.
Protection Level	From the drop down, choose Call, Connect, None, Pkt, Pkt Integrity, or Pkt Privacy.
Connect Timeout	Specify the amount of time (in seconds) for a connection to establish.
Access Permissions	Specify the access permissions on the database, such as Read, ReadWrite, Share Deny None, Share Deny Read, and Share Deny Write.

Note: In case of SQL Server Native Client 11.0 provider, only the **Connect Timeout** setting is available.

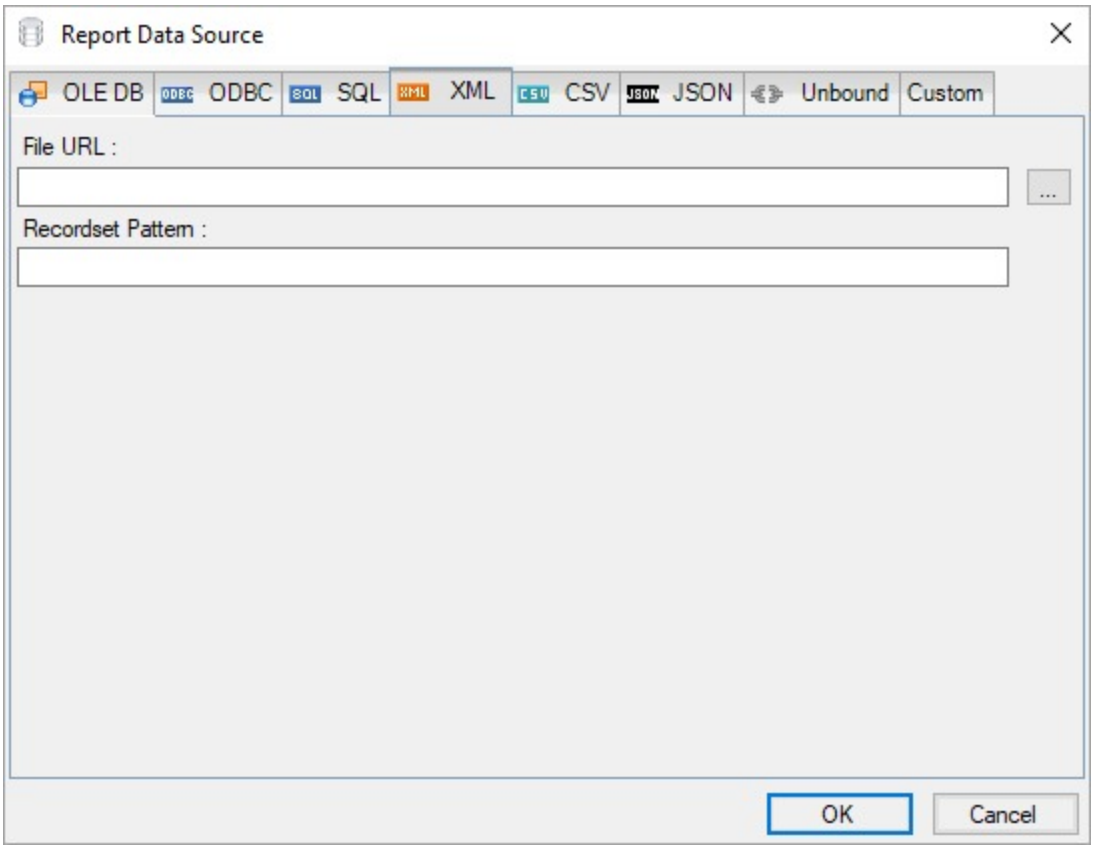
The **All** tab lets you view and edit the initialization properties available for the selected OLEDB provider. These properties can differ based on the OLEDB provider you want to use.

XML Provider

This article explains connecting a section report to an XML data source.

Connect to an XML Data Source

1. In the designer, select the **Data Source** icon on the Detail section of the report or click the gray area around the design surface and select the **Edit Data Source** link in the Properties pane.
2. In the **Report Data Source** dialog that appears, select the **XML** tab to connect to an XML data source.



3. Click the Ellipsis (...) button next to **File URL** field select the <Browse...> option to specify the XML file path. For example, you can connect to the Factbook.xml sample data source which can be downloaded from GitHub.
4. In the **Recordset Pattern** field, enter a valid XPath expression. For example,

XPath Expression
`//countries`

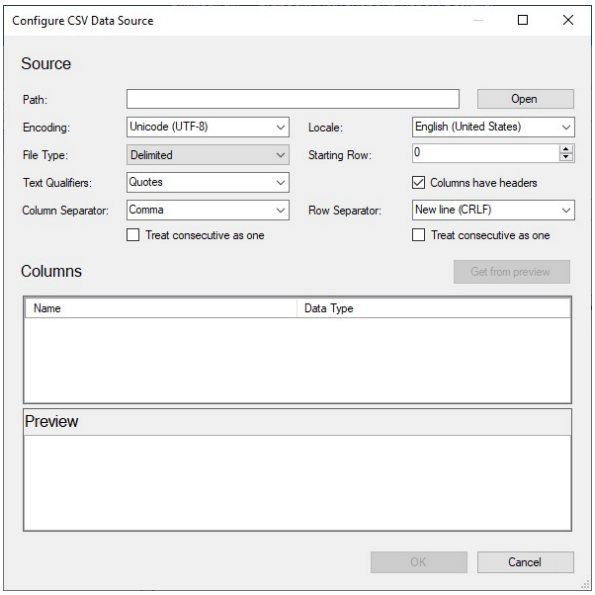
   For more information about the XPath expressions, please see https://www.w3schools.com/xml/xpath_intro.asp.
5. Click the **OK** button to close the **Report Data Source** dialog. Your report is now connected to the XML data source successfully.

CSV Provider

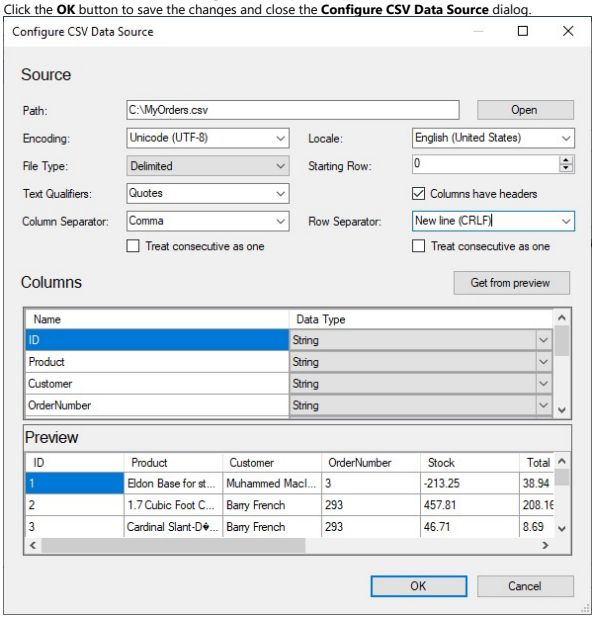
This article explains connecting a section report to a CSV data source.

Connect to a CSV Data Source

1. In the designer, select the **Data Source** icon on the Detail section of the report or click the gray area around the design surface and select the **Edit Data Source** link in the Properties pane.
2. In the **Report Data Source** dialog that appears, select the **CSV** tab to connect to a CSV data source.
3. Click the **Build** button next to the Connection String section to open the **Configure CSV Data Source** dialog box.



4. To specify the **Path** of the file, click the **Open** button and navigate to the desired folder on your system. For example, you can connect to the MyOrders.csv sample data source which can be downloaded from GitHub.
5. Select the **Column Separator** as **Comma** from the drop-down menu.
6. Click the **Get from Preview** button to fill the Columns area with the column names and their corresponding data types (string by default) present in the CSV file. This allows you to modify the name and data type (like String, Boolean, DateTime, Integer, Float, Decimal, Double, or Long) for the columns.
   For more information, see the **Configuration Properties for CSV Data Source** section.
7. Click the **OK** button to save the changes and close the **Configure CSV Data Source** dialog.



The **Connection String** section displays the generated connection string as shown below:

Connection String
`Path=C:\\MyOrders.csv;Locale=en-IN;TextQualifier=";ColumnsSeparator=,;RowsSeparator=\r\n;Columns=ID,Product,Customer,OrderNumber,Stock,Total(Decimal),UnitPrice(Decimal),City,ProductLine,Discount(Decimal);HasHeaders=True`

8. Click the **OK** button to close the **Report Data Source** dialog. Your report is now connected to the CSV data source successfully.

Configuration Settings for CSV Data Source

The CSV Data Provider provides the following configuration settings in the **Configure CSV Data Source** dialog. Based on the defined configuration settings, the CSV connection string is generated.

Setting	Description	Example
Path	Path to the CSV file - both local and relative; or a URL for centrally located CSV data sources.	C:\MyOrders.csv
Encoding	Specify the character encoding used in the CSV file.	Unicode (UTF-8)
Locale	Specify the locale used in the CSV file.	English (United States)
File Type	Define the type of CSV file. You can choose from Fixed and Delimited options.	Delimited
Starting Row	Row number to start fetching data.	0
Text Qualifiers	Character to specify where the text begins and ends, that is, the character that encloses values in the CSV file. You can choose from Quotes and Single quotes options.	Quotes
Columns have headers	Specify whether the CSV file has columns with headers or not.	Checked

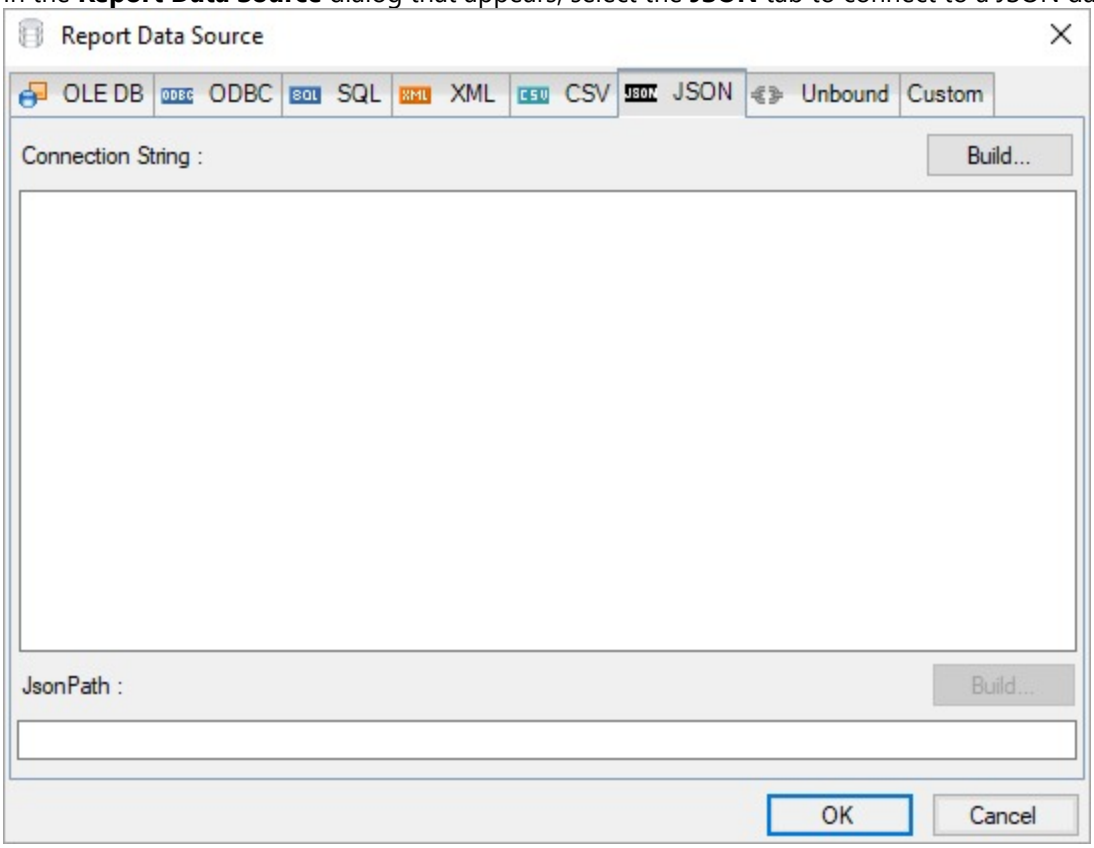
Column Separator	Specify the symbol used to separate the columns in the CSV file. You can choose from Comma, Semicolon, Tab, and Space options.	Comma
Row Separator	Symbol used to separate the rows in the CSV file. You can choose from CRLF (carriage return and line feed), CR (carriage return), and LF (line feed) new line formats.	New line (CRLF)
Treat consecutive as one	Specify whether to join the column separators or row separators as one.	Checked for Column separator Unchecked for Row Separator
Get from preview	Fills the Columns area with names and data types (string by default) for columns present in the CSV file. This allows you to modify the name and data type (like String, Boolean, DateTime, Integer, Float, Decimal, Double, or Long) for the columns.	UnitPrice(Decimal)
Note: Text Qualifiers, **Column Separator, Row Separator**, and **Treat Consecutive as one** options are not available for **Fixed** file type.		

JSON Provider

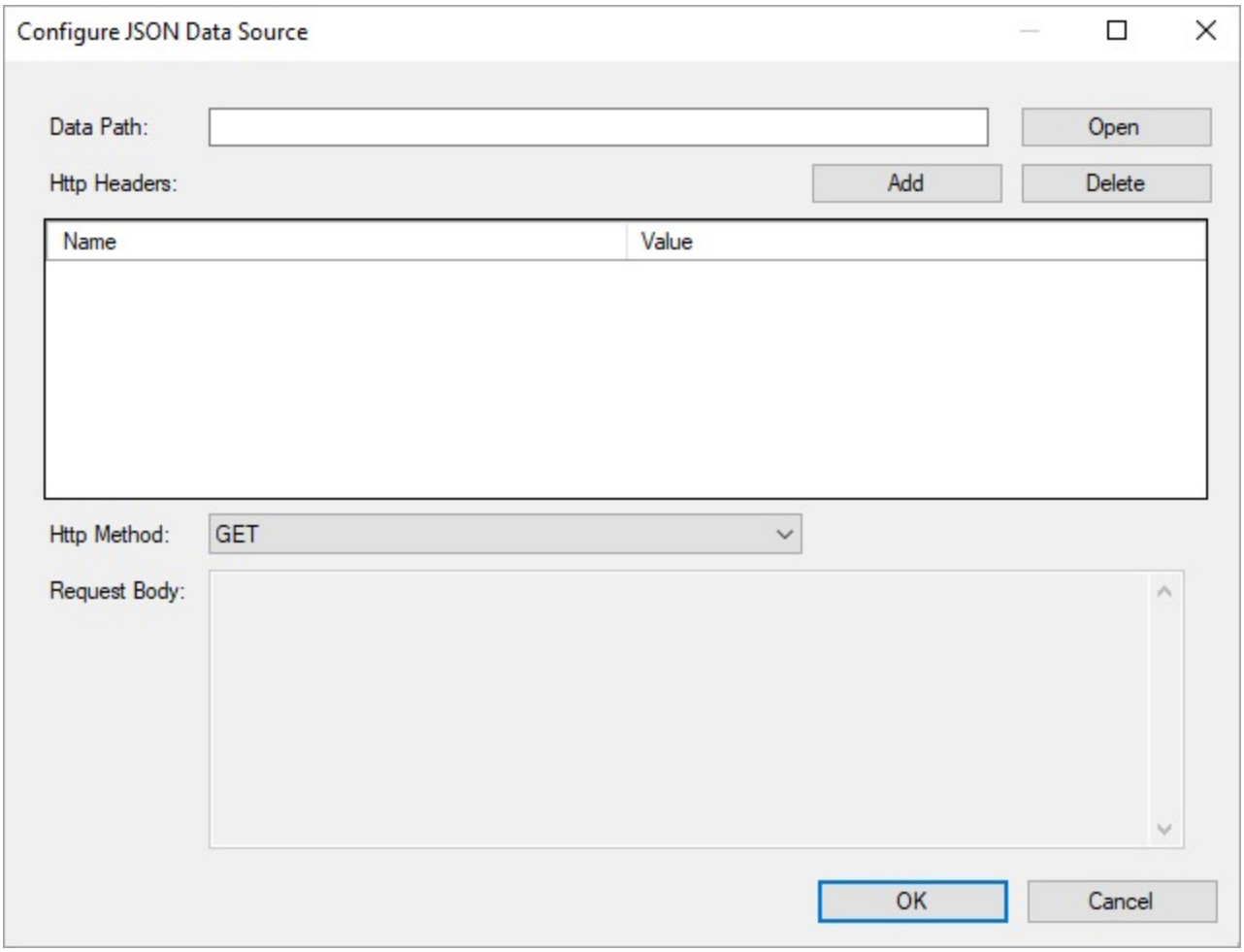
This article explains connecting a Section report to a JSON data source.

Connect to a JSON Data Source

1. In the designer, select the **Data Source** icon on the Detail section of the report or click the gray area around the design surface and select the **Edit Data Source** link in the Properties pane.
2. In the **Report Data Source** dialog that appears, select the **JSON** tab to connect to a JSON data source.



3. Click the **Build** button next to the Connection String section to open the **Configure JSON Data Source** dialog box.



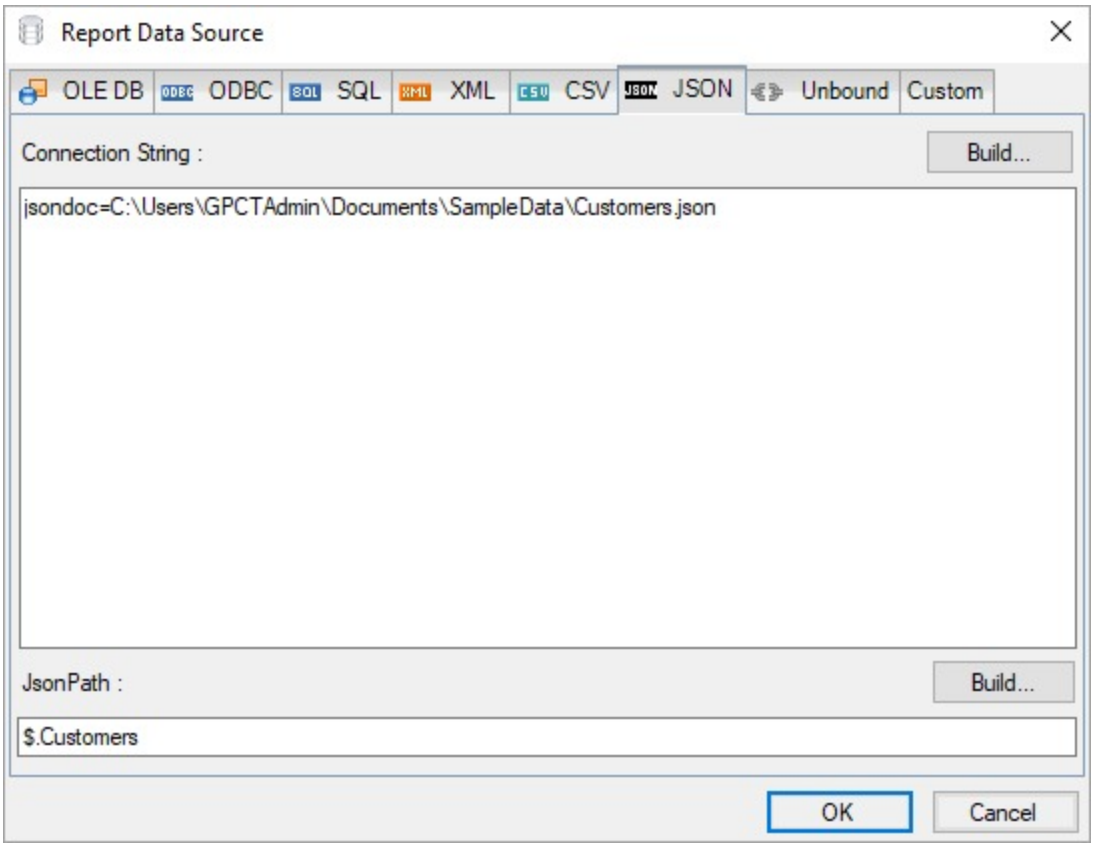
4. To specify the **Data Path** of the file, click the **Open** button and navigate to the desired folder on your system. For example, you can connect to the 'Customers.json' sample data source which can be downloaded from GitHub.
   For more information, see the **Configuration Settings for JSON Data Source**.
5. Click **OK** to save the changes and close the **Configure JSON Data Source** dialog box.
   The Connection String section displays the generated connection string as shown below.

Connection String
`jsondoc=C:\Customers.json`

6. In the **JSON Path** field, enter a valid JSON Path expression or click the **Build** button to generate the path using the JSON query designer.

JSON Path
`$.Customers`

   For more information on JSON Path expressions, please visit this article.
7. Click the **OK** button to save the changes.



Configuration Settings for JSON Data Source

The JSON Data Provider provides the following configuration settings in the **Configure JSON Data Source** dialog box.

- **Data Path**: Path or URL of an external JSON data file or select the file available on the system using the **Open** button. You can also pass parameters (including report parameters) to the URL. For example, `https://demodata.mescius.io/northwind/odata/v1/customers?$top=7&$orderby=Country`. The connection string generated starts with the keyword **jsondoc**.
- **HTTP Headers**: Contains information related to authorization. Here you provide the expected content type, bearer token, etc. necessary for establishing the connection. For example, providing credentials to log in to a service/URI to retrieve data.
- **HTTP Method**: The HTTP request method, it can either be set to GET (included in URL) or POST (included in Request Body). POST requests allow defining a body as an expression to obtain data.
- **Request Body**: Body for the POST request method. The text area is enabled only when the **HTTP Method** is set to POST.

Unbound Provider

The data source and data set for Unbound Provider data type can be set at run time. For more information, see Bind Data Set and Unbound Data to Section Report at Run Time.

Design Reports

An important milestone in the process of report authoring is designing a report. ActiveReports .NET lets you design

Page/RDLX as well as Section reports. This topic gives you an overview about the tutorials, aimed at designing different types of reports in ActiveReports.

Design Page/RDLX Reports

Learn how to design Page/RDLX reports by following tutorials that explain designing Page/RDLX reports from various aspects.

Design Section Reports

Learn how to design Section reports. In this section, you will find various tutorials on designing Section report, including sophisticated scenarios.

Design Page/RDLX Reports

ActiveReports Designer supports designing useful Page/RDLX reports. This section covers a wide range of topics to transform data into meaningful insights:

Layout

Learn the different types of layout options for your Page/RDLX reports.

Report Dialog

Learn about the features available through report dialog in Page/RDLX reports.

Master Report (RDLX Report)

Learn about using master reports.

Layers

Learn about layers that can overlay with other group of controls on a Page or an RDLX report.

Expressions

Learn about expressions that you can use in reports.

Data Visualizers

Learn about Data Visualizers that let you create small graphs to make data easily comprehensible.

Filters

Learn about filters in Page and RDLX reports,

Group Data

Learn about grouping data in Page and RDLX reports,

Interactivity

Learn about the features in RDLX and Page reports that support interactive capabilities.

Report Appearance

Learn about customizing the appearance of reports.

Tutorials: Report Controls in Page/RDLX Reports

Learn how to design the Page and RDLX reports in the Report Designer.

Tutorials: Page/RDLX report Scenarios

Learn how to use Page/RDLX report tutorials to create different types of report scenarios.

Layout

ActiveReports provides a wide range of layout options for your Page/RDLX report that you can easily control by using the specific properties.

Page/RDLX reports allow to create a very powerful layout. Here we show a few base use-cases and tutorials. ActiveReports provides a wide range of layout options for your Page/RDLX report that you can easily control by using the specific properties.

In this section

Keep Groups Together in Data Regions
: You can keep the contents of a group together on a single page by setting the **Keep Together on one page if possible** option (Table group) or the **KeepTogether** property (Tablix row group).

Set Page Size, Margins, and Orientation in Page Reports
: Learn about the options to control page layout properties such as page size, page margins, and page orientation. These options allow you to define the layout settings for each page in a Page report.

Set Page Layout in Z- or N-Order
: A report may have horizontally or vertically growing data rendered across multiple pages during the preview. In this case you can use the **LayoutPagesOrder** property of the report to specify the page layout order for a Page or an RDLX report. ActiveReports supports two types of page layout order, Z-Order or N-Order.

Repeat Blank Rows in a Table Data Region in Page Reports
: Learn about the **RepeatBlankRows** property and how to use it when creating different report layouts.

Page Breaks in Data Regions
: You can control page breaks in data groups of data regions by using the **NewPage** property for data regions and its combination with the **PageBreakAtStart**, **PageBreakAtEnd**, and **BreakLocation** properties. These properties help you tune page breaks after or before a data group in your report and on which page to continue displaying the report content.

Skip Page Generation in Page Reports
: The topic shows how to hide or show a page when you preview a Page report.

Hide or Show Sections in RDLX and RDLX Dashboard Reports
: The topic shows how to hide or show the sections in reports with multiple sections.

Keep Groups Together in Data Regions

In Table and Tablix data regions of Page and RDLX reports, you can choose to keep the contents of a group together on a single page.

- For the Table data region, this feature is available as the **Keep Together on one page if possible** option for a group's layout.
- For the Tablix data region, this property is available as **KeepTogether** property in the properties panel for a row group's layout.

With the option selected or the property set to true, you can keep the content of a group (details row along with the group header and the group footer) together on the same page. For example, you can choose to print the grouped data - all on one page instead of printing across multiple pages. The group is attempted to be printed on a single page with no page breaks if possible. In case the group is too large to fit on the current page or the next page, the feature is ignored and the groups split across the report pages.

Table Data Region

In the following image, **Keep Together on one page if possible** option is selected for a table group in Page report, hence the table groups appear together (if possible) across the pages.

(For an enlarged image view, open the image in a separate tab.)

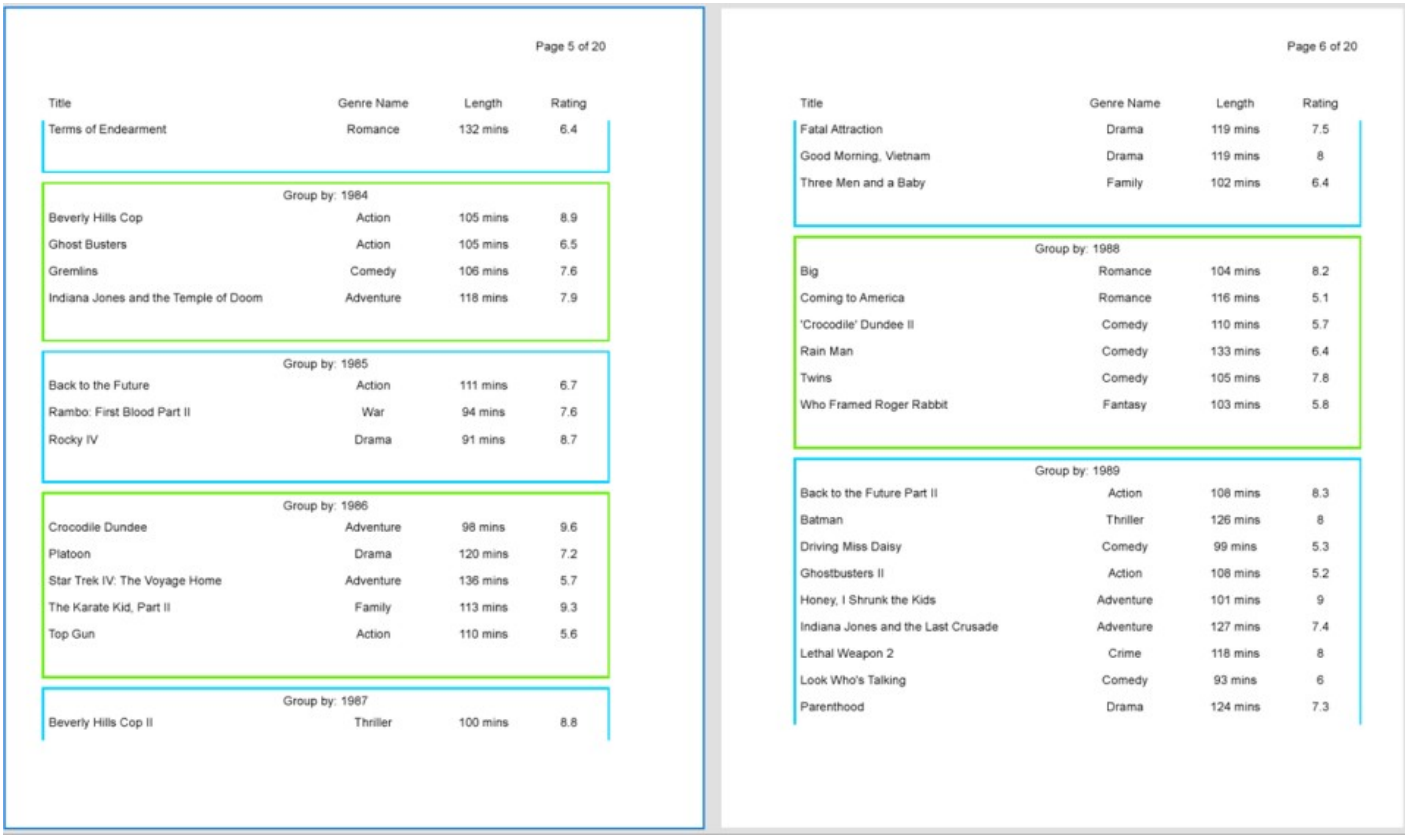
Page 5 of 25

Title	Genre Name	Length	Rating
Group by: 1982			
An Officer and a Gentleman	Romance	122 mins	8.6
E.T. the Extra-Terrestrial	Drama	120 mins	7.3
Porky's	Comedy	94 mins	9.1
Rocky III	Action	99 mins	5.4
Tootsie	Romance	119 mins	8.8
Group by: 1983			
Star Wars: Episode VI - Return of the Jedi	Fantasy	134 mins	7.6
Terms of Endearment	Romance	132 mins	6.4
Group by: 1984			
Beverly Hills Cop	Action	105 mins	8.9
Ghost Busters	Action	105 mins	6.5
Gremlins	Comedy	106 mins	7.6
Indiana Jones and the Temple of Doom	Adventure	118 mins	7.9
Group by: 1985			
Back to the Future	Action	111 mins	6.7
Rambo: First Blood Part II	War	94 mins	7.6
Rocky IV	Drama	91 mins	8.7

Page 6 of 25

Title	Genre Name	Length	Rating
Group by: 1986			
Crocodile Dundee	Adventure	98 mins	9.6
Platoon	Drama	120 mins	7.2
Star Trek IV: The Voyage Home	Adventure	136 mins	5.7
The Karate Kid, Part II	Family	113 mins	9.3
Top Gun	Action	110 mins	5.6
Group by: 1987			
Beverly Hills Cop II	Thriller	100 mins	8.8
Fatal Attraction	Drama	119 mins	7.5
Good Morning, Vietnam	Drama	119 mins	8
Three Men and a Baby	Family	102 mins	6.4
Group by: 1988			
Big	Romance	104 mins	8.2
Coming to America	Romance	116 mins	5.1
'Crocodile' Dundee II	Comedy	110 mins	5.7
Rain Man	Comedy	133 mins	6.4
Twins	Comedy	105 mins	7.8
Who Framed Roger Rabbit	Fantasy	103 mins	5.8

In the following image, **Keep Together on one page if possible** option is unselected for a table group in Page report, hence the table groups split across the pages.
(For an enlarged image view, open the image in a separate tab.)



Page 5 of 20

Title	Genre Name	Length	Rating
Terms of Endearment	Romance	132 mins	6.4
Group by: 1984			
Beverly Hills Cop	Action	105 mins	8.9
Ghost Busters	Action	105 mins	6.5
Gremlins	Comedy	106 mins	7.6
Indiana Jones and the Temple of Doom	Adventure	118 mins	7.9
Group by: 1985			
Back to the Future	Action	111 mins	6.7
Rambo: First Blood Part II	War	94 mins	7.6
Rocky IV	Drama	91 mins	8.7
Group by: 1986			
Crocodile Dundee	Adventure	98 mins	9.6
Platoon	Drama	120 mins	7.2
Star Trek IV: The Voyage Home	Adventure	136 mins	5.7
The Karate Kid, Part II	Family	113 mins	9.3
Top Gun	Action	110 mins	5.6
Group by: 1987			
Beverly Hills Cop II	Thriller	100 mins	8.8

Page 6 of 20

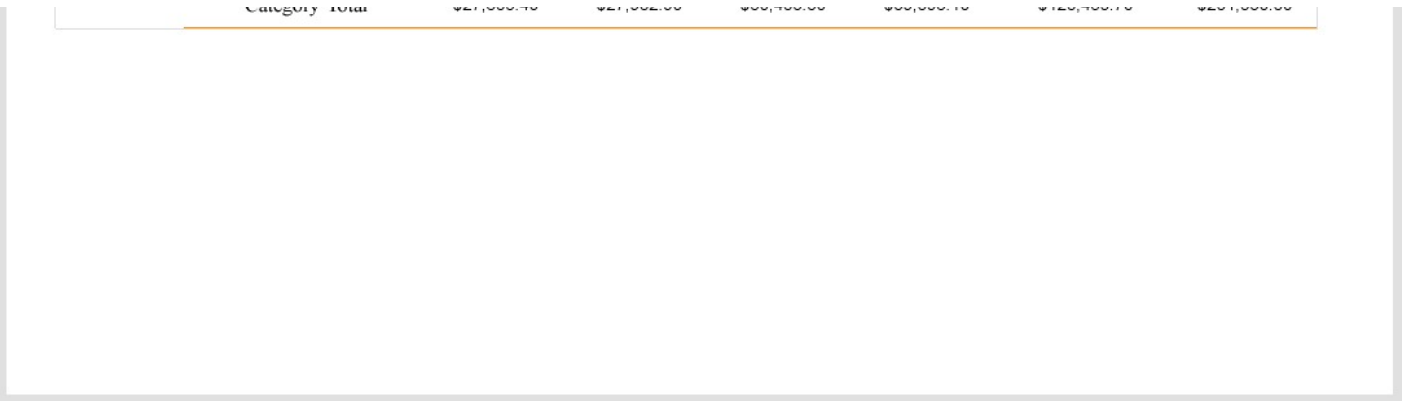
Title	Genre Name	Length	Rating
Fatal Attraction	Drama	119 mins	7.5
Good Morning, Vietnam	Drama	119 mins	8
Three Men and a Baby	Family	102 mins	6.4
Group by: 1988			
Big	Romance	104 mins	8.2
Coming to America	Romance	116 mins	5.1
'Crocodile' Dundee II	Comedy	110 mins	5.7
Rain Man	Comedy	133 mins	6.4
Twins	Comedy	105 mins	7.8
Who Framed Roger Rabbit	Fantasy	103 mins	5.8
Group by: 1989			
Back to the Future Part II	Action	108 mins	8.3
Batman	Thriller	126 mins	8
Driving Miss Daisy	Comedy	99 mins	5.3
Ghostbusters II	Action	108 mins	5.2
Honey, I Shrunk the Kids	Adventure	101 mins	9
Indiana Jones and the Last Crusade	Adventure	127 mins	7.4
Lethal Weapon 2	Crime	118 mins	8
Look Who's Talking	Comedy	93 mins	6
Parenthood	Drama	124 mins	7.3

Tablix Data Region

In the following image, **KeepTogether** property for the parent row group of tablix in Page report is set to True, hence the tablix row group appear together (if possible) across the pages.

Orders by Category, Quarter, and Year		1995					Product Total
		Q 1	Q 2	Q 3	Q 4	Total for the Year	
Grains/Cereals	Singaporean Hokkien Fried Mee	$448.00	$1,484.00	$1,862.00	$1,736.00	$5,530.00	$9,332.40
	Gustaf's Knäckebröd	$201.60	$504.00	$3,003.00	$420.00	$4,128.60	$7,232.40
	Ravioli Angelo	$546.00	$97.50	$585.00	$1,053.00	$2,281.50	$7,807.80
	Gnocchi di nonna Alice	$6,870.40	$8,177.60	$8,626.00	$8,436.00	$32,110.00	$45,121.20
	Wimmers gute Semmelknödel	$2,872.80	$1,010.80	$2,094.75	$2,227.75	$8,206.10	$23,009.00
	Filo Mix	$308.00	$42.00	$931.00	$812.00	$2,093.00	$3,383.80
	Tunnbröd	$720.00	$1,141.20	$90.00	$729.00	$2,680.20	$4,840.20
	Category Total	$11,966.80	$12,457.10	$17,191.75	$15,413.75	$57,029.40	$100,726.80

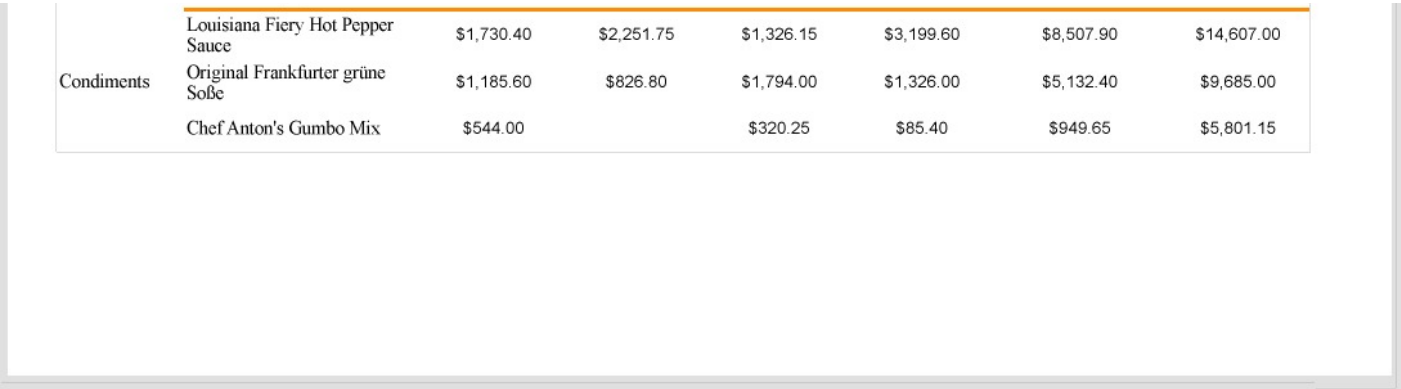
Orders by Category, Quarter, and Year		1995					Product Total
		Q 1	Q 2	Q 3	Q 4	Total for the Year	
Dairy Products	Mozzarella di Giovanni	$3,808.60	$2,609.00	$4,906.80	$1,461.60	$12,786.00	$25,738.80
	Queso Cabrales	$1,209.60	$2,457.00	$1,554.00	$420.00	$5,640.60	$13,902.00
	Geitost	$398.00	$196.50	$170.00	$170.00	$934.50	$1,713.50
	Camembert Pierrot	$4,651.20	$6,426.00	$6,086.00	$7,140.00	$24,303.20	$50,286.00
	Gorgonzola Telino	$1,810.00	$2,237.50	$2,137.50	$1,612.50	$7,797.50	$16,172.50
	Raclette Courdavault	$7,172.00	$8,283.00	$7,535.00	$11,880.00	$34,870.00	$76,296.00
	Mascarpone Fabioli		$2,368.00	$320.00	$448.00	$3,136.00	$9,171.20
	Queso Manchego La Pastora	$456.00	$1,710.00	$1,368.00	$5,320.00	$8,854.00	$12,866.80
	Flotemysost	$5,469.60	$399.90	$2,236.00	$3,096.00	$11,201.50	$20,876.50
	Gudbrandsdalsost	$2,678.40	$1,296.00	$4,140.00	$7,848.00	$15,962.40	$24,307.20
	Category Total	$27,653.40	$27,982.90	$30,453.30	$39,396.10	$125,485.70	$251,330.50



In the following image, **KeepTogether** property for the parent row group of tablix in Page report is set to False, hence the tablix group splits across the pages.

Orders by Category, Quarter, and Year		1995					Product Total
		Q 1	Q 2	Q 3	Q 4	Total for the Year	
Grains/Cereals	Singaporean Hokkien Fried Mee	$448.00	$1,484.00	$1,862.00	$1,736.00	$5,530.00	$9,332.40
	Gustaf's Knäckebröd	$201.60	$504.00	$3,003.00	$420.00	$4,128.60	$7,232.40
	Ravioli Angelo	$546.00	$97.50	$585.00	$1,053.00	$2,281.50	$7,807.80
	Gnocchi di nonna Alice	$6,870.40	$8,177.60	$8,626.00	$8,436.00	$32,110.00	$45,121.20
	Wimmers gute Semmelknödel	$2,872.80	$1,010.80	$2,094.75	$2,227.75	$8,206.10	$23,009.00
	Filo Mix	$308.00	$42.00	$931.00	$812.00	$2,093.00	$3,383.80
	Tunnbröd	$720.00	$1,141.20	$90.00	$729.00	$2,680.20	$4,840.20
	Category Total	$11,966.80	$12,457.10	$17,191.75	$15,413.75	$57,029.40	$100,726.80
Dairy Products	Mozzarella di Giovanni	$3,808.60	$2,609.00	$4,906.80	$1,461.60	$12,786.00	$25,738.80
	Queso Cabrales	$1,209.60	$2,457.00	$1,554.00	$420.00	$5,640.60	$13,902.00
	Geitost	$398.00	$196.50	$170.00	$170.00	$934.50	$1,713.50
	Camembert Pierrot	$4,651.20	$6,426.00	$6,086.00	$7,140.00	$24,303.20	$50,286.00
	Gorgonzola Telino	$1,810.00	$2,237.50	$2,137.50	$1,612.50	$7,797.50	$16,172.50
	Raclette Courdavault	$7,172.00	$8,283.00	$7,535.00	$11,880.00	$34,870.00	$76,296.00

Orders by Category, Quarter, and Year		1995					Product Total
		Q 1	Q 2	Q 3	Q 4	Total for the Year	
Dairy Products	Mascarpone Fabioli		$2,368.00	$320.00	$448.00	$3,136.00	$9,171.20
	Queso Manchego La Pastora	$456.00	$1,710.00	$1,368.00	$5,320.00	$8,854.00	$12,866.80
	Flotemysost	$5,469.60	$399.90	$2,236.00	$3,096.00	$11,201.50	$20,876.50
	Gudbrandsdalsost	$2,678.40	$1,296.00	$4,140.00	$7,848.00	$15,962.40	$24,307.20
	Category Total	$27,653.40	$27,982.90	$30,453.30	$39,396.10	$125,485.70	$251,330.50
Produce	Tofu	$2,306.40	$1,850.70	$1,860.00	$1,255.50	$7,272.60	$8,630.40
	Manjimup Dried Apples	$763.20	$4,812.40	$5,724.00	$5,565.00	$16,864.60	$44,742.60
	Longlife Tofu	$1,032.00	$128.00		$50.00	$1,210.00	$2,566.00
	Uncle Bob's Organic Dried Pears	$624.00	$2,520.00	$2,700.00	$2,700.00	$8,544.00	$22,464.00
	Rössle Sauerkraut	$4,550.00	$3,984.00	$2,052.00	$3,876.00	$14,462.00	$26,865.60
	Category Total	$9,275.60	$13,295.10	$12,336.00	$13,446.50	$48,353.20	$105,268.60



	Louisiana Fiery Hot Pepper Sauce	$1,730.40	$2,251.75	$1,326.15	$3,199.60	$8,507.90	$14,607.00
Condiments	Original Frankfurter grüne Soße	$1,185.60	$826.80	$1,794.00	$1,326.00	$5,132.40	$9,685.00
	Chef Anton's Gumbo Mix	$544.00		$320.25	$85.40	$949.65	$5,801.15

Set Page Size, Margins, and Orientation in Page Reports

Page layout properties such as page size, page margins, and page orientation define how the report pages appear when previewed or printed. While a report's layout can be set for a report, you can also control the layout settings for each page in a Page report.

You should note that Page reports make use of Overflow Place Holder control to handle overflowing data on multiple pages, with a data region placed on the first page of the report and the OverflowPlaceHolder controls on subsequent pages. See how data in a Page report flows from the first page to the next in Create Columnar Reports with OverflowPlaceholder Control tutorial.

Page Size

You can change the default page size using the **PageSize** property of the page from the Properties panel and set it to the standard sizes such as Letter, Tabloid, Legal, Statement, Executive, A3, A4, A5, B4 (JIS), and B5 (JIS). You can also manually input the values in the **Width** and **Height** properties to customize the page size.

Page Margins

Page margin is a space between the edge and the content on your report page. You can set the page margins using the **Margins** property of the page from the Properties panel. By default, the value of page margins (left, right, top, and bottom) is set to 1 in.

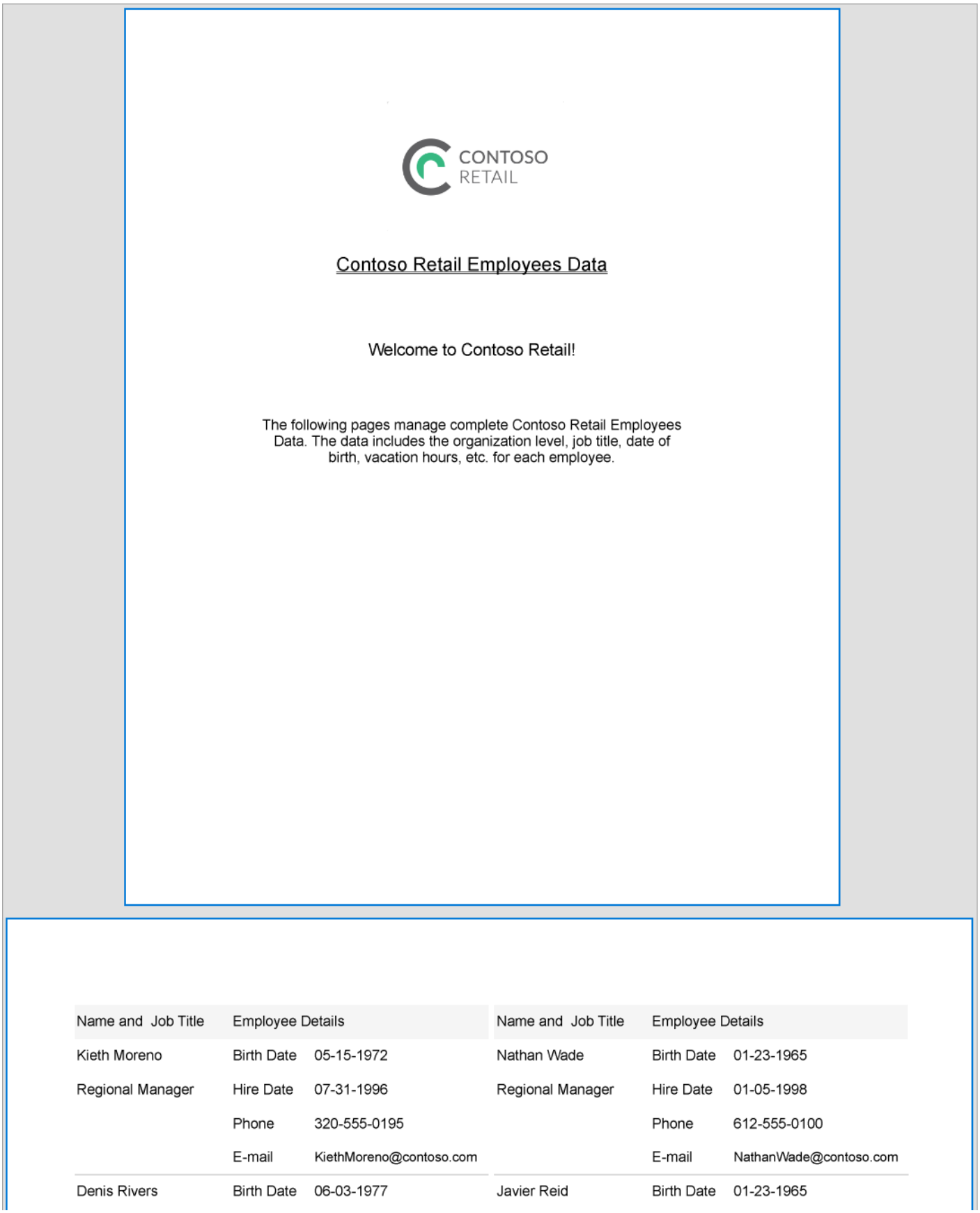
Page Orientation

The Page or paper orientation is how the rectangular report page is oriented. While **Portrait** orientation is suitable for the text expanding in a vertical direction, **Landscape** orientation is suitable for horizontally expanding content like in Tablix data regions. You can specify the report page orientation using the **PaperOrientation** property of the page from the Properties panel. By default, the **PaperOrientation** property is set to Portrait.

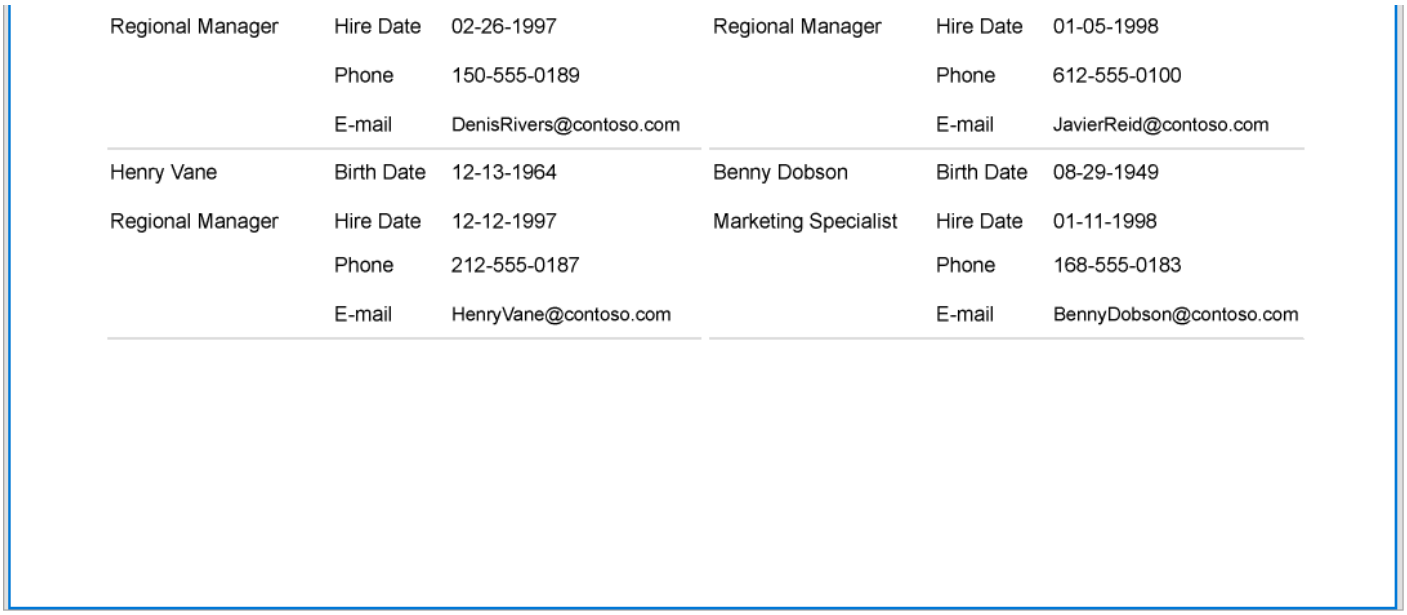
Mixed Page Orientation

You can mix Portrait and Landscape page orientations in a report depending on the data being rendered on the pages. For example, in the employee database handbook report shown below, the front page contains a general introduction inside TextBox controls about the following pages, which contain details on the employees. While the front page looks clear in Portrait, the rest of the pages where the data is wider with a Table data region and an OverflowPlaceholder

control side-by-side, the Landscape orientation makes more sense.



Name and Job Title	Employee Details		Name and Job Title	Employee Details	
Kieth Moreno	Birth Date	05-15-1972	Nathan Wade	Birth Date	01-23-1965
Regional Manager	Hire Date	07-31-1996	Regional Manager	Hire Date	01-05-1998
	Phone	320-555-0195		Phone	612-555-0100
	E-mail	KiethMoreno@contoso.com		E-mail	NathanWade@contoso.com
Denis Rivers	Birth Date	06-03-1977	Javier Reid	Birth Date	01-23-1965



Regional Manager	Hire Date	02-26-1997	Regional Manager	Hire Date	01-05-1998
	Phone	150-555-0189		Phone	612-555-0100
	E-mail	DenisRivers@contoso.com		E-mail	JavierReid@contoso.com
Henry Vane	Birth Date	12-13-1964	Benny Dobson	Birth Date	08-29-1949
Regional Manager	Hire Date	12-12-1997	Marketing Specialist	Hire Date	01-11-1998
	Phone	212-555-0187		Phone	168-555-0183
	E-mail	HenryVane@contoso.com		E-mail	BennyDobson@contoso.com

Set Page Layout in Z- or N-Order

If a report has horizontally or vertically growing data rendered across multiple pages during the preview, you may want the preview to show the report pages in a particular order.

Using the **LayoutPagesOrder** property of the report, you can specify the page layout order for a Page or an RDLX report. There are two types of page layout order, Z-Order or N-Order, supported in ActiveReports as explained below.

Z-Order

The Z-order renders the report in the shape of the alphabet 'Z'. This means the horizontally expanding data in the report is rendered first on the upcoming pages followed by the vertically expanding data.

A report in Multipage view with page layout order set to Z-order is as shown. Note the page layout order: 1A -> 1B -> 1C -> 2A -> 2B -> 2C ->3A -> 3B -> 3C.

Note: The **LayoutPagesOrder** property does not affect the page layout when exporting to CSV, JSON, XML, CSV, Excel Data, or DOCX file format.

N-Order

The N-order renders the report in the shape of the alphabet 'N'. This means the vertically expanding data in the report is rendered first on the upcoming pages followed by the horizontally expanding data.

A report in Multipage view with page layout order set to N-order is as shown. Note the page layout order: 1A -> 2A -> 3A -> 1B -> 2B -> 3B -> 1C -> 2C -> 3C.

Layout Order in RDLX Reports

In RDLX reports, controls or data regions grow vertically or horizontally to accommodate data. In an RDLX report with a Tablix data region, if the tablix data is too large to fit inside a single page, the data expands over multiple pages in both horizontal and vertical directions. The following image shows the preview of such a report in Z-order.

(For an enlarged image view, open the image in a separate tab.)

Ship Country	Ship Name	1996			1997	
		Q3	Q4	Total	Q1	Q2
France	Vins et alcools Chevalier	$39.54		$39.54		
	Victuailles en stock	$41.34	$8.56	$49.90	$37.13	$194.72
	Blondel père et fils	$61.02	$131.70	$192.72	$209.96	$155.59
	Du monde entier	$24.69		$24.69		
	Bon app'		$272.54	$272.54	$64.56	$361.70
	La maison d'Asie		$84.28	$84.28	$106.33	$53.32
	Folies gourmandes				$12.61	
	France restauration					
	Spécialités du monde					
	La corne d'abondance					
Germany	Toms Spezialitäten	$11.61		$11.61	$22.92	$1.43
	Ottilies Käseladen	$55.09		$55.09	$91.48	$201.85
	Frankenversand	$208.58	$298.44	$507.02	$4.93	$36.65
	QUICK-Stop	$384.10	$432.23	$816.33	$206.64	$1,425.25
	Morgenstern Gesundkost	$125.77		$125.77		$127.34
	Lehmanns Marktstand	$102.39	$110.37	$212.76		$265.08

1997			1998			Total Freight Charges
Q3	Q4	Total	Q1	Q2	Total	
	$18.87	$18.87				$58.41
	$22.11	$253.96	$189.39		$189.39	$493.25
$58.30		$423.85	$7.09		$7.09	$623.66
$6.25		$6.25	$32.76		$32.76	$63.70
$113.15	$117.00	$656.41	$390.64	$38.28	$428.92	$1,357.87
$27.65	$249.93	$437.23	$111.52	$2.79	$114.31	$635.82
$487.38	$137.95	$637.94				$637.94
$30.34		$30.34	$141.08		$141.08	$171.42
	$2.91	$2.91	$96.57	$8.80	$105.37	$108.28
			$87.49		$87.49	$87.49
$27.79		$52.14	$62.22		$62.22	$125.97
$145.63	$157.55	$596.51	$71.49	$139.65	$211.14	$862.74
$425.76	$76.10	$543.44	$110.03	$242.95	$352.98	$1,403.44
$558.82	$1,346.29	$3,537.00	$915.49	$336.81	$1,252.30	$5,605.63
	$58.71	$186.05	$10.22		$10.22	$322.04
$206.30	$91.28	$562.66	$105.61	$136.00	$241.61	$1,017.03

Germany	Die Wandernde Kuh	$85.34	$37.49	$122.83		$105.65
	Königlich Essen		$69.74	$69.74	$63.81	$32.14
	Drachenblut Delikatessen		$35.99	$35.99		
	Blauer See Delikatessen					$36.71
	Alfreds Futterkiste					
	Alfred's Futterkiste					
Brazil	Hanari Carnes	$124.00		$124.00		$68.65
	Wellington Importadora	$13.97		$13.97	$44.12	
	Que Delicia	$9.45	$45.03	$54.48	$99.23	
	Ricardo Adocicados	$42.52		$42.52	$132.99	$60.43
	Comércio Mineiro	$79.70		$79.70	$11.93	$65.99
	Tradiçao Hipermercados	$1.35		$1.35		$46.77
	Familia Arquibaldo		$17.09	$17.09	$21.48	$6.54
	Queen Cozinha		$890.78	$890.78	$179.11	
	Gourmet Lanchonetes				$24.50	

$132.75		$238.40		$71.64	$71.64	$432.87
$32.35	$201.64	$329.94	$384.41	$29.59	$414.00	$813.68
	$33.35	$33.35	$79.25	$157.45	$236.70	$306.04
$1.93		$38.64	$98.48	$31.14	$129.62	$168.26
$29.46		$29.46				$29.46
	$84.96	$84.96	$109.95	$1.21	$111.16	$196.12
$12.41	$146.10	$227.16	$300.08	$73.53	$373.61	$724.77
$13.55	$55.23	$112.90	$67.84		$67.84	$194.71
$108.06	$31.02	$238.31	$34.76		$34.76	$327.55
$65.22		$258.64	$245.99	$85.80	$331.79	$632.95
		$77.92	$0.21	$29.99	$30.20	$187.82
$79.40		$126.17	$147.04		$147.04	$274.56
$176.81	$10.83	$215.66				$232.75
$307.10	$173.98	$660.19	$349.98	$81.75	$431.73	$1,982.70
$40.89	$243.67	$309.06	$4.98	$8.34	$13.32	$322.38

Belgium	Suprêmes délices	$57.57		$57.57	$230.36	
	Maison Dewey					$66.69
Switzerland	Chop-suey Chinese	$22.98	$1.17	$24.15		$91.76

	$1.59	$231.95	$501.95	$29.76	$531.71	$821.23
$6.20	$155.64	$228.53	$185.54	$44.84	$230.38	$458.91
	$128.08	$219.84	$27.19	$96.06	$123.25	$367.24

The following image shows the preview of the above report in N-order.

(For an enlarged image view, open the image in a separate tab.)

Ship Country	Ship Name	1996			1997	
		Q3	Q4	Total	Q1	Q2
France	Vins et alcools Chevalier	$39.54		$39.54		
	Victuailles en stock	$41.34	$8.56	$49.90	$37.13	$194.72
	Blondel père et fils	$61.02	$131.70	$192.72	$209.96	$155.59
	Du monde entier	$24.69		$24.69		
	Bon app'		$272.54	$272.54	$64.56	$361.70
	La maison d'Asie		$84.28	$84.28	$106.33	$53.32
	Folies gourmandes				$12.61	
	France restauration					
	Spécialités du monde					
	La corne d'abondance					
Germany	Toms Spezialitäten	$11.61		$11.61	$22.92	$1.43
	Ottilies Käseladen	$55.09		$55.09	$91.48	$201.85
	Frankenversand	$208.58	$298.44	$507.02	$4.93	$36.65
	QUICK-Stop	$384.10	$432.23	$816.33	$206.64	$1,425.25
	Morgenstern Gesundkost	$125.77		$125.77		$127.34
	Lehmanns Marktstand	$102.39	$110.37	$212.76		$265.08

Germany	Die Wandernde Kuh	$85.34	$37.49	$122.83		$105.65
	Königlich Essen		$69.74	$69.74	$63.81	$32.14
	Drachenblut Delikatessen		$35.99	$35.99		
	Blauer See Delikatessen					$36.71
	Alfreds Futterkiste					
	Alfred's Futterkiste					
Brazil	Hanari Carnes	$124.00		$124.00		$68.65
	Wellington Importadora	$13.97		$13.97	$44.12	
	Que Delicia	$9.45	$45.03	$54.48	$99.23	
	Ricardo Adocicados	$42.52		$42.52	$132.99	$60.43
	Comércio Mineiro	$79.70		$79.70	$11.93	$65.99
	Tradiçao Hipermercados	$1.35		$1.35		$46.77
	Familia Arquibaldo		$17.09	$17.09	$21.48	$6.54
	Queen Cozinha		$890.78	$890.78	$179.11	
	Gourmet Lanchonetes				$24.50	
Belgium	Suprêmes délices	$57.57		$57.57	$230.36	
	Maison Dewey					$66.69
Switzerland	Chop-suey Chinese	$22.98	$1.17	$24.15		$91.76

Switzerland	Richter Supermarkt	$148.33		$148.33	$137.35	$78.85
Venezuela	HILARION-Abastos	$81.91	$184.41	$266.32	$245.13	$112.97
	GROSELLA-Restaurante	$66.29		$66.29		
	LILA-Supermercado	$84.93	$55.62	$140.55	$148.61	$150.19
	LINO-Delicateses				$99.27	
Austria	Ernst Handel	$286.57	$485.43	$772.00	$648.39	$816.01
	Piccolo und mehr		$483.09	$483.09	$36.58	$339.22
Mexico	Centro comercial Moctezuma	$3.25		$3.25		
	Tortuga Restaurante	$98.81	$64.50	$163.31		$236.71
	Ana Trujillo Emparedados y helados	$1.61		$1.61		
	Pericles Comidas clásicas		$54.20	$54.20	$83.49	$69.32

USA	Save-a-lot Markets		$429.99	$429.99	$226.79	$620.12
	Hungry Coyote Import Store		$50.46	$50.46	$0.20	
	Lazy K Kountry Store				$7.48	$11.92
	Great Lakes Food Market					$3.35
	Trail's Head Gourmet Provisioners					$63.01
	Let's Stop N Shop					$13.73
	The Cracker Box					
Sweden	Folk och fä HB	$3.67	$68.80	$72.47	$34.19	$430.25
	Berglunds snabbköp	$101.67	$168.64	$270.31	$12.80	$361.22
Finland	Wartian Herkku	$162.27	$35.16	$197.43	$226.91	$215.36
	Wilman Kala					

	Antonio Moreno Taqueria		$22.00	$22.00		$147.93
USA	Rattlesnake Canyon Grocery	$517.89	$142.08	$659.97	$721.46	$72.73
	White Clover Markets	$4.56	$23.29	$27.85	$75.46	$59.13
	Split Rail Beer & Ale	$4.54	$426.94	$431.48	$4.34	
	Old World Delicatessen	$257.62	$84.21	$341.83	$73.02	
	Lonesome Pine Restaurant	$13.25		$13.25		$24.91
	The Big Cheese	$17.52		$17.52		

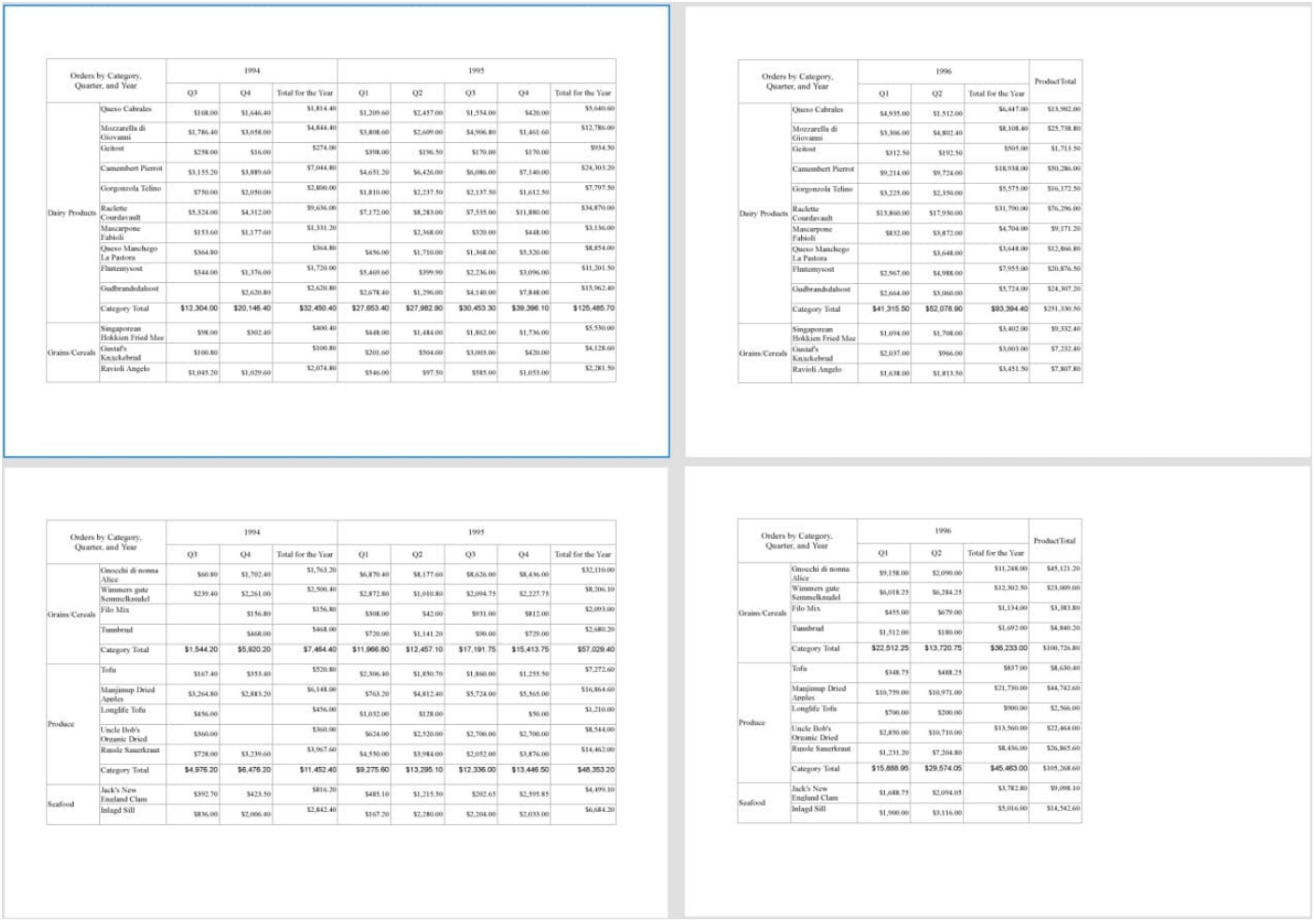
Italy	Magazzini Alimentari Riuniti	$44.61		$44.61	$160.90	
	Reggiani Caseifici	$7.45		$7.45	$25.04	$22.95
	Franchi S.p.A.				$3.02	
Spain	Romero y tomillo	$23.19		$23.19		
	Godos Cocina Tipica	$107.83		$107.83		$4.32
	Bólido Comidas preparadas		$77.92	$77.92		
	Galería del gastronómo		$10.14	$10.14	$18.69	$6.54

Layout Order in Page Reports

In a Page report, controls or data regions do not change in size based on the data. To handle expanding or overflowing data on multiple pages, Page reports use Overflow Place Holder control, with a data region placed on the first page of the report and the OverflowPlaceHolder controls on subsequent pages.

The following image shows the preview of a Page report with a Tablix data region on one page and an Overflow Place Holder control on the next page in Z-order.

(For an enlarged image view, open the image in a separate tab.)



Orders by Category, Quarter, and Year		1994			1995				
		Q3	Q4	Total for the Year	Q1	Q2	Q3	Q4	Total for the Year
Dairy Products	Queso Cabrales	$168.00	$1,646.40	$1,814.40	$1,209.60	$2,457.00	$1,554.00	$420.00	$5,640.60
	Mozzarella di Giovanni	$1,786.40	$3,058.00	$4,844.40	$3,808.60	$2,609.00	$4,906.80	$1,461.60	$12,786.00
	Geitost	$258.00	$16.00	$274.00	$398.00	$196.50	$170.00	$170.00	$934.50
	Camembert Pierrot	$3,155.20	$3,889.60	$7,044.80	$4,651.20	$6,426.00	$6,086.00	$7,140.00	$24,303.20
	Gorgonzola Telino	$750.00	$2,050.00	$2,800.00	$1,810.00	$2,237.50	$2,137.50	$1,612.50	$7,797.50
	Raclette Courdavault	$5,324.00	$4,312.00	$9,636.00	$7,172.00	$8,283.00	$7,535.00	$11,880.00	$34,870.00
	Mascarpone Fabioli	$153.60	$1,177.60	$1,331.20		$2,368.00	$320.00	$448.00	$3,136.00
	Queso Manchego La Pastora	$364.80		$364.80	$456.00	$1,710.00	$1,368.00	$5,320.00	$8,854.00
	Flotemysost	$344.00	$1,376.00	$1,720.00	$5,469.60	$399.90	$2,236.00	$3,096.00	$11,201.50
	Gudbrandsdalsost		$2,620.80	$2,620.80	$2,678.40	$1,296.00	$4,140.00	$7,848.00	$15,962.40
	Category Total	$12,304.00	$20,146.40	$32,450.40	$27,653.40	$27,982.90	$30,453.30	$39,396.10	$125,485.70
Grains/Cereals	Singaporean Hokkien Fried Mee	$98.00	$302.40	$400.40	$448.00	$1,484.00	$1,862.00	$1,736.00	$5,530.00
	Gustaf's Knäckebröd	$100.80		$100.80	$201.60	$504.00	$3,003.00	$420.00	$4,128.60
	Ravioli Angelo	$1,045.20	$1,029.60	$2,074.80	$546.00	$97.50	$585.00	$1,053.00	$2,281.50

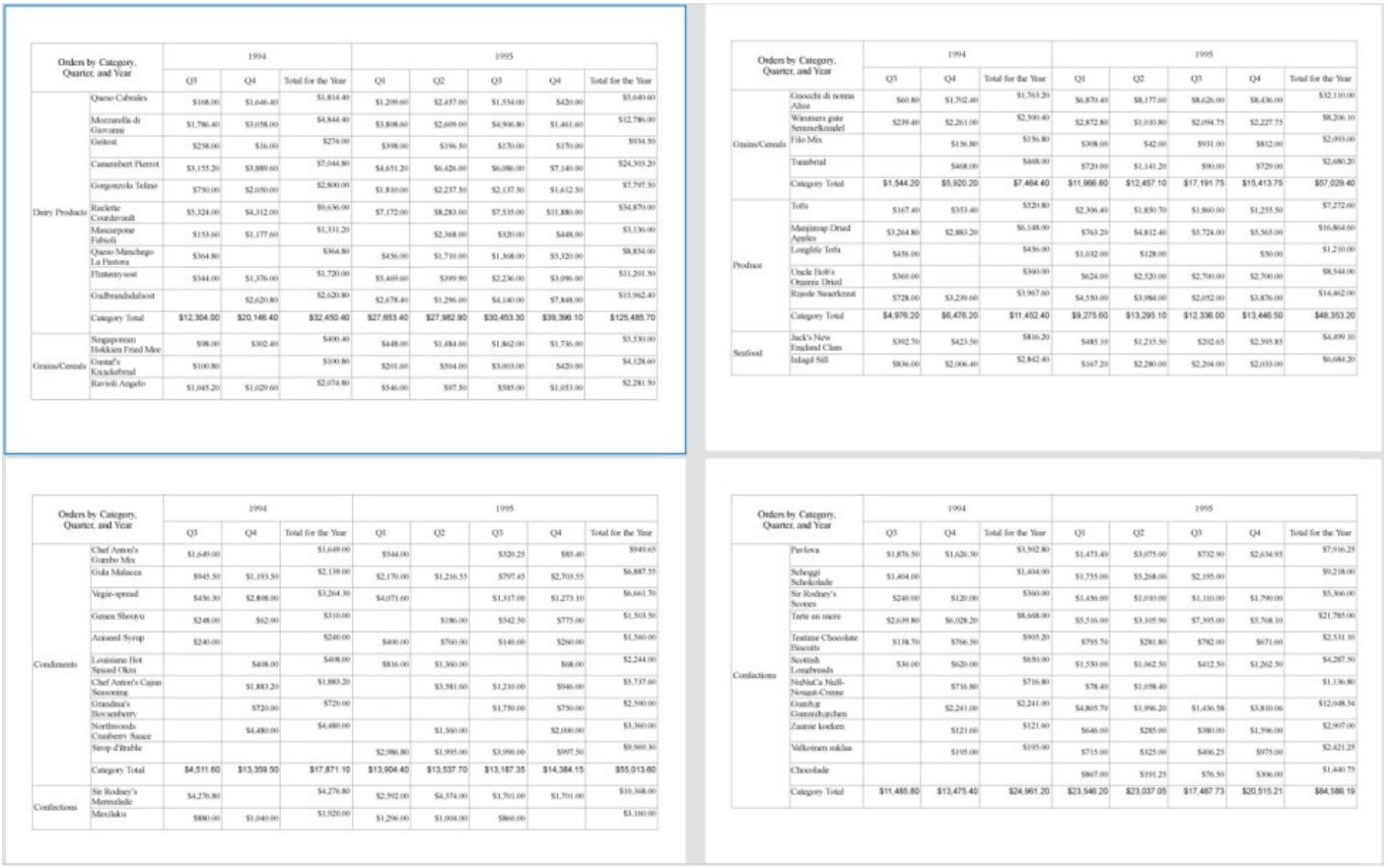
Orders by Category, Quarter, and Year		1996			ProductTotal
		Q1	Q2	Total for the Year	
Dairy Products	Queso Cabrales	$4,935.00	$1,512.00	$6,447.00	$13,902.00
	Mozzarella di Giovanni	$3,306.00	$4,802.40	$8,108.40	$25,738.80
	Geitost	$312.50	$192.50	$505.00	$1,713.50
	Camembert Pierrot	$9,214.00	$9,724.00	$18,938.00	$50,286.00
	Gorgonzola Telino	$3,225.00	$2,350.00	$5,575.00	$16,172.50
	Raclette Courdavault	$13,860.00	$17,930.00	$31,790.00	$76,296.00
	Mascarpone Fabioli	$832.00	$3,872.00	$4,704.00	$9,171.20
	Queso Manchego La Pastora		$3,648.00	$3,648.00	$12,866.80
	Flotemysost	$2,967.00	$4,988.00	$7,955.00	$20,876.50
	Gudbrandsdalsost	$2,664.00	$3,060.00	$5,724.00	$24,307.20
	Category Total	$41,315.50	$52,078.90	$93,394.40	$291,330.50
Grains/Cereals	Singaporean Hokkien Fried Mee	$1,694.00	$1,708.00	$3,402.00	$9,332.40
	Gustaf's Knäckebröd	$2,037.00	$966.00	$3,003.00	$7,232.40
	Ravioli Angelo	$1,638.00	$1,813.50	$3,451.50	$7,807.80

Orders by Category, Quarter, and Year		1994			1995				
		Q3	Q4	Total for the Year	Q1	Q2	Q3	Q4	Total for the Year
Grains/Cereals	Gnocchi di nonna Alice	$60.80	$1,702.40	$1,763.20	$6,870.40	$8,177.60	$8,626.00	$8,436.00	$32,110.00
	Wimmers gute Semmelknudel	$239.40	$2,261.00	$2,500.40	$2,872.80	$1,010.80	$2,094.75	$2,227.75	$8,206.10
	Filo Mix		$156.80	$156.80	$308.00	$42.00	$931.00	$812.00	$2,093.00
	Tunnbrod		$468.00	$468.00	$720.00	$1,141.20	$90.00	$729.00	$2,680.20
	Category Total	$1,544.20	$5,920.20	$7,464.40	$11,966.80	$12,457.10	$17,191.75	$15,413.75	$57,029.40
Produce	Tofu	$167.40	$353.40	$520.80	$2,306.40	$1,850.70	$1,860.00	$1,255.50	$7,272.60
	Manjimup Dried Apples	$3,264.80	$2,883.20	$6,148.00	$763.20	$4,812.40	$5,724.00	$5,565.00	$16,864.60
	Longlife Tofu	$456.00		$456.00	$1,032.00	$128.00		$50.00	$1,210.00
	Uncle Bob's Organic Dried	$360.00		$360.00	$624.00	$2,520.00	$2,700.00	$2,700.00	$8,544.00
	Rmole Sauerkraut	$728.00	$3,239.60	$3,967.60	$4,550.00	$3,984.00	$2,052.00	$3,876.00	$14,462.00
	Category Total	$4,976.20	$6,476.20	$11,452.40	$9,275.60	$13,295.10	$12,336.00	$13,446.50	$48,353.20
Seafood	Jack's New England Clam	$392.70	$423.50	$816.20	$485.10	$1,215.50	$202.65	$2,595.85	$4,499.10
	Inlagd Sill	$836.00	$2,006.40	$2,842.40	$167.20	$2,280.00	$2,204.00	$2,033.00	$6,684.20

Orders by Category, Quarter, and Year		1996			ProductTotal
		Q1	Q2	Total for the Year	
Grains/Cereals	Gnocchi di nonna Alice	$9,158.00	$2,090.00	$11,248.00	$45,121.20
	Wimmers gute Semmelknudel	$6,018.25	$6,284.25	$12,302.50	$23,009.00
	Filo Mix	$455.00	$679.00	$1,134.00	$3,383.80
	Tunnbrod	$1,512.00	$180.00	$1,692.00	$4,840.20
	Category Total	$22,512.25	$13,720.75	$36,233.00	$100,726.80
Produce	Tofu	$348.75	$488.25	$837.00	$8,630.40
	Manjimup Dried Apples	$10,759.00	$10,971.00	$21,730.00	$44,742.60
	Longlife Tofu	$700.00	$200.00	$900.00	$2,566.00
	Uncle Bob's Organic Dried	$2,850.00	$10,710.00	$13,560.00	$22,464.00
	Rmole Sauerkraut	$1,231.20	$7,204.80	$8,436.00	$26,865.60
	Category Total	$15,888.95	$29,574.05	$45,463.00	$105,268.60
Seafood	Jack's New England Clam	$1,688.75	$2,094.05	$3,782.80	$9,098.10
	Inlagd Sill	$1,900.00	$3,116.00	$5,016.00	$14,542.60

The following image shows the preview of the above report in N-order.

(For an enlarged image view, open the image in a separate tab.)



To Set a Z- or N-Order in Report

1. From the Report Explorer, select the **Report** node.
2. Go to the Properties panel, set the **LayoutPagesOrder** property to 'ZOrder' or 'NOrder'.
   Or,

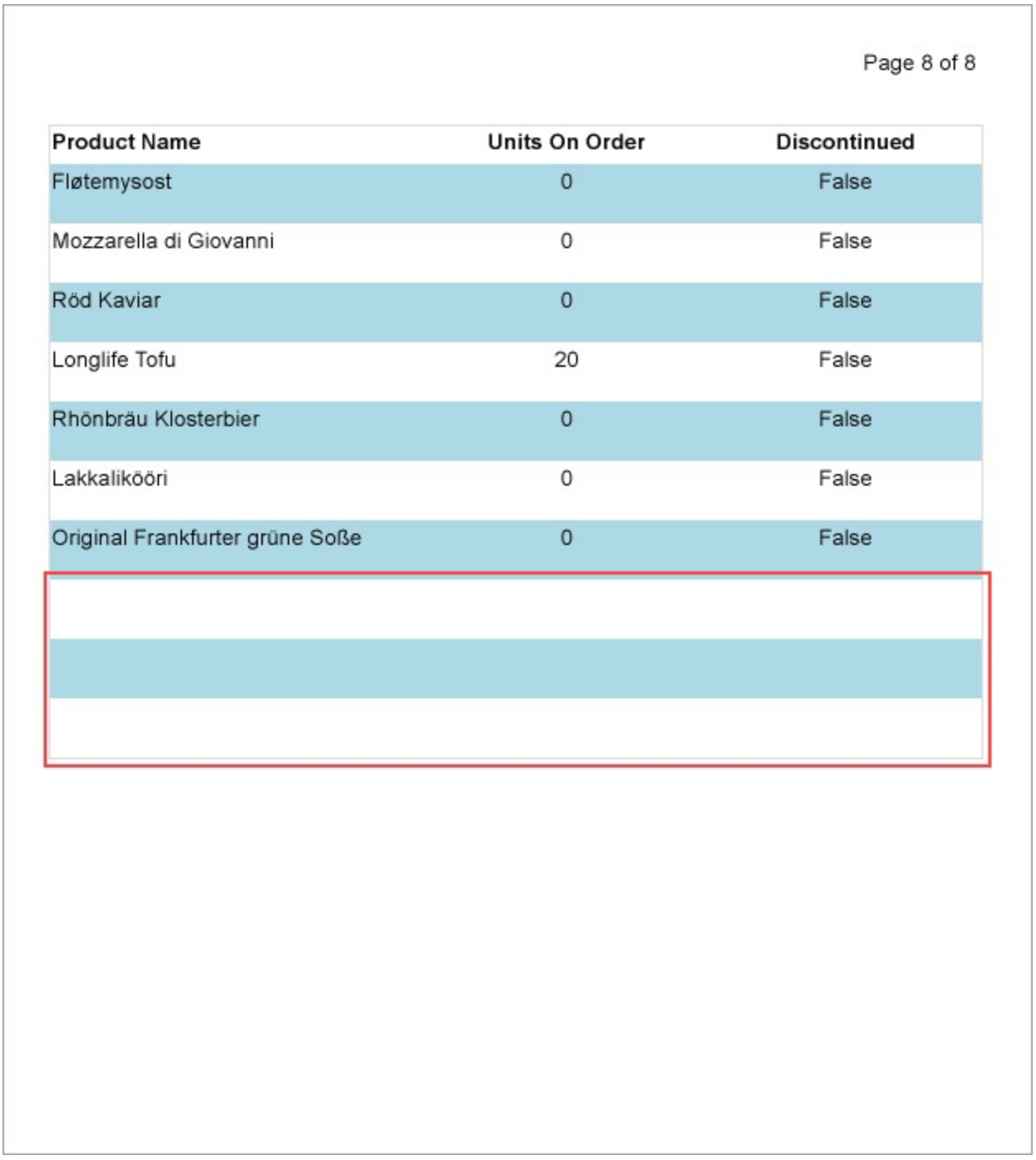
1. From the Report Explorer, select the **Report** node.
2. Click the **Property dialog...** to open the **Report** dialog
3. From the **Appearance** page, set **Pages layout order** to 'Z-order' or 'N-order'.

Repeat Blank Rows in a Table Data Region in Page Reports

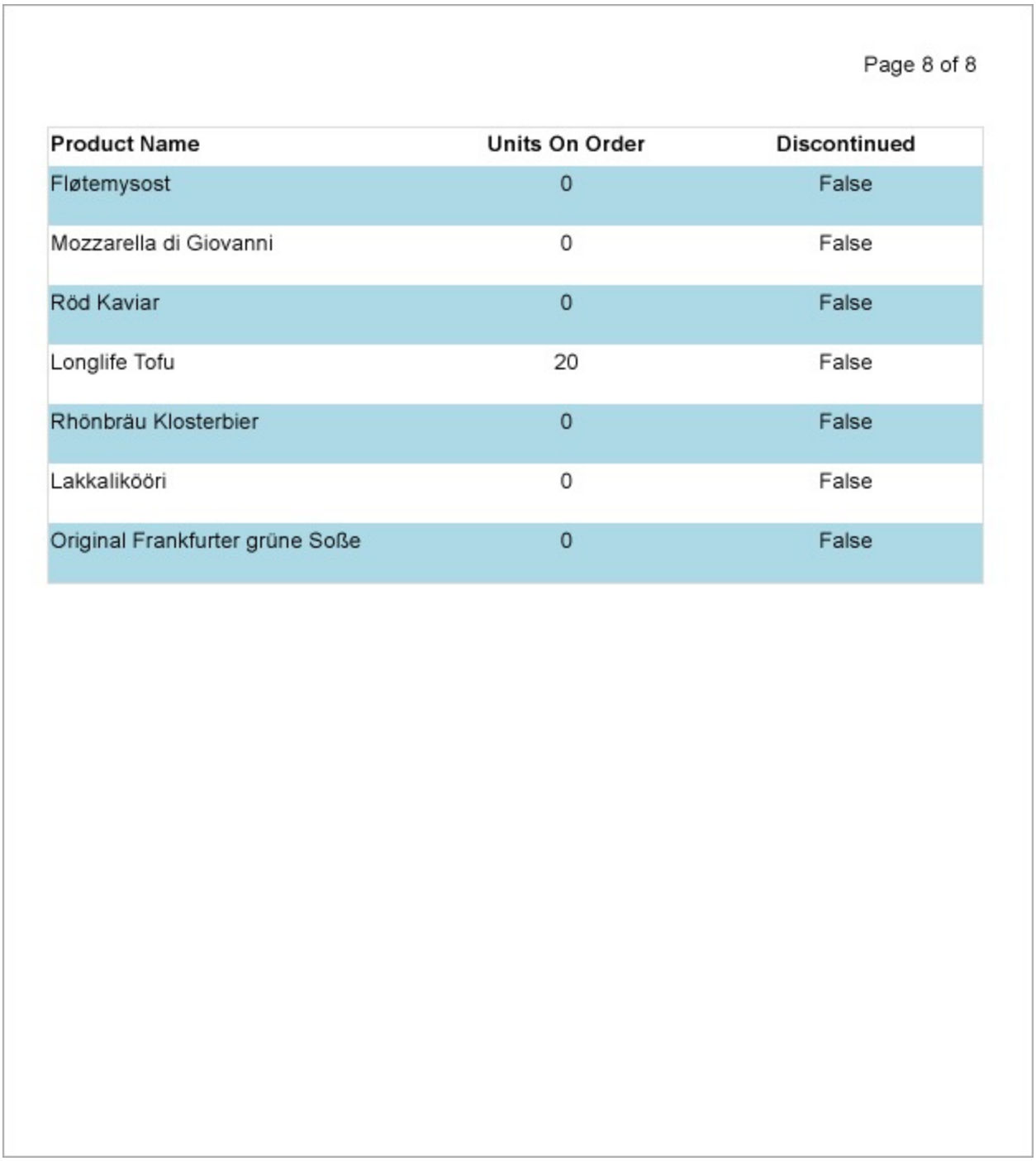
Table data region in Page report has **RepeatBlankRows** property that lets you control the behavior of blank rows with the help of options:

- **None**: indicates no filling of extra space with blank rows.
- **FillPage**: fills the empty space within a page with blank rows, to reach the table's **FixedHeight**. Note that this setting is equivalent to property **RepeatToFill** property (obsolete) set to 'True'.
- **FillGroup**: fills the table with blank rows within a group.

The following image shows the last page of a report with Table data region's **RepeatBlankRows** property set to 'FillPage'.



The following image shows the last page of a report with Table data region's **RepeatBlankRows** property set to 'None'.



Page Breaks in Data Regions

You can control page breaks in data groups of data regions by using the **NewPage** property for data regions and its combination with the **PageBreakAtStart**, **PageBreakAtEnd**, and **BreakLocation** properties. These properties help you tune page breaks after or before a data group in your report and on which page to continue displaying the report content.

The **NewPage** property works for the data regions - List, Table, BandedList, Tablix, Container, TOC, and Chart, and has

the following options.

- Next - a default value that makes a new group start from the immediate next page of the report.
- Odd - a new group starts from the next odd page of the report.
- Even - a new group starts from the next even page of the report.

You can use the **NewPage** property for the following data region groups:

- Table: Groups, Detail groups
- Tablix: Row groups, Adjacent Row groups, Child Row groups
- List: Detail groups
- BandedList: Groups, Details section

NewPage and PageBreakAtStart / PageBreakAtEnd

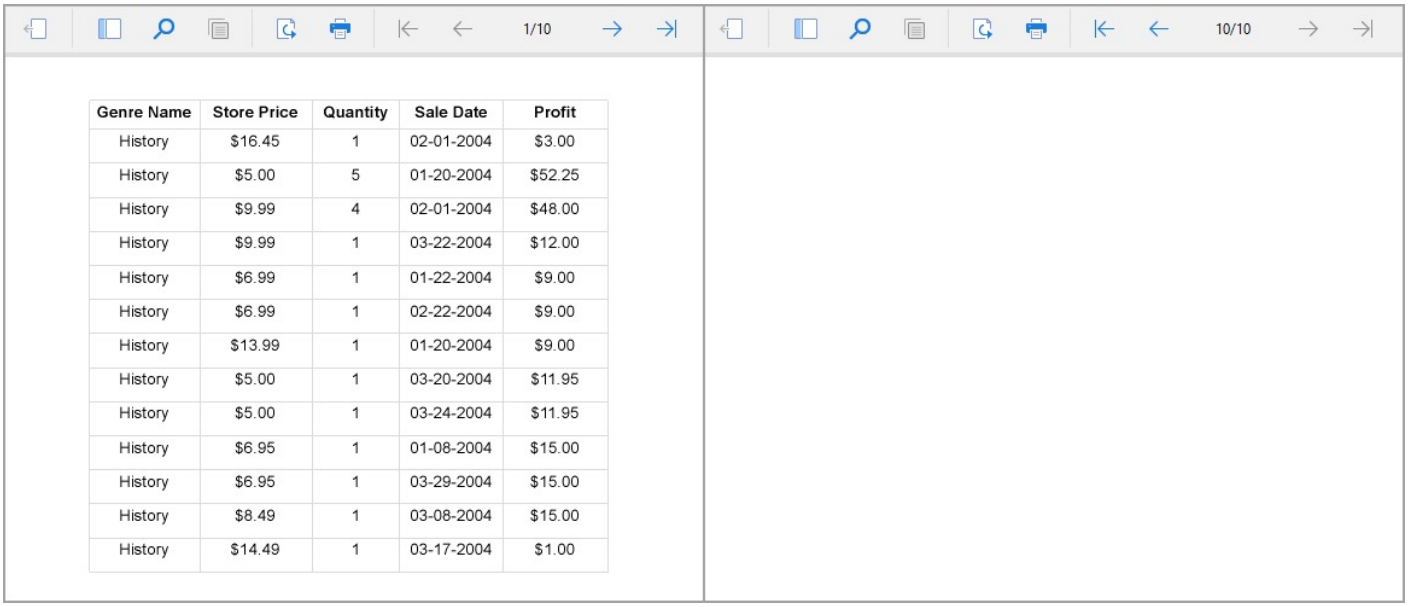
See the possible combinations of NewPage with the PageBreakAtStart and PageBreakAtEnd properties.

PageBreakAtStart	PageBreaktEnd	NewPage > Next	NewPage > Odd	NewPage > Even
False	False	No new page	No new page	No new page
False	True	A new page is added after the data region.	A new page is added after the data region and at odd page of the report.	A new page is added after the data region and at even page of the report.
True	False	A new page is added before the data region.	A new page is added before the data region and at odd page of the report.	A new page is added before the data region and at even page of the report.
True	True	A new page is added before and after the data region.	A new page is added before and after the data region and at odd page of the report.	A new page is added before and after the data region and at even page of the report.

The image below uses a report that has a Table data region with the following properties:

- PageBreakAtEnd > True
- PageBreakAtStart > False
- NewPage > Even

As a result, the report renders 10 pages, a page break is added after a table, and a new page is added after the table at an even page (Page 10).



Genre Name	Store Price	Quantity	Sale Date	Profit
History	$16.45	1	02-01-2004	$3.00
History	$5.00	5	01-20-2004	$52.25
History	$9.99	4	02-01-2004	$48.00
History	$9.99	1	03-22-2004	$12.00
History	$6.99	1	01-22-2004	$9.00
History	$6.99	1	02-22-2004	$9.00
History	$13.99	1	01-20-2004	$9.00
History	$5.00	1	03-20-2004	$11.95
History	$5.00	1	03-24-2004	$11.95
History	$6.95	1	01-08-2004	$15.00
History	$6.95	1	03-29-2004	$15.00
History	$8.49	1	03-08-2004	$15.00
History	$14.49	1	03-17-2004	$1.00

After the same properties are changed as

- PageBreakAtEnd > False
- PageBreakAtStart > True
- NewPage > Odd

As a result, the report renders 11 pages, a page break is added before a table (Page 2), and a new page is added at odd page of the report (Page 1).

NewPage and BreakLocation

The **BreakLocation** property controls the break location of a data group. See the possible combination of the **NewPage** with the **BreakLocation** property.

	NewPage > Next	**NewPage > Odd**	**NewPage > Even**
BreakLocation > None	No page break		
BreakLocation > Start	A page break is added before a new group. A new group starts from the next page of the report.	A page break is added before a new group. A new group starts from the next odd page of the report.	A page break is added before a new group. A new group starts from the next even page of the report.
BreakLocation > End	A page break is added after a current group. A new group starts from the next page of the report. [1]	A page break is added after a current group. A new group starts from the next odd page of the report. [1]	A page break is added before a new group. A new group starts from the next even page of the report. [1]
BreakLocation >	A page break is added before and after a current group. A	A page break is added before and after a current group. A new	A page break is added before a new group. A

StartAndEnd	new group starts from the next page of the report.	group starts from the next odd page of the report.	new group starts from the next even page of the report.
BreakLocation > Between	A page break is between groups. A new group starts from the next page of the report. [1]	A page break is between groups. A new group starts from the next odd page of the report. [1]	A page break is added before a new group. A new group starts from the next even page of the report. [1]

[1] The first group always starts on the first page.

Multiple Data Regions Behavior

In a report with multiple data regions, the behavior of the **NewPage** property is as follows:

- If at least one data region in a report has the **NewPage > Even**, page breaks are added with the value **Even** for all report's data regions.
- If at least one data region in a report has the **NewPage > Odd**, page breaks are added with the value **Odd** for all report's data regions.
- If NewPage is set to different values in data regions of a report, page breaks are added with the value **Next** for all data regions.

> **Page report Limitation:** When multiple data regions are added to the same page, the data regions take the **Next** value; the other values (Odd, Even) are ignored.

Skip Page Generation in Page Reports

In ActiveReports, you can easily hide the pages in a Page report at run time using the visibility properties.

The visibility properties for a report page can be set using the **Hidden** property.

In the rendered report, the page generation for the hidden pages is skipped.

Hidden Property

This property controls the visibility of the report page based on the expression that you specify or the value you set, that is, True or False. If you want to hide the report page, set the property to True. In case you want to conditionally hide the report page, enter a suitable expression.

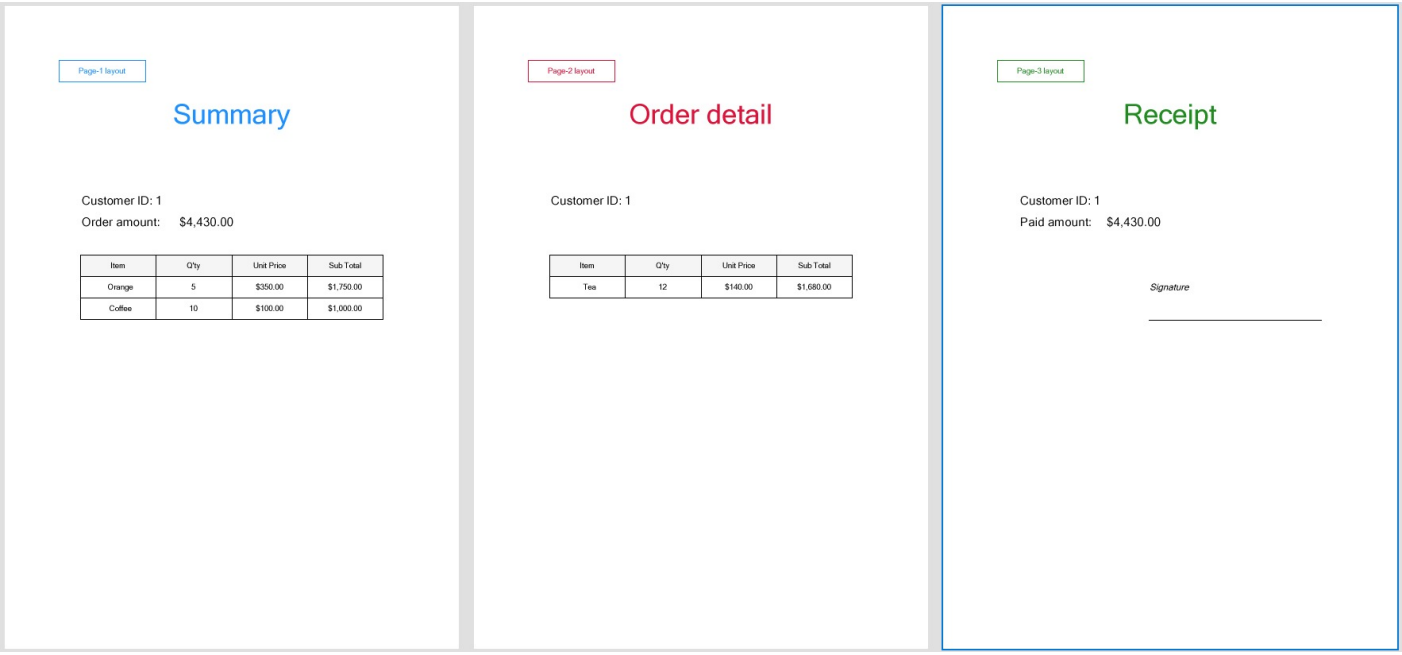
Consider a scenario where a Page report contains three pages: Page 1, Page 2, and Page 3.

- Page 1 of the report consists of the report header and order summary details,
- Page 2 of the report contains the overflowing content of Page 1, that is, order summary details, and
- Page 3 consists of a receipt. You want the page generation for Page 3 to skip in case the order amount or the amount paid is zero.

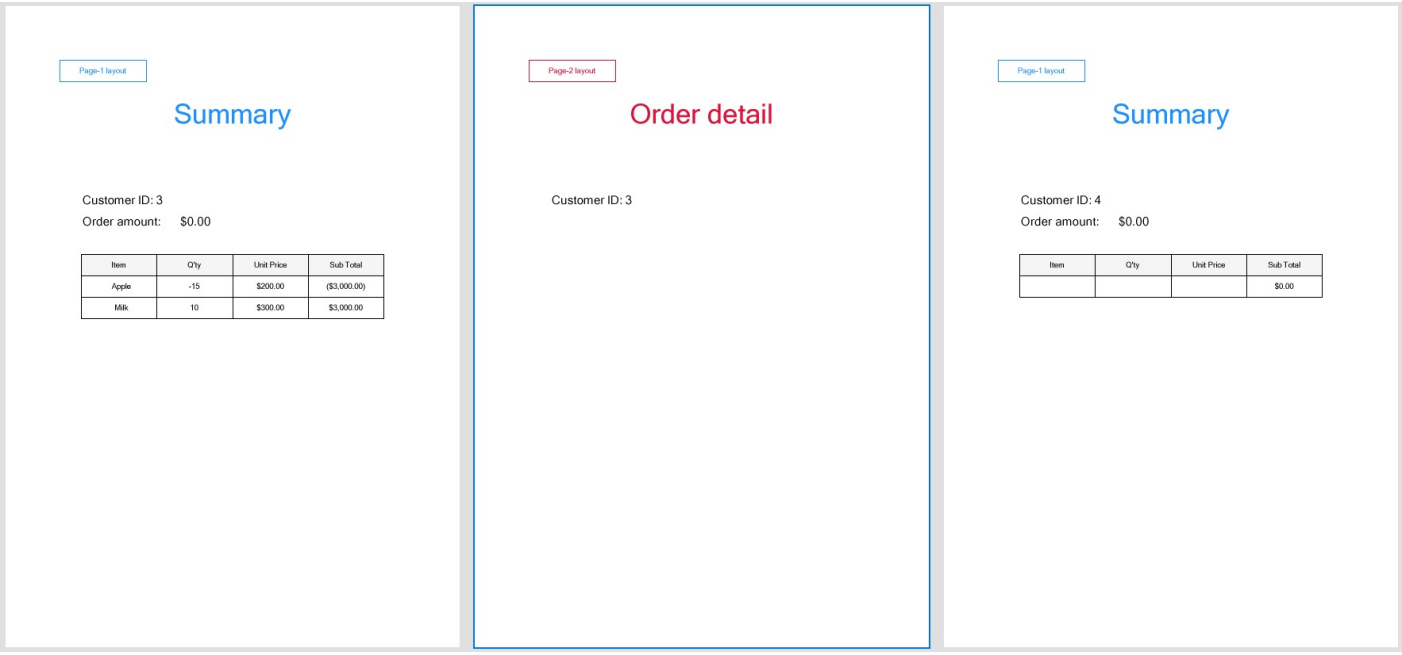
For the report pages to appear in this manner, you need to set the **Hidden** property for Page 3 to the expression like follows, which calculates the order amount and hides the receipt page in case the order amount calculates to zero:

```
=iif(Sum(Fields!Quantity.Value*Fields!UnitPrice.Value)=0, True, False)
```

This is how the three pages of an order in a report are rendered when the order amount is not zero:



When the order amount is zero, only the first two pages of the order are rendered and the receipt page is skipped:



Hide or Show Sections in RDLX and RDLX Dashboard Reports

The RDLX and RDLX Dashboard Reports are reports with multiple sections. You can control the visibility of the initial state of a section using **Hidden** property.

Hidden property

This property controls the visibility of the report sections based on the expression that you specify or the value you set, that is, True or False. If you want to hide the report page, set the property to True. In case you want to conditionally hide the report page, enter a suitable expression.

Conditionally hide/show a report section

Let us assume that the report has few sections and we want to hide one of the sections. The procedure to hide a section in RDLX and RDLX Dashboard Reports based on user input is described below.

1. Create a report parameter. In the **Report - Parameters** dialog, set the following:
    - in General tab:
        - **Name:** ShowHideParam
        - **Data type**: Boolean
        - **Text for prompting users for a value**: Select whether to hide the section
    - in Default Values tab:
        - select **Non-queried** option
        - **Value**: false
2. Select the ReportSection/Section tab of the report that you want to hide.
3. Go to the **Hidden** property of the section and enter the expression as shown:

```
=IIF(Parameters!ShowHideParam.Value=True, True, False)
```

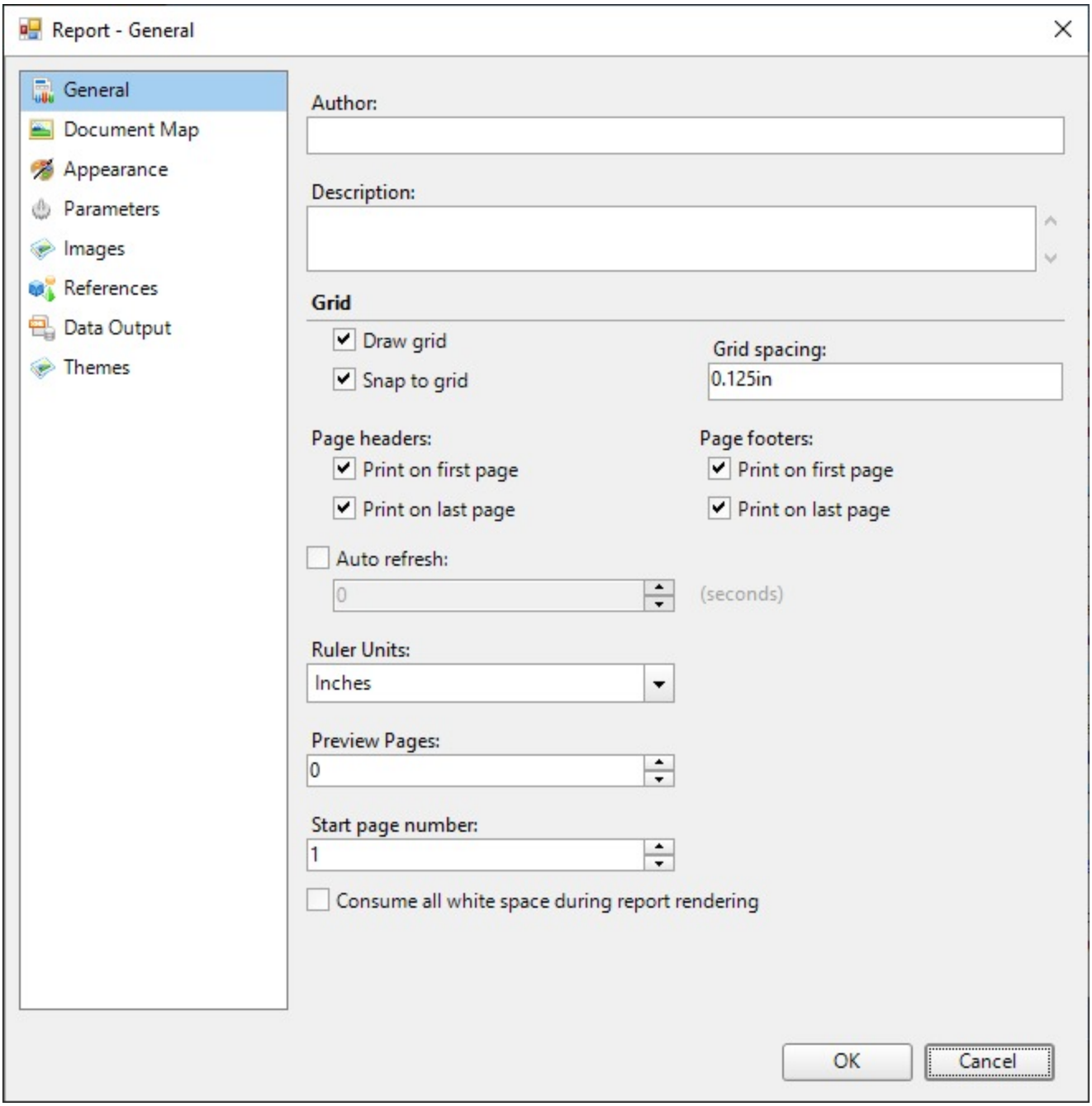
4. Preview the report.
5. From the Parameters panel, select 'True' to hide the section.
   Similarly, for the hidden section to show, select 'False'.

Report Dialog

In a Page Report or an RDLX Report, you can set the basic report properties in the Report dialog. You can access this dialog by doing one of the following:

- Go to the Visual Studio Report menu and select **Report Properties**.
- In the Report Explorer, right-click the Report node and from the context menu that appears, select **Report Properties**.
- In the Report Explorer, select the Report node and in the Commands section at the bottom of the Properties Panel, click the **Property dialog** command.
- Right-click the gray area outside the design surface to select the Report and in the Commands section at the bottom of the Properties Panel, click the **Property dialog** command.

The Report dialog provides the following pages where you can set report properties:



General

The General page of the Report dialog allows you to control the following items:

- **Author**: Enter the name of the report author here.
- **Description**: Enter a description of the report here.
- **Draw grid**: Clear this check box to remove the grid lines from the report design surface.
- **Snap to grid**: Clear this check box to allow free placement of report items on the report design surface instead of the automatic alignment of report items with grid lines.
- **Grid spacing**: Enter the spacing between grid lines in inches. The default value is 0.125 inches.
- **Page headers**: These options are enabled when you set a Page Header in a Report Definition Language (RDLX) report.
  - **Print on first page** - Select this check box to set the page header on the first page of the report.
  - **Print on last page** - Select this check box to set the page header on the last page of the report.
- **Page footers**: These options are enabled when you set a Page Footer in a RDLX report.
  - **Print on first page** - Select this check box to set the page footer on the first page of the report.
  - **Print on last page** - Select this check box to set the page footer on the last page of the report.
- **Auto refresh**: Select this check box to automatically refresh the pages of the report at regular intervals. When this check box is selected, you can supply the interval in seconds.
- **Ruler Units**: Set the ruler units in Inches or Centimeters.
- **Preview Pages**: Set the number of pages to display in the Preview tab. Minimum values is 0 and maximum is 10000 pages. If the value is set to 0, the Preview tab displays all the pages. By default, the Preview tab displays all the pages.
- **Start page number**: Set the number from which the page numbering should start.
- **Consume all white space during report rendering**: In RDLX report, selecting this option consumes all white spaces in report during report rendering.

Document Map

The Document Map page of the Report dialog allows you to control the following items:

- **Source**: The document map source. You can choose the source as Labels, Headings, or both Labels and Headings.
- **Numbering Style**: Specifies numbering style for headings like 1, 2, 3, 4 or a, b, c, d, etc.

See Document Map for more information.

Appearance

The Appearance page of the Report dialog allows you to control the page layout for your report.

Columns (in RDLX reports)

- **Number of columns**: Enter the number of columns you want to use in your report.
- **Spacing**: Enter the number of inches of space to use between columns.

Page Layout

- **Paper Size**: Select one among the standard paper sizes from the dropdown.
- **Width**: Specify the width of the layout.
- **Height**: Specify the height of the layout.
- **Left margin**: Specify the Left margin for the layout.
- **Right margin**: Specify the Right margin for the layout.
- **Top margin**: Specify the Top margin for the layout.
- **Bottom margin**: Specify the Bottom margin for the layout.
- **Orientation**: Select one among **Portrait** and **Landscape** as your page orientation.
- **Pages layout order**: See Set Page Layout in Z- or N-Order for more information.
  Z-order: Horizontally expanding data in the report is rendered first on the upcoming pages followed by the vertically expanding data.
  N-order: Vertically expanding data in the report is rendered first on the upcoming pages followed by the horizontally expanding data.

Design

- **Output background only at design-time**: Select the checkbox to display background (for example, background image or background color) in the design tab only.

Stylesheet

- **Source**: Select one among **Embedded** and **External** as your style sheet source.
- **Value**: Select the style sheet to apply to the report. Following options are available:

  Expression - opens the Expression Editor dialog to set an expression.

  New - opens the **New Stylesheet Editor** dialog for creating an external or embedded style sheet.

  Open file - opens the **Open Stylesheet from file** dialog for navigating to a local style sheet file. This option is only available for external style sheets.
  In case of **embedded style sheets**, a list of available style sheets in the report is provided.

Parameters

The Parameters page of the Report dialog allows you to control how a user interface is presented to your users for each parameter. See Parameters for further information.

Images

The Images page of the Report dialog allows you to add and modify images for your report. Click the **Open** button located at the top left of the page to display the Open file dialog, where you can navigate to an image. Once you select an image file and click the **Open** button, a thumbnail of the image is displayed in the Image column, and the Name and MIME Type values are automatically populated in their respective columns.

You can use the images you add here in the Image control. The **MIME Type** column provides a combo-box with a list of image file extensions, where you can change default file filter of the added image.

You can also use the **Remove** button located at the top right of the page to remove any added image.

References

The References page of the Report dialog allows you to add references to assemblies and classes so that you can call methods from them in expressions throughout your report. You can also access the References dialog from the Properties Panel by selecting the **Classes** (Collection) or **References** (Collection) property of a report and clicking the ellipsis button that appears.

Assembly Name

This is a list of the assemblies available for use in your report. You can delete assemblies using the **Remove** button, or add them using the **Open** button which presents the Open file dialog.

Classes

This is a list of instance-based classes you can create for use in your report.

- **Class name**: Enter the namespace and name of the class here. (i.e. Invoicing.GetDueDate)
- **Instance name**: Enter a name for the instance of the class here. (i.e. m_myGetDueDate)

Data Output

The Data Output page of the Report dialog allows you to control how the report's data is rendered in XML exports.

- **Element name**: Enter the name you want to appear as the top level data element in your exported XML file.
- **Data transform (.xsl file)**: Enter the name of the XSL file you want to use as a style sheet for the exported XML file.
- **Data schema**: Enter the schema or namespace to use for validating data types in the exported XML file.
- **Render textboxes as**: Choose whether to render textboxes as Attributes or Elements in the exported XML file.
    - Attributes example: `<table1 textbox3="Report created on: 7/26/2005 1:13:00 PM">`
    - Elements example: `<table1> <textbox3>Report created on: 7/26/2005 1:13:28 PM</textbox3>`

Themes

The Themes page of the Report dialog displays the report's themes. This dialog allows you to create a new theme, add, modify or remove an existing one, as well as rearrange the order of themes if a report has many themes. When you select to create or modify a theme, the **Theme Editor** is opened. See Themes for further information.

Master Report (RDLX Report)

Master Reports are like dynamic templates you can design for use with content reports. This assists users in creating reports that share common elements such as a logo in the page header or a web site link in the page footer. You design the master report with controls, code, data sources, and layout properties that cannot be modified from content reports.

Master Reports differ from templates in that they are loaded each time the report is executed. Therefore, you can modify a master report and the changes to the master report automatically appear in any reports that reference it.

In an RDLX report, you can create a master report and apply it to any number of content reports to keep common styles in one easy-to-maintain location.

Master reports help you perform the following tasks:

- Create a predefined layout with limited modification possibilities.
- Hide details of data sources.

Design a Master Report

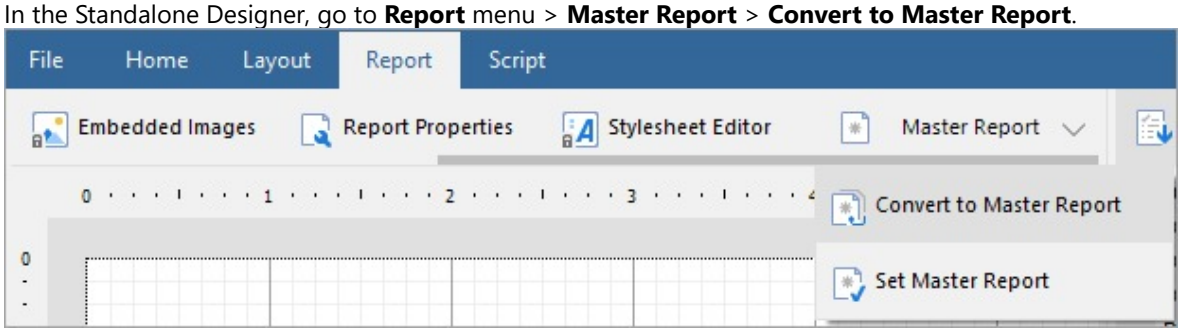
When designing a master report, you use controls, code, data sources, and layout properties in the same way that you do in a normal report. A master report is valid on its own, and can be run without a content report. To prevent end users from modifying a master report, you can set permissions on the file to **Read Only** for that user or group.

In an RDLX Report, you can create a master report by saving it as an RDLX-master file. You can then apply it like a template to content reports.

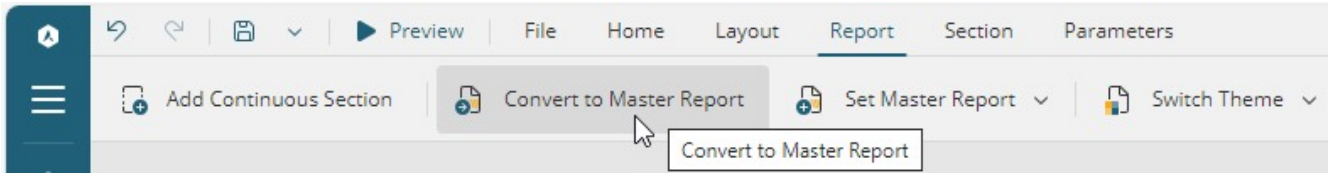
A ContentPlaceHolder control appears in the toolbox when you convert an RDLX report to a Master Report. This control defines regions where users can add content after applying a master report template.

Note: These steps assume that you have already created a new RDLX report and connected it to a data source. The Master Report feature is only available with RDLX Reports.

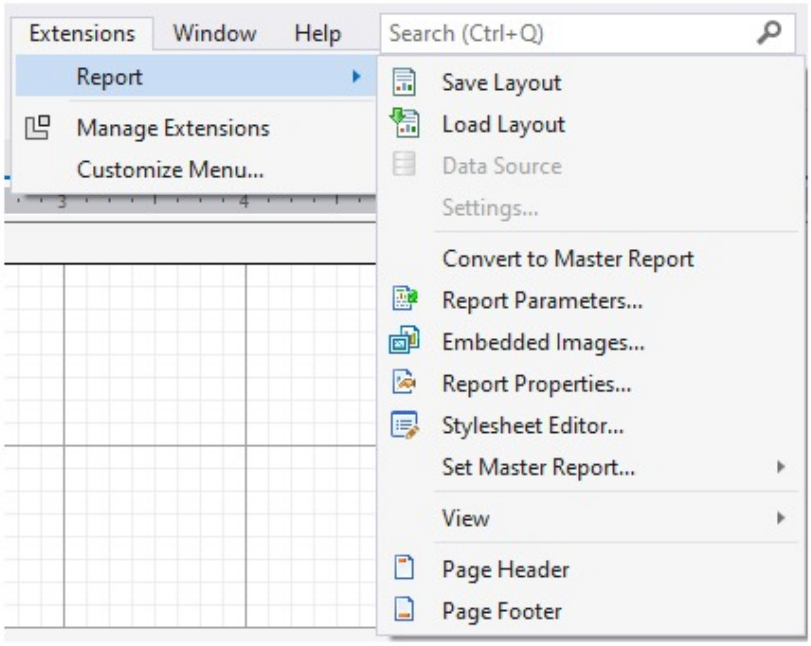
1. In the Standalone Designer, go to **Report** menu > **Master Report** > **Convert to Master Report**.



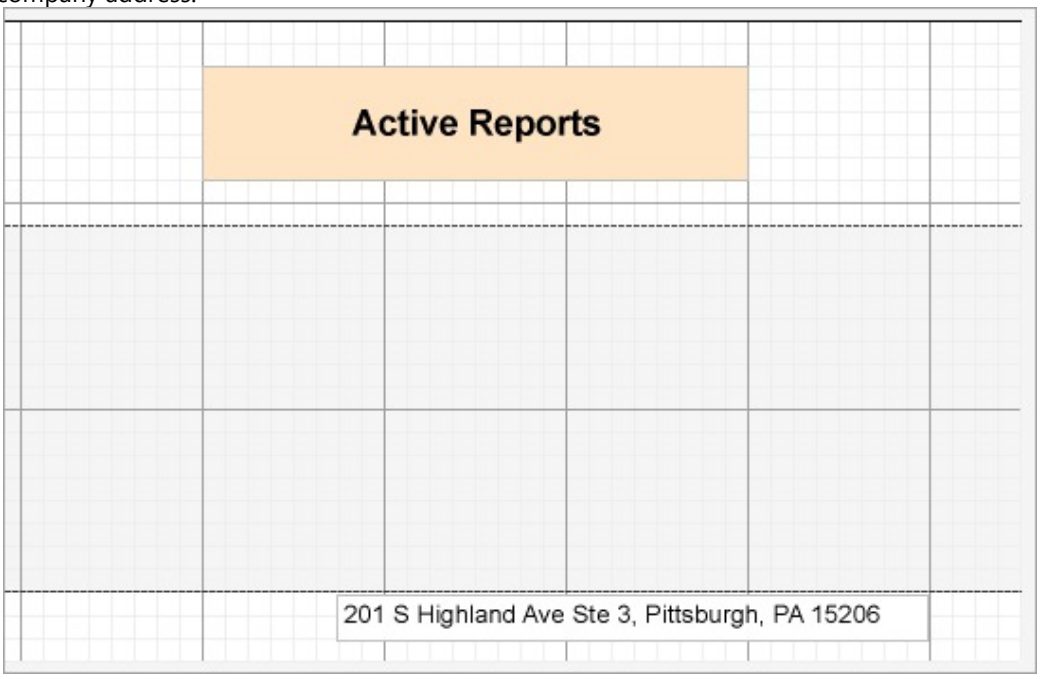
In WebDesigner, go to **Report** menu > **Convert to Master Report**.



The same option in Visual Studio Integrated Designer can be accessed from **Extensions** > **Report**.



2. Right-click the ruler area to the top or left of the report and choose **Page header.** Repeat and choose **Page footer**.
3. From the toolbox, drag and drop controls into the page header and footer sections. These controls appear on every page of a content report when you apply the master report. For example, a company logo image, or a textbox with the company address.



4. A **ContentPlaceHolder** control appears in the toolbox when you convert an RDLX report to a Master Report. This control defines regions where users can add content. Use the following instructions to add the ContentPlaceHolder control in the master report.

    1. From the toolbox, drag and drop the ContentPlaceHolder control onto the report design surface.
    2. Right-click the control and from the context menu that appears, select **Properties** to open the properties window.
    3. Set the following properties for the ContentPlaceHolder control.

Property	Description
Location	To position the control with respect to the top left corner of the container.
Size	To set the control size for determining the space available to design a content report.
Text	To add instructive text for the user. E.g. "Design your content report here." This caption appears in the design view and not in the final output.

5. Save the master report locally on your system. In the standalone designer, the option is in the **File** menu.

    1. In Visual Studio, select the report, and from the **Extensions** menu > **Report** > **Save Layout**.
    2. In the **Save As** dialog that appears, navigate to the location where you want to save the report and click **Save** to save the report in rdlx-master file format.

Use a Master Report to Create a Content report

The reports to which you apply the master report are content reports. A content report is not valid on its own, and cannot be run without its specified master report.

When the user creates a new report and sets a master report on it, the design view is effectively the opposite of the design view of the master report. Any report controls overlaid by ContentPlaceHolder controls are not visible in the content report at design time, but are visible at run time. These are the only areas where users can add report controls.

1. With focus on the report to which you want to apply the master report, from the Report menu, select **Set Master Report**, then **Open Local File**.
2. In the Open dialog that appears, navigate to the location where you saved the master report and open it. The master report layout is applied to the content report with all areas locked except for the region with the ContentPlaceHolder, which is available for use.

While designing a Content report, user can:

- Add elements that do not exist in the master report.
- Add new data sources that do not exist in the master report.
- Add new datasets from a data source in the master report.
- Add images to the EmbeddedImages collection.
- Add parameters to the ReportParameter collection.
- Add any number of report controls into placeholder rectangles designated by the master report.
- Modify the report name and description.
- Add new custom code that does not exist in the master report.

While designing a Content report, user cannot:

- Modify or remove elements that exist in the master report (disabled grey area).
- Remove a master report data source.
- Remove a master report dataset or modify its query.
- Modify the sort or filter on a master report dataset.
- Remove images from the EmbeddedImages collection.

- Remove parameters from the ReportParameter collection.
- Modify the margins or page settings of the master report.

> **Note:** Code in the master report is hidden in the content report, so in order to allow content report users to access code, the master report developer must provide information.

Run-Time Sequence of Events

This is what happens behind the scenes when you run a content report.

1. ActiveReports loads the content report.
2. The loader parses the master report tag on the content report and requests the master report from the resource resolver.
3. The master report is loaded into the definition.
4. As each ContentPlaceHolder in the content report is parsed, it finds the corresponding placeholder in the master report and loads the content from the content report into it.
5. Data sources, datasets, and fields are merged. The master report has higher priority if there is a conflict.
6. Themes are merged. The master report has higher priority if there is a conflict.
7. Report properties from the content report are added to those of the master report. For the following properties, the content report has a higher priority in case of conflict:
    - Report Description
    - Report Author
    - Report AutoRefresh
    - Report Custom
    - Report Language
    - Report DataTransform
    - Report DataSchema
    - Report ElementName
    - Report DataElementStyle
    - Dataset filters
    - Report Theme
    - Report Code
    - All content inside the ContentPlaceHolder controls

Modifying an Aggregated Report Definition

When you run a content report, the content report and its master combine to form an aggregated report definition. Using the ReportDefinition API, you can save this aggregate at run time as a third report definition which has no master or content report. Once this aggregate is saved as a normal report definition (*.rdlx file) you can edit it like any other report definition.

Advantages of a Master Report

- Implement common report functionality such as adding consistent page headers and footers within master reports.
- Apply company-wide changes in information such as address changes to a single master instead of modifying each report individually.
- Apply widespread data-related changes (such as data source location) to a single master report instead of modifying each report.
- Create code, data sources, themes and page layouts that are shared across the application or enterprise.
- Hide report complexity from end users who can use the standalone designer application to create content reports.

Advantages of a Shared Master Report

Shared master reports offer the advantages of a local master report, plus:

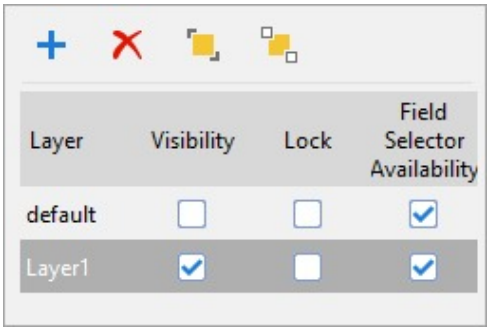
- They make your reports portable
- They allow multiple authors to use them

Layers

Layers can be understood as a group of controls that can overlay with other groups of controls on a Page or an RDLX report. The controls are editable only on the currently selected layer. You can toggle between the layers and create a design in each layer. You can lock or unlock, add or remove, and show or hide these layers. When you create a new report, a default layer is automatically added to it.

This feature is available in both Page and RDLX reports.

Elements of Layers Editor



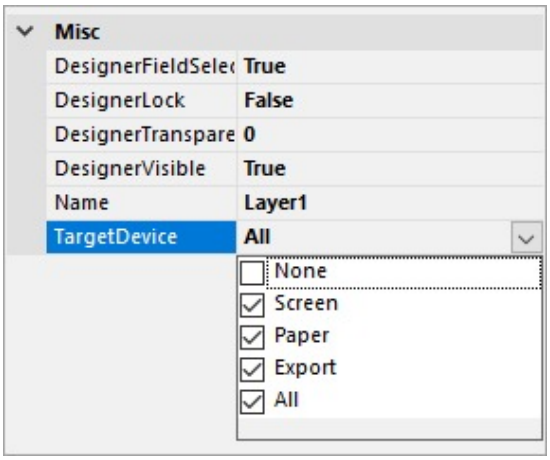
The Layers Editor can be accessed from the Toolbox > Layers on the left panel of the designer. The following table lists the actions you can perform through the Layers Editor.

Layers Element	Description
New	Adds a layer to the layer collection.
Remove	Removes the selected layer from the layer collection.
Bring to Front	Brings the group of controls placed on a selected Layer to the front of the controls on other layers.
Send to Back	Brings the group of controls placed on a selected Layer to the back of the controls on other layers.
Visibility	Toggles the visibility of the selected layer at design time. This lets you see what controls are there on each layer.
Lock	Toggles locking and unlocking of the layers. Locking a layer locks all of the controls on it to prevent accidental changes.
Field Selector Availability	Controls the visibility of the data field icons for the entire report or certain controls placed in a specific layer.

	Note: The property is only applicable on selected controls, which are: TextBox, CheckBox, InputField, Image, Table, and Tablix.

Important Properties

Selecting any Layer from the Layers editor reveals the following properties in the Properties panel.



Property	Value	Description
DesignerFieldSelectorAvailability	True/False	Control the visibility of the data field icons for the entire report or certain controls placed in a specific layer. This property can also be set using the checkbox for **Field Selector Availability** in the Layers Editor.
DesignerLock	True/False	Locks or unlocks controls placed on a Layer. You cannot move or resize the controls placed on the design surface of a locked Layer through a keyboard or a mouse. Other editing functions like cut, copy, or paste and addition or deletion of controls are possible. This property can also be set using the checkbox for **Lock** in the Layers Editor.
DesignerTransparency	0 to 1	Sets the transparency of the controls on a Layer at design time to a value between 0 and 1. A Layer with transparency set to 1 is not visible on the designer.
DesignerVisible	True/False	Determines if the controls placed on a Layer are visible on the designer or not. This property can also be set using the checkbox for **Visibility** in the Layers Editor.
Name	Layer Name (string)	Sets the name of a Layer (except the Default Layer).
TargetDevice	None, Screen, Paper, Export, All	Specifies or limits the visibility of controls placed on a Layer based on the selected target. See **View, Export, or Print Layers** for information on TargetDevice-

Property	Value	Description
		specific outputs.

View, Export, or Print Layers

The **TargetDevice ('TargetDevice Property' in the on-line documentation)** property determines whether you can view, export, or print the controls placed on a Layer. This property allows you to show or hide the controls that belong to a Layer on a specific target device. For example, if you want to display controls placed on Layer1 in the WinViewer and export the output to PDF, you can select the **Screen** and **Export** options of the TargetDevice property simultaneously.

> **Note:** TargetDevice property only determines the target (i.e. Screen, Export, or Print) where the group of controls placed on a Layer appears. In order to get the output on a viewer or export or print the controls, you need to add code for exporting or printing, or adding a viewer to the application.

TargetDevice property allows you to select from the following options:

- **None**
- **Screen**
- **Paper**
- **Export**
- **All**

Setting the TargetDevice Property

Use the following steps to set or change the **TargetDevice** property of a Layer:

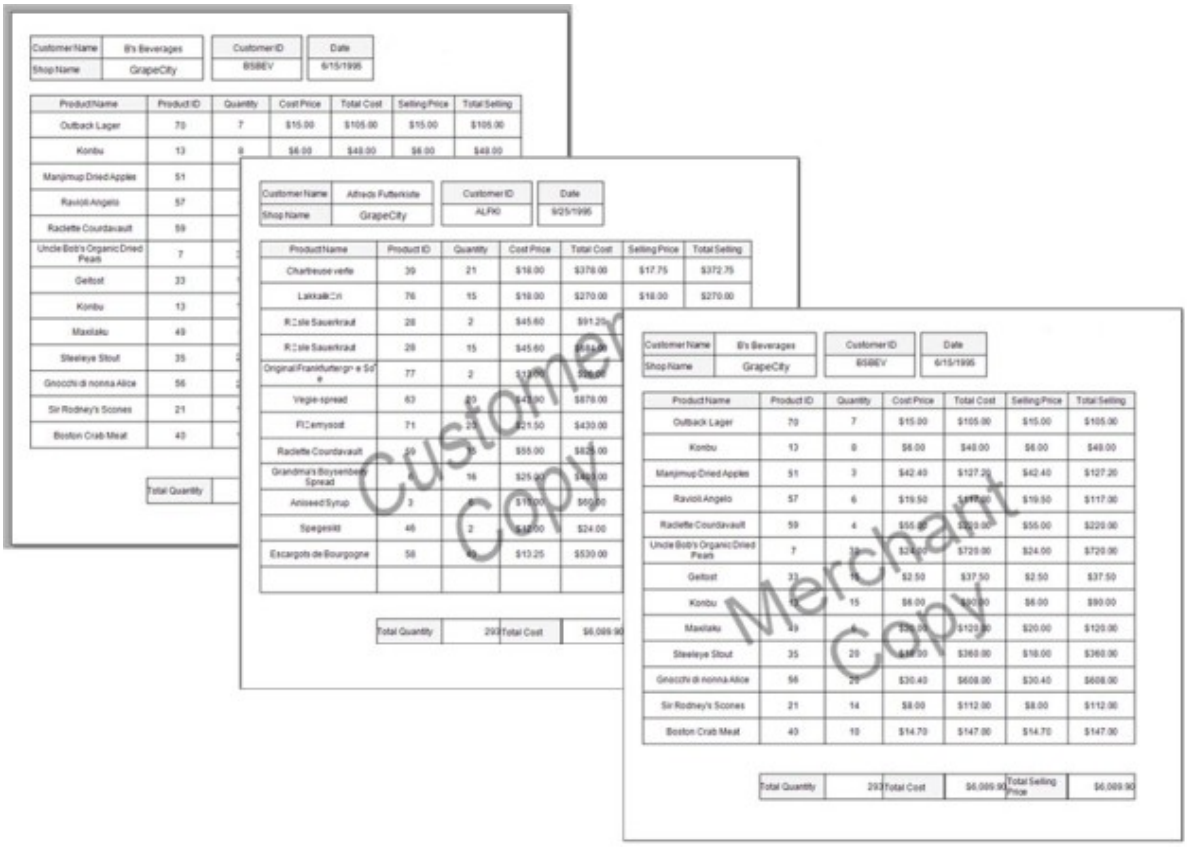
1. Select a Layer from the Layers List window. This becomes the Active Layer of the control.
2. In the Properties window, select the **TargetDevice** property.
3. From the dropdown, select out of the options: **None, Screen, Paper, Export**, or **All**.
   You can also select more than one option out of Screen, Paper and Export at the same time.

TargetDevice	Output Type	Description
None	-	Controls placed on a Layer cannot be viewed, exported or printed.
Screen	WinViewer	Controls placed on a Layer can be viewed on any of the supported viewers.
	WebViewer	
	WPF Viewer	
Paper	Physical/ Virtual Printer	Controls placed on a Layer are can be printed to physical or virtual printers.
Export		Controls placed on a Layer are exported to any of the supported file formats. See Export in Desktop Viewer for more information on the export formats.
All	All Outputs	Controls placed on a Layer can be viewed, exported and printed.

Add a watermark when your report is exported

Leverage the advantage of Layers in scenarios where you do not want to make changes to an existing report but want to perform minor modifications to the layout. With Layers, it is possible to make modifications to the same report without changing the original report layout. Let's take an example of a sales receipt to see how Layers can help in this scenario.

The requirement in this scenario is that a hard copy of the report is printed with a **Customer Copy** watermark and the soft copy of the same report is exported to an export format in a PDF format with a **Merchant Copy** watermark.



We have created the original sales receipt layout in the default layer and locked the layer by checking the Lock element. This step is necessary to make sure that the existing layout is not modified while making edits in the other layer.

After that, we added two Layers, one layer to contain the Customer Copy watermark image, and the other to contain the Merchant Copy watermark image. The **TargetDevice** property for the Customer Layer is set to **Paper** for printing a hard copy of the sales receipt and for the Merchant Layer to **Export** for exporting it to an export format.

> **Note**: PDF Export has the WatermarkPrintOnly setting that allows adding a watermark to an exported PDF file when printed.

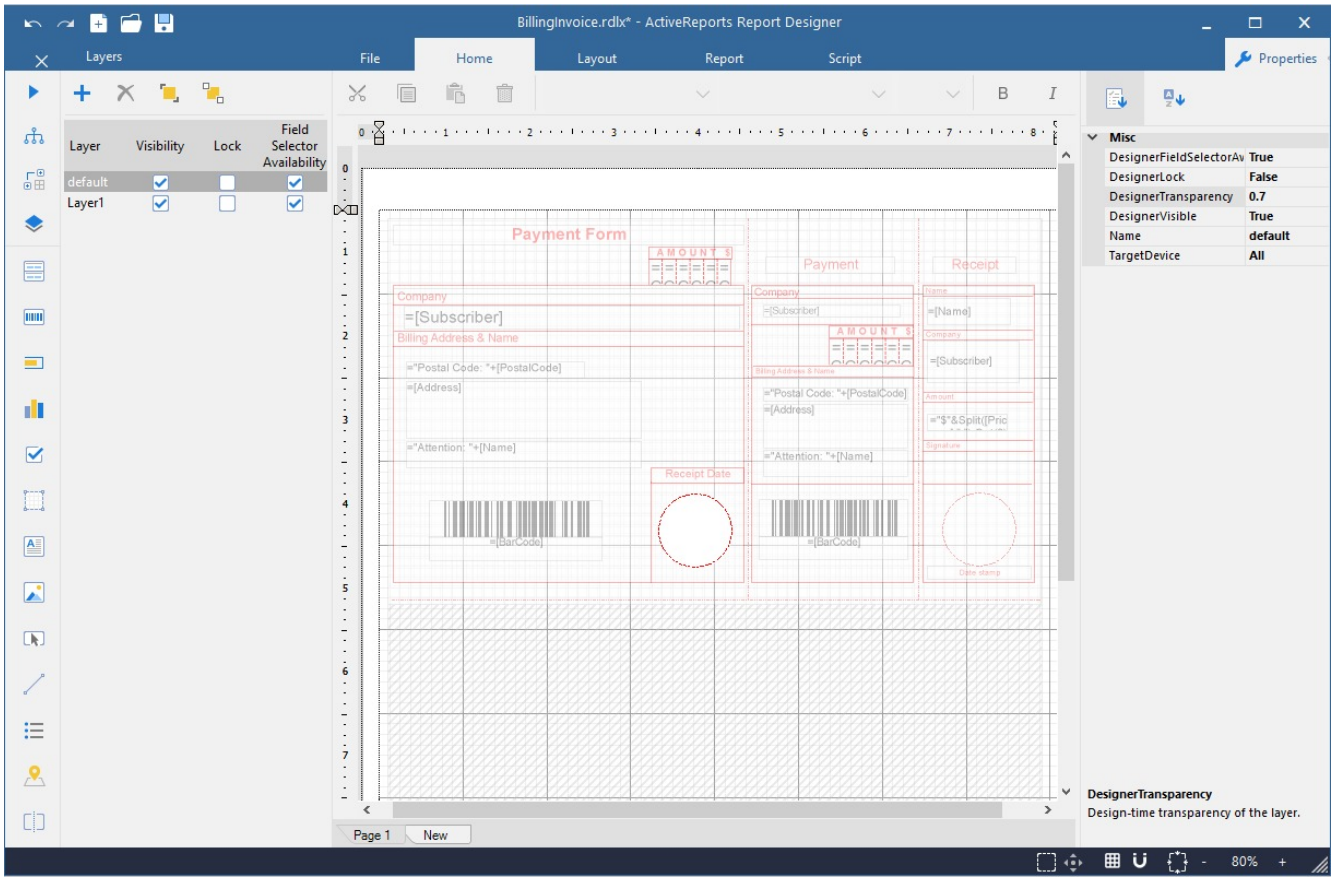
Create Pre-Printed Form

Layers can be used to create a layout of a pre-printed form, a form with some fixed data. This is particularly useful when you

want to replicate a layout of a pre-printed form that is either uneditable or unavailable in soft copy.

Let us take an example of an order invoice sent to the customers to see how Layers can help replicate the layout of the scanned image easily. You can also follow the following steps to create the layout of your desired report.

1. Create a report layout on the Default Layer. We created the invoice report layout by adding controls on the Default Layer.
2. Set the **DesignerLock** property of this Layer to **True** to make sure the controls are not modified or changed by mistake.
3. Set the **DesignerTransparency** property of the Default Layer to 0.5 to display the scanned image placed on Default Layer in the background.
4. Place a date stamp, for example, on Layer1, and set the **TargetDevice** property to **All**.



Expressions

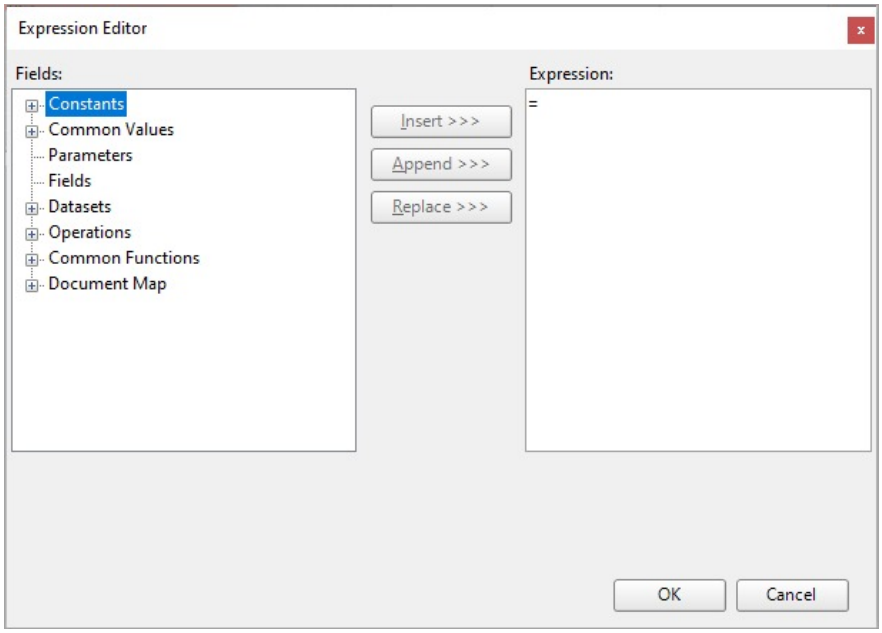
In ActiveReports, you can use an expression to set the value of a control in the report, or set conditions under which certain styles apply. You can set Microsoft Visual Basic® .NET in expressions through,

- Properties in the properties window
- Expression Editor dialog

All expressions begin with an equal sign (=). Even the expression for a field value for a TextBox is set as follows:
`=Fields!LastName.Value`

Expression Editor Dialog

You can build expressions quickly using the Expression Editor dialog. This dialog allows you to choose from a number of fields available to the report as well as to a particular property. You can access the Expression Editor by selecting nearly any property of a control and choosing **<Expression...>** from the drop-down list.



There are the following types of fields available in the Expression Editor:

- **Constants**
  Constants available for properties which have enumerated values such as TextDecoration or BorderStyle.
- **Common Values**
  Run time values available to every property in every report. There are two variables in this list which come from the User collection: User ID and User Language. See **Common Values** for further information.
- **Parameters**
  Parameters fields available in reports which contain report parameters. If available, you can choose a parameter from this field to retrieve the current value of the parameter.
- **Fields (*DataSet name*)**
  All fields from a dataset which is linked to the report control.
- **Datasets**
  All fields in each dataset associated with the report. However, the report retrieves only the sum or the first value of any field that is not within the current dataset scope.
- **Operations**
  Arithmetic, comparison, concatenation, logical/bitwise, bit shift operators for creating custom expressions.
- **Common Functions**
  Predefined Visual Basic .NET functions for which ActiveReports provides intrinsic support. See **Common Functions** for more information.
- **Document Map**
  The DocumentMap.Path expression defines labels for the report's TableOfContents members. The example of such expression is `=DocumentMap.Path & "General Information"`. If this expression is defined in the **Label** property of the report's control associated with the report's TableOfContents, **General Information** will be displayed as the label of the corresponding report's TableOfContents member.

Create an Expression in the Expression Editor

The Expression Editor dialog is composed of two panes, Fields and Expression.

1. From the Fields pane, select a field you want to use in your expression.
2. Click the Replace, Insert or Append button to add the field to the Expression pane. The expression pane shows the fields in a valid expression format.
3. Click OK to close the dialog.

The expression appears as the property value in the properties grid.

> **Tip:** While building an expression, you can directly add the entire expression or part of it in the Expression pane of the Expression Editor. Then use the Insert or Append buttons to create a complete expression.

Using Expressions in Reports

In the raw form, your data may not be ideally suited for display in a report. You can customize it and bring it into shape using expressions. Following are some examples of how expressions are set in different scenarios.

Concatenating Fields and Strings

You can concatenate fields with strings and with other fields. For e.g., use the following expression to get a result like **Customer Name: Bossert, Lewis**.

```
="Customer Name: " & Fields!LastName.Value & "," & Fields!FirstName.Value
```

Conditional Formatting

You can use expressions in properties like Color, Font, Border etc. on specific field values based on a condition, to highlight a part of data. The formula for conditional formatting is:
=iif(Fields!*YourFieldName*.Value *operator* "*Value to compare*", "*If condition is met, use this value.*", "*If not, use this one.*")

For e.g., if you enter the following expression in the **Font > FontWeight** property of a textbox that displays names of people, you get the name "Denise" in bold.

```
=iif(Fields!FirstName.Value = "Denise", "Bold", "Normal")
```

Functions

You can use a number of aggregate and other functions in your expressions. ActiveReports includes a range of functions, including running value, population standard variance, standard deviation, count, minimum and maximum. For e.g., use the following expression to get a count of employees.

```
=Count(Fields!EmployeeID.Value, Nothing)
```

As you design the report, the full text of an expression can get very long. ActiveReports makes expressions easier to read by shortening them.

When an expression is in the form:
=Fields!<FieldName>.Value

On the design surface, you see this text inside that TextBox:
=[<FieldName>]

Double-click the TextBox to view the full expression in edit mode.

For aggregates too, when the Expression value is:
=<Aggregate>(Fields!<FieldName>.Value)

On the design surface, you see this text inside the TextBox:
=<Aggregate>([<FieldName>])

This shortened expression value is only a visual change to allow you to see the field name easily. It shows up in both the TextBox on the design surface as well as any dropdown boxes inside the dialogs.

> **Note:** You can type the format as listed above for either field name values or aggregates on field names. This evaluates the full expression when the report is viewed.

Besides the shorthand for field names, you can also type shorthand like [@Param] for parameters and [&Value] for Globals such as [&PageNumber] on the design surface. Please note that you cannot use shorthand in the Expression Editor.

Common Values

Common Values are run time values available to every property in every report. You can directly drag and drop these common values from the Report Explorer onto the design surface or add and modify the values from the Expression Editor. Following is a list of the values that you can see under the Common Values node in the Report Explorer and in the Expression Editor.

Value	Description	Expression
Page N of M	Gets both the current page and the total number of pages in the report.	`="Page " & Globals!PageNumber & " of" & Globals!TotalPages`
Page N of M (Section)	Gets both the current page and the total number of pages in the report section.	`="Page " & Globals!PageNumberInSection & " of " & Globals!TotalPagesInSection`
Page N of M (Cumulative)	Gets both the current page and the total number of cumulative pages in a report.	`="Page " & Globals!CumulativePageNumber & " of " & Globals!CumulativeTotalPages`
Current Date and Time	Gets the date and time when the report began to run.	`=Globals!ExecutionTime`
User ID	Gets the machine name/user name of the current user.	`=User!UserID`
Page Number	Gets the current page number in the report.	`=Globals!PageNumber`

Page Number (Section)	Gets the current page number in the report section.	`=Globals!PageNumberInSection`
Total Pages	Gets the total number of pages in the report.	`=Globals!TotalPages`
Total Pages (Section)	Gets the total number of pages in the report section.	`=Globals!TotalPagesInSection`
Cumulative Page Number	Gets the current cumulative page number.	`=Globals!CumulativePageNumber`
Cumulative Total Pages	Gets the total number of cumulative pages in the report.	`=Globals!CumulativeTotalPages`
Report Folder	Gets the name of the folder containing the report.	`=Globals!ReportFolder`
Report Name	Gets the name of the report.	`=Globals!ReportName`
User Language	Gets the language settings of the current user.	`=User!Language`

Note: Page N of M (Section), Page Number (Section) or Total Pages (Section) is applied to page numbering when you set grouping in a report. Each section represents a group, not to be confused with sections in a Section report.

Note: Page N of M (Cumulative), Page Number (Cumulative) or Total Pages (Cumulative) is applied to page numbering when you use collation in a report.

Common Functions

You can use a function in an expression to perform actions on data in data regions, groups and datasets. You can access these functions in the Expression Editor dialog. In any property that accepts expressions, you can drop down the property and select **<Expression...>** to open the dialog.

Within the Expression Editor dialog, there is a tree view of Fields. Expand the **Common Functions** node to view the available functions. The following tables contain details about each of the functions included in ActiveReports for use in property expressions.

Date & Time

These are all methods from the DateAndTime class in Visual Basic. Please see the msdn DateAndTime Class topic for information on overloads for each method.

These are all the available aggregate functions:

Function	Description	Syntax and Example
DateAdd	Returns a date and time value that is the result of adding the interval to the date and time field of the specified unit.	`DateAdd(<DateInterval>,<Number>,<DateTime>)` `=DateAdd("d", 5, Fields!SaleDate.Value); =DateAdd(DateInterval.Day, 5, Fields!SaleDate.Value)`
DateDiff	Returns the difference between the start date and time and end date and time of the specified unit.	`DateDiff(<DateInterval>,<DateTime1>,<DateTime2>[,<DayOfWeek>[,WeekOfYear]]))` `=DateDiff("yyyy"), Fields!SaleDate.Value,"1/1/2015");=DateDiff(DateInterval.Year,Fields!SaleDate.Value,"1/1/2015")`
DatePart	Returns the Integer value that represents the specified part of the given date.	`DatePart(<DateInterval>,<DateTime1>[,<FirstDayOfWeek>[,FirstWeekOfYear]]))` `=DatePart("m", Fields!SaleDate.Value)`
DateSerial	Returns a Date value that represents a specified year, month, and a day, with the time information set to midnight (00:00:00).	`DateSerial(<Year Number>,<Month Number>,<Day Number>)` `=DateSerial(DatePart("yyyy", Fields!SaleDate.Value)-10, DatePart("m", Fields!SaleDate.Value)+5, DatePart("d", Fields!SaleDate.Value)-1)`

DateString	Returns the String value that represents the current date in your system.	`DateString()` `=DateString()`
DateValue	Returns a Date value that contains the information on date represented by a string, with the time set to midnight (00:00:00).	`DateValue(<StringDate>)` `=DateValue("December 12, 2015")`
Now	Returns the current date and time in your system.	`Now()` `=Now()`
Today	Returns a Date value that contains the current date in your system.	`Today()` `=Today()`
Day	Returns an Integer value from 1 through 31 that represents the day of the month.	`Day(<DateTime>)` `=Day(Fields!SaleDate.Value)`
Hour	Returns an Integer value from 0 through 23 that represents the hour of the day.	`Hour(<DateTime>)` `=Hour(Fields!SaleDate.Value)`
Minute	Returns an Integer value from 0 through 59 that represents the minute of the hour.	`Minute(<DateTime>)` `=Minute(Fields!SaleDate.Value)`
Month	Returns an Integer value from 0 through 12 that represents the month of the year.	`Month(<DateTime>)` `=Month(Fields!SaleDate.Value)`
MonthName	Returns the name of the month specified in the date as a String.	`MonthName(<Month Number>[,<Abbreviate>])` `=MonthName(Fields!SaleDate.Value)`
Second	Returns an Integer value from 0 through 59 that represents the second of the minute.	`Second(<DateTime>)` `=Second(Fields!SaleDate.Value)`
TimeSerial	Returns a Date value that represents a specified hour, minute, and	`TimeSerial(<Hour Number>, <Minute Number>, <Second Number>)` `=TimeSerial(DatePart("h", Fields!SaleDate.Value), DatePart("n", Fields!SaleDate.Value), DatePart("s", Fields!SalesDate.Value))`

	second, with the date information set relative to January 1 of the year 0001.	
TimeValue	Returns a Date value that contains the information on time represented by a string, with the date set to January 1 of the year 0001.	`TimeValue(<StringTime>)` `=TimeValue("15:25:45"); TimeValue(Fields!SaleDate.Value)`
TimeOfDay	Returns a Date value containing the current time of day in your system.	`TimeOfDay()` `=TimeOfDay()`
Timer	Returns a Double value that represents the number of seconds elapsed since midnight.	`Timer()` `=Timer()`
TimeString	Returns the String value that represents the current time of day in your system.	`TimeString()` `=TimeString()`
Weekday	Returns an Integer value that contains a number representing the day of the week.	`Weekday(<DateTime[,<DayOfWeek>])` `=Weekday(Fields!SaleDate.Value,0)`
WeekdayName	Returns a String value that contains the name of the specified weekday.	`WeekdayName(<WeekDay>[,<Abbreviate[, <FirstDayOfWeek>]])` `=WeekdayName(3, True, 0); =WeekDayName("w", Fields!SaleDate.Value), True, 0)`
Year	Returns an Integer value from 1 through 9999 representing the year.	`Year(<DateTime>)` `=Year(Fields!SaleDate.Value)`
Quarter	Returns an Integer value from 1 through 4 representing the quarter number.	`Quarter(<DateTime>)` `=Quarter(Fields!SaleDate.Value)`
QuarterName	Returns a string value representing the quarter name.	`QuarterName(<DateTime>)` `=QuarterName(Fields!SaleDate.Value)`

Math

These are all methods and fields from the System.Math class. Please see the msdn Math Class topic for information on overloads for each method.

Function	Description	Syntax and Example
Abs	Returns the absolute or positive value of a single-precision floating-point number.	`Abs(<Number>)` `=Abs(-5.5);=Abs(Fields!YearlyIncome.Value-80000)`
Acos	Returns the angle whose cosine is the specified number.	`Acos(<Number>)` `=Acos(.5); =Acos(Fields!Angle.Value)`
Asin	Returns the angle whose sine is the specified number	`Asin(<Number>)` `=Asin(.5); =Asin(Fields!Angle.Value)`
Atan	Returns the angle whose tangent is the specified number.	`Atan(<Number>)` `=Atan(.5); =Atan(Fields!Angle.Value)`

Atan2	Returns the angle whose tangent is the quotient of two specifed numbers.	`Atan2(<Number1>,<Number2>)`
		`=Atan2(3,7);`
		`=Atan2(Fields!CoordinateY.Value,Fields!CoordinateX.Value)`

BigMul	Returns the multiplication of two 32-bit numbers.	`BigMul(<Number1>,<Number2>)`
		`=BigMul(4294967295,-2147483647);`
		`=BigMul(Fields!Int32Value.Value, Fields!Int32Value.Value)`

Ceiling	Returns the smallest integer greater than or equal to the specified double-precision floating-point number.	`Ceiling(<Number>)` `=Ceiling(98.4331); =Ceiling(Fields!AnnualSales.Value /6)`
Cos	Returns the smallest integer greater than or equal to the specifed double-precision floating-point number.	`Cos(<Number>)` `=Cos(60)`
Cosh	Returns the hyperbolic cosine of the specified angle.	`Cosh(<Number>)` `=Cosh(60)`
E	Returns the value of E, which is 2.71828182845905.	`E` `=E*2`
Exp	Returns e raised to the specified ^, where is Euler s number. It is the inverse of the Log function.	`Exp(<Number>)` `=Exp(3); =Exp(Fields!IntegerCounter.Value)`
Fix	Returns the integer portion of a number.	`Fix(<Number>)` `=Fix(-7.15); =Fix(Fields!AnnualSales.Value /-5)`
Floor	Returns the longest integer less than or equal to the specified double-precision floating-point number.	`Floor(<Number>)` `=Floor(4.67); =Floor(Fields!AnnualSales.Value/ 12)`
IEEERemainder	Returns the remainder after division of one number by another according to IEEE satndards.	`IEEERemainder(<Number1>,<Number2>)` `=IEEERemainder(9,8)`
Log	Returns the logarithm of the specified number.	`Log(<Number>)` `=Log(20.5); =Log(Fields!NunberValue.Value)`
Log10	Returns the logarithm of the specified number to the base 10.	`Log10(<Number>)` `=Log10(20.5); =Log10(Fields!NumberValue.Value)`
Max	Returns the maximum non-null value from the specified expression.	`Max(<Value>)` `=Max(Fields!OrderTotal.Value)`
Min	Returns the minimum non-null value from the specified expression.	`Min(<Value>)` `=Min(Fields!OrderTotal.Value)`
PI	Returns the value of PI, which is 3.14159265358979.	`PI` `=2 * PI * Fields!Radius.Value`
Pow	Returns one number raised to the ^ of another number.	`Pow(<Number1,<Number2>)` `=Pow(Fields!Quantity.Value, 2)`
Round	Returns the round-off of a decimal number to the nearest	`Round(<Number>)`

	integer or to the nearest decimal number up to the specified digits.	`=Round(12.456); =Round(Fields!AnnualSales.Value / 12.3)`
Sign	Returns a value indicating the sign of an 8-bit signed integer.	`Sign(<Number>)` `=Sign(Fields!AnnualSales.Value-60000)`
Sin	Returns the sine of the specified number.	`Sin(<Number>)` `=Sin(60)`
Sinh	Returns the hyperbolic sine of the specified angle.	`Sinh(<Number>)` `=Sinh(60)`
Sqrt	Returns the square root of the specified number.	`Sqrt(<Number>)` `=Sqrt(121)`
Tan	Returns the tangent of the specified number.	`Tan(<Number>)` `=Tan(60)`
Tanh	Returns the hyperbolic tangent of the specified angle.	`Tanh(<Number>)` `=Tanh(60)`

Inspection

These are all methods from the DateAndTime class in Visual Basic. Please see the msdn DateAndTime Class topic for information on overloads for each method.

Function	Description	Syntax and Example
IsArray	Returns True if the expression can be evaluated as an array.	`IsArray(<Expression>)` `=IsArray(Parameters!Initials.Value)`
IsDate	Returns True if the expression represents a valid Date value.	`IsDate(<Expression>)` `=IsDate(Fields!BirthDate.Value); =IsDate("31/12/2010")`
IsDBNull	Returns True if the expression evaluates to a null.	`IsDBNull(<Expression>)` `=IsDBNull(Fields!MonthlySales.Value)`
IsError	Returns True if the expression evaluates to an error.	`IsError(<Expression>)` `=IsError(Fields!AnnualSales.Value = 80000)`
Isnothing	Returns True if the expression evaluates to nothing.	`IsNothing(<Expression>)` `=IsNothing(Fields!MiddleInitial.Value)`
IsNumeric	Returns True if the expression can be evaluated as a number.	`IsNumeric(<Expression>)` `=IsNumeric(Fields!AnnualSales.Value)`

ProgramFlow

These are all methods from the Interaction class in Visual Basic. Please see the msdn Interaction Class topic for more information.

Function	Description	Syntax and Example
Choose	Returns a value from a list of arguments.	`Choose(<Index>,<Value>[, <Value2>,...[, <Value N>]])` `=Choose(3, "10", "15", "20", "25")`
IIF	Returns the value if the expression evaluates to True, and the second value if the expression evaluates to False.	`IIF(<Condition>, <TruePart>, <FalsePart>)` `=IIF(Fields!AnnualSales.Value >= 80000, "Above Average", "Below Average")`
Partition	Returns a string (in the form x : y) that represents the calculated range based on the specified interval containing the specified number.	`Partition(<Value>, <Start>, <End>, <Interval>)` `=Partition(1999, 1980, 2000, 10)`
Switch	Returns the value of the first expression that evaluates to True among a list of expressions.	`Switch(<Condition1>, <Value1>[, <Condition2>, <Value2>,...[, <ConditionN>, <ValueN>]])` `=Switch(Fields!FirstName.Value = "Abraham", "Adria",`

		`Fields!FirstName.Value = "Charelotte", "Cherrie")`

Aggregate

You can use aggregate functions within report control value expressions to accrue data. ActiveReports supports aggregate functions from RDLX 2005, plus some proprietary extended set of functions. For all of the functions, you can add an optional <Scope> parameter.

These are all the available aggregate functions:

Function	Description	Syntax and Example
AggregateIf	Decides whether to calculate a custom aggregate from the data provider of the values returned by the expression based on a Boolean expression.	`AggregateIf(<Condition>, <AggregateFunction>, <AggregateArguments>)` `=AggregateIf(Fields!Discontinued.Value=True, Sum, Fields!InStock.Value)`
Avg	Calculates the average of the non-null values returned by the expression.	`Avg(<Values>)` `=Avg(Fields!Cost.Value, Nothing)`
Count	Calculates the number of non-null values returned by the expression.	`Count(<Values>)` `=Count(Fields!EmployeeID.Value, Nothing)`
CountDistinct	Calculates the number of non-repeated values returned by the expression.	`CountDistinct(<Values>)` `=CountDistinct(Fields!ManagerID.Value, "Department")`
CountRows	Calculates the number of rows in the scope returned by the expression.	`CountRows()` `=CountRows("Department")`
CrossAggregate	Calculates the specified function with the specified expression as an argument in the cross of the specified row and column.	`CrossAggregate(<Expression>, <FunctionName>, <ColumnGroupName>, <RowGroupName>)` `=CrossAggregate(Fields!Count.Value, "Sum", "MonthGroup", "ProductGroup")`
CumulativeTotal	Calculates the sum of page-level aggregates returned by the expression for current and previous pages.	`CumulativeTotal(<Expression>, <Aggregate>)` `=CumulativeTotal(Fields!OrderID.Value, Count)`
DistinctSum	Calculates the sum of the values returned by an expression using only the rows when the value of another expression is not repeated.	`DistinctSum(<Values>, <Value>)` `=DistinctSum(Fields!OrderID.Value, Fields!OrderFreight.Value, "Order")`
First	Shows the first value returned by the expression.	`First(<Values>)` `=First(Fields!ProductNumber.Value, "Category")`
Last	Shows the last value returned by the expression.	`Last(<Values>)` `=Last(Fields!ProductNumber.Value, "Category")`
Max	Shows the largest non-null value returned by the expression.	`Max(<Values>)` `=Max(Fields!OrderTotal.Value, "Year")`
Median	Shows the value that is the mid-point of the values returned by the expression. Half of the values returned will be above this value and half will be below it.	`Median(<Values>)` `=Median(Fields!OrderTotal.Value)`
Min	Shows the smallest non-null value returned by the expression	`Min(<Values>)` `=Min(Fields!OrderTotal.Value)`
Mode	Shows the value that appears most frequently in the values returned by the expression.	`Mode(<Values>)` `=Mode(Fields!OrderTotal.Value)`
RunningValue	Shows a running aggregate of values returned by the expression (Takes one of the other aggregate functions as a parameter),	`RunningValue(<Values>, <AggregateFunction>)` `=RunningValue(Fields!Cost.Value, Sum, Nothing)`
StDev	Calculates the dispersion (standard deviation) of all non-null values returned by the expression.	`StDev(<Values>)` `=StDev(Fields!LineTotal.Value, "Order")`
StDevP	Calculates the population dispersion (population standard	`StDevP(<Values>)`

	deviation) of all non-null values returned by the expression.	`=StDevP(Fields!LineTotal.Value, "Order")`
Sum	Calculates the sum of the values returned by the expression.	`Sum(<Values>)` `=Sum(Fields!LineTotal.Value, "Order")`
Var	Calculates the variance (standard deviation squared) of all non-null values returned by the expression.	`Var(<Values>)` `=Var(Fields!LineTotal.Value, "Order")`
VarP	Calculates the population variance (population standard deviation squared) of all non-null values returned by the expression.	`VarP(<Values>)` `=VarP(Fields!LineTotal.Value, "Order")`

Conversion

These are all methods from the Convert class in the .NET Framework. Please see the msdn Convert Class topic for more information.

Function	Description	Syntax and Example
ToBoolean	Converts the specified value to Boolean.	`ToBoolean(<Value>)` `=ToBoolean(Fields!HouseOwnerFlag.Value)`
ToByte	Converts the specified value to Byte.	`ToByte(<Value>)` `=ToByte(Fields!ProductNumber.Value)`
ToDateTime	Converts the specified value to a Date and Time value.	`ToDateTime(<Value>)` `=ToDateTime(Fields!SaleDate.Value); =ToDateTime("1 January, 2017")`
ToDouble	Converts the specified value to Double.	`ToDouble(<Value>)` `=ToDouble(Fields!AnnualSales.Value); =ToDouble(535.85 * .2691 * 67483)`
ToInt16	Converts the specified value to a 16-bit signed Integer.	`ToInt16(<Value>)` `=ToInt16(Fields!AnnualSales.Value); =ToInt16(535.85)`
ToInt32	Converts the specified value to a 32-bit signed Integer.	`ToInt32(<Value>)` `=ToInt32(Fields!AnnualSales.Value)`
ToInt64	Converts the specified value to a 64-bit signed Integer.	`ToInt64(<Value>)` `=ToInt64(Fields!AnnualSales.Value)`
ToSingle	Converts the specified value to a single-precision floating-point number.	`ToSingle(<Value>)` `=ToSingle(Fields!AnnualSales.Value); =ToSingle(15.857692134)`
ToUInt16	Converts the specified value to a 16-bit unsigned Integer.	`ToUInt16(<Value>)` `=ToUInt16(Fields!AnnualSales.Value)`
ToUInt32	Converts the specified value to a 32-bit unsigned Integer.	`ToUInt32(<Value>)` `=ToUInt32(Fields!AnnualSales.Value)`
ToUInt64	Converts the specified value to a 64-bit unsigned Integer.	`ToUInt64(<Value>)` `=ToUInt64(Fields!AnnualSales.Value)`

Miscelleneous

ActiveReports also offers several functions which do not aggregate data, but which you can use with an IIf function to help determine which data to display or how to display it.

The first four are miscellaneous functions from the RDLX 2005 specifications. GetFields is a proprietary function to extend RDLX specifications.

Function	Description	Syntax and Example
InScope	Determines whether the current value is in the indicated scope.	`InScope(<Scope>)` `=InScope("Order")`
Level	Returns the level of the current value in a recursive hierarchy.	`Level()` `=Level()`

Previous	Returns the previous value within the indicated scope.	`Previous(<Value>)` `=Previous(Fields!OrderID.Value)`
RowNumber	Shows a running count of all the rows in the scope returned by the expression.	`RowNumber()` `=RowNumber()`

GetFields	Returns an IDictionary<string,Field> object that contains the current contents of the Fields collection. Only valid when used within a data region. This function makes it easier to write code that deals with complex conditionals. To write the equivalent function without GetFields() would require passing each of the queried field values into the method which could be prohibitive when dealing with many fields.	`GetFields()`
		`=Code.DisplayAccountID(GetFields())`
		Custom function. Paste in the Code tab.
		`'Within the Code tab, add this function.`
		`Public Function DisplayAccountID( flds as Object) as Object`
		`  If flds("FieldType").Value = "ParentAccount" Then`
		`    Return flds("AccountID").Value`
		`  Else`
		`    Return flds("ParentAccountID").Value`
		`  End If`
		`End Function`

Lookup	Returns the first matching value for the specified name from the dataset with pairs of name and value. For more information, see Report Builder Functions - Lookup Function.	`Lookup(<SourceExpression>, <DestinationExpression>, <ResultExpression>, <LookupDataset>)` `=Lookup(Fields!ProductID.Value, Fields!ProductID.Value, Fields!Quantity.Value, "DataSet2")`
LookupSet	Returns multiple row values from a specified dataset and can be used for the 1-to-many relationship. For more information, see Report Builder Functions - LookupSet Function.	`LookupSet(source_expression, destination_expression, result_expression, dataset)` `=LookupSet(Fields!CategoryID.Value, Fields!CategoryID.Value, Fields!UnitsInStock.Value, "Products")`
MapPoint	Allows displaying simple data directly on the Map as a map Point Layer.	`MapPoint(<Latitude>, <Longitude>)` `=MapPoint(Fields!Latitude.Value, Fields!Longitude.Value)`
GroupIndex	Returns the index of the element in the current group.	`=GroupIndex()`
GroupIndex (with scope)	Returns the index of the element in the specified group.	`=GroupIndex(<Group>)`

Scope

All functions have a Scope parameter which determines the grouping, data region, or dataset to be considered when calculating the aggregate or other function. Within a data region, the Scope parameter's default value is the innermost grouping to which the report control belongs. Alternately, you can specify the name of another grouping, dataset, or data region, or you can specify **`Nothing`**, which sets it to the outermost data region to which the report control belongs.

The Scope parameter must be a data region, grouping, or dataset that directly or indirectly contains the report control using the function in its expression. If the report control is outside of a data region, the Scope parameter refers to a dataset. If there is only one dataset in the report, you can omit the Scope parameter. If there are multiple datasets, you must specify which one to use to avoid ambiguity.

Note: You cannot set the Scope parameter to `Nothing` outside of a data region.

LookupSet Function in Data Regions

The LookupSet function returns multiple row values from a specified dataset, so this function is used for one-to-many relationship in a data. The fields of the dataset returned by the LookupSet function behave as regular dataset fields that you can use in functions/aggregates within the scope of the data region.

The following data regions can use the LookupSet function in the **Value** property - Tablix, Table, Classic Chart, BandedList, List, and Sparkline.

Syntax

The basic syntax of the Lookup expression is as follows.

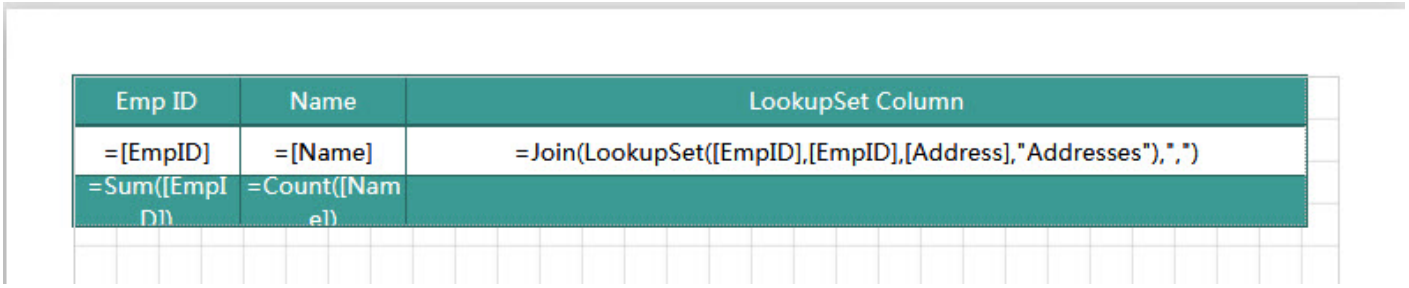
```
LookupSet(<SourceExpression>, <DestinationExpression>, <ResultExpression>, <LookupSetDataset>)
```

- **SourceExpression**: An expression that is evaluated in the current scope and that specifies the name or key to look up.
- **DestinationField**: An expression that is evaluated for each row in a dataset and that specifies the name or key to match on.
- **ResultExpression**: An expression that is evaluated for the row in the dataset where source_expression = destination_expression, and that specifies the value to retrieve.
- **LookupSetDataset**: A constant that specifies the name of a dataset in the report. For example, "ContactInformation".

The report below shows information on addresses for each employee and displays the addresses as string separated by commas. To display all addresses for each employee in a string separated by commas, we need to use the Join function in the expression with the LookupSet function.

For example:

```
=LookupSet(Fields!CategoryID.Value, Fields!CategoryID.Value, Fields!UnitsInStock.Value,
"Products")
=Join(LookupSet(Fields!CategoryID.Value, Fields!CategoryID.Value, Fields!UnitsInStock.Value,
"Products"), ",")
```



Emp ID	Name	LookupSet Column
10001	Davolio	Address01,Address02,Address03
10002	Fuller	Address04
20003	2	

Usage

- The data type of the SourceExpression and the DestinationExpression should be same.
- When the Lookup function is used as a value expression in a data region, the expression is evaluated for each row or repeated data of the data region's dataset.
- The Lookup function returns one value if found, and null if no rows are found in the Lookup dataset.
- The Lookup expressions can be a part of aggregated expressions. A user can use the Lookup function in a table group or table header or footer, and sum all values for the table.

Limitations

- Only "=" comparison is supported between SourceExpression and DestinationExpression.
- Non-aggregate expressions such as multiply, mod, AND and OR, are not allowed in the comparison criteria.
- Only one level of Lookup is allowed, that is, nested Lookup functions are not supported.

Data Visualizers

Several report controls support a type of expression called Data Visualizers that lets you create small graphs to make your data easy to comprehend for the user. For instance, you can red flag an overdue account using the Flags Icon Set as a background image in the TextBox control.

There are several types of Data Visualizers available through a dialog linked to properties on the report controls. Let's explore each type of data visualizers in depth.

Image Data Visualizers

These Data Visualizers are supported in the **Value** property of the image report control, as well as in the **BackgroundImage - Value** property of TextBox, CheckBox, Shape, and Container report controls. See the topics below to learn more.

> ⚠ **Caution**: In the following topics, the terms "argument" and "parameter" may seem interchangeable, but within an expression, an argument refers to the returned value of a parameter, while a parameter may be just a variable.

Icon Set
: Learn about the included image strips from which you can select using arguments. These include traffic lights, arrows, flags, ratings, symbols, and more, plus you can create your own custom image strips.

Range Bar
: Learn how you can provide minimum, maximum, and length arguments to render a 96 by 96 dpi bar image in line with your text to show a quick visual representation of your data values.

Range Bar Progress
: Learn how you can use a second bar to show progress along with the data range.

Data Bar
: Learn about data bars, which are similar to range bars with a few different arguments.

Gradient
: Learn how to apply gradient style to the background of a control.

Hatch
: Learn how to apply hatch style to the background of a control.

Background Color Data Visualizer

These Data Visualizers are available in the **BackgroundColor** property of the TextBox and CheckBox report

controls. See the following topics to learn more.

Color Scale 2
: Learn about displaying TextBoxes with a range of background colors that are keyed to the value of the data.

Color Scale 3
: Learn about color scales with an additional middle color value.

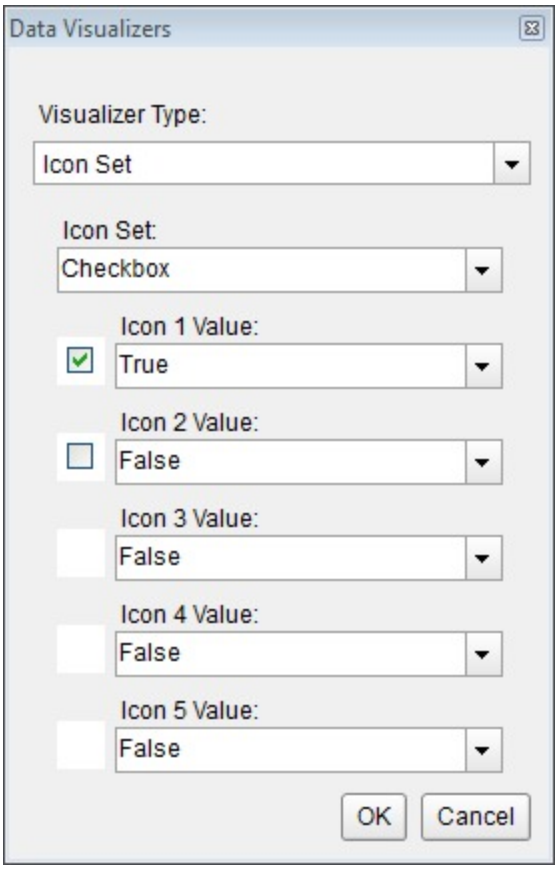
Icon Set

The Icon Set data visualization allows you to use arguments to select an image from an image strip and display it either as a BackgroundImage, or as a Value. You can use the standard image strips included with ActiveReports, or create custom image strips.

The Icon Set Data Visualizer is supported in the Image report control **Value** property, and also in the TextBox, CheckBox, Shape, and Container report controls' **BackgroundImage - Value** property.

Product ID	In Stock	Re Order Level	Difference	Icon Set
1000	5	7	-2	
1001	5	5	0	
1002	14	6	8	
1003	15	4	11	
1004	3	10	-7	
1005	5	8	-3	
1006	0	2	-2	
1007	6	7	-1	
1008	7	5	2	
1009	8	1	7	
1010	1	1	0	
1011	2	9	-7	
1012	4	1	3	
1013	0	8	-8	
1014	17	7	10	
1015	19	5	14	
1016	11	5	6	

To open the dialog, drop down the **BackgroundImage** property of TextBox, CheckBox, Shape, and Container report controls, or the **Value** property of Image report control, and select **<Data Visualizer...>**. To build the data visualizer expression, select the appropriate values for each of the options in the dialog.



Parameters

- **Icon Set**. This designates the name of the icon set to use.
- **Icon 1 Value**. A Boolean expression that, if it evaluates to True, renders this icon from the strip.
- **Icon 2 Value**. A Boolean expression that, if it evaluates to True, renders this icon from the strip.
- **Icon 3 Value**. A Boolean expression that, if it evaluates to True, renders this icon from the strip.
- **Icon 4 Value**. A Boolean expression that, if it evaluates to True, renders this icon from the strip.
- **Icon 5 Value**. A Boolean expression that, if it evaluates to True, renders this icon from the strip.

You can use static values or any expression that evaluates to a Boolean value. For icon sets with fewer than five icons, set the unused values to False.

Standard Image Strips

Name	Image
Checkbox	
3TrafficLights	
Arrows	
Blank	

Flags	
GrayArrows	
Quarters	
Ratings	
RedToBlack	
Signs	
Symbols1	
Symbols2	
TrafficLights	

Note: When using icon sets, you must set the **Source** property to **Database,** otherwise it may throw exception.

Syntax

```
=IconSet("Flags", False, True, False, False, False)
```

Usage

Following the Icon Set argument, there are five Boolean arguments. The first argument to evaluate to True displays the corresponding image. Use data expressions that evaluate to a Boolean value to replace the literal values in the code above.

Example

This expression displays the first symbol (the green flag) on each row in which the Difference exceeds 10, displays the second symbol (the yellow flag) on each row in which the quantity is greater than 0, and displays the third symbol (the red flag) on each row in which the quantity is equal to or below 0. Notice that we provide literal False values in the fourth and fifth arguments, which have no images in this strip.

Paste in the BackgroundImage Value property of a Textbox

```
=IconSet("Flags",Fields!TaxPercent.Value > 10,Fields!TaxPercent.Value >
0,Fields!TaxPercent.Value <= 0,False,False)
```

In several of the included image strips, the last spots are empty. When using the Checkbox, 3TrafficLights, Flags, RedToBlack, Signs, Symbols1, Symbols2, or TrafficLights image strip, it generally makes sense to set the Boolean values for all of the unused icon spaces to False.

Custom Image Strips

The Blank image strip is included so that you can customize it. Drop down the section below for details.

Custom image strips must conform to the following rules.

1. The format must be of a type that is handled by the .NET framework.
2. The size of the strip must be 120 x 24 pixels.
3. Each image must be 24 x 24 pixels in size.
4. There must be no more than five images in the strip.
5. If there are fewer than five images in the strip, there must be blank spaces in the image to fill in.

Types of Custom Images

Here is the syntax for various types of custom images, followed by examples of each.

External image syntax

```
=IconSet(location of image strip, condition#1, condition#2, condition#3, condition#4,
condition#5)
```

External image path example
`=IconSet("C:\Images\customstrip.bmp", 4 > 9, 5 > 9, 10 > 9, False, False)`

External image URL example
`=IconSet("http://mysite.com/images/customstrip.gif", 4 > 9, 5 > 9, 10 > 9, False, False)`

Image from an assembly resource syntax

```
=IconSet("res:[Assembly Name]/Resource name", condition#1, condition#2, condition#3,
condition#4, condition#5)
```

Assembly resource image example

`=IconSet("res:ReportAssembly,`
`Version=1.1.1.1./ReportAssembly.Resources.Images.CustomImage.png",`
`4 > 9, 5 > 9, 10 > 9, False, False)`

Embedded image syntax

```
=IconSet("embeddedImage:ImageName", condition#1, condition#2, condition#3, condition#4,
condition#5)
```

Embedded image example
`=IconSet("embeddedImage:Grades", Fields!Score.Value >=90, Fields!Score.Value >=80,` `Fields!Score.Value >=70, Fields!Score.Value >=60, True)`

Theme image syntax

```
=IconSet("theme:ThemeImageName", condition#1, condition#2, condition#3, condition#4,
condition#5)
```

Theme image example

`=IconSet("theme:Grades", Fields!Score.Value >=90, Fields!Score.Value >=80, Fields!Score.Value`
`>=70,`
`Fields!Score.Value >=60, True)`

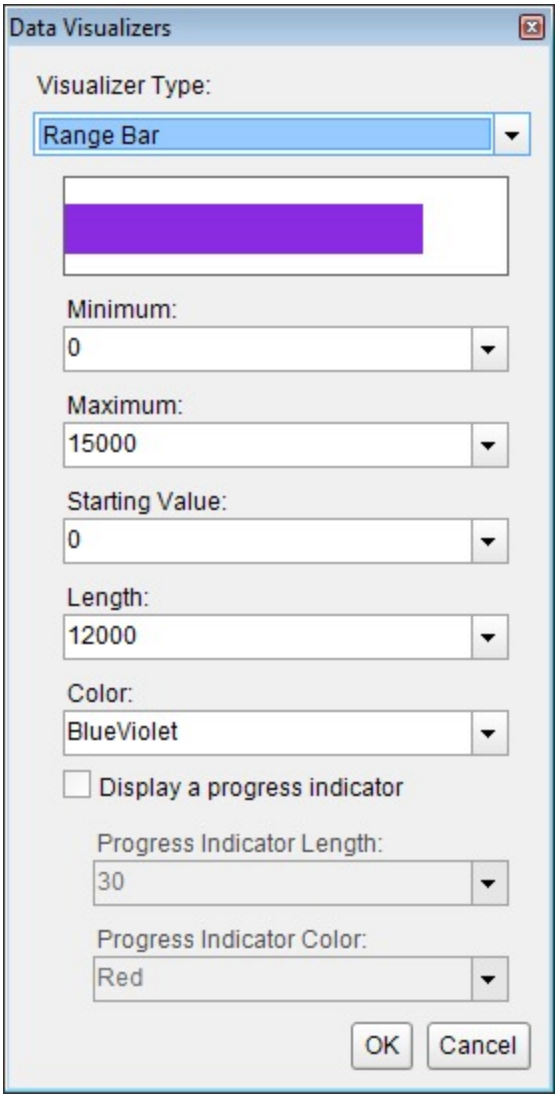
Range Bar

The Range Bar data visualization displays a 96 by 96 dpi bar image. The colored bar renders as half the height of the image, centered vertically. The amount of colored bar to render to the right of the Start argument (or to the left in the case of a negative value) is based on the Length argument. If the Length argument is zero, a diamond renders.

The Minimum and Maximum arguments determine the range of data. The area between the Length argument and the Maximum argument is transparent (or between the Length and the Minimum in the case of a negative value).

The Range Bar Data Visualizer is supported in the Image report control **Value** property, and also in the TextBox, CheckBox, Shape, and Container report controls' **BackgroundImage - Value** property.

When you select a TextBox, CheckBox, Shape, or a Container control on your report, in the Properties window or Properties dialog, you can drop down the **BackGroundImage Value** property and select **<Data Visualizer...>** to launch the dialog. The same is true if you select an Image control and drop down the **Value** property. To build the data visualizer expression, select the appropriate values for each of the options in the dialog.



Parameters

- **Minimum**. The lowest value in the range of data. This value corresponds to the leftmost edge of the image. If this argument is greater than the Start argument, Start becomes equal to Minimum. The data type is Single.
- **Maximum**. The highest value in the range of data. This value corresponds to the rightmost edge of the image. If this argument is less than the Start argument, Start becomes equal to Maximum. The data type is Single.
- **Color**. The HTML color string to use in rendering the Length in the bar image.
- **Start**. The point from which the Range Bar begins to be rendered. The data type is Single.
- **Length**. The width of the bar to render within the control. Setting this value to 0 renders a diamond shape instead of a bar. The data type is Single.

You can use static values or aggregate functions (e.g. Min or Max) to set the parameters. For more information on these and other aggregate functions, see the Common Functions topic.

Syntax

```
=RangeBar(Minimum, Maximum, Color, Start, Length)
```

Usage

Use this data visualizer to render a bar in the color specified, the length of which changes depending on the number returned by the Length parameter, in the case of the simple example, GrossProfit. If your data contains only positive values, Start corresponds with Minimum at the left edge of the DataBar. The area between the Length and the Maximum is transparent.

Simple Example

Set the Length parameter to the value of a field in your dataset to display the field values visually.

Paste into a TextBox BackgroundImage property

```
=RangeBar(0,15000,"BlueViolet",0,Fields!GrossProfit.Value)
```

Store Name	Gross Profit	Range Bar
Store #1000	$12,602.57	
Store #1001	$10,348.09	
Store #1002	$13,939.84	
Store #1003	$12,201.39	
Store #1004	$11,383.74	
Store #1005	$10,979.59	
Store #1006	$12,709.01	

Example Using Negative Values

When your data contains negative as well as positive values, you can use an Immediate If expression for the Color parameter. In this example, if the Projected Stock value is negative, it renders in Crimson, while positive values render in BlueViolet. You can also see that negative values render to the left of Zero and positive values render to the right. A Length value of exactly zero renders as a diamond.

Paste into a TextBox BackgroundImage property

```
=RangeBar(-5,20,IIf((Fields!InStock.Value - 5) < 0, "Crimson",
"BlueViolet"),0,Fields!InStock.Value-5)
```

Title	In Stock	Projected Sales	Projected Stock	Range Bar
Snow White and the Seven Dwarfs	5	5	0	
Gone with the Wind	14	5	9	
Bambi	5	5	0	
One Hundred and One Dalmatians	7	5	2	
Mary Poppins	2	5	-3	
Doctor Zhivago	17	5	12	
The Jungle Book	7	5	2	
Butch Cassidy and the Sundance Kid	15	5	10	
Love Story	16	5	11	
The Godfather	11	5	6	
The Sting	8	5	3	
The Towering Inferno	6	5	1	
Blazing Saddles	7	5	2	
Jaws	11	5	6	
One Flew Over the Cuckoo's Nest	3	5	-2	

Default Behavior

The function returns **null** (i.e. no image is rendered) in the following cases:

1. The **Maximum** is less than or equal to the **Minimum**.
2. The expression is placed in a property which does not take an image.
3. The **Source** property of the image is not set to **Database**.

The Start value changes in the following cases:

1. If the **Start** value is less than the **Minimum** value, **Start** is the same as **Minimum**.
2. If the **Start** value is greater than the **Maximum** value, **Start** is the same as **Maximum**.

The Length value changes in the following cases:

1. If the **Start** value plus the **Length** value is less than the **Minimum** value, **Length** becomes **Minimum** minus **Start**.
2. If the **Start** value plus the **Length** value is greater than the **Maximum** value, **Length** becomes **Maximum** minus **Start**.

If the argument for any of the parameters cannot be converted to the required data type, the default value is used instead.

Parameter	Default Value
Minimum	0

Maximum	0
Color	Green
Start	0
Length	0

Range Bar Progress

The Range Bar Progress data visualization displays a 96 by 96 dpi double bar image. The first colored bar renders as half the height of the image, centered vertically. The amount of colored bar to render to the right of the Start argument (or to the left in the case of a negative value) is based on the Length argument. If the Length argument is zero, a diamond renders.

The second colored bar renders using the ProgressColor as one fourth of the height of the image, centered vertically over the Length bar. The amount of colored bar to render to the right of the Start argument (or to the left in the case of a negative value) is based on the Progress argument. If the Progress argument is zero, a smaller diamond renders.

The Minimum and Maximum arguments determine the range of data. The area between the Length and Progress arguments and the Maximum argument is transparent (or between the Length and Progress and the Minimum in the case of a negative value).

The Range Bar Progress Data Visualizer is supported in the Image report control **Value** property, and also in the TextBox, CheckBox, Shape, and Container report controls' **BackgroundImage - Value** property.

Parameters

- **Minimum**. The lowest value in the range of data. This value corresponds to the leftmost edge of the image. If this argument is greater than the Start argument, Start becomes equal to Minimum. The data type is Single.
- **Maximum**. The highest value in the range of data. This value corresponds to the rightmost edge of the image. If this argument is less than the Start argument, Start becomes equal to Maximum. The data type is Single.
- **Color**. The HTML color string to use in rendering the Length, the thicker bar in the bar image.
- **Start**. The point from which the Range Bar Progress begins to be rendered. The data type is Single.
- **Length**. The length of the thicker bar to render within the control. Setting this value to 0 renders a diamond shape instead of a bar. The data type is Single.
- **ProgressColor**. The HTML color string to use in rendering the Progress, the thinner bar in the bar image.
- **Progress**. The length of the thinner bar to render within the control. Setting this value to 0 renders a diamond shape instead of a bar. The data type is Single.

You can use static values or aggregate functions (e.g. Min or Max) to set the parameters. For more information on these and other aggregate functions, see the Expressions topic.

Syntax

```
=RangeBarProgress(Minimum, Maximum, Color, Start, Length, ProgressColor, Progress)
```

Usage

Use this data visualizer to render a double bar in the colors specified, the length of which changes depending on the number returned by the Length parameter for the thick bar, in the case of the simple example, GrossSales. The thin bar

length is based on the value returned by the Progress parameter, in this case, GrossProfit. If your data contains only positive values, Start corresponds with Minimum at the left edge of the Range Bar. The area between the Length or Progress and the Maximum is transparent.

Simple Example

Set the Length and Progress parameters to the values of fields in your dataset to display the field values visually.

Paste into a TextBox BackgroundImage property

```
=RangeBarProgress(0,30000,"BlueViolet",0,Fields!GrossSales.Value,"Gold",Fields!GrossProfit.Value)
```

Store Name	Gross Sales	Gross Profit	Range Bar
Store #1000	$22,797.30	$12,602.57	
Store #1001	$18,889.32	$10,348.09	
Store #1002	$25,625.24	$13,939.84	
Store #1003	$22,386.34	$12,201.39	
Store #1004	$19,893.03	$11,383.74	
Store #1005	$19,775.52	$10,979.59	
Store #1006	$22,400.15	$12,709.01	

Example Using Negative Values

When your data contains negative as well as positive values, you can use an Immediate If expression for the Color parameter. In the example below, if the Difference value is negative, it is rendered in red, while positive values are rendered in gold. You can also see that negative values are rendered to the left of Zero and positive values are rendered to the right. A Length value of exactly zero is rendered as a diamond. The thicker blue violet bar represents the InStock value.

Paste into a TextBox BackgroundImage property

```
=RangeBarProgress(-10,20,"BlueViolet",0,Fields!InStock.Value,IIf(Fields!Difference.Value < 0,
"Red", "Gold"),Fields!Difference.Value)
```

Product ID	In Stock	Re Order Level	Difference	Range Bar
1000	5	7	-2	
1001	5	5	0	
1002	14	6	8	
1003	15	4	11	
1004	3	10	-7	
1005	5	8	-3	
1006	0	2	-2	
1007	6	7	-1	
1008	7	5	2	
1009	8	1	7	
1010	1	1	0	
1011	2	9	-7	

Default Behavior

The function returns **null** (i.e. no image is rendered) in any of the following cases:

1. The **Maximum** is less than or equal to the **Minimum**.
2. The expression is placed in a property which does not take an image.
3. The **Source** property of the image is not set to **Database**.

The Start value changes in the following cases:

1. If the **Start** value is less than the **Minimum** value, **Start** is the same as **Minimum**.
2. If the **Start** value is greater than the **Maximum** value, **Start** is the same as **Maximum**.

The Length value changes in the following cases:

1. If the **Start** value plus the **Length** value is less than the **Minimum** value, **Length** becomes **Minimum** minus **Start**.
2. If the **Start** value plus the **Length** value is greater than the **Maximum** value, **Length** becomes **Maximum** minus **Start**.

The Progress value changes in the following cases:

1. If the **Start** value plus the **Progress** value is less than the **Minimum** value, **Progress** becomes **Minimum** minus **Start**.
2. If the **Start** value plus the **Progress** value is greater than the **Maximum** value, **Progress** becomes **Maximum** minus **Start**.

If the argument for any of the parameters cannot be converted to the required data type, the default value is used instead.

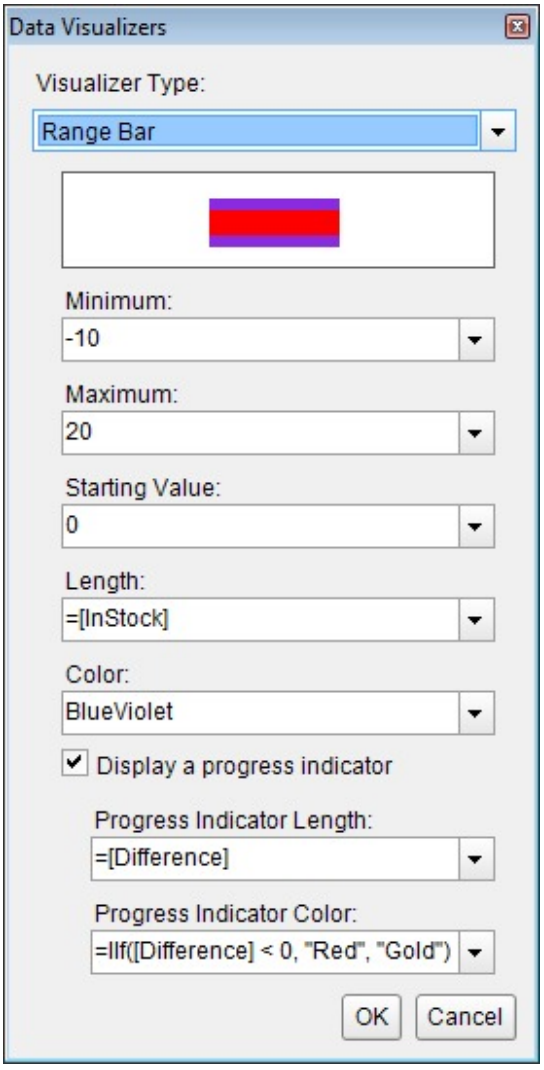
Parameter	Default Value
Minimum	0
Maximum	0
Color	Green

Start	0
Length	0
ProgressColor	Red
Progress	0

Dialog

When you select a TextBox, CheckBox, Shape, or a Container control on your report, in the Properties window or Properties dialog, you can drop down the **BackgroundImage Value** property and select **<Data Visualizer...>** to launch the dialog. To build the data visualizer expression, select the appropriate values for each of the options on the dialog.

For a Range Bar Progress expression, be sure to select the **Display a progress indicator** check box. This enables the progress options.



Data Bar

The Data Bar data visualization displays a 96 by 96 dpi bar image. The colored bar fills the Image top to bottom, while the Value argument determines the amount of colored bar to render to the right of the Zero argument (or to the left in

the case of a negative value).

The Minimum and Maximum arguments determine the range of data. The area between the Value argument and the Maximum argument is transparent (or between the Value and the Minimum in the case of a negative value).

The Data Bar Data Visualizer is supported in the Image report control **Value** property, and also in the TextBox, CheckBox, Shape, and Container report controls' **BackgroundImage - Value** property.

Parameters

- **Value**. This is the field value in the report to be evaluated. The data type is Single.
- **Minimum**. The lowest value in the range of data against which the Value argument is compared. This value corresponds to the leftmost edge of the image. If this argument is greater than the Zero argument, Zero becomes equal to Minimum. The data type is Single.
- **Maximum**. The highest value in the range of data against which the Value argument is compared. This value corresponds to the rightmost edge of the image. If this argument is less than the Zero argument, Zero becomes equal to Maximum. The data type is Single.
- **Zero**. This value determines the zero point to the left of which negative data is rendered, and to the right of which positive data is rendered. The data type is Single.
- **Color**. The HTML color string to use in rendering the Value in the bar image.
- **Alternate Color**. The HTML color string to use when the Value is less than the Zero value (optional).

Select the **Use Alternate Color when Value is less than Zero Value** check box to enable the Alternate Color parameter. You can use static values or aggregate functions (e.g. Min or Max) to set parameters. For more information on these and other aggregate functions, see the Expressions topic.

Syntax

```
=DataBar(Value, Minimum, Maximum, Zero, Color)
=DataBar(Value, Minimum, Maximum, Zero, Color, Alternate Color)
```

Usage

Use this data visualizer to render a bar in the color specified, the length of which changes depending on the number returned by the Value parameter, in the case of the simple example, InStock. If your data contains only positive values, Zero corresponds with Minimum at the left edge of the Data Bar. The area between the Value and the Maximum is transparent.

Simple Example

Set the Value parameter to the value of a field in your dataset to display the field values visually.

Paste into the BackgroundImage Value property of a TextBox
`=DataBar(Fields!InStock.Value,0,20,0,"BlueViolet")`

Product ID	In Stock	Data Bar
1000	5	
1001	5	
1002	14	
1003	15	
1004	3	
1005	5	
1006	0	
1007	6	
1008	7	
1009	8	
1010	1	
1011	2	
1012	4	

Example Using Negative Values

When your data contains negative as well as positive values, you can select the **Use Alternate Color when Value is less than Zero Value** check box, and then select an Alternate Color. In this example, if the Difference value is negative, it is rendered in Crimson, while positive values are rendered in BlueViolet. You can also see that negative values are rendered to the left of Zero and positive values are rendered to the right.

Paste in the BackgroundImage Value property of a TextBox

```
=DataBar(Fields!Difference.Value,-10,20,0,"BlueViolet","Crimson")
```

Product ID	In Stock	Re Order Level	Difference	Data Bar
1000	5	7	-2	
1001	5	5	0	
1002	14	6	8	
1003	15	4	11	
1004	3	10	-7	
1005	5	8	-3	
1006	0	2	-2	
1007	6	7	-1	
1008	7	5	2	
1009	8	1	7	
1010	1	1	0	
1011	2	9	-7	
1012	4	1	3	

Default Behavior

The function returns **null** (i.e. no image is rendered) in any of the following cases:

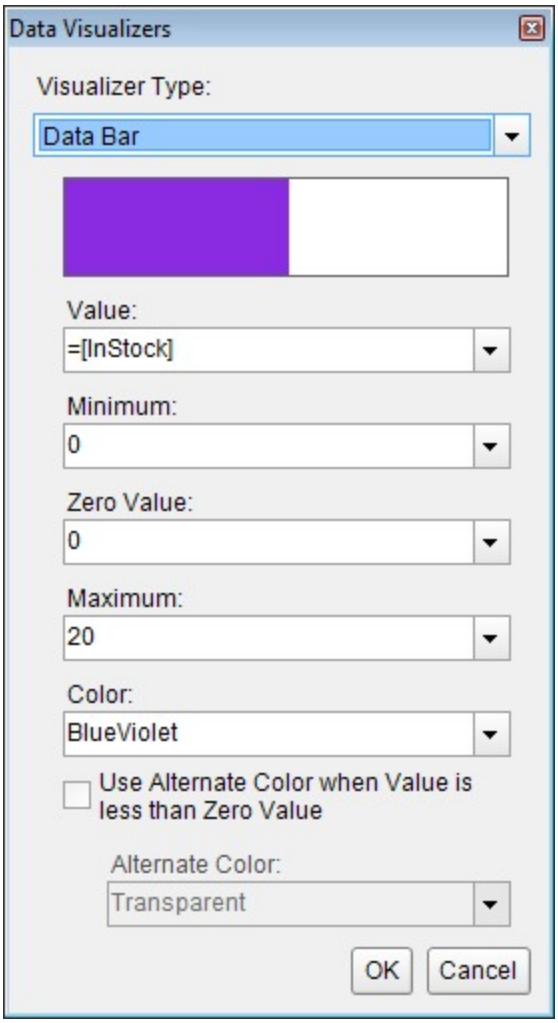
1. The **Maximum** is less than or equal to the **Minimum**.
2. The expression is placed in a property which does not take an image.
3. The **Source** property of the image is not set to **Database**.

If the argument for any of the parameters cannot be converted to the required data type, the default value is used instead.

Parameter	Default Value
Value	0
Minimum	0
Maximum	0
Zero	0
Color	Green
Alternate Color	*null*

Dialog

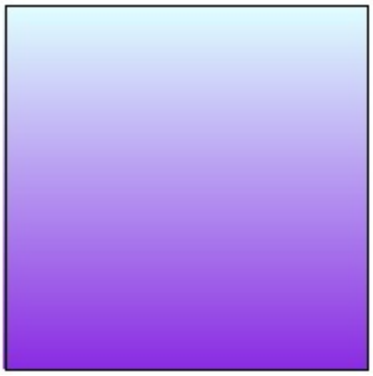
When you select a TextBox, CheckBox, Shape, or a Container control on your report, in the Properties window or Properties dialog, you can drop down the **BackGroundImage Value** property and select **<Data Visualizer...>** to launch the dialog. The same is true if you select an Image control and drop down the **Value** property. To build the data visualizer expression, select the appropriate values for each of the options in the dialog.



Gradient

The Gradient data visualization displays a gradient of color that transitions from one color to another.

When you select a Shape control on your report, in the Properties window, drop down the BackGroundImage - Value property and select <Data Visualizer...> to launch the dialog. The same is true if you select an Image control and drop down the Value property. To build the data visualizer expression, select the appropriate values for each of the options in the dialog.



Parameters

- **Gradient type**. The available gradient types are Horizontal, Vertical, DiagonalUp, DiagonalDown, FromCenter, and FromCorner.
- **Color1**. The start (primary) color of the gradient.
- **Color2**. The end (secondary) color of the gradient.

Syntax

```
=Gradient(Gradient type, Color1, Color2)
```

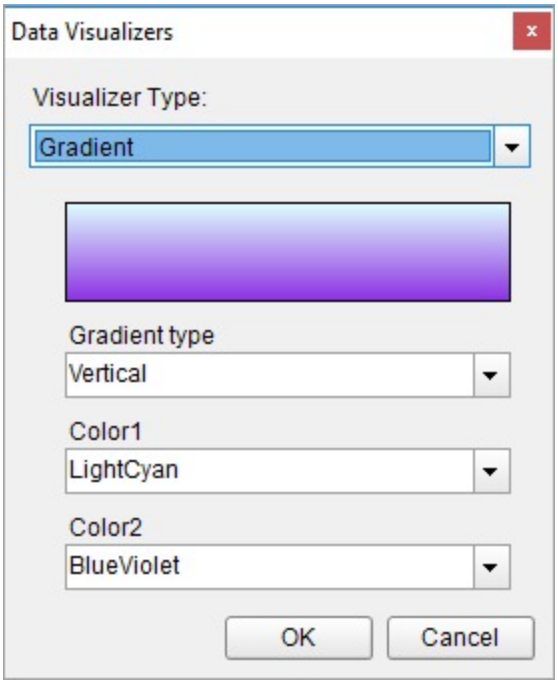
Usage

Using Data Visualizers, the gradient can be applied in TextBox, Shape, CheckBox, and Container controls from **BackgroundImage - Value** property and in Image control from **Value** property. The following example shows how Gradient background style can be simply applied to a control.

Example

Set the following expression in the BackgroundImage.Value property of the Shape .

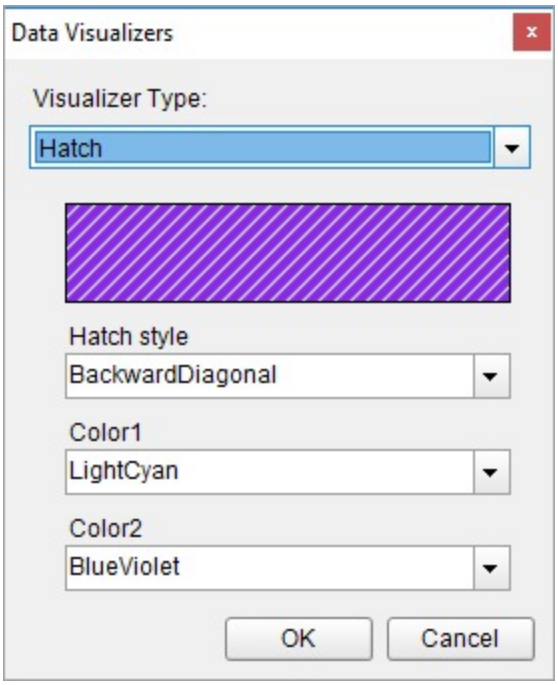
Paste into the BackgroundImage Value property of a Shape
`=Gradient("Vertical","LightCyan","BlueVoilet")`



Hatch

The Hatch data visualization displays the geometric hatch pattern.

When you select a Shape "images/design-reports/ on your report, in the Properties window, drop down the BackGroundImage - Value property and select <Data Visualizer...> to launch the dialog. The same is true if you select an Image control and drop down the Value property. To build the data visualizer expression, select the appropriate values for each of the options in the dialog.



Parameters

- **Hatch style**. The available hatch styles are Horizontal, Vertical, ForwardDiagonal, BackwardDiagonal, LargeGrid, and many more.
- **Color1**. The color of pattern or foreground color.
- **Color2**. The background color.

Syntax

```
=Hatch(Hatch style,Color1,Color2)
```

Usage

Using Data Visualizers, the hatch can be applied in TextBox, Shape, CheckBox, and Container control's from **BackgroundImage - Value** property and in Image control from **Value** property. The following example shows how Hatch background style can be simply applied to a control.

Example

Set the following expression in the Shape control's BackgroundImage - Value property.

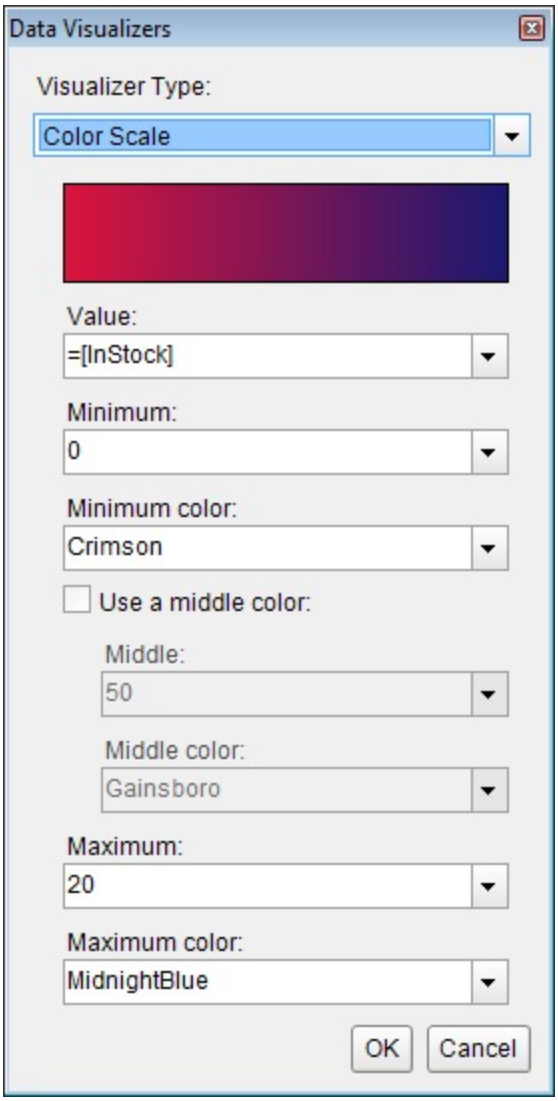
Paste into BackgroundImage property
`=Hatch("BackwardDiagonal","LightCyan","BlueViolet")`

Color Scale 2

The ColorScale2 data visualization displays a background color in a range of colors to indicate minimum and maximum values, and all shades in between. The Color Scale Data Visualizer is available for TextBox and CheckBox controls.

When you select a TextBox or a CheckBox control on your report, in the Properties window or Properties dialog, you can drop down the **BackGroundColor** property and select **<Data Visualizer...>** to launch the dialog. To build the data visualizer expression, select the appropriate values for each of the options in the dialog.

> **Note**: If you select the **Use a middle color** check box, the expression used in the BackgroundColor property changes to ColorScale3. For more information, see ColorScale3.



Parameters

- **Value**. This is the field value in the report to evaluate. The data type is Single.
- **Minimum**. If the Value evaluates to this number, the StartColor renders.
- **Maximum**. If the Value evaluates to this number, the EndColor renders.
- **StartColor**. The HTML color string to use if the Value evaluates to the Minimum value.
- **EndColor**. The HTML color string to use if the Value evaluates to the Maximum value.

You can use static values or aggregate functions (e.g. Min or Max) to set the Minimum and Maximum parameters. For more information on these and other aggregate functions, see the Expressions topic.

Syntax

```
=ColorScale2(Value, Minimum, Maximum, StartColor, EndColor)
```

Usage

Use an expression with this syntax in the **BackgroundColor** property of a Textbox or a CheckBox control. This causes

the background color to change depending on the value of the field you specified in the **Value** parameter, in the case of the example, InStock. Any values falling between the **Minimum** value and the **Maximum** render with a color between the **StartColor** and **EndColor**.

Example

Set the Value parameter to the value of a field in your dataset to display the field values visually.

Paste into the BackgroundColor property of a TextBox
`=ColorScale2(Fields!InStock.Value,0,20,"Crimson","MidnightBlue")`

Product ID	In Stock	Color Scale 2
1000	5	
1001	5	
1002	14	
1003	15	
1004	3	
1005	5	
1006	0	
1007	6	
1008	7	
1009	8	
1010	1	
1011	2	
1012	4	
1013	0	
1014	17	
1015	19	

Default Behavior

The function returns **Transparent** in any of the following cases:

1. The **Value** is out of range (i.e. does not fall between the **Minimum** and **Maximum** values).
2. The **Maximum** is less than the **Minimum**.

If the argument for any of the parameters cannot be converted to the required data type, the default value is used instead.

Parameter	Default Value
Value	0
Minimum	0

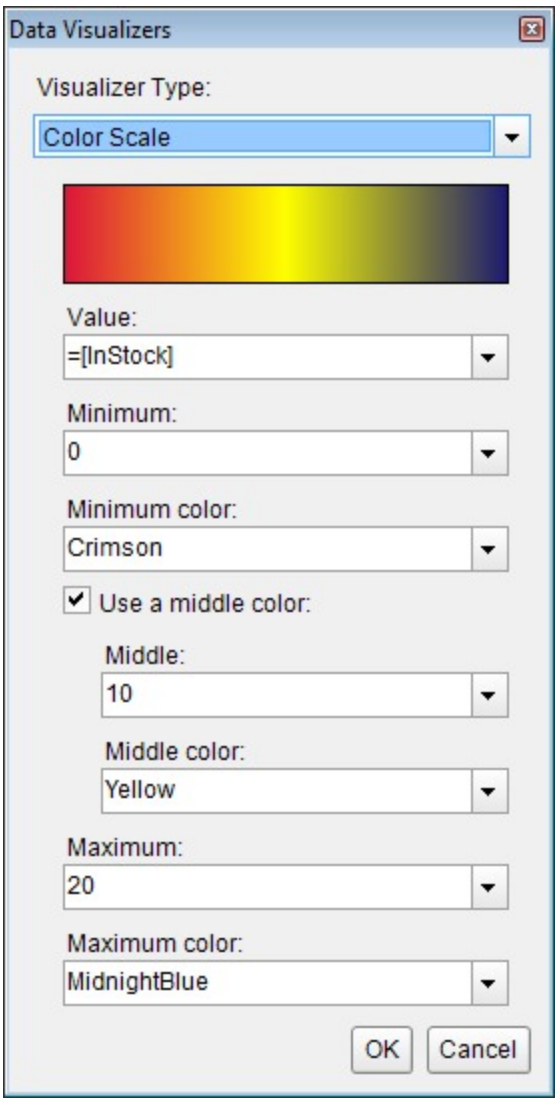
Maximum	0
StartColor	Silver
EndColor	WhiteSmoke

Color Scale 3

The ColorScale3 data visualization displays a background color in a range of colors to indicate minimum, middle, and maximum values, and all shades in between. The Color Scale Data Visualizer is available for TextBox and CheckBox controls.

When you select a TextBox or a CheckBox control on your report, in the Properties window or Properties dialog, you can drop down the **BackGroundColor** property and select **<Data Visualizer...>** to launch the dialog. To build the data visualizer expression, select the appropriate values for each of the options in the dialog.

> **Note**: If you clear the **Use a middle color** check box, the expression used in the BackgroundColor property changes to ColorScale2. For more information, see ColorScale2.



Parameters

- **Value**. This is the field value in the report to be evaluated. The data type is Single.
- **Minimum**. If the Value evaluates to this number, the StartColor is rendered.
- **Middle**. If the Value evaluates to this number, the MiddleColor is rendered.
- **Maximum**. If the Value evaluates to this number, the EndColor is rendered.
- **StartColor**. The HTML color string to use if the Value evaluates to the Minimum value.
- **MiddleColor**. The HTML color string to use if the Value evaluates to the Middlevalue.
- **EndColor**. The HTML color string to use if the Value evaluates to the Maximum value.

You can use static values or aggregate functions (e.g. Min, Avg, or Max) to set the Minimum, Middle, and Maximum parameters. For more information on these and other aggregate functions, see the Expressions topic.

Syntax

```
=ColorScale3(Value, Minimum, Middle, Maximum, StartColor, MiddleColor, EndColor)
```

Usage

Use an expression with this syntax in the **BackgroundColor** property of a Textbox or a CheckBox control. This causes the background color to change depending on the value of the field you specified in the **Value** parameter, in the case of the example, InStock. Any values falling between the **Minimum** value and the **Middle** value render with a gradient scale color between the **StartColor** and **MiddleColor**. The closer the value is to the **Minimum**, the closer to Crimson the color renders. In the same way, values falling between the **Middle** and **Maximum** render with a color between the **MiddleColor** and **EndColor**, in this case, varying shades of yellow-green.

Example

Set the Value parameter to the value of a field in your dataset to display the field values visually.

Paste into the BackgroundColor property of a TextBox
`=ColorScale3(Fields!InStock.Value,0,10,20,"Crimson","Yellow","MidnightBlue")`

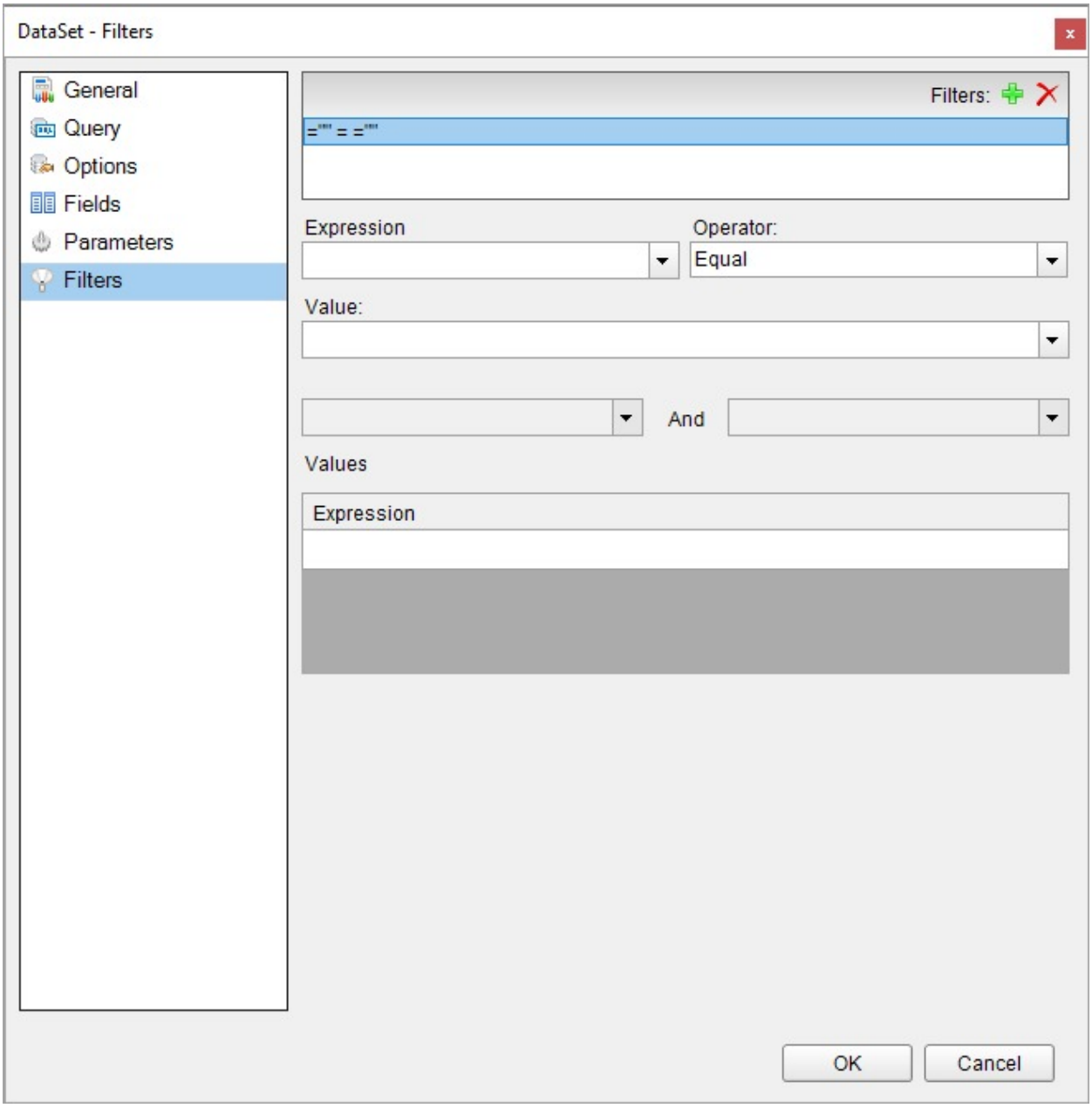
Product ID	In Stock	Color Scale 3
1000	5	
1001	5	
1002	14	
1003	15	
1004	3	
1005	5	
1006	0	
1007	6	
1008	7	
1009	8	
1010	1	
1011	2	
1012	4	
1013	0	
1014	17	
1015	19	
1016	11	

Filters

In Page and RDLX reports, ActiveReports allows you to set filters on a large set of data that has already been retrieved from the data source and use them with datasets or data regions to limit the information you want to display on your report.

Although not as efficient performance-wise as query parameters which filter data at the source, there are still scenarios which demand filters. The obvious case is when the data source does not support query parameters. Another case for using filters is when users who require different sets of data view in the same report.

You can set filters on a **Filters** page similar to the one in the following image.



A filter consists of the three major elements:

- **Expression:** Type or use the expression editor to provide the expression on which to filter data. See Expressions page on more information.
- **Operator:** Select the operator to compare the expression results with the Value.
- **Value:** Enter the value with which to compare the expression results.

For example, in the filter `=Fields!YearReleased.Value = 1997` applied on a dataset from the Movies table of the Reels.db database, `=Fields!YearReleased.Value` is set under expression, **=** is the operator and **1997** is the value on which filter is set.

You can also use multiple values with the **In** and **Between** operators. Two fields with an *And* in the middle appear for the Between operator, and another Expression field is available at the bottom of the Filters page or tab for the In operator. The following table lists all available filtering operators.

Filtering Operators

Filter	Description
Equal	Select this operator if you want to choose data for which the value on the left is equal to the value on the right.
Like	Select this operator if you want to choose data for which the value on the left is similar to the value on the right. See the MSDN Web site for more information on the Like operator.
NotEqual	Select this operator if you want to choose data for which the value on the left is not equal to the value on the right.
GreaterThan	Select this operator if you want to choose data for which the value on the left is greater than the value on the right.
GreaterThanOrEqual	Select this operator if you want to choose data for which the value on the left is greater than or equal to the value on the right.
LessThan	Select this operator if you want to choose data for which the value on the left is less than the value on the right.
LessThanOrEqual	Select this operator if you want to choose data for which the value on the left is less than or equal to the value on the right.
TopN	Select this operator if you want to choose items from the value on the left which are the top number specified in the value on the right.
BottomN	Select this operator if you want to choose items from the value on the left which are the bottom number specified in the value on the right.
TopPercent	Select this operator if you want to choose items from the value on the left which are the top percent specified in the value on the right.
BottomPercent	Select this operator if you want to choose items from the value on the left which are the bottom percent specified in the value on the right.
In	Select this operator if you want to choose items from the value on the left which are in the array of values on the right. This operator enables the Values list at the bottom of the Filters page.
Between	Select this operator if you want to choose items from the value on the left which fall between pair of values you specify on the right. This operator enables two Value boxes instead of one.

You can set filters on the following:

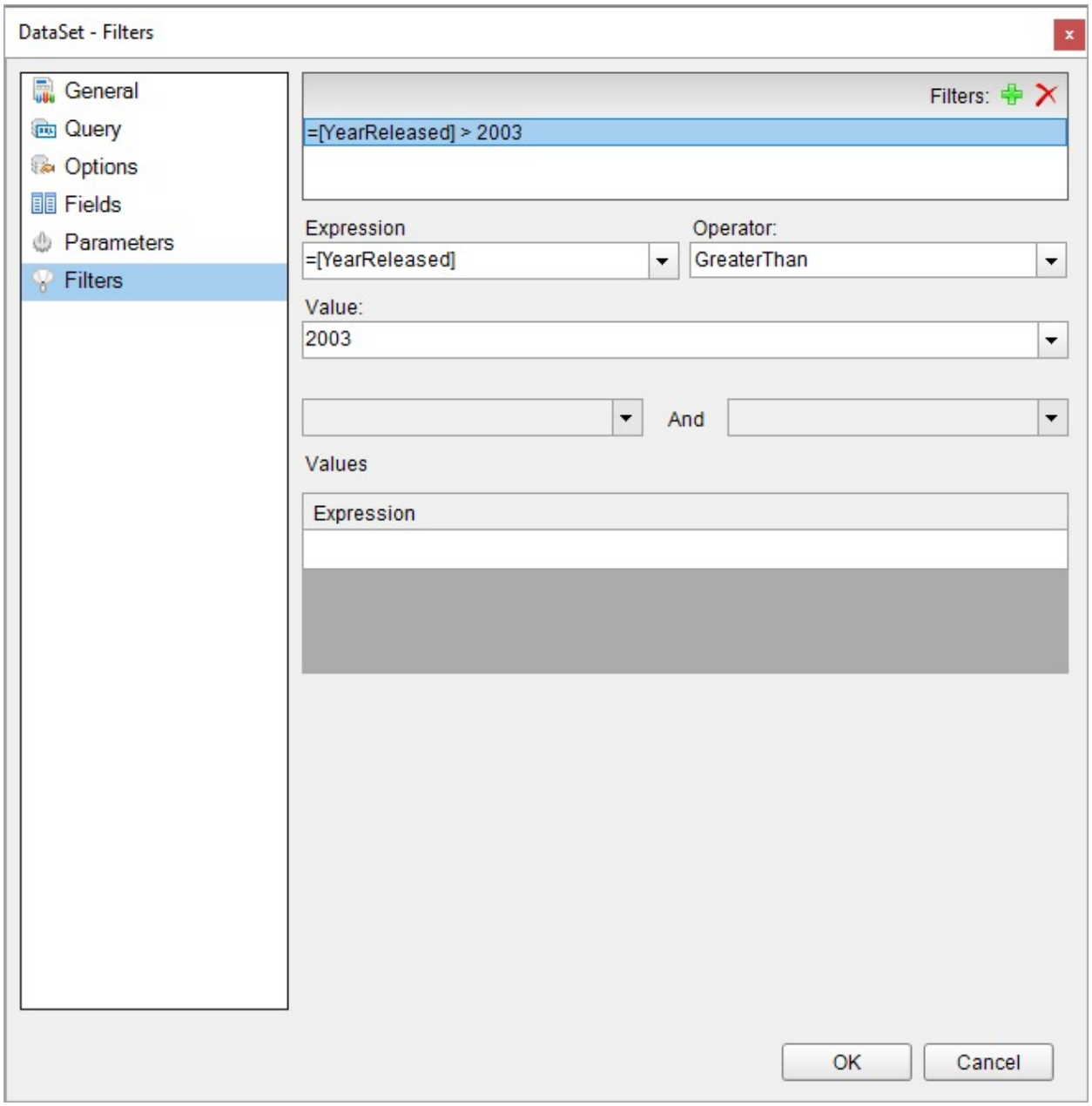
- DataSet
- Data Region
- Groups in a Data Region

Set a Filter on a DataSet

When you set a filter on a dataset, any control you add to the design surface can use this filtered data. Let us elaborate the dataset filtering using the 'Movie' table from Reels.db data source on GitHub.

1. In the Report Explorer, right-click the DataSet node and select **Edit**.
2. In the **DataSet** dialog that appears, go to the **Filters** page and click the Add (+) icon to add a new filter for the data set. By default, an empty filter expression gets added to the filter list.

3. Under **Expression**, enter an expression or use the Expression Editor to provide the expression on which to filter data. For example, `=Fields!YearReleased.Value`
4. Under **Operator**, select an operator from the list to decide how to compare the 'Expression' with the 'Value'. For example, set a **GreaterThan** operator on the expression above.
5. Under **Value**, enter a value or set an expression using the Expression Editor with which to compare the expression results. For example, **2003** to represent the year 2003. The resultant filter looks as follows.
   `=Fields!YearReleased.Value > 2003`



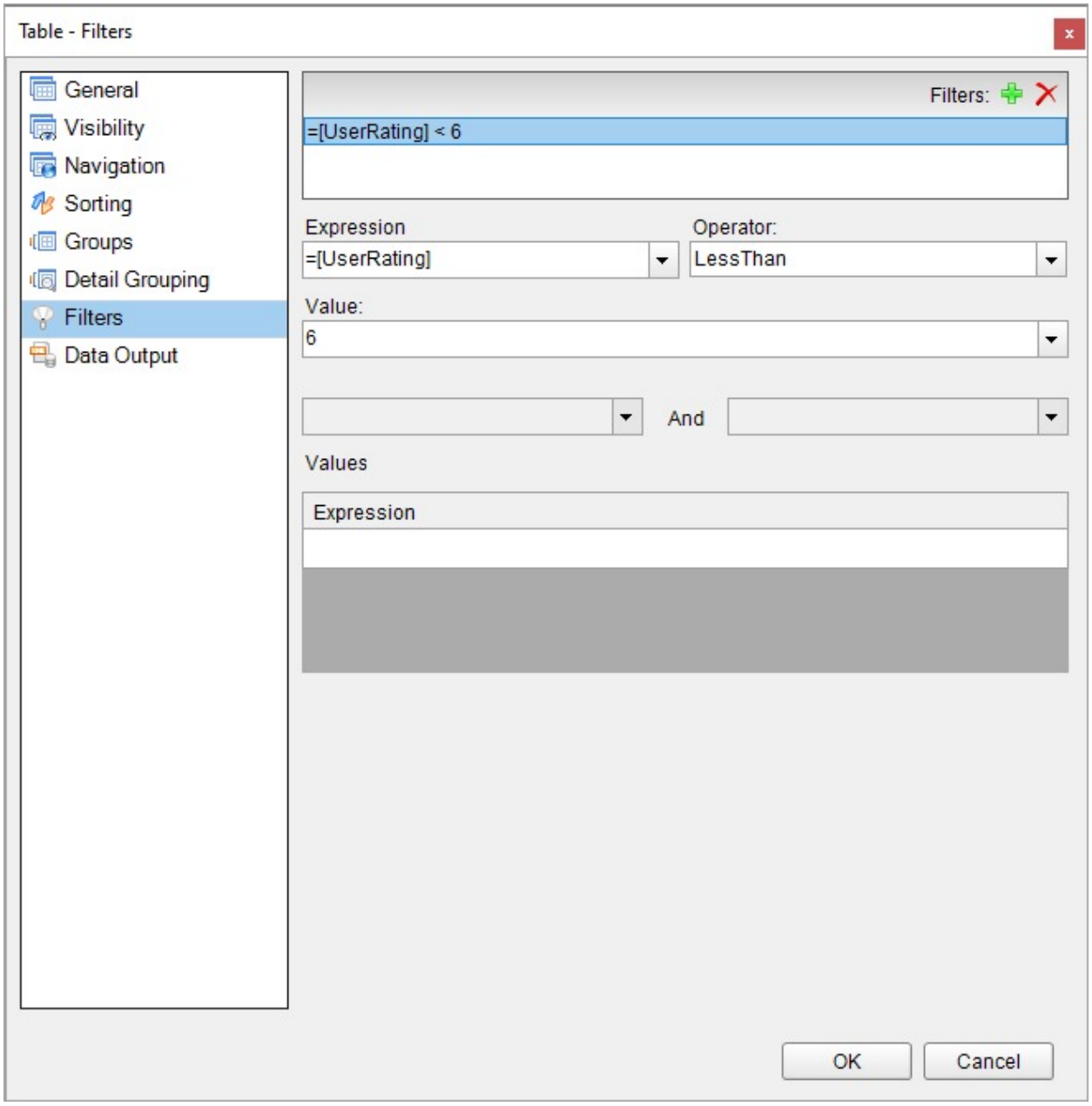
6. Now that the dataset filter is configured, to view the filtered data, let us add a **Table** data region and drag-drop fields such as 'Title', 'Year Released', and 'User Rating' on its Details row.
7. Preview the report. You see that the table displays the records only for the years where 'Year Released' is greater than '2003'.

Title	Year Released	User Rating
Shrek 2	2004	7,1
Star Wars: Episode III - Revenge of the Sith	2005	8,5
Spider-Man 2	2004	8,8
The Passion of the Christ	2004	8,8
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	2005	9
Harry Potter and the Goblet of Fire	2005	9,5
Meet the Fockers	2004	7,6
The Incredibles	2004	8,3
Harry Potter and the Prisoner of Azkaban	2004	7,3
War of the Worlds	2005	6
King Kong	2005	6,3

Set a Filter in a Data Region

When you set a filter in a data region, you can limit the amount of data available for use inside that data region. The following steps are applicable to **Table** and **List** data regions, and the data regions are bound to the 'Movie' table from Reels.db data source on data source on GitHub.

1. With the data region selected on the report, under the Properties window, click the **Property dialog** link to open the data region dialog.
2. In the data region dialog that appears, select the **Filters** page and click the **Add (+)** icon to add a new filter for the data region. By default, an empty filter expression gets added to the filter list.
3. Under **Expression**, enter an expression or use the Expression Editor to provide the expression on which to filter data. For example, `=Fields!UserRating.Value`
4. Under **Operator**, select an operator from the list to decide how to compare the Expression with the Value. For example, set a **LessThan** operator on the Expression above.
5. Under **Value**, enter a value or set an expression using the Expression Editor with which to compare the expression results. For example, **6** to represent the Rating 6. The resultant filter looks like the following.
   `=Fields!UserRating.Value < 6`

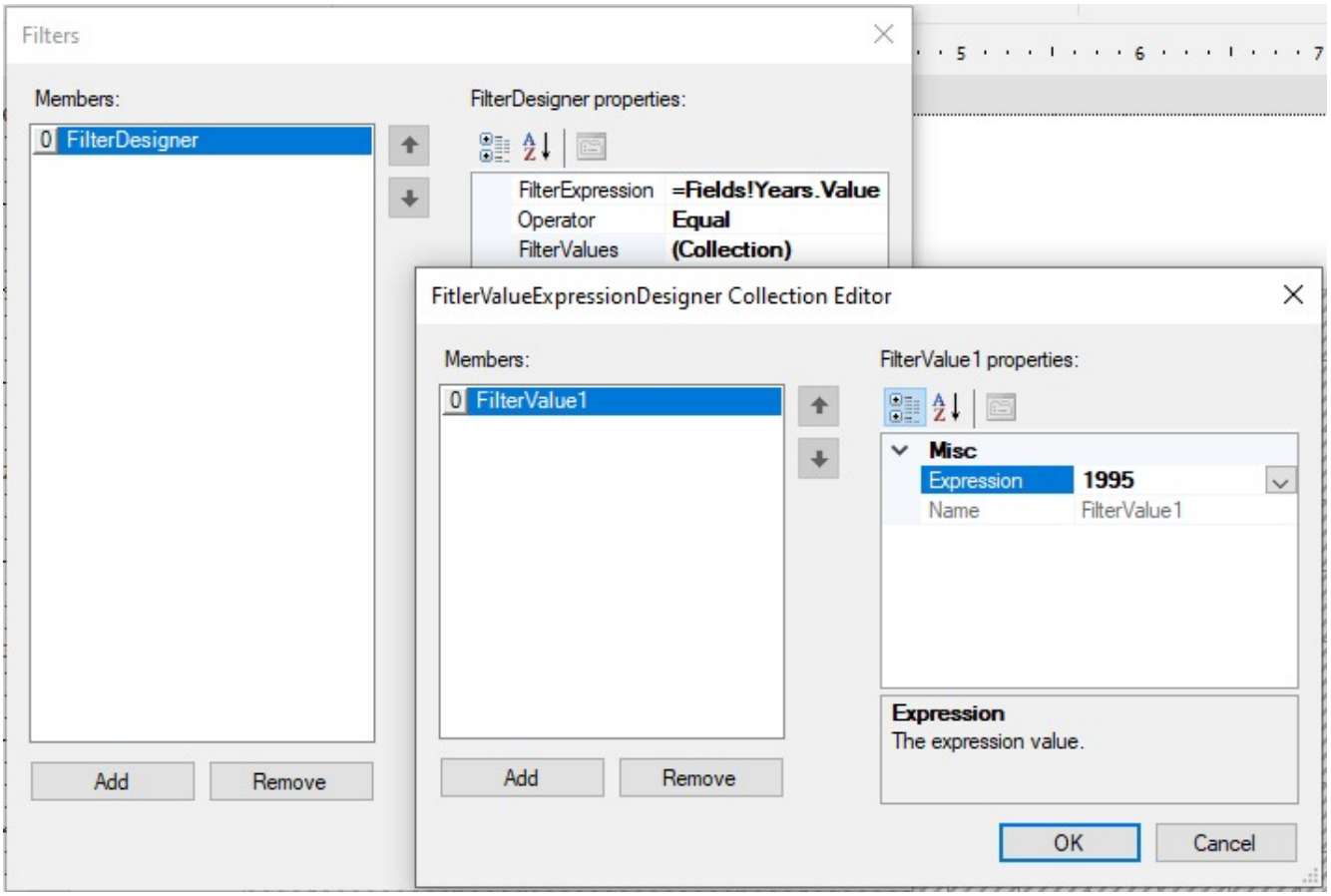


6. Preview the report to get the filtered data.

For **Tablix** data region, follow these steps to add a filter. Let us demonstrate applying a filter on the Tablix data region using the 'TablixSample.rdlx' report that you can find in Samples18 folder on GitHub. In this report, we will apply filter such that only the records for the year '1995' are shown.

1. With the **Tablix** data region selected, go to the **Filters** property in the Properties window.
2. Click the ellipses next to '(Collection)' to open the Filter dialog.
3. Click **Add** to add a filter, and in the FilterDesigner properties, enter the **FilterExpression** to `=Fields!Years.Value` and **Operator** to `Equal`.
4. In the **FilterValues** property, click the ellipses next to '(Collection)' to open the **FilterValueExpressionDesigner** Collection Editor.
5. Click **Add** to add the filter values and set the filter values properties.
6. In **Expression**, specify the value to which the filter expression should be evaluated, in our case, set **Expression** to

'1995'.



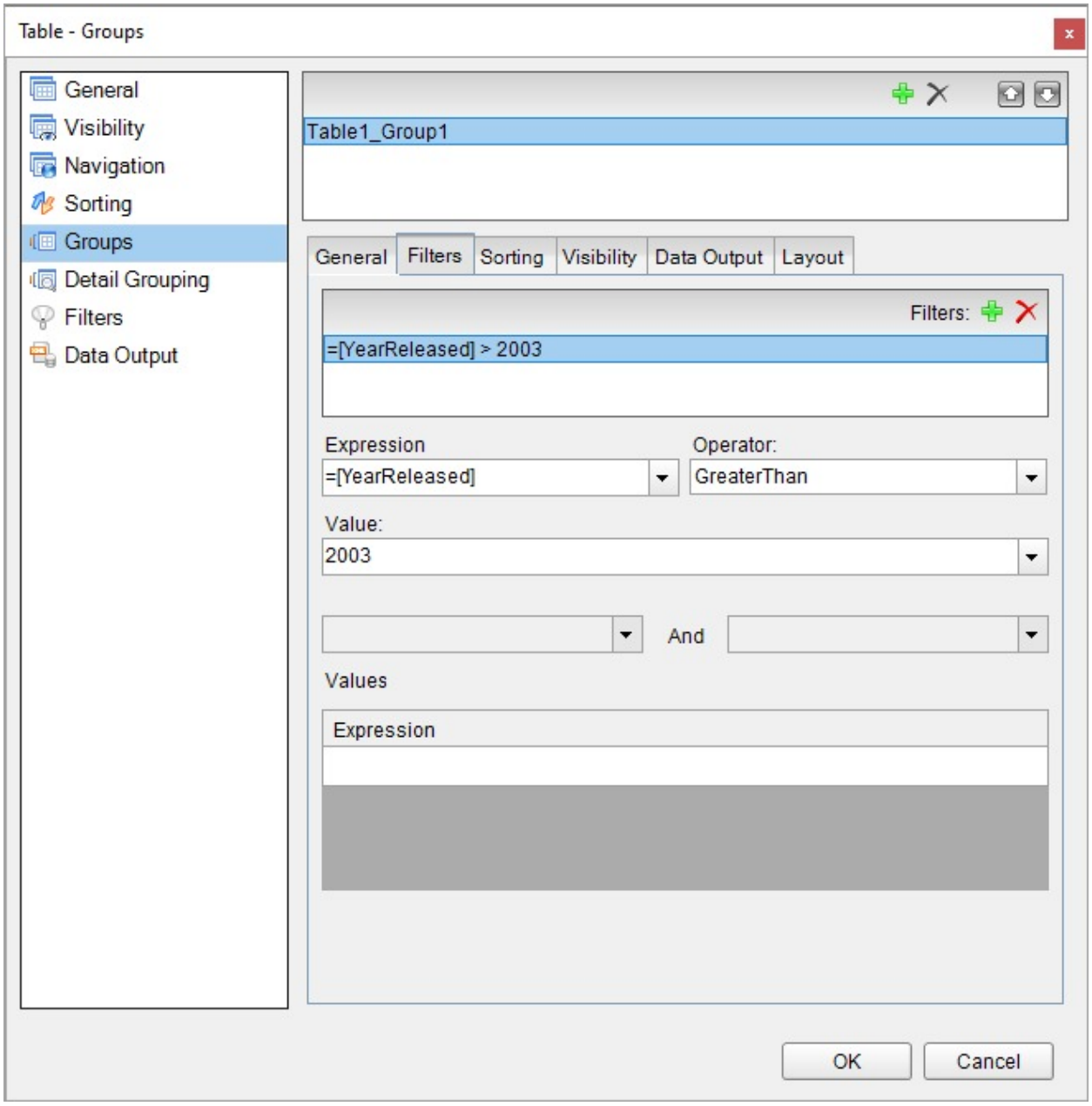
7. Click OK to complete setting up the filter value and then again to finish adding the filter.

Orders by Category, Quarter, and Year		1995					ProductTotal
		Q1	Q2	Q3	Q4	Total for the Year	
Meat/Poultry	Thüringer Rostbratwurst	$7,821.00	$6,238.44	$4,456.44	$12,131.42	$30,647.30	$30,647.30
	Tourtière	$926.30	$655.65	$625.80	$1,124.95	$3,332.70	$3,332.70
	Perth Pasties	$5,764.00	$1,422.40	$2,624.00	$4,362.40	$14,172.80	$14,172.80
	Pâté chinois	$6,240.00	$950.40	$1,920.00	$1,032.00	$10,142.40	$10,142.40
	Alice Mutton	$3,338.40	$2,847.00	$5,226.00	$4,095.00	$15,506.40	$15,506.40
	Mishi Kobe Niku	$1,552.00	$1,552.00		$5,432.00	$8,536.00	$8,536.00
	CategoryTotal	$25,641.70	$13,665.89	$14,852.24	$28,177.77	$82,337.60	$82,337.60
Grains/Cereals	Gnocchi di nonna Alice	$6,870.40	$8,177.60	$8,626.00	$8,436.00	$32,110.00	$32,110.00
	Wimmers gute Semmelknödel	$2,872.80	$1,010.80	$2,094.75	$2,227.75	$8,206.10	$8,206.10
	Filo Mix	$308.00	$42.00	$931.00	$812.00	$2,093.00	$2,093.00
	Tunnbröd	$720.00	$1,141.20	$90.00	$729.00	$2,680.20	$2,680.20
	Singaporean Hokkien Fried	$448.00	$1,484.00	$1,862.00	$1,736.00	$5,530.00	$5,530.00

Set a Filter on Groups in a Data Region

You can also set filters on grouped data in a data region. The following example uses the Table data region to show filtering on groups.

1. In the report, set grouping on a data region. For example, on a Table data region, set grouping on the `=Fields!YearReleased.Value` field.
2. With the data region selected on the report, under the Properties window, click the **Property dialog** link to open the data region dialog.
3. In the Table dialog, go to the **Groups** tab and select the Group.
4. After selecting the group, go to the **Filters** tab and click the Add (+) icon to add a new filter. By default, an empty filter expression gets added to the filter list.
5. Under **Expression**, enter an expression or use the Expression Editor to provide the expression on which to filter data. For example, `=Fields!YearReleased.Value`
6. Under **Operator**, select an operator from the list to decide how to compare the Expression with the Value. For example, **GreaterThan** operator set on the Expression above.
7. Under **Value**, enter a value or set an expression using the Expression Editor with which to compare the expression results. For example, **2005** to represent the year 2005. The resultant filter looks like the following.
   `=Fields!YearReleased.Value > 2005`

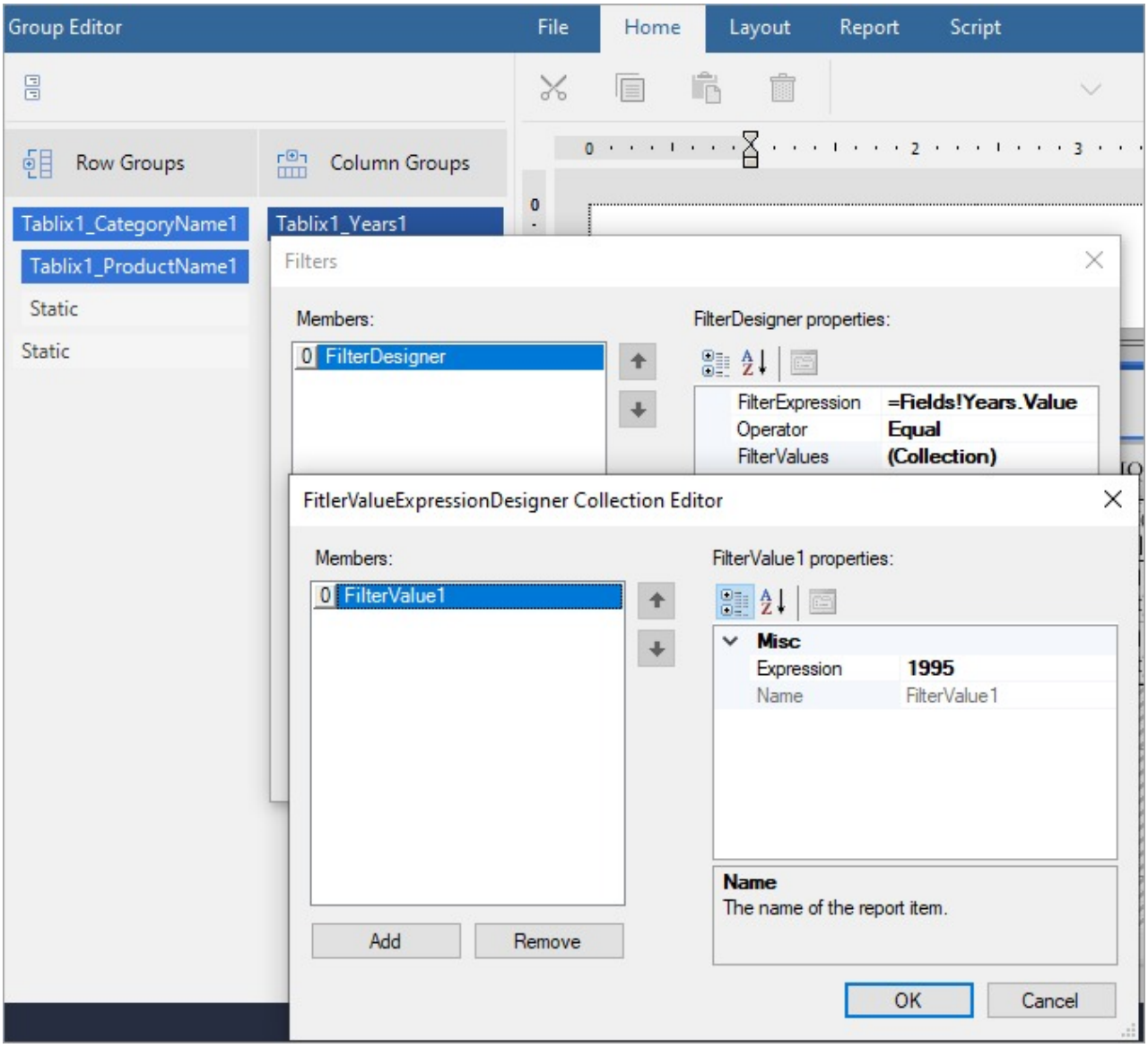


8. Preview the report to get the filtered data.

Title	Year Released	User Rating
2004		
Shrek 2	2004	7,1
Spider-Man 2	2004	8,8
The Passion of the Christ	2004	8,8
Meet the Fockers	2004	7,6
The Incredibles	2004	8,3
Harry Potter and the Prisoner of Azkaban	2004	7,3
The Day After Tomorrow	2004	7,5
The Bourne Supremacy	2004	8
The Polar Express	2004	9,9
National Treasure	2004	9
Shark Tale	2004	6
I, Robot	2004	6,5
Troy	2004	7,1
Ocean's Twelve	2004	5,8
50 First Dates	2004	8,6
Van Helsing	2004	5,9
Fahrenheit 9/11	2004	9,8
Lemony Snicket's A Series of Unfortunate Events	2004	9,3
Dodgeball: A True Underdog Story	2004	9,7
The Village	2004	5,1
The Grudge	2004	9,7
The Aviator	2004	9,4
Million Dollar Baby	2004	5,8
Collateral	2004	9,5
2005		
Star Wars: Episode III - Revenge of the Sith	2005	8,5
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	2005	9
Harry Potter and the Goblet of Fire	2005	9,5

For **Tablix** data region, follow these steps to add a filter. Let us demonstrate applying a filter on the Tablix data region using the 'TablixSample.rdlx' report that you can find in Samples18 folder on GitHub.

1. With the **Tablix** data region selected, go to the **Filters** property in the Properties window.
2. Click the ellipses next to '(Collection)' to open the Filter dialog.
3. Click **Add** to add a filter, and in the FilterDesigner properties, enter the **FilterExpression** to `=Fields!Years.Value` and **Operator** to `Equal`.
4. In the **FilterValues** property, click the ellipses next to '(Collection)' to open the **FilterValueExpressionDesigner** Collection Editor.
5. Click **Add** to add the filter values and set the filter values properties.
6. In **Expression**, specify the value to which the filter expression should be evaluated, in our case, set **Expression** to '`1995`'.



7. Click OK to complete setting up the filter value and then again to finish adding the filter.

Orders by Category, Quarter, and Year		1995					ProductTotal
		Q1	Q2	Q3	Q4	Total for the Year	
Meat/Poultry	Thüringer Rostbratwurst	$7,821.00	$6,238.44	$4,456.44	$12,131.42	$30,647.30	$30,647.30
	Tourtière	$926.30	$655.65	$625.80	$1,124.95	$3,332.70	$3,332.70
	Perth Pasties	$5,764.00	$1,422.40	$2,624.00	$4,362.40	$14,172.80	$14,172.80
	Pâté chinois	$6,240.00	$950.40	$1,920.00	$1,032.00	$10,142.40	$10,142.40
	Alice Mutton	$3,338.40	$2,847.00	$5,226.00	$4,095.00	$15,506.40	$15,506.40
	Mishi Kobe Niku	$1,552.00	$1,552.00		$5,432.00	$8,536.00	$8,536.00
	CategoryTotal	$25,641.70	$13,665.89	$14,852.24	$28,177.77	$82,337.60	$82,337.60
Grains/Cereals	Gnocchi di nonna Alice	$6,870.40	$8,177.60	$8,626.00	$8,436.00	$32,110.00	$32,110.00
	Wimmers gute Semmelknödel	$2,872.80	$1,010.80	$2,094.75	$2,227.75	$8,206.10	$8,206.10
	Filo Mix	$308.00	$42.00	$931.00	$812.00	$2,093.00	$2,093.00
	Tunnbröd	$720.00	$1,141.20	$90.00	$729.00	$2,680.20	$2,680.20
	Singaporean Hokkien Fried	$448.00	$1,484.00	$1,862.00	$1,736.00	$5,530.00	$5,530.00

Group Data

In a Page report and RDLX report, you can set grouping to organize data in your reports. The most common grouping scenario is to create groups by fields or expressions in a data region.
Depending on the data region you select, you can group data in one of the following ways:

- In a Table or BandedList, you can add group header and footer rows. You can also set detail grouping in the Table data region.
- In a List, you can set detail grouping.
- In a Tablix, you can add columns and rows either dynamically or manually to group data.
- In a Classic Chart , you can group data by categories or series.

Group in a FixedPage

In a Page report, you can group your data on the fixed page. A group set on the fixed page, applies to the entire report including the controls within the report. Therefore, once you set grouping here, you may decide not to group individual data regions. See Page Report topic for more information on properties available for grouping data.

Use the following steps to understand how to apply grouping on a fixed page. These steps assume that you have already added a Page report template, connected it to a data source and created a dataset.

1. Right-click the gray area outside the design surface and select **Fixed Layout Settings** or with the fixed page selected, under the Properties Panel, click the **Property dialog** link to open the FixedPage dialog.

2. On the **Grouping** page, in the General tab, under **Group on** enter the field name or expression on which you want to group the data. For example, `=Fields!YearReleased.Value`.

   > **Note:** A default group name like FixedPage1_Group appears under **Name**, once you set the group. To modify the group name, add a field name or expression under **Group on** to enable the Name option and enter a new name.

3. Under the **Document map label** field, you can optionally set a label to add the item to the document map.
4. Click **OK** to close the dialog.
5. Drag and drop a data region, for example, a Table data region, onto the design surface and set data inside its cells.
6. Go to the **Preview Tab** to view the result. You'll notice that the data appears in groups sorted according the **YearReleased** field on each report page.

Detail Grouping

Detail grouping is available in the List and Table data regions. It is useful when you do not want to repeat values within the details. When you set detail grouping, the value repeats for each distinct result of the grouping expression instead of each row of data.

For example, in Table topic, if you do not set detail grouping, you see each year as many times as there are movies from that year.

Title	MPAA	User Rating	Year Released
Snow White and the Seven Dwarfs	Approved	8.7	1937
Gone with the Wind	Approved	9.3	1939
Bambi	Approved	8.2	1942
One Hundred and One Dalmatians	G	5.3	1961
Mary Poppins	G	8.9	1964
The Sound of Music	G	7.4	1965
Doctor Zhivago	PG-13	9.8	1965
The Jungle Book	G	5.1	1967
The Graduate	Approved	6.8	1967
Butch Cassidy and the Sundance Kid	M	6.4	1969
Love Story	PG	5.8	1970
Airport	G	9.2	1970

If you set detail grouping to `=Fields!YearReleased.Value,` each year appears only once.

Title	MPAA	User Rating	Year Released
Snow White and the Seven Dwarfs	Approved	8.7	1937
Gone with the Wind	Approved	9.3	1939
Bambi	Approved	8.2	1942
One Hundred and One Dalmatians	G	5.3	1961
Mary Poppins	G	8.9	1964
The Sound of Music	G	7.4	1965
The Jungle Book	G	5.1	1967
Butch Cassidy and the Sundance Kid	M	6.4	1969
Love Story	PG	5.8	1970
The Godfather	R	5.3	1972
The Exorcist	R	6.7	1973
Blazing Saddles	R	5.5	1974

Recursive Hierarchies

If you want to display parent-child relationships in your data, you can create a recursive hierarchy. To do this, you need a unique ID field for the child group and an ID field for the parent group.

For example, if you have pulled data from the Reels database using the following SQL query:

SQL Query

```
SELECT EmployeePosition.*, Employee.*, Employee_1.PositionID AS ManagerPosition FROM Employee AS Employee_1 RIGHT JOIN (EmployeePosition INNER JOIN Employee ON EmployeePosition.PositionID = Employee.PositionID) ON Employee_1.EmployeeID = Employee.ManagementID;
```

You can set Detail Grouping in a Table data region using the `=Fields.Item("EmployeePosition.PositionID").Value` field, and the `=Fields!ManagerPosition.Value` field as the parent group to display parent-child relationships in your data. To view the hierarchy level, add another column in the table and add expression `=Level()`.

Recursive Hierarchy

Title	Management Role	Salary	Department ID	Hierarchy Level
President	Senior Management	32000	1	0
VP Country Manager	Senior Management	17000	1	1
VP Information Systems	Senior Management	8000	2	1
HQ Information Systems	Middle Management	3000	2	2
VP Human Resources	Senior Management	9000	4	1
HQ Human Resources	Middle Management	3000	4	2
VP Finance	Senior Management	17000	5	1
HQ Finance and Accounting	Middle Management	3000	5	2
Store Manager	Store Management	6000	6	2
Store Assistant Manager	Store Management	6000	6	3
Store Shift Supervisor	Store Management	4000	6	4
Store Cashier	Store Full Time Staff	4000	8	5
Store Stocker	Store Full Time Staff	2000	9	5
Store Information Systems	Store Full Time Staff	3000	7	4
HQ Marketing	Middle Management	2000	3	1

Note: You can use only one group expression when you set a parent group.

Level Function

To better visualize data in a recursive hierarchy, you can use the Level function, which indents text and further clarifies the relationships between parent and child data. To do this, you set an expression in the **Padding - Left** property of the

text box you want to indent.

For example, in a Table data region, for the recursive hierarchy example above, you can set the following expression in the Padding - Left property of the text box that contains the 'Title' to indent values according to levels:

```
=Convert.ToString(2 + (Level()*10)) & "pt"
```

Recursive Hierarchy

Title	Management Role	Salary	Department ID
President	Senior Management	32000	1
VP Country Manager	Senior Management	17000	1
VP Information Systems	Senior Management	8000	2
HQ Information Systems	Middle Management	3000	2
VP Human Resources	Senior Management	9000	4
HQ Human Resources	Middle Management	3000	4
VP Finance	Senior Management	17000	5
HQ Finance and Accounting	Middle Management	3000	5
Store Manager	Store Management	6000	6
Store Assistant Manager	Store Management	6000	6
Store Shift Supervisor	Store Management	4000	6
Store Cashier	Store Full Time Staff	4000	8
Store Stocker	Store Full Time Staff	2000	9
Store Information Systems	Store Full Time Staff	3000	7
HQ Marketing	Middle Management	2000	3

Interactivity

ActiveReports supports a number of features like parameters, filters, drill-down, links, document map and sorting to provide reports with interactive capabilities.

ActiveReports supports features like parameters, filters, drill-down, links, document map and sorting to provide an

interactive look to your report at run time.

Parameters

ActiveReports allows you to set parameters in your report to filter the data displayed. Parameters make navigation of the report easier for the user at run time. See Parameters for further details.

Drill-Down Links

When you open a report with drill-down features, part of the data is hidden so that you only see high-level data until you request more detail. See Drill-Down Links for more information.

Bookmark, Hyperlinks, and Drill-Through Links

Bookmark Links

When you click a bookmark link, the viewer navigates to a bookmarked item within the current report.

See Bookmarks for further details.

Hyperlinks

When you click a hyperlink, your machine's default internet browser opens to display a Web page.

See Hyperlinks for further details.

Drill-Through

Using the drill-through link feature in your report you can navigate to another report for details about the item you clicked.

See Drill-Through Links for further details.

Sorting

The sorting feature allows you to organize your data and present it in a logical order at run-time. Using this feature you can sort the data alphabetically or numerically in ascending or descending order. See Sort Data for further details.

> **Note:** You cannot use the interactive features in the following cases:
>
> - With Adobe Acrobat Reader and RawHTML types of the WebViewer.
> - With reports that use the collation feature, i.e. reports that have two or more themes.

Document Map

The Document Map (Table of Contents) feature allows you to navigate to a particular item in a report. See Document Map for further details.

Parameters

ActiveReports allows you to use parameters to filter or add the data to display in reports. You can either prompt users for parameters so that they control the output, or supply the parameters behind the scenes.

In a Page or an RDLX report, the easiest way to build queries with parameters is to use the **Visual Query Designer**, as it

automatically sets up each parameter.

In the DataSet dialog, click on  to access Visual Query Designer to create SQL queries. See Query Builder in Microsoft SQL Client and OLEDB Providers for further information on creating a parameterized query using the interactive query designer.

However, if you would like to create parameterized query manually, you must enter each parameter in three locations:

- The Report Parameters dialog,
- The Parameters page of the DataSet Dialog, and
- The Query page of the DataSet Dialog.

Report - Parameters Dialog

The **Report - Parameters** dialog allows you to control how and whether a user interface is presented to your users for each parameter. Parameter values are collected in the order they appear in the **Report Parameters** collection. You can change the order using the arrows in the Report - Parameters dialog.

The Report - Parameters dialog contains a parameters page with a list of parameters and three tabs to set parameter properties. You need to set the following properties in the dialog to create a parameter:

1. In the General tab, enter a report parameter name. Each report parameter in the collection must have a unique name, and the name must match the name you enter or call in the Parameters page of the DataSet dialog.
2. Set the data type, the text used to prompt the user, whether to allow null, blank, multiple values or multiline text, and whether to hide the user interface. For parameters with Date or DateTime data types, you have the choice to select the display format.
3. Go to the Available tab to populate a list of available values from which users can choose, or the Default tab to set the default value.

The tabs in the Report - Parameters Dialog are explained below.

General

- **Name**: Set the name for the parameter in this field. The value you supply here appears in the parameters list and must match the corresponding query parameter.
- **Data type**: Set the data type for your parameter which must match the data type of the field that it filters. The interface presented might also differ depending on the data type.
  - **Boolean**: Presents the user with two options True or False
  - **Date**: Presents the user with a calendar picker to select a date if you do not supply a default value or a drop-down selection of available values
  - **DateTime**: Presents the user with a calendar picker to select a date and a time picker to select the time in cases where you do not supply a default value or a drop-down selection of available values
  - **Integer**: Presents the user with a text box or a drop-down selection of available values
  - **Float**: Presents the user with a text box or a drop-down selection of available values
  - **String**: Presents the user with a text box or a drop-down selection of available values
- **Text for prompting users for a value**: Enter the text you want to see on the user interface to request information from the user in this field. By default this is the same as the Name property.
- **Allow null value**: Select this check box if you want to allow null values to be passed for the parameter. It is not selected by default.
- **Allow blank value**: Select this check box if you want to allow blank values to be passed for the parameter. It is not selected by default.
- **Multivalue**: Select this check box to allow the user to select multiple items in the available values list. For a multi-value parameter, you can also allow user to specify special value for 'Select all' option in the **Value for 'Select All'** property.
- **Multiline**: Select this check box to allow multiline values in the parameter. The control will automatically adjust to accommodate multiple lines.
- **Hidden**: Select this check box to hide the parameter interface from the user and instead provide a default value or pass in values from a subreport or drill-through link. Please note that if you hide the user interface and do not provide a default value, the report will not run.
- **Display format**: Choose a format for a Date or DateTime data type. The selected format is displayed in the preview in the Parameters pane. The Display format applies to parameters with available values specified by a query and the default parameter value.

Available Values

These values are used to fill a drop-down list from which the end user can choose.

- **Non-queried**: You can supply Labels and Values by typing in static values or using expressions.
- **From query**: You can select a Dataset from which to select a Value field and Label field.

Default Values

This is the value that you give for the parameter if the user does not supply one, or if you hide the parameter user interface. You can choose from 'Non-queried' and 'From query' options.

Note: In the Available Values tab the **Value** is what is passed to the query parameter, and the **Label** is what is shown to the user. For example, if the Value is an Employee Number, you might want to supply a more user-friendly Label showing Employee Names.

To access the Report - Parameters Dialog

You can access the Report - Parameters dialog through any one of the following:

- In the Report Explorer, click the Add (+) icon and select the **Parameter** option.
- In the Report Explorer, right-click the Parameters node and select **Add Parameter**.
- In the Report Explorer, right-click the Report node and select **Report Parameters**.
- From the Report Menu, select **Report Parameters**.

Parameters Page of the DataSet Dialog

On the Parameters page of the DataSet Dialog, pass a Report Parameter into the parameter in your query. You can click the Add (+) icon at the top of the parameters list, enter parameter name, and supply a value like:

```
=Parameters!MPAA.Value
```

Query Page of the DataSet Dialog

On the Query page of the DataSet Dialog, enter the parameter in the SQL query. Use the syntax specific to your data source type to create a parameter. For example, with an OleDB data source, add a query like the following for a multi-value Movie Rating parameter:

```
SELECT * FROM Movie WHERE (MPAA = ? AND YearReleased = ?)
```

If you want to run a report without prompting the user for a value at run time, you need to set a default value for each parameter, and the **Hidden** check box should be selected in the Report - Parameters dialog > General tab.

Subreport parameters are also considered hidden parameters, as a user can easily synchronize a subreport's data with that of the parent report. See Subreport for further details.

Drill-Through parameters are also hidden parameters, as drill-through links are used to navigate from one report to another. When you select **Jump to report** for the action, the parameters list is enabled.

Parameter in SQL query for Data Sources

A query parameter can get its value from the Report Parameters collection (entered by the user or from a value you supply), a field in another dataset, or an expression. Syntax for adding a parameter in your query might differ depending upon the data source that you are using. Use the syntax specific to your data source type to create a parameter.

Parameterized query for different data sources are as follows:

Data Source	Parameter Syntax	Example
OleDB	(?)	`SELECT * FROM Customer WHERE (CustomerID = ? AND AccountNumber = ?)`
ODBC	(?)	`SELECT * FROM Customer WHERE (CustomerID = ? AND AccountNumber = ?)`
SQL Client	@ParameterName	`SELECT * FROM Customer WHERE (CustomerID = @CustomerID AND AccountNumber = @AccountNumber)`

OracleDB	:ParameterName	`SELECT * FROM Customer WHERE CustomerID = :CustomerID AND AccountNumber = :AccountNumber`

Multi-Value Parameter

A multi-value parameter allows you to input one or more than one parameter values in a report. You can choose a few options from the list or simply choose 'Select all' to select all options.

You need to check the **Multivalue** option to make a parameter multi-value, and optionally set **Value for 'Select All'**. If there are large number of options to choose from, choosing 'Select all' option creates an SQL query too long for an SQL Command to run. In such a case, you can specify a value to the multi-value parameter in **Value for 'Select All'** option.

The following procedure takes you through a step by step process of how to create a multi-value parameter and specify a value for selecting all options from the list. The report binds to the 'Products' table from 'NWIND.db' data source available on GitHub. It is a SQLite Provider, a custom data provider that works if System.Data.SQLite package is added and referenced in the ActiveReports.config file. See setting up the dependencies and configuration file as described in the topic Custom Data Providers.

Create a Report

In the ActiveReports Designer, create a new Page or an RDLX report.

Bind Report to Data

As you create a new report, **Report Datasource dialog** appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the **Report Explorer** and then selecting the **Add Data Source** option.

Connect to a Data Source

1. In the Report Data Source dialog, select the **General** page and enter the name of the data source.
2. Under Type, select 'SQLite Provider'.
3. In the **Connection String**, enter the path of the .db, here, 'NWIND.db', for example `Data Source=C:\Data\NWIND.db`
4. Click OK to close the dialog and complete the data source connection.

Create Dataset to Populate Parameter Values

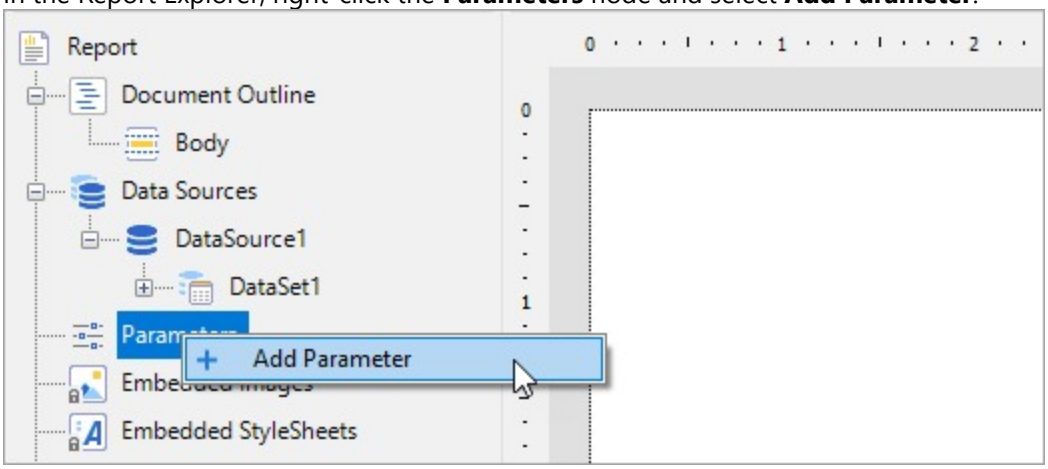
1. In the Report Explorer, right-click the Data Source (DataSource1 by default) node and select **Add Data Set**. See Add Dataset for more information.
2. In the DataSet dialog that appears, select the **Query** page.
3. Enter an SQL query like the following into the **Query** text box:

   ```
   SELECT DISTINCT productName FROM Products
   ```

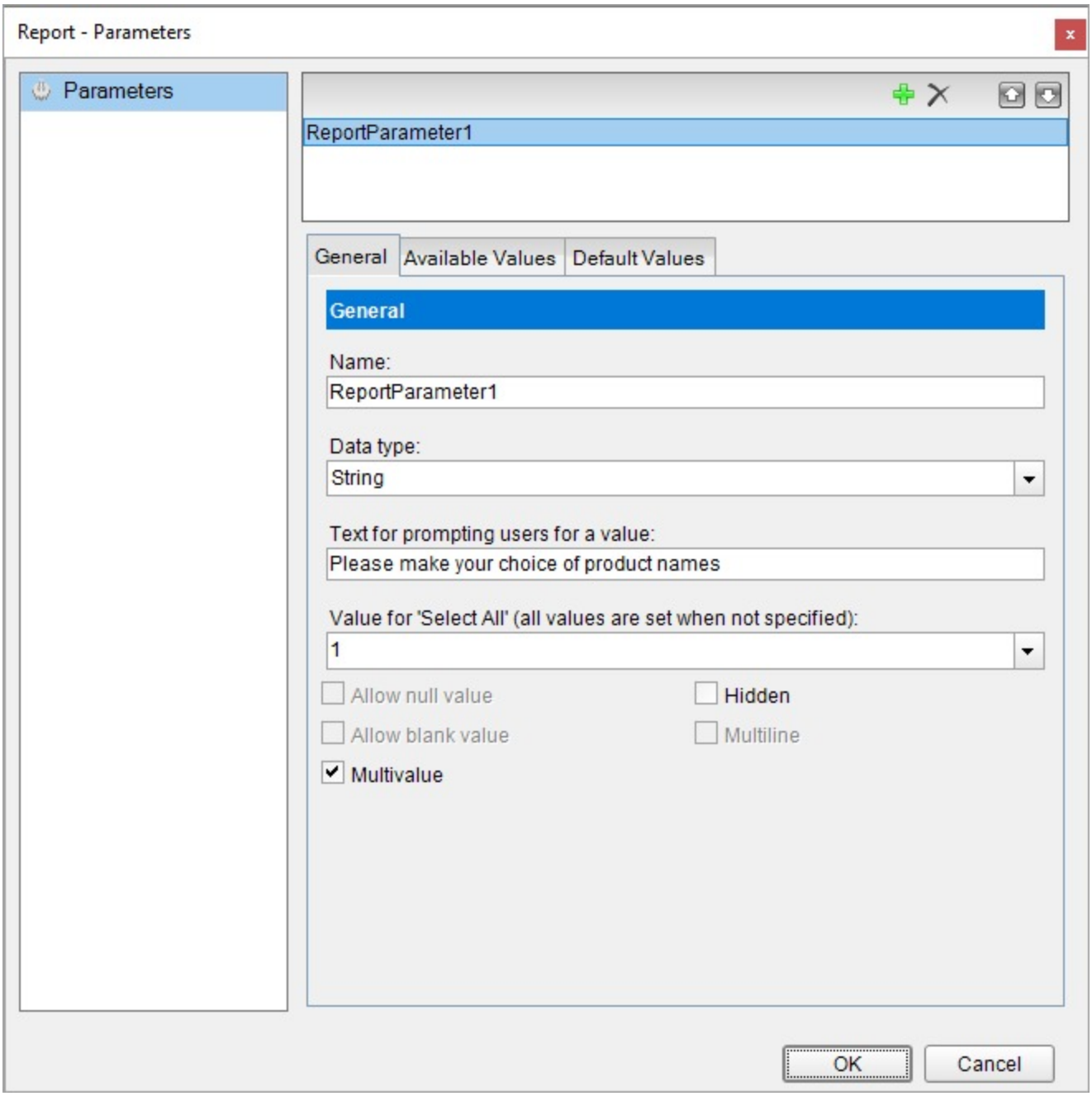
   See Query Builder in Microsoft SQL Client and OLEDB Providers for more information on building SQL queries.
4. Click the **OK** to close the dialog. You see the data set, **DataSet1**, and the field, **productName**, in the Report Explorer.

Add Report Parameter

1. In the Report Explorer, right-click the **Parameters** node and select **Add Parameter**.



2. In the **Report - Parameters** dialog that appears, add a name for the parameter, **ReportParameter1**.
3. Ensure that the **Data type** matches that of the field (String for ProductName).
4. Enter some text in the **Text for prompting users for a value** field.
5. Select the check box next to **Multivalue** to allow users to select more than one item from the list.
6. In the **Value for 'Select All'** option, enter '1'.



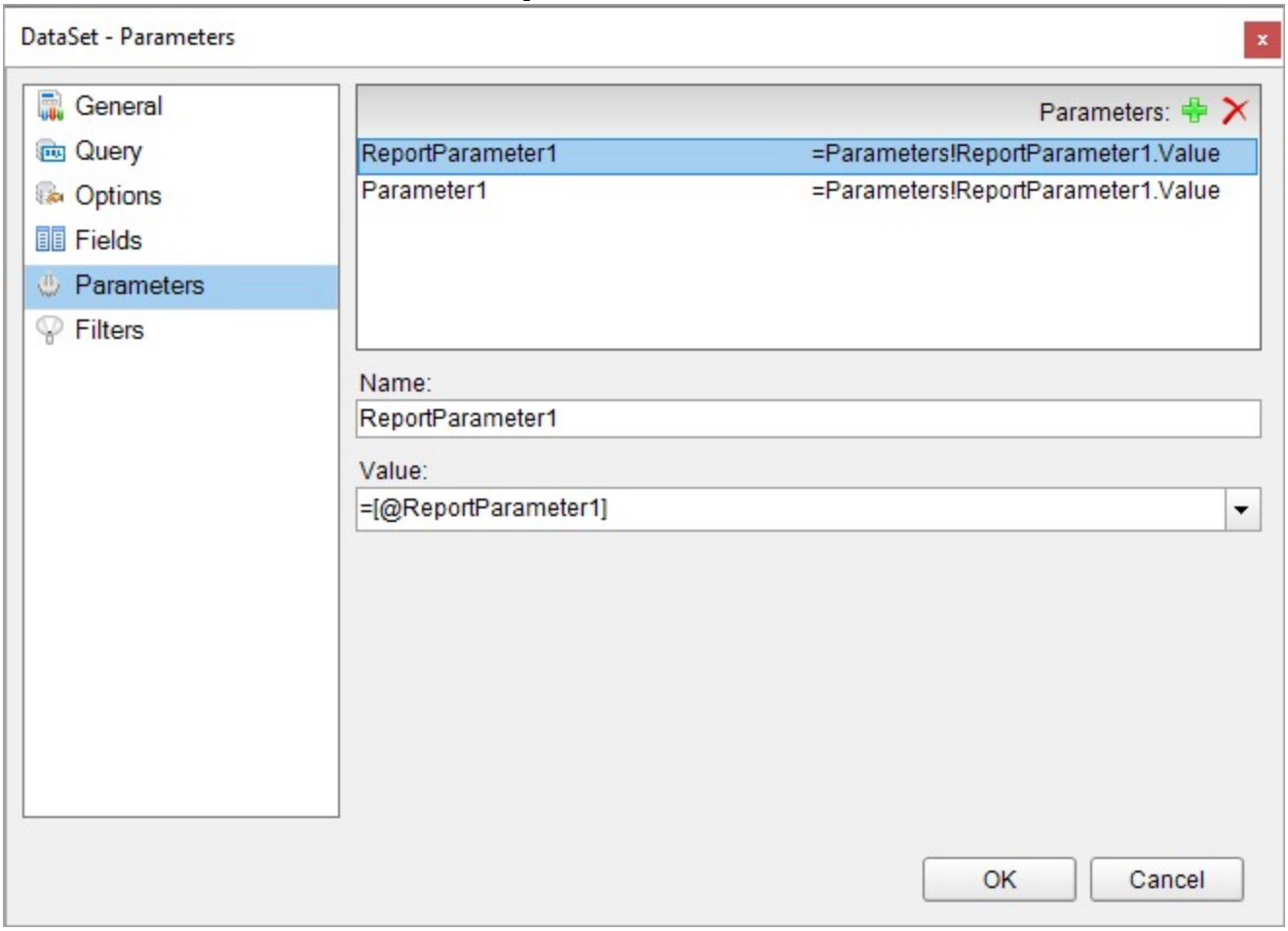
Provide List of Values for Report Parameter

1. In the **Report - Parameters** dialog, go to the **Available Values** tab and select the **From query** radio button.
2. Under the **Dataset** field, select the dataset created in the previous steps (DataSet1).
3. Under the **Value** and **Label** fields, use the drop-down to select the field, `productName`.
4. Click **OK** to close the dialog and add the parameter to the collection.

Add Dataset with Parameter

1. In the Report Explorer, right-click the Data Source (DataSource1) node and select **Add Data Set**.
2. In the DataSet dialog, on the **Parameters** page, click the **Add (+)** icon above the parameters list and add the

following to the dataset to provide values for the parameters we add to the query in the step 3 below.
Name: ReportParameter1 **Value**: `=Parameters!ReportParameter1.Value`
Name: Parameter1 **Value**: `=Parameters!ReportParameter1.Value`



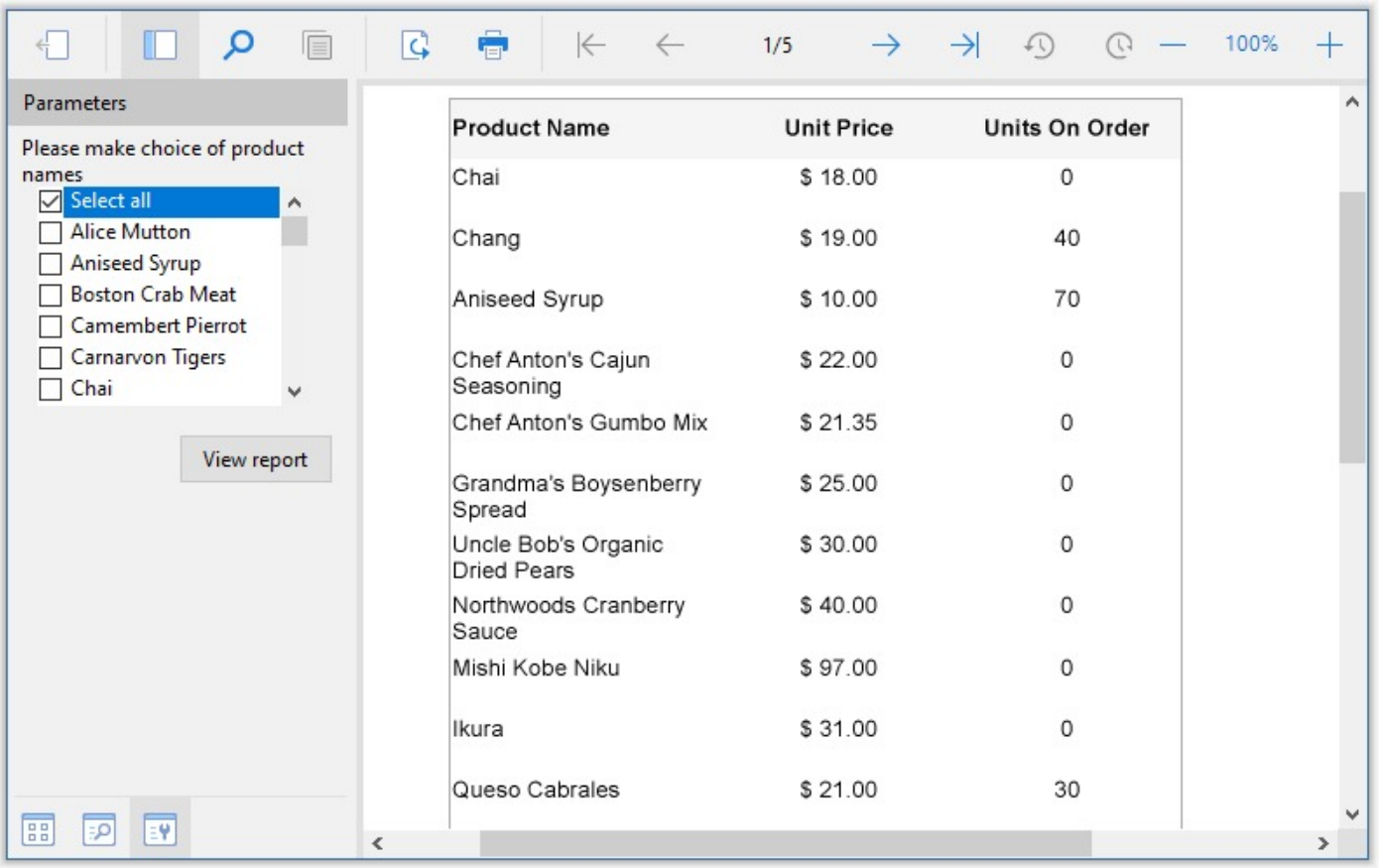
3. On the **Query** page, enter a SQL query like the following in the **Query** text box:

   ```
   SELECT * FROM products WHERE ProductName in (?) OR '1' in (?)
   ```

   At run time, this query matches the selected product name and fetches data accordingly. If the user chooses 'Select all' (for which we have specified value '1'), then query after 'OR' is evaluated and data is fetched for all products.
4. Click the **Validate DataSet** icon to validate the query and to populate the Fields list.
5. Click the **OK** to close the dialog. You see the data set, **DataSet2**, and the fields in the Report Explorer.

Design report

1. Drag-drop **Table** data region and bind it to the **DataSet2**.
2. Fill-in the table with some fields, for example, [ProductName], [UnitPrice], and [UnitsOnOrder]. See page on Table data region for more information.
3. Preview the report and observe the 'Parameters panel in the sidebar with **Select all** option at the top. For more information on Parameters pane, see Windows Forms Viewer.



Note: If the **Available Values** (queried or non-queried) for a parameter contain only some values from database and the **Select all** value is specified for the parameter, then at report preview, selecting the 'Select all' checkbox shows all records from the database instead of only those present in the parameter. For example, if the database has four records and in the **Available values** there are only two records, with the **Select all** value specified, then on previewing report and selecting 'Select all' checkbox, all four records are shown instead of only two.

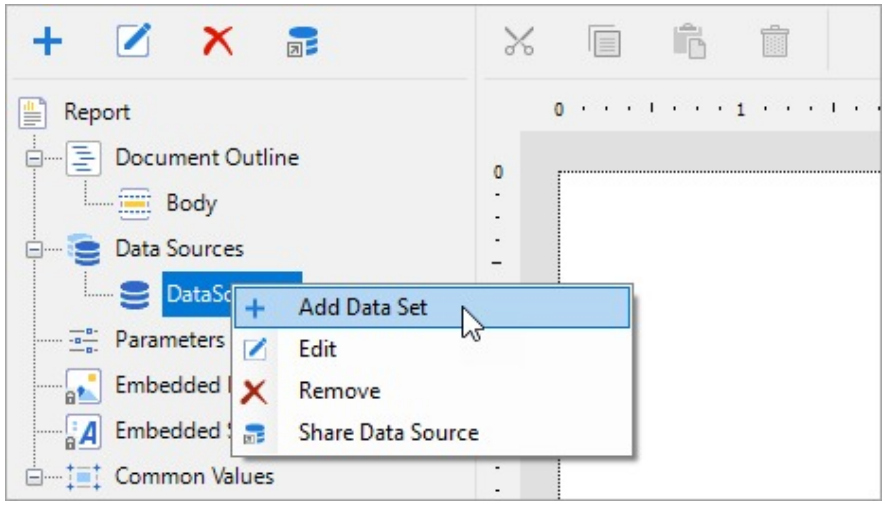
Cascading Parameter

When a parameter's value list depends on the value of another parameter, the report collects the required parameter value and uses it to create the value list for the second parameter. This cascade of parameter values is sometimes also called dependent or hierarchical parameters.

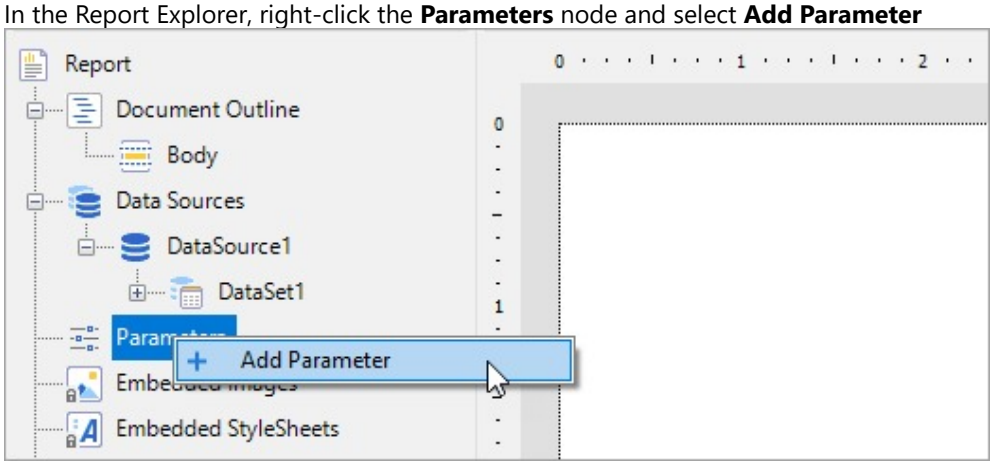
You can create cascading parameters in a Page or an RDLX report using the following steps.

Note: This topic uses the Reels database. The Reels.db file can be downloaded from GitHub.

1. In the Report Explorer, right-click the Data Source (DataSource1 by default) node and select **Add Data Set** to create a dataset named **Regions**.

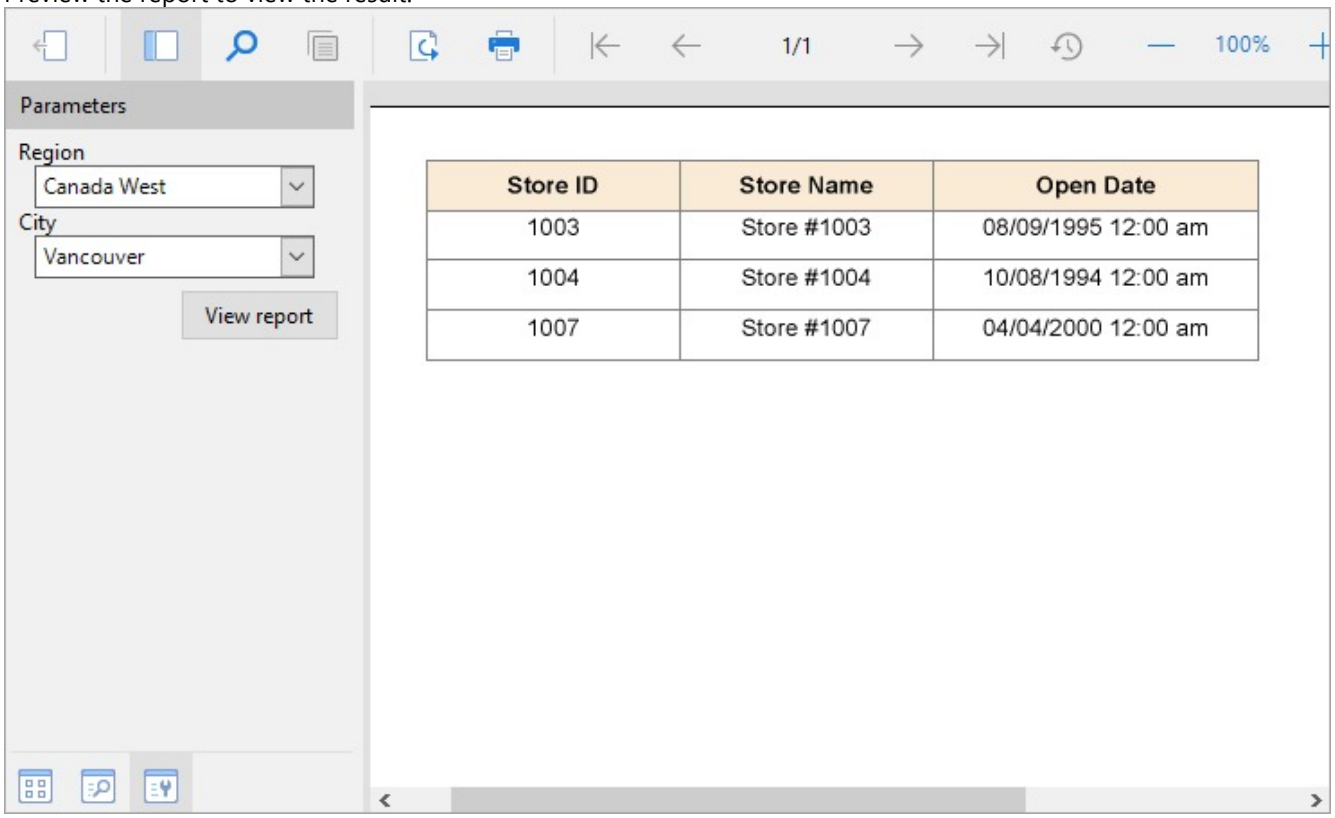


2. On the Query page of the **DataSet Dialog**, use the following SQL Query to fetch data from the Regions table.
   ```
   SELECT RegionID, Region FROM Regions
   ```
3. Click **OK** to close the dialog.
4. Follow the step 1 to create another dataset named **Districts** and on the **Parameters** page of the **DataSet** Dialog, click the **Add(+)** icon to add a parameter named **Region** with the value set to:
   ```
   =Parameters!Region.Value
   ```
   This parameter is added to the Report Parameters collection later.
5. In the **Districts** dataset dialog, on the **Query** page, add the following SQL query to fetch data from the Districts table. This query depends on the Region parameter.
   ```
   SELECT DistrictID, District FROM Districts WHERE Region = ?
   ```
6. Click **OK** to close the Districts DataSet dialog.
7. Follow the step 1 and create another dataset named **StoreNames** and on the **Parameters** page of the DataSet Dialog, click the **Add(+)** icon to add a parameter named **DistrictID** with the value set to:
   ```
   =Parameters!DistrictID.Value
   ```
   This parameter is added to the Report Parameters collection later.
8. In the **StoreNames** dataset, on the **Query** page, add the following SQL query to retrieve data for the selected region from the selected district. This query depends on the DistrictID parameter.
   ```
   SELECT StoreID, StoreName, OpenDate FROM Store WHERE NOT StoreID = 0 AND DistrictID = ?
   ```
9. Click **OK** to close the StoreNames DataSet dialog.
10. In the Report Explorer, right-click the **Parameters** node and select **Add Parameter**



11. In the **Report - Parameters** dialog that appears, add a parameter named **Region** with an Integer data type. On the **Available Values** tab, select **From query** and set the **Dataset** to Regions, the **Value** field to RegionID, and the **Label** field to Region.
12. Click **OK** to close the Report - Parameters dialog.

13. Follow the same process as steps 10 and 11 to add a second parameter named **DistrictID** with an Integer data type. On the **Available Values** tab, select **From query** and set the **Dataset** to Districts, DistrictID for the **Value** field, and District for the **Label** field.
14. Drag and drop a Table data region (or any other data region) onto the design surface, and drag the **StoreID**, **StoreName** and **OpenDate** fields onto the Details row.
15. Preview the report to view the result.



Notice that the two drop down lists, for regions and districts appear in the Parameters sidebar while the second drop down list remains disabled until a region is selected. Click the **View Report** button to see the StoreID, StoreName and OpenDate values returned for the selected region and district.

Note: In a Page report, when you have multiple datasets in the report, you need to set the **DataSet** property on the **General** tab of the FixedPage dialog in order to specify which dataset is used to display data in the report.

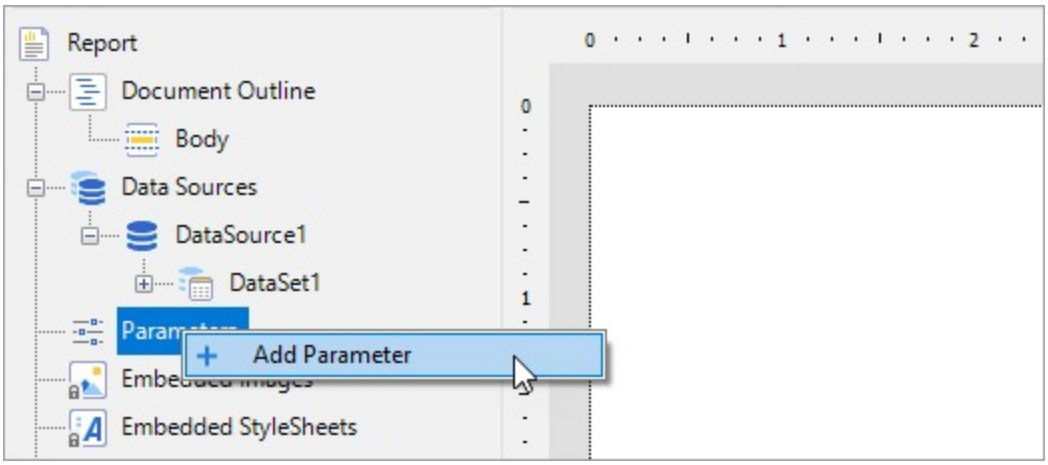
Hidden Parameter

If you want to run a report without prompting the user for a value at run time, you need to set a default value for each parameter. The report collects the required parameter value from the default value and uses it to generate the report.

Default values can be queried or non-queried. A non-queried default value can be a static value or an expression. A queried default value is a field value from a dataset.

Note: This topic uses the DVDStock table in the Reels database. The Reels.db file can be downloaded from GitHub.

1. In the Report Explorer, right-click the **Parameters** node and select **Add Parameter**.



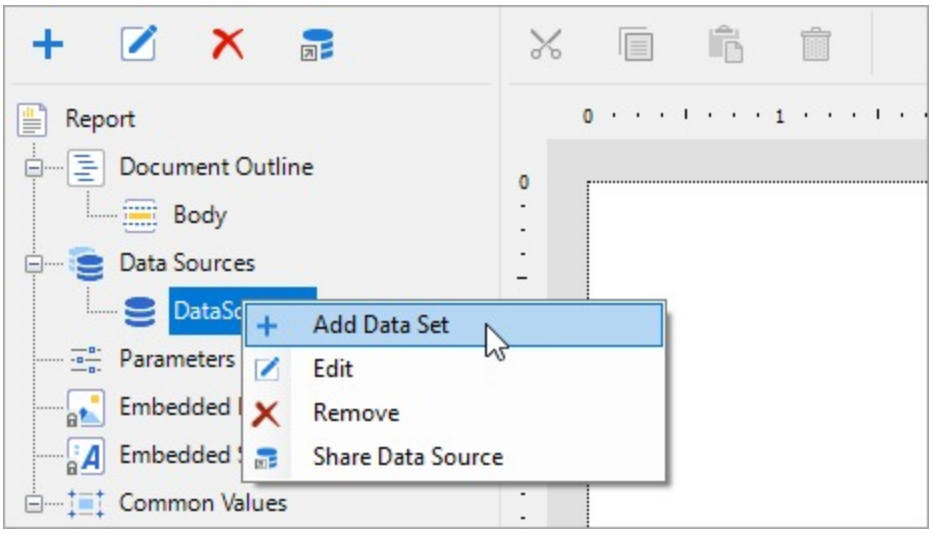
2. In the **Report - Parameters** dialog that appears, add a parameter named **StorePrice** with an **Integer** data type. Click the checkbox next to **Hidden** to hide the parameter UI at run time.
3. On the Default Values tab, select **Non-queried** and click the Add(+) icon to add an empty expression for the value.

> **Note**: When you use **From query** to provide a default value, only the first returned row value is used as the default value.

4. In the **Value** field enter 5 and click **OK** to close the Report - Parameters dialog.

> **Note**: When adding multiple default values, in the Report - Parameters dialog, General tab, check the **Multivalue** check box, otherwise the report collects only the first default value from the list and uses it to generate the report.

5. In the Report Explorer, right-click Data Source (DataSource1 by default) node and select **Add Data Set** to create a dataset.



6. In the DataSet Dialog that appears, on the Parameters page, click the Add(+) icon to add an empty expression for the parameter.
7. In the Name field, enter the same parameter name (**StorePrice**) you had added in the steps above and set its value to:
   `=Parameters!StorePrice.Value`
8. On the Query page of the DataSet Dialog, use the following SQL query to fetch data from the DvDStock table.
9. From the Toolbox, drag and drop a Table data region (or any other data region) onto the design surface, and

from the Report Explorer, drag the **Title**, **StorePrice** and **In Stock** fields onto the table Details row.

10. Preview the report to view the result.

Notice that the report collects the required parameter value from the default value (i.e. 5) and uses it to display the list of Movie DVDs with Store Price $5.

Title	Store Price	In Stock
The Towering Inferno	$5.00	6
Jaws	$5.00	11
Stir Crazy	$5.00	23
Beverly Hills Cop II	$5.00	20
Terminator 2: Judgment Day	$5.00	4
Beauty and the Beast	$5.00	22
The Lion King	$5.00	16
Interview with the Vampire: The Vampire Chronicles	$5.00	9
Casper	$5.00	10

Actionable Parameters

The 'Apply Parameters' action is available for any report item that has the **Action** property.

A parameter action modifies a parameter value through user interaction, like a mouse click, and displays the data for the currently selected data. You must define where to apply a parameter action to see the outcome (for example, filter in a chart), where the action needs to take place (unselect/select check boxes), and the action type to define the behavior of the action.

When the **Apply Parameters** action is performed at runtime, the data is re-queried and the viewer re-renders the report with a new set of parameters. This way, a report author can implement cross filters. Cross filters allow you to dynamically filter the data at runtime, making the report interactive.

> **Note**: The toggle and sorting states are not preserved when Apply Parameters action takes place.

Parameters Action Type

You can specify any of the following actions for the parameter action:

- **Set**: Sets the value of a specified parameter.
- **Reset**: Resets the parameter to its default value.
- **Toggle**: Adds/removes value to/from multi-value parameter values collection.

Let us describe few scenarios where actionable parameters are useful.

Apply Cross-Filters in Report

The general procedure for applying cross-filters in a report using the 'Apply Parameters' action is described as follows:

1. Create a report parameter. The report parameter can be a Multi-Value Parameter.
2. Select the report control or the data region where the mouse click action will take place on the preview. (eg., List).
3. With the report control or the data region selected, go to the **Properties** panel and click the ellipses near the **Action** property to open the **Navigation** or the **Action** page in a dialog.
4. In the **Action** property, select 'Apply Parameters' and fill in the following fields:
    - Select the **Name** of the parameter.
    - Specify the parameters action **Type**.
    - Enter the **Value** of the parameter. The Value is the expression for the new parameter.
5. Select the data region where the action (to filter data) should take place. For eg., to update the Chart data region on user action, select chart. It is here that the cross-filter will be applied.
   Note that there can be more than one data region where the cross-filter needs to be applied.
6. Go to the **Filters** property of the selected data region (here, chart). The filter expression should use the same parameter.

See the Create RDLX Dashboard Report tutorial that showcases applying cross-filters in a report in detail. The action performed on the list toggles the GenreID, which applies a new set of parameters based on the selection, and updates the chart.

Drill-Down Links

With the drill-down feature, you can temporarily hide a part of your report. That hidden part can be controls, groups, columns or rows. When you open a drill-down report, part of the data is hidden so that you can only see high-level data until you request for more detail. In such reports you find an expand icon (plus-sign image) next to the toggle item in the report. Clicking the toggle image, or plus sign, expands hidden content into view and the expand icon changes to a collapse icon (minus-sign). When you click the minus-sign image, it hides the content and returns the report to its previous state.

To create a drill-down report, use the **Visibility** properties of controls, groups, columns, or rows. Simply set the Visibility-hidden property to **True** and set the toggle item to the name of another item in the report, usually a text box in the group containing the hidden item. At run time, this puts a plus sign next to the toggle item which the user can click to display the hidden data.

If you export a drill-down report, any content which is hidden at the time of export remains hidden in the exported file. If you want all of the content to appear in the exported file, you must first expand all hidden data.

Create Drill-Down Report

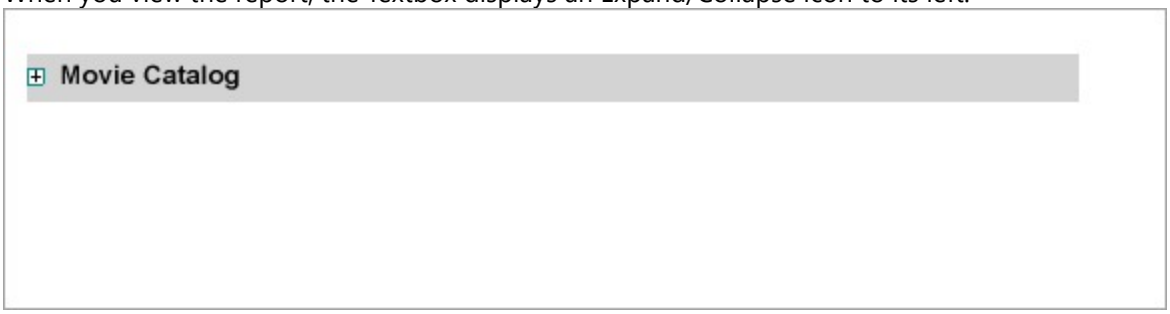
In a Page report, you can set up data regions, report controls, table rows, and tablix row and column groups to collapse, so that users can drill down into the data they choose to view.

In order to collapse an item, you use the **Visibility** settings available in the Properties Panel or in the control dialog.

You set the initial visibility of the report controls to Hidden and allow the user to toggle them by clicking other report controls (usually a TextBox).

When the report is initially displayed at run-time, the toggle items display with plus sign icons that you can click to display the detail data. Follow these steps to set a drill-down link.

1. Drag and drop a **TextBox** control and a **Table** data region onto the report design surface.
2. Place the TextBox control such that it appears as a header on your report.
3. From the Report Explorer, expand your data set and drag fields and place them inside the detail row of the Table data region. Expressions for these fields appear in the detail row, and labels appear in the table header row.
4. With the Table data region selected on the design surface, under the Properties window, click the **Property dialog** link to open the respective control's dialog.
5. In the Table dialog that appears, go to the **Visibility** page, change the **Initial visibility** to **Hidden**, and select the checkbox next to Visibility can be toggled by another report item.
6. From the drop-down list that appears, select the TextBox that you added in step 1. The TextBox is now used to toggle items in the Table and show detail data.
7. Click **OK** to save the changes.
8. When you view the report, the Textbox displays an Expand/Collapse icon to its left.



9. Click the Expand/Collapse icon to view the hidden data.

Title	Year Released	User Rating
Titanic	1997	9.1
Star Wars	1977	8
Shrek 2	2004	7.1
E.T. the Extra-Terrestrial	1982	7.3
Star Wars: Episode I - The Phantom Menace	1999	9.1
Spider-Man	2002	8.6
Star Wars: Episode III - Revenge of the Sith	2005	8.5
The Lord of the Rings: The Return of the King	2003	6.5
Spider-Man 2	2004	8.8
The Passion of the Christ	2004	8.8
Jurassic Park	1993	6.8

Bookmarks

A bookmark link is similar to a hyperlink, except that it moves the viewer to another area in the report instead of linking to a web page. You can create these links on a control using a Bookmark ID that connects to another target control.

Bookmarks are displayed at preview.

Add Bookmark

A bookmark link is similar to a hyperlink, except that it moves the viewer to another area in the report instead of opening a web page. You can add bookmarks in a two-step process:

- Identify the place (target control) where you want to allow a user to jump to with the help of a Bookmark ID.
- Share that Bookmark ID with another control that links to the target control.

Use the following steps to create a Bookmark ID on a Textbox control and create a bookmark link on another Textbox control at the bottom of the page in a Page / RDLX report.

To add a Bookmark ID on a control

Bookmark ID is like a URL that provides information required by the report viewer to locate the report control. You need to provide a Bookmark ID for any control to which you want to allow users to jump to via a Bookmark Link.

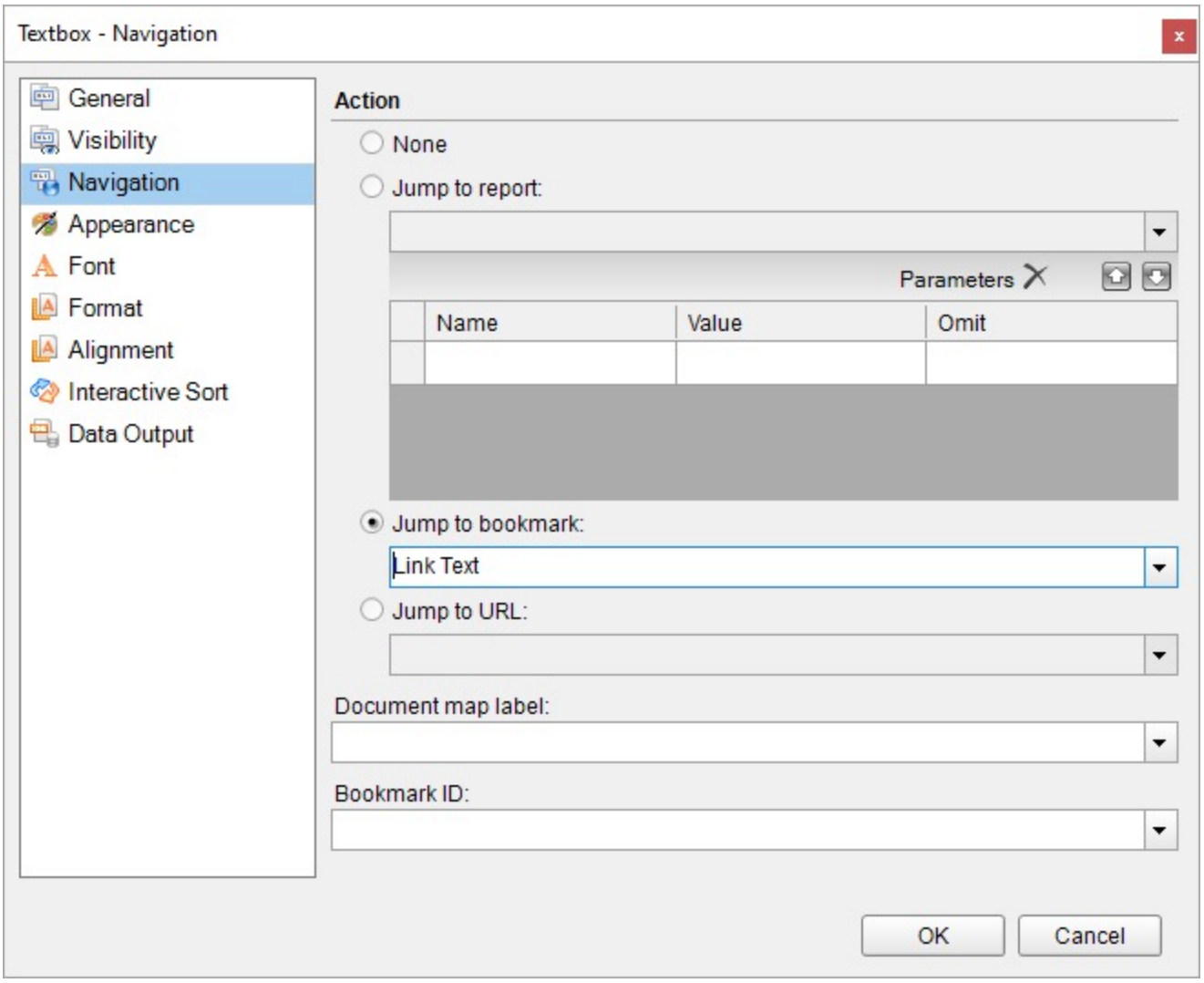
1. Drag and drop a Textbox control onto the design surface.
2. Select the Textbox to view its properties in the Properties window and enter any text in the **Value** property (For e.g., Top).
3. Click the **Property dialog** link below the Properties window to open the Textbox dialog.
4. In the TextBox dialog that appears, select the **Navigation** page and in the **Bookmark ID** field enter text like *Link Text*.
5. Click **OK** to close the dialog.

> **Tip:** You can also set the Bookmark ID through the **Bookmark** property in the Properties window.

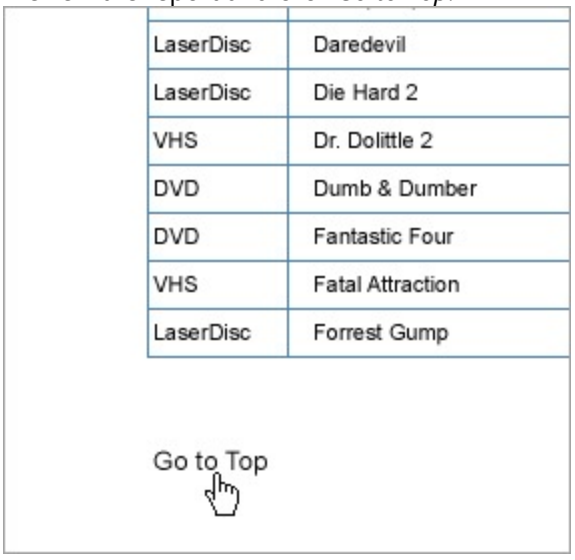
To set a Bookmark Link

Bookmark Link is a simple link you create to jump to the location where the Bookmark ID is set.

1. Drag and drop another Textbox control onto the design surface. Place it at the bottom of the page for this example.
2. Select the Textbox to view its properties in the Properties window and in the **Value** property enter *Go to Top*.
3. Click the **Property dialog** link below the Properties window to open the Textbox dialog.
4. In the Textbox dialog that appears, click on the **Navigation** page and select the **Jump To Bookmark** radio button to activate it.
5. Under **Jump To Bookmark**, enter the same text (i.e. *Link Text)* you assigned as Bookmark ID in the steps above.



6. Click **OK** to close the dialog.
7. Preview the report and click *Go to Top*.



You move to the top of the page where the Bookmark ID was set on the control.

> **Tip:** You can also access the Navigation page of a control to set the bookmark link through the ellipsis button next to the **Action** property in the Properties window.

Hyperlinks

Hyperlinks take you to a web page that opens in the default browser of the system. You can set hyperlinks in the Textbox, Image, Chart, and Map controls to access a Web page from your report.

Hyperlinks are displayed when you preview a Page or an RDLX report, export a report in HTML , PDF, RTF, and Excel formats.

Add Hyperlink to Textbox or Image

Let's set a hyperlink on the TextBox or Image controls to access a Web page from your a Page or an RDLX report.

1. Drag and drop a Textbox or Image control to the design surface.
2. With the control selected, under the Properties window, click the **Property dialog** link to open the control's dialog and go to the **Navigation** page.
   OR
   With the control selected, go to the Properties window and click the ellipses near the **Action** property to open the **Navigation** page in the dialog.
3. On the **Navigation** page, select the **Jump to URL** radio button to enable the field below it.
4. Type or use the expression editor to provide a valid Web page address. For example, https://developer.mescius.com/activereportsnet.
5. Click **OK** to close the dialog.
6. In the Properties window, enter text in the **Value** property of the respective control to set the display text for the Web page hyperlink. For example, 'Company'.

Add Hyperlink to Map Layer Elements

Map layer elements like point, polygon and line provides you with a functionality to set hyperlinks on them to access a Web page from your report.

1. On the design surface, click the map until the map panes appear.
2. In the layers pane, right-click the layer in use and select **Edit**.
3. In the selected layer's dialog that appears, go to the **Navigation** page.
4. On the Navigation page, select the **Jump to URL** radio button to enable the field below it.
5. Type or use the expression editor to provide a valid Web page address. For example, https://developer.mescius.com/activereportsnet.
6. Click **OK** to close the dialog.

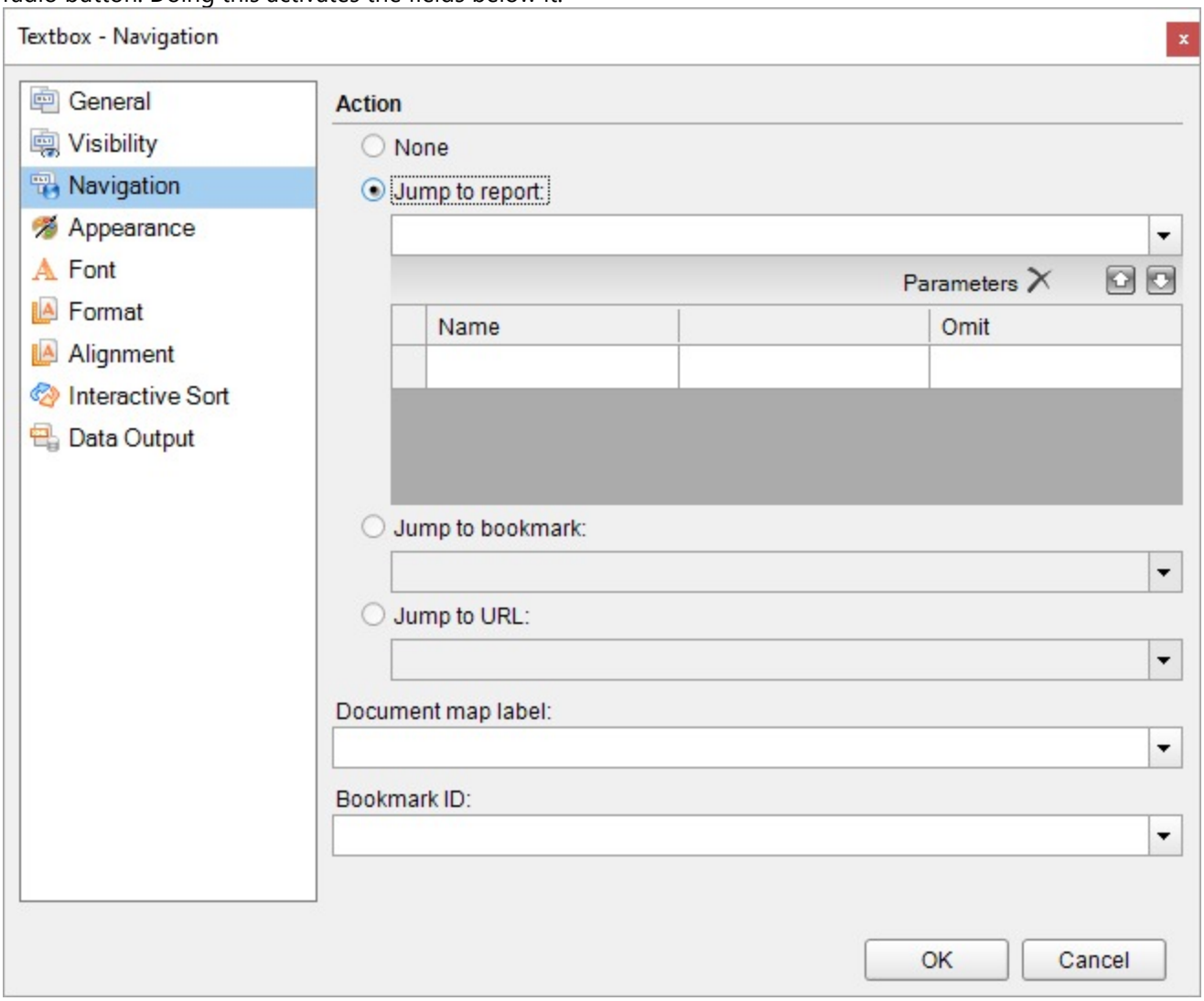
Drill-Through Links

A drill-through link takes you to another report with more detail. Drill-through links appear as a hyperlink that you can click to move to a completely different report. You can also create more complex links where you pass parameters to the report called by the link.

Drill-through links are displayed when you preview a Page/RDLX report.

Follow these steps to set a drill-through link.

1. On the design surface, select a report control (like a TextBox) on which you want to set the link and under the Properties window, click the **Property dialog** link to open the TextBox dialog.
2. In the control dialog that appears, go to the **Navigation** page and under **Action**, select the **Jump to report** radio button. Doing this activates the fields below it.



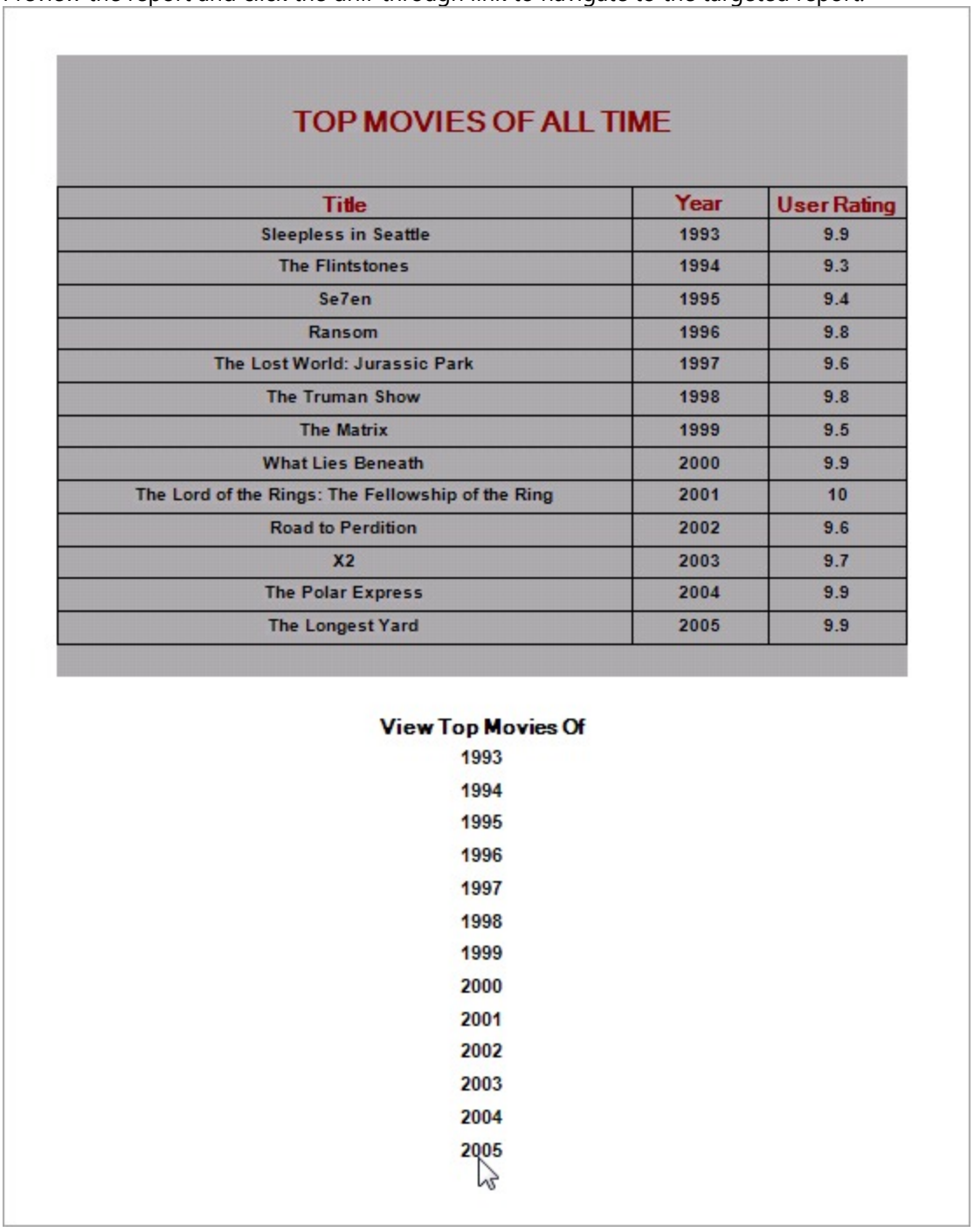
3. Under the **Jump to Report** field, enter the name of the report (like BasicReport.rdlx) that you want to navigate to on clicking the drill-through link. You can also use expressions to create drill-through links.

> **Note:** In the **Jump to Report** field, enter just the report name if the targeted report is in the same directory as the parent report. Or you can enter a relative path to the report. Use the Custom Resource Locator to jump to a report in your connected database.

4. After setting the detail report to drill-through, on the **Navigation** page, under **Parameters**, you can optionally enter a valid parameter **Name** and **Value** to pass to the detail report. This value must evaluate to a valid value for the parameter. By setting parameters you can jump right to the desired information. For example, if your summary report contains a list of invoice numbers and delivery dates for each customer, you could use a drill-through link with the invoice number as the parameter to allow the user to jump to the relevant invoice.

> ⚠ **Caution:** The Parameter Name must exactly match the name of the parameter in the detail report. If any parameter is spelled differently, capitalized differently, or if an expected parameter is not supplied, the drill-through report fails.

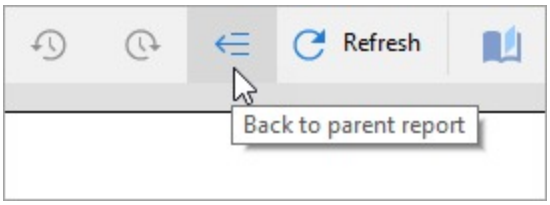
5. Preview the report and click the drill-through link to navigate to the targeted report.



TOP MOVIES OF ALL TIME

Title	Year	User Rating
Sleepless in Seattle	1993	9.9
The Flintstones	1994	9.3
Se7en	1995	9.4
Ransom	1996	9.8
The Lost World: Jurassic Park	1997	9.6
The Truman Show	1998	9.8
The Matrix	1999	9.5
What Lies Beneath	2000	9.9
The Lord of the Rings: The Fellowship of the Ring	2001	10
Road to Perdition	2002	9.6
X2	2003	9.7
The Polar Express	2004	9.9
The Longest Yard	2005	9.9

6. On the Viewer toolbar, click the **Back to Parent Report** button to return to the main report.



Sort Data

In order to better organize and present the data in your report, you can sort it alphabetically or numerically in ascending or descending order. You can also use sorting with grouped data to present an easy to understand, comprehensive view of report data.

In a Page or an RDLX report, you can sort data in a data region, in a data grouping, or on a fixed page in a Page report. You can also sort the data directly in the SQL query. In addition, you can also set interactive sorting for your data on a TextBox control.

ActiveReports provides a **Sorting** page in the dialogs of a data region, grouped data and fixed page to determine where you want to display sorted data.

Sorting Data in a Data Region

In Table and List data regions, you can sort data within the data region. To sort data within these data regions, set sorting in the **Sorting** page of the specific data region's dialog.

In Tablix, BandedList and Chart data regions, sorting is only possible on grouped data; therefore, there is no independent Sorting page available in their specific dialogs.

Sorting Grouped Data

A **Sorting** tab is available inside the **Groups** page of all the data region dialogs and the **Detail Grouping** page of the **List** dialog. It allows you to set the sort order of grouped data. This tab is enabled once grouping is set inside the data region.

Sorting on a Fixed Page

In a Page report, sorting is also possible on a fixed page grouped on a dynamic value. Sorting data on a fixed page is similar to sorting grouped data in a data region. The only difference is when you sort data on the fixed page you apply sorting to all the data regions that are placed on the design surface.

Sorting Data via SQL Query

When you connect to a data source and create a data set to fetch data for your report, you define a query. Set the ORDER BY keyword in the query to sort data in ascending or descending order.

By default, the ORDER BY keyword usually sorts the data in ascending order, but you can include the DESC keyword in your query to sort data in descending order. For example, if you want to fetch data from the 'Movie' table of the 'Reels.db' data source and sort it on the `Title` field, your query looks like the following:

```
<>SELECT * FROM Movie ORDER BY Title
```

or

```
SELECT * FROM Movie ORDER BY Title ASC
```

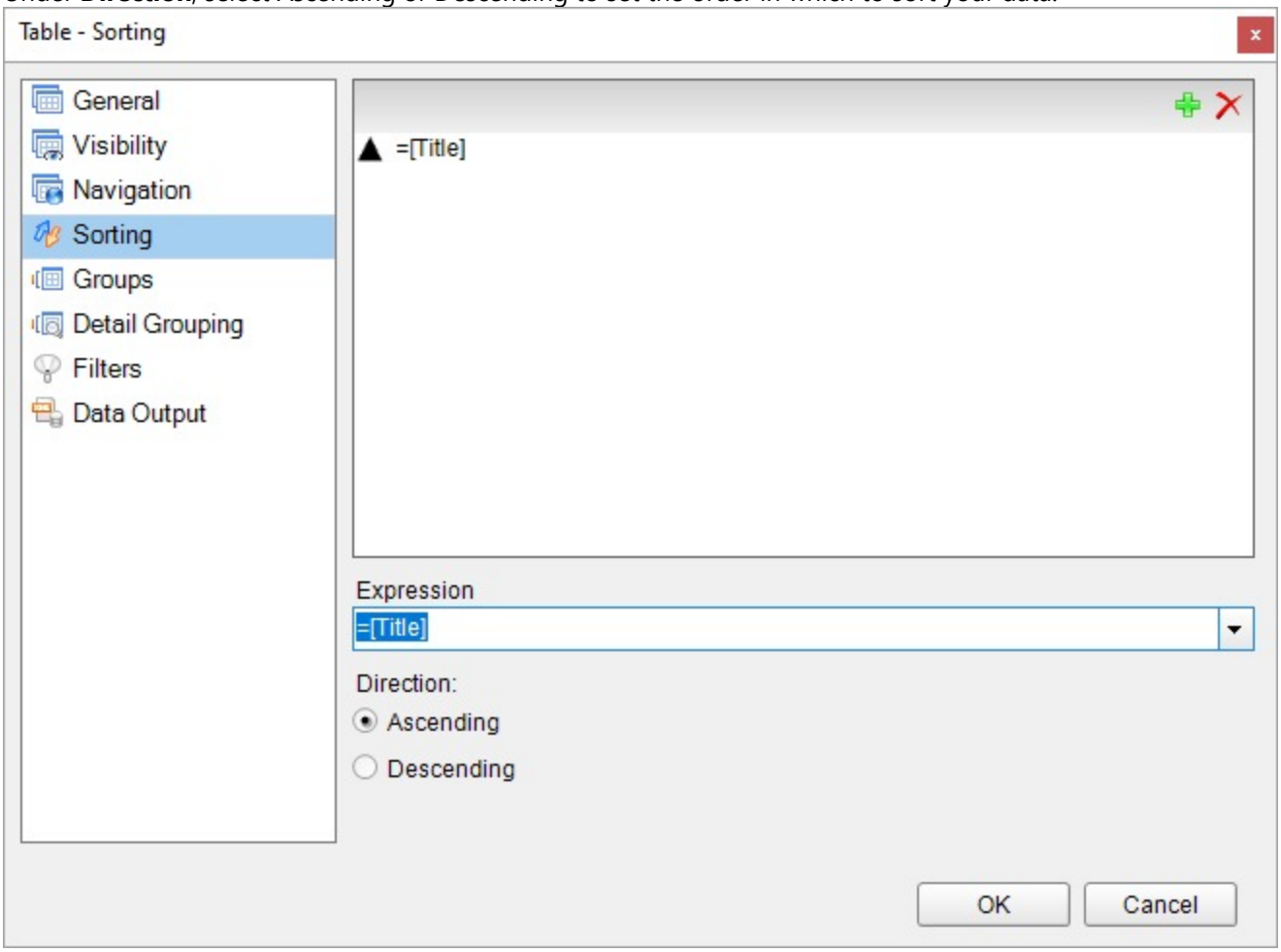
In case you want the *Title* field sorted in descending order, your query looks like the following:

```
SELECT * FROM Movie ORDER BY Title DESC
```

Sorting in Data Region

You can set the sorting expression on the **Sorting** page of the data region dialog. Let us elaborate on the sorting in a data region in a report that uses the 'Movie' table from 'Reels.db' data source on GitHub.

1. Right-click the data region and select **Properties** to open the Properties window. Select the Property dialog link under properties where the commands are displayed to open the data region dialog.
2. In the dialog that appears, go to the **Sorting** page and click the Add(+) icon to add an empty expression to the sorting list below.
3. In the Expression field, enter the expression directly or use <Expression...> from the dropdown to open the Expression Editor and select the field on which you want to sort your data.
4. Under **Direction**, select Ascending or Descending to set the order in which to sort your data.



5. Click **OK** to close the dialog.
6. From the Report Explorer, drag and drop the field on which the sorting expression is set.
7. Preview the report view the result.

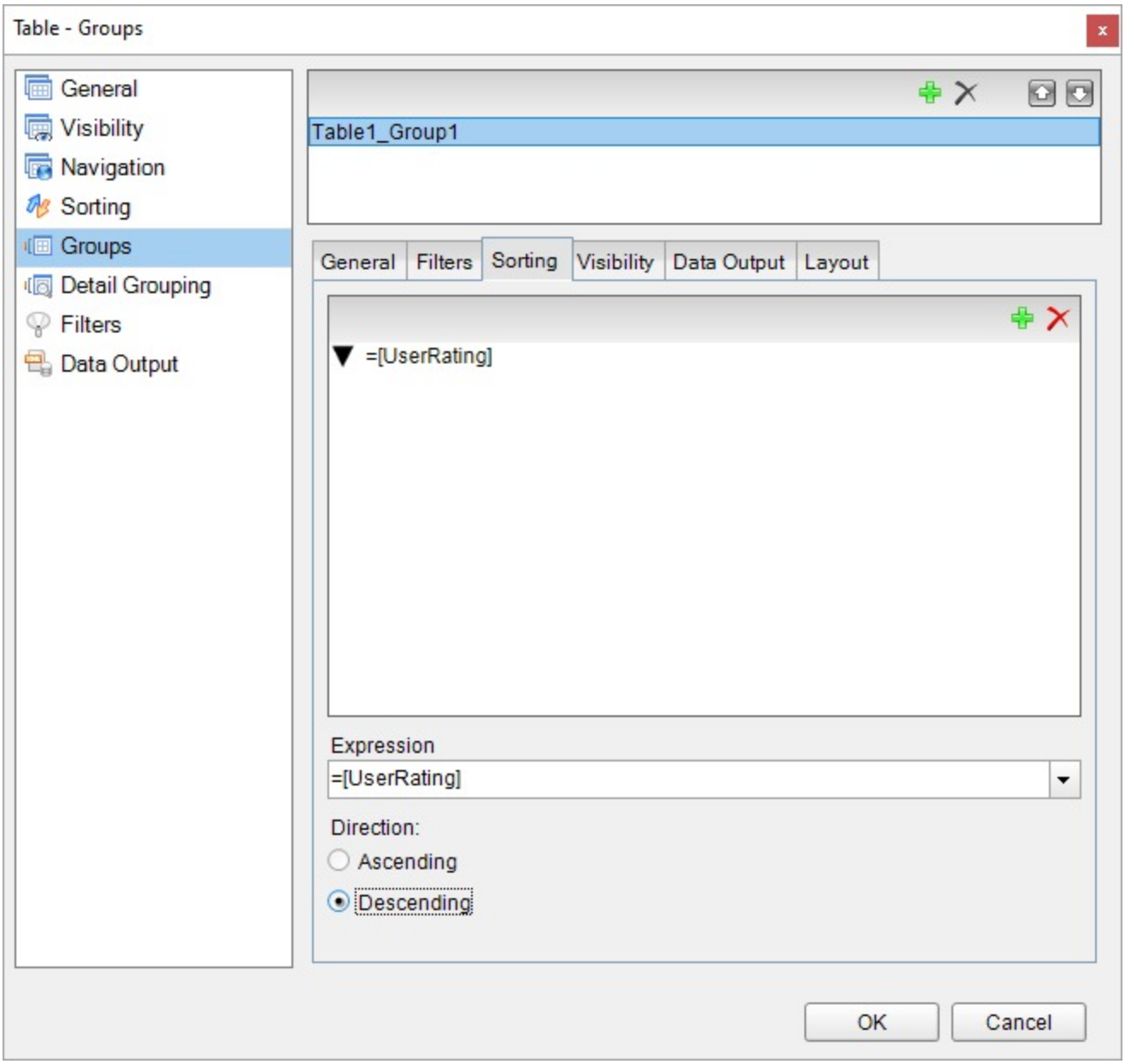
Title	Year Released	User Rating
101 Dalmatians	1996	9
2 Fast 2 Furious	2003	7.8
50 First Dates	2004	8.6
8 Mile	2002	9.1
A Beautiful Mind	2001	9.3
A Bug's Life	1998	8.7
A Few Good Men	1992	8.8
A League of Their Own	1992	9
A Time to Kill	1996	6
Ace Ventura: When Nature Calls	1995	7.1
Air Force One	1997	6

Sorting on Grouped Data

You can sort the order of groups through the **Sorting** tab of the **Groups** page or the **Detail Grouping** page of the List data region. The data region is bound to the 'Movie' table from 'Reels.db' data source on GitHub.

> **Note:** In a **Chart** data region dialog, the **Sorting** tab is available on the **Category Groups** and **Series Groups** pages.

1. On the **Groups** or the **Detail Grouping** page of the data region dialog, select the **Sorting** tab.
2. In the **Sorting** tab, click the Add(+) icon to add an empty expression to the sorting list.
3. In the Expression field, enter the expression directly or use <Expression...> from the dropdown to open the Expression Editor and select the field on which you want to sort your data. The expression set here should be the same as the grouping expression.
4. Under **Direction**, select Ascending or Descending to set the order in which to sort your data.



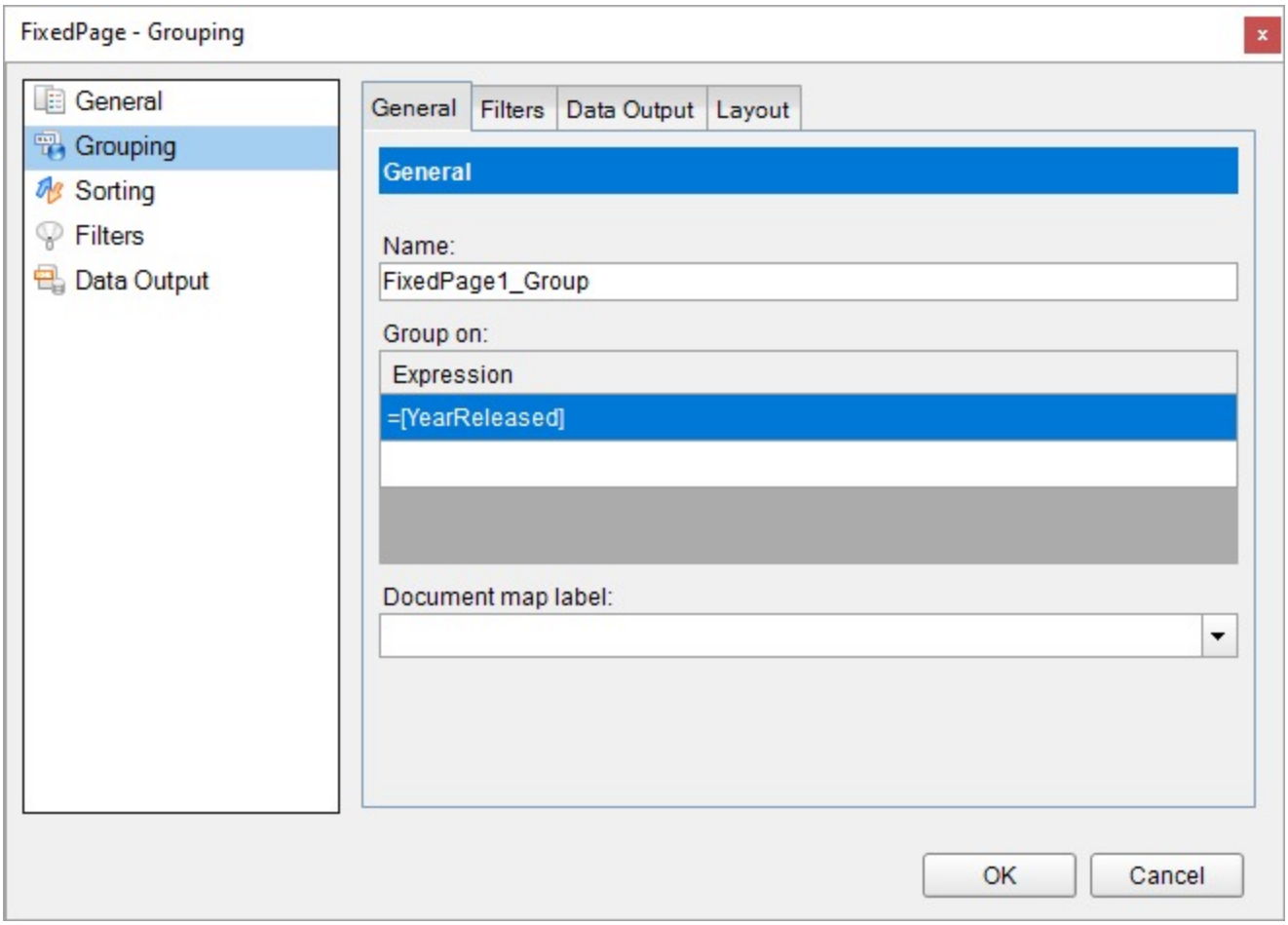
5. Click **OK** to close the dialog.
6. Preview the report to view the result.

Title	Year Released	User Rating
The Lord of the Rings: The Fellowship of the Ring	2001	10
Lethal Weapon 3	1992	10
Superman	1978	10
Mission: Impossible II	2000	9.9
The Polar Express	2004	9.9
The Longest Yard	2005	9.9
What Lies Beneath	2000	9.9
The Silence of the Lambs	1991	9.9
Sleepless in Seattle	1993	9.9
Rocky	1976	9.9
How the Grinch Stole Christmas	2000	9.8
Home Alone 2: Lost in New York	1992	9.8
Ransom	1996	9.8

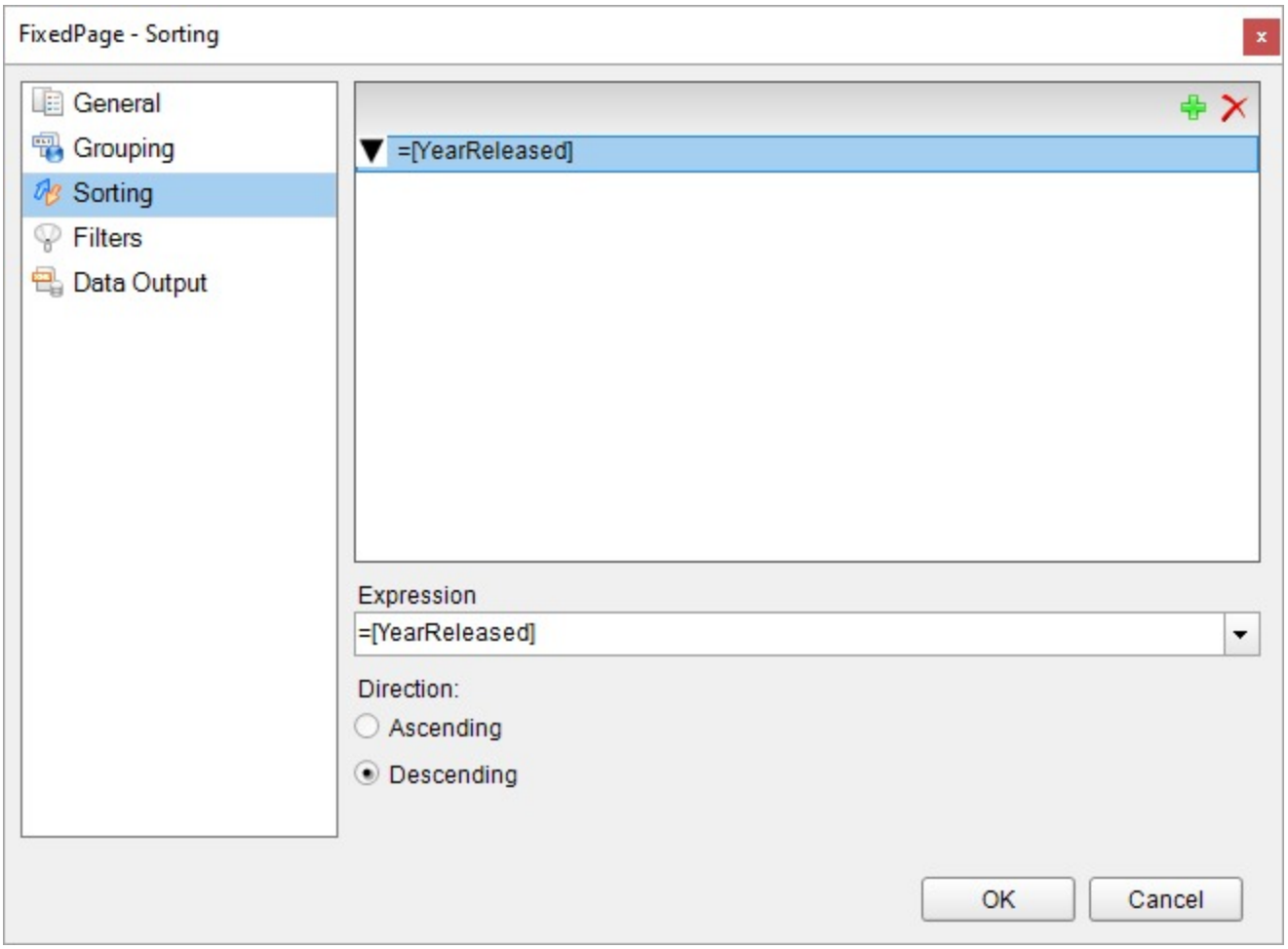
Sorting on Fixed Page

In a Page report, if the fixed page is grouped on a dynamic value, you can sort the order of the groups through the **Sorting** page of the **FixedPage** Dialog. Follow these steps to set up sorting on a fixed page in a report bound to the 'Movie' table from 'Reels.db' data source on GitHub:

1. Right-click the gray area of the report and select **Fixed Layout Settings** to open the **FixedPage** dialog.
   OR
   Right-click the report page and select Properties to open the Properties window. Select the **Property dialog** link under properties where the commands are displayed to open the **FixedPage** dialog.
2. In the **FixedPage** dialog that appears, go to the **Grouping** page and in the Expression field, enter the expression directly or use <Expression...> from the dropdown to open the Expression Editor and select the field on which you want to group your data.



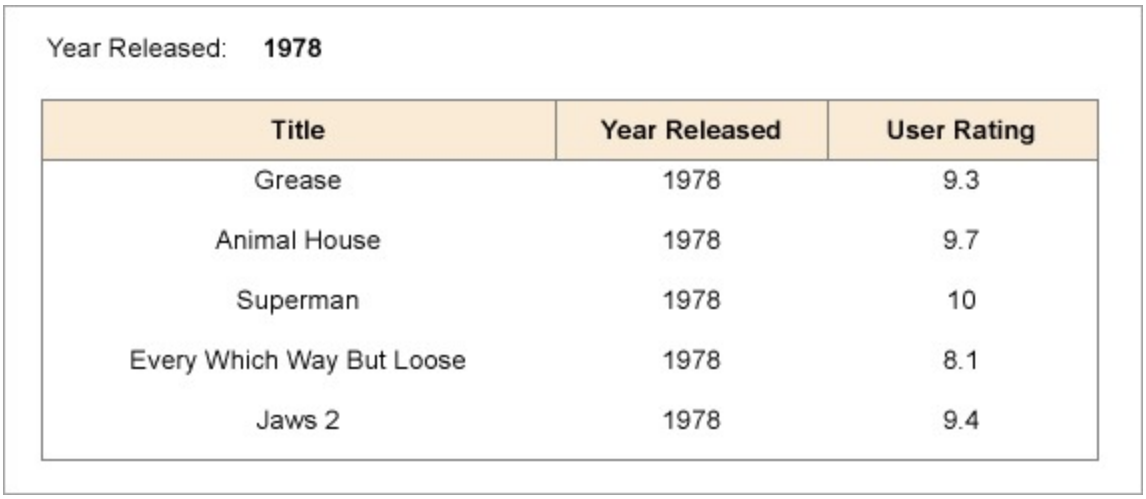
3. In the **FixedPage** dialog, now go to the **Sorting** page and click the Add(+) icon to add an empty expression to the sorting list below and in the Expression field enter the same expression you used for grouping the data.
4. Under **Direction**, select Ascending or Descending to set the order in which to sort your data.
5. Click **OK** to close and apply the settings.



6. From the Report Explorer, drag and drop the field on which sorting is set.
7. Preview the report to view the result.

Year Released: **1977**

Title	Year Released	User Rating
Star Wars	1977	8
Saturday Night Fever	1977	9.7
Close Encounters of the Third Kind	1977	5.1
Smokey and the Bandit	1977	8


Year Released: **1978**

Title	Year Released	User Rating
Grease	1978	9.3
Animal House	1978	9.7
Superman	1978	10
Every Which Way But Loose	1978	8.1
Jaws 2	1978	9.4

Note: The difference in setting sorting on a Fixed Page is that it affects every data region placed on the report layout, whereas sorting on a data region is limited to the data region only.

Interactive Sorting

You can add interactive sorting to a TextBox control to allow users to sort columns of data within a data region on a published report.

The interactive sorting feature is set through the **Interactive Sort** page that is available in the TextBox dialog.

Once you set interactive sorting on a TextBox control, while previewing the report, the textbox control displays a sort icon inside it. A user can sort data that appears inside the textbox in ascending or descending order by clicking the icons.

On the Interactive Sort page of the TextBox dialog you can find the following fields for entering values:

- **Sort Expression:** An expression specifying the sort value for data contained in the column.
- **Data region or group to sort:** Select the grouping level or data region within the report to sort. The default value is Current scope, but you may also choose an alternate data region or grouping.
- **Evaluate sort expression in this scope:** Select the grouping level within the report on which to evaluate an aggregate sorting expression. The default value is Current scope, but you may also choose an alternate data region or grouping.

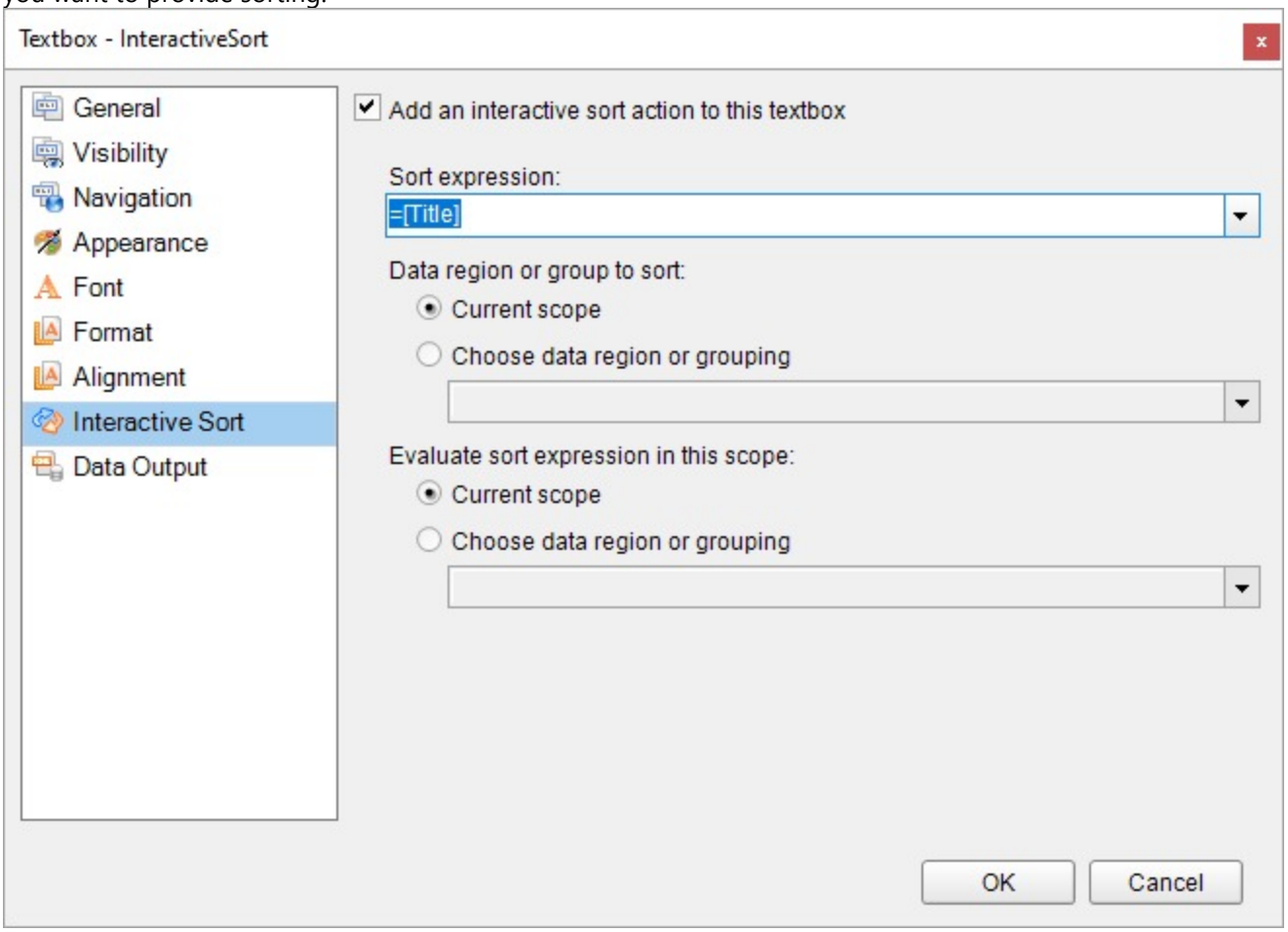
Set Interactive Sorting on TextBox

Let's set up the interactive sorting in a report that is bound to the 'Movie' table from 'Reels.db' data source on GitHub and displays movies with the title, release year and user rating information. We will add the interactive sorting to the Title textbox, so that you can view the list of movies by the title, changing the descending or ascending order of movies, from A to Z and vice versa.

1. From the Toolbox, drag and drop a Table data region onto the report.
2. From the Report Explorer, drag fields into the detail row of the table. Labels appear in the table header row, and expressions appear in the detail row.
3. Click to select a TextBox in the header row on which you want to allow users to sort, and in the Commands section at the bottom of the Properties Panel, click the **Property dialog** command.
4. In the **TextBox** dialog that appears, select the **Interactive Sort** page.
5. Select the checkbox next to **Add an interactive sort action to this textbox** and enable the other properties on

the page.

6. Under **Sort expression**, drop down the list and select the expression containing the value of the field on which you want to provide sorting.



> **Note**: Under **Data region or group to sort**, and **Evaluate sort expression in this scope**, you can optionally choose a different a scope from the **Choose data region or grouping** drop down list.

7. Click **OK** to close the dialog and accept the changes.
8. Preview the report. At preview, you can see a sort icon next to the TextBox control, the **Title** header in this case.

Title	Year Released	User Rating
Chicken Run	2000	6.9
Chicken Little	2005	5.6
Chicago	2002	7.7
Cheaper by the Dozen	2003	5.6
Charlie's Angels: Full Throttle	2003	9.5
Charlie's Angels	2000	6.4
Charlie and the Chocolate Factory	2005	5.9
Catch Me If You Can	2002	7.4
Cast Away	2000	5.7
Casper	1995	8.4
Butch Cassidy and the Sundance Kid	1969	6.4
Bruce Almighty	2003	7.9
Bringing Down the House	2003	9.3
Blazing Saddles	1974	5.5
Black Hawk Down	2001	9.6
Big Momma's House	2000	8.3

You can click the icon to sort in descending order. The icon changes to an up arrow that you can click to sort in ascending order.

Title	Year Released	User Rating
101 Dalmatians	1996	9
2 Fast 2 Furious	2003	7.8
50 First Dates	2004	8.6
8 Mile	2002	9.1
A Beautiful Mind	2001	9.3
A Bug's Life	1998	8.7
A Few Good Men	1992	8.8
A League of Their Own	1992	9
A Time to Kill	1996	6
Ace Ventura: When Nature Calls	1995	7.1
Air Force One	1997	6
Airport	1970	9.2
Aladdin	1992	5.1
American Beauty	1999	8.4

Document Map

When you click an item in the document map, the viewer jumps to that item in the report.

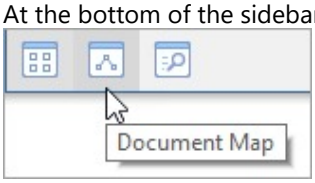
A document map functions as a table of contents and provides a convenient way to navigate a lengthy report.

In a Page or an RDLX report, you can add report controls, data regions, groups and detail groups to the document map by:

- Assigning a value to the **Document map label** on the **Navigation** page of the corresponding dialog.
- Setting the value of the **Label** property in the properties window.
- Setting the value of the **HeadingLevel** property in the properties window.

Viewing the Document Map

1. On the Viewer toolbar, click the Side Panel button to display the sidebar.

2. At the bottom of the sidebar pane, click the **Document map** button to display the document map.


If there is no document map associated with the report, the button does not appear at the bottom of the sidebar pane.
3. In the Document map that appears, click the item you want to view in the report.

Exporting Document Map

At preview, the document map appears in a sidebar to the left of the report, but when you export your report to various file formats, they handle document maps differently.

Export Filter	Effect on Document Map
HTML	A .toc file containing the document map is exported along with the HTML report.
PDF	Document map appears in the bookmarks panel.
Text	Document map does not appear in the exported report.
Rich Text Format	Document map does not appear in the exported report.
TIFF	Document map does not appear in the exported report.
Excel	Document map does not appear in the exported report.

Note: For printing and rendering purposes use Table of Contents control in your Page report and RDLX report.

To add a control to the Document Map

Using Document Map Label or Label Property

1. On the design surface, select a control you want to add to the Document map and right-click to choose Properties from the context menu.
2. In the command section of the Properties Panel, click the **Property dialog** open the control's dialog.
3. In the dialog that appears, go to the **Navigation** page and under the **Document map label**, enter a text or an expression representing the control in the Document map.

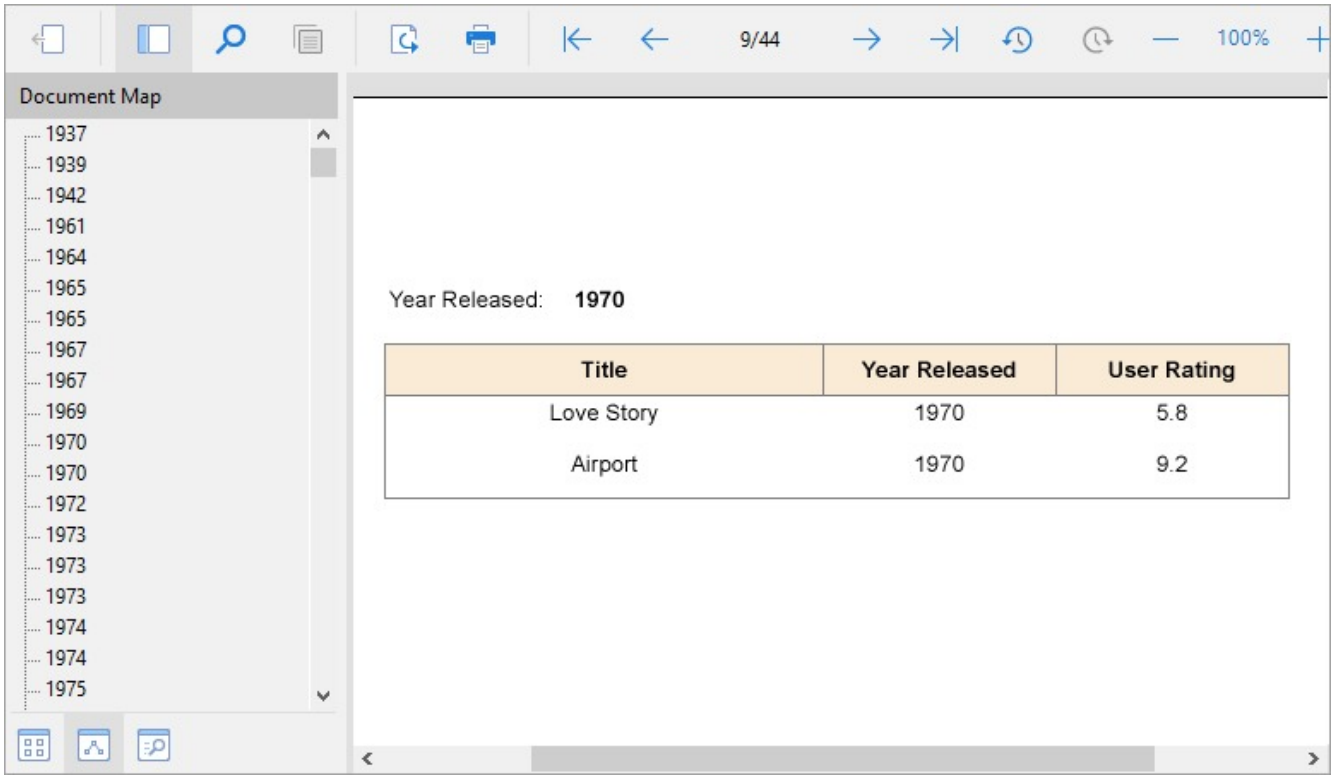
Alternatively,

1. On the design surface, select the report control you want to add to the Document map and right-click to choose Properties from the context menu.
2. In the Properties window that appears, enter a text or an expression in the **Label** property to represent the report control in the Document map.

Using HeadingLevel Property

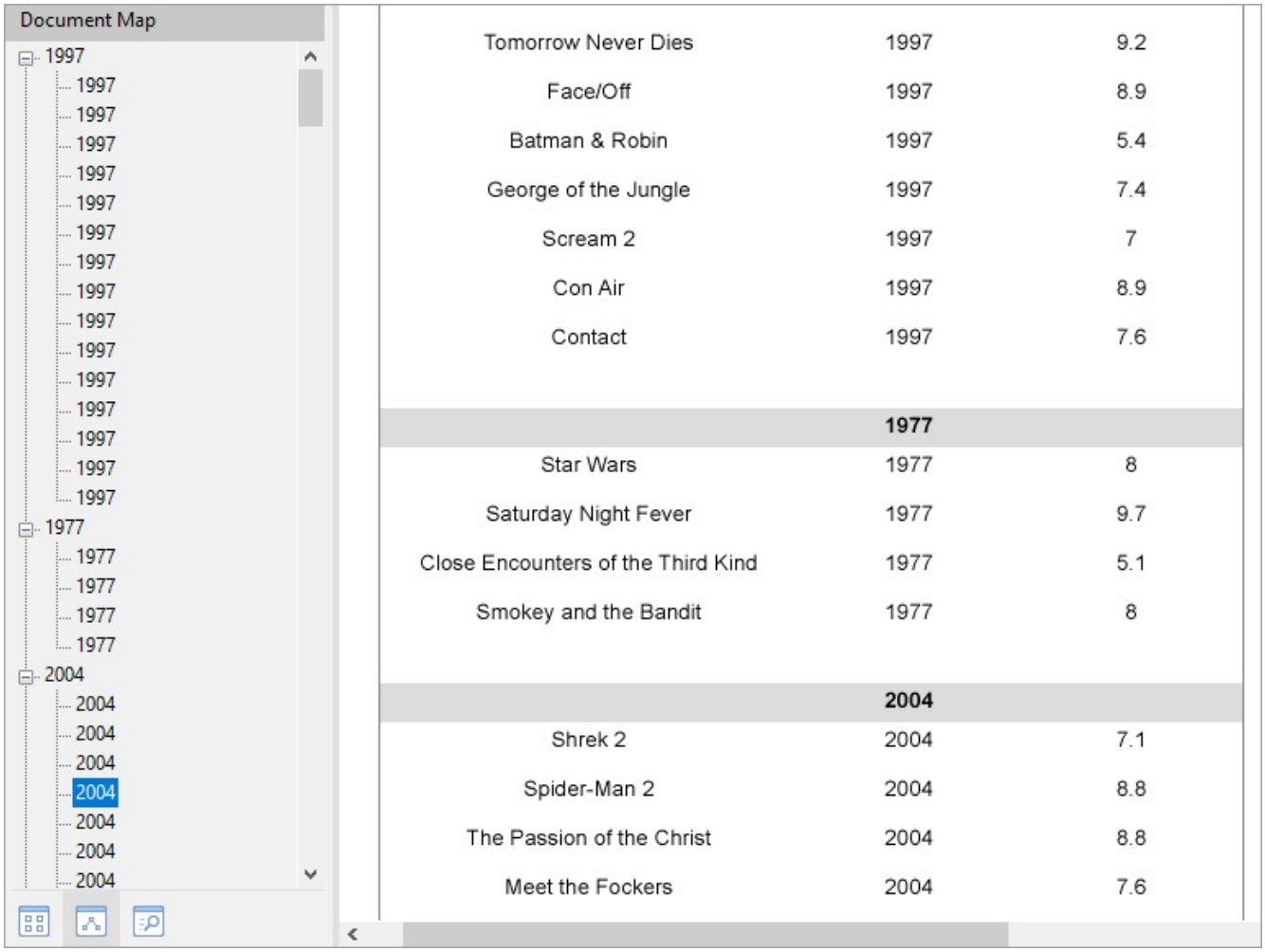
1. On the design surface, select the report control you want to add to the Document map (e.g. TextBox).
2. In the Properties window that appears, set the **HeadingLevel** property of the report control.

   Note The **HeadingLevel** property defines the DocumentMap level for that control.
3. Preview the report to view the document map.



To create a hierarchical Document Map

1. Open a report that contains a data region like a Table.
2. Select the data region and in the Properties Panel that appears, select the **Property dialog** command at the bottom of the window to open the data region dialog.
3. On the data region dialog, on the **Groups** page, set the grouping expression. For example, group the data on YearReleased (`=Fields!YearReleased.Value`).
4. On the same **Groups** page, set the Document map label to the value of the grouping expression. For example, `=Fields!YearReleased.Value`.
5. On the design surface, select a control inside the data region. For example, the TextBox in the detail row of the Table data region.
6. Right-click the control and select properties to open the Properties Panel. In the command section of the Properties Panel, click **Property dialog**.
7. In the dialog that appears, go to the **Navigation** page and under the Document map label, enter a text or an expression representing the control in the Document map.
8. Preview the report to view the document map.



Note: In an Page report, you can also set grouping and set the document map label on the **FixedPage dialog > Groups** page.

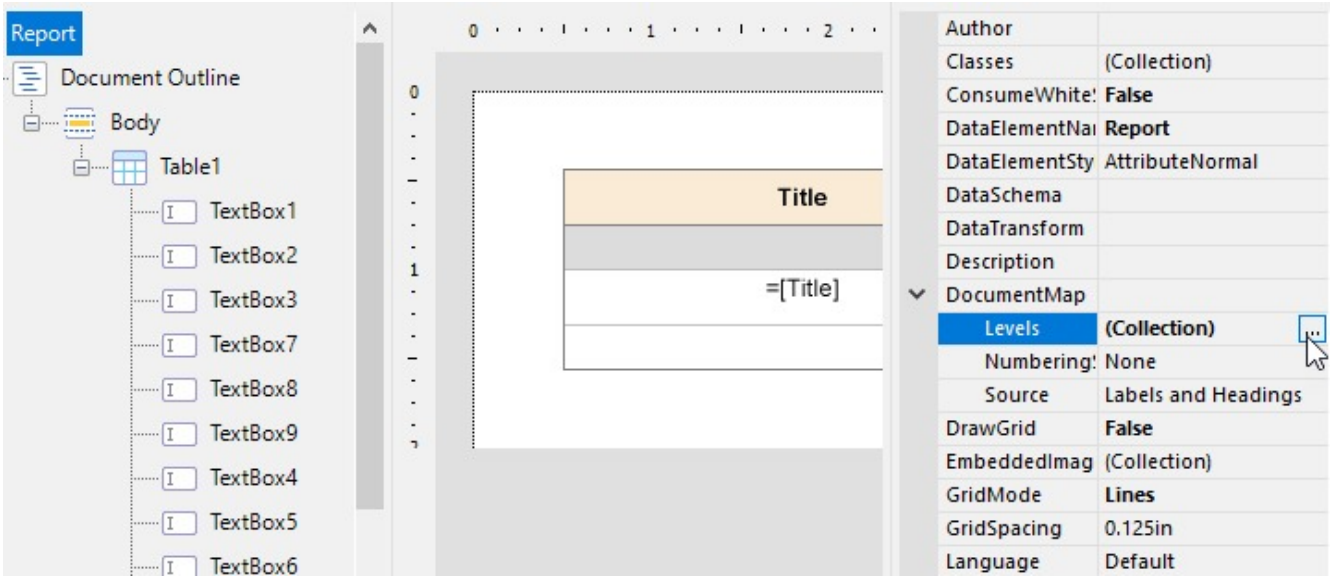
To set the numbering style for Document Map levels

You can the set the numbering style for all document map levels using the **Report dialog** or in the Properties Panel using the **DocumentMap** property.

1. In the Report Explorer select the **Report** item.
2. In the command section of the Properties Panel, click **Property dialog** to open the Report dialog.
3. In the Report dialog that appears, go to the **Document Map** page and set the **Source** property to one of the available value.
4. Set the **Numbering Style** to one of the available formats. The selected style would get applied to all the document levels and only to the selected source that you had set above.
5. Click **OK** to close the dialog.

You can also set separate numbering style to different document map levels using the **DocumentMapLevelDesigner Collection Editor** dialog.

1. In the Report Explorer select the **Report** item.
2. Go to the Properties Panel, click the **Levels (Collection)** property and then click the ellipsis button that appears.



3. In the **DocumentMapLevelDesigner Collection Editor** dialog that appears, consider the hierarchy of report bookmarks that appears in Document Map and add as many levels required using the **Add** button.
4. Select each level and set its **NumberingStyle** to any of the available format.
5. Click **OK** to close the dialog.

Note: Any customization made here gets directly applied to the Table of Contents control.

Report Appearance

Customize the appearance of reports by applying styles and themes, as described in this section:

- Themes
- Styles

Themes

Work with ActiveReports to create and add themes, customize and apply themes, use constant expressions in a theme, and much more. A theme is a collection of properties that defines the appearance of a report. A theme includes colors, fonts, images, and constant expressions that you can apply to report elements once you add a theme to a report.

You can add one or many themes to a report. If a report has multiple themes, you can use the report's **CollateBy ('CollateBy Enumeration' in the on-line documentation)** enumeration to control the page order in a report. See **Set Up Collation** for more information.

The Theme Editor and the **Report - Themes** dialog allow you to manage themes in a report.

The theme saved as an .rdlx-theme file on your local machine can be added in your report using the **Report – Themes** dialog. Or, you can add a dynamic theme using an expression.
Also, in the File menu, select **Open** to open and modify an existing theme and select **Save** or **Save As** to save the changes on your local machine.

To access the Report - Themes dialog

1. In the Designer, click the gray area around the report page to select a report.
2. Do one of the following:

- In the Properties window, select the **Themes** property and click the ellipsis (...) button to open the Report - Themes dialog.
- With the report selected, in the Properties window under properties where the commands are displayed, click the **Property dialog** link. In the Report dialog that appears, go to **Themes**.
- On the **Report** menu, select **Report Properties** and go to Themes in the Report dialog that appears.

Create and add themes

A theme is a collection of properties that defines the appearance of a report. A theme includes colors, fonts, images, and expressions that you can apply to report elements once you add a theme to a report.

You can add one or many themes to a report. If a report has multiple themes, you can use the report's **CollateBy ('CollateBy Property' in the on-line documentation)** property to control the page order in a report.

Use the following instructions to create and add themes.

Create a new theme

1. Open **Theme Editor**.
2. In the Theme Editor that opens, define the colors, fonts, images, and constant expressions properties for your new theme under the corresponding tabs.
3. On the **File** menu, select **Save**.
4. Choose a directory on your local machine and enter the name of a new theme, then click **Save**.

Add a theme to the report

1. In the Designer, click the gray area around the report page to select a report.
2. In the Properties window, select the **Themes** property and click the ellipsis (...) button to open the Report - Themes dialog.
3. In the Report - Themes dialog that opens, click the **Add...** icon.
4. From the **Value** drop-down, select <**Open File...**>.
5. In the Open dialog that appears, select a theme file from your local files, click **Open**, and then select **OK.**

Customize and apply themes

Use the following instructions to customize an existing theme and apply it to your report.

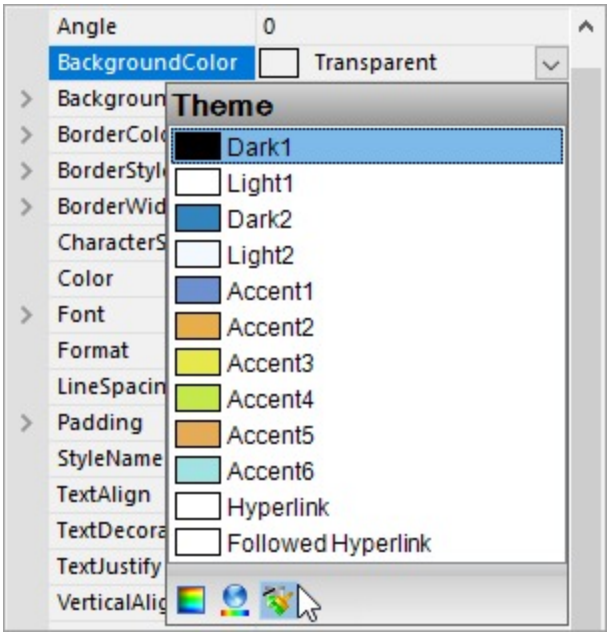
Modify a theme

1. In the designer, click the gray area around the report page to select a report.
2. In the Properties panel, select the **Themes** property and click the ellipsis (...) button to open the Report - Themes dialog.
3. In the Report - Themes dialog that opens, select an existing report theme.
4. Click the **Edit...** icon above the list of themes.
5. In the Theme Editor that opens, modify the theme properties and click **OK** to close the dialog.

Apply a theme color

1. In the designer, select the report's control (for example, a **TextBox**).
2. In the Properties panel, go to the color-related property (for example, the **BackgroundColor** property) and click the arrow to display the drop-down list of values.

3. In the list that appears, go to the **Theme** tab and select the color you want.



Apply a theme font

1. In the designer, select the report's control (for example, a **TextBox**).
2. In the Properties window, go to a property from the Font properties group (for example, the **Font** property) and click the arrow to display the drop-down list of values.
3. In the values list that appears, select a font defined in a theme (for example, **=Theme.Fonts!MinorFont.Family**).

Use Constant Expressions in a theme

In the Theme Editor, you can define constant expressions to be used in a theme. Later, you can apply a constant expression to the report's control by selecting it in the Value field of that control.

Constant expressions allow you to define a name and an associated value to be used in themes.

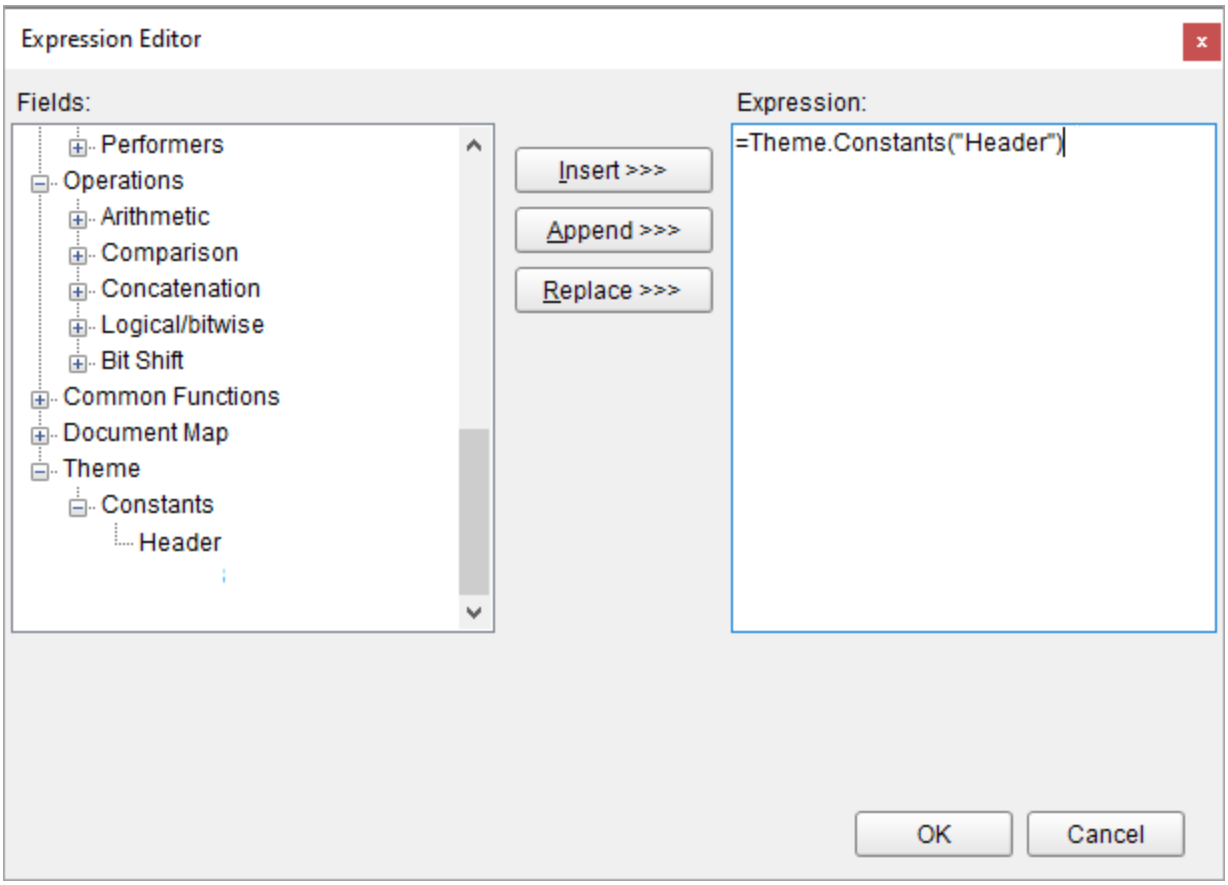
Use the following instructions to create and use constant expressions in themes.

Define a constant expression

1. In the Theme Editor, go to **Constants**.
2. Double-click the field under **Name** and enter the Constant name (for example, **Header**).
3. In the next field to the right, under **Value**, enter the Constant value (for example, **Invoice#**).

Use a constant expression

1. In the Designer, select the report's control (for example, a **TextBox**).
2. In the Properties Panel, go to the **Values** field and select the **<Expression>** option from the drop-down list to open the Expression Editor.
3. In the Expression Editor, expand the **Themes** node with the constant expressions defined in the report theme.
4. In the Themes node, select a constant and then click the **Replace** or **Insert** button.
5. Click **OK** to add the constant expression in the TextBox.



Use Dynamic Expressions in a theme

Dynamic expressions allow you to define themes based on some condition that is evaluated at run-time.

To add a dynamic expression in a theme

1. In the Designer, click the gray area around the report page to select a report.
2. In the Properties window, select the **Themes** property and click the ellipsis (...) button to open the **Report - Themes** dialog.
3. In the Report - Themes dialog that opens, click **Value** and then **<Expression..>** icon.
4. Enter the expression with dynamic condition and click **OK**.

See tutorial on how to Apply Theme at Runtime Using Dynamic Expression.

Set Up Collation

You can add multiple themes to a report. In this case, the report renders a combination of multiple outputs for each theme. For example, if a report has two themes, then the report output includes a combination of the first and the second themes, applied to each report page. You can control the combination rules of the report output in the **CollateBy** property.

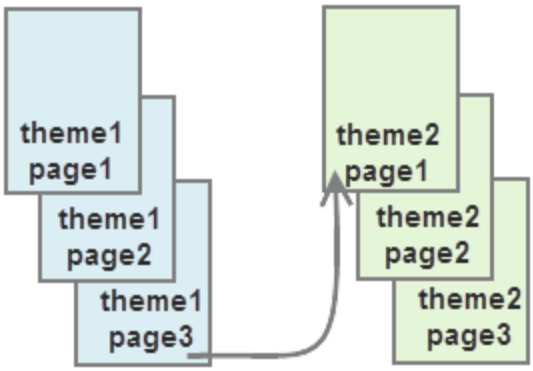
> ⚠ **Caution:** If you are using collation in a report, you cannot use interactive features, such as drill down, links,

> document map, and sorting.

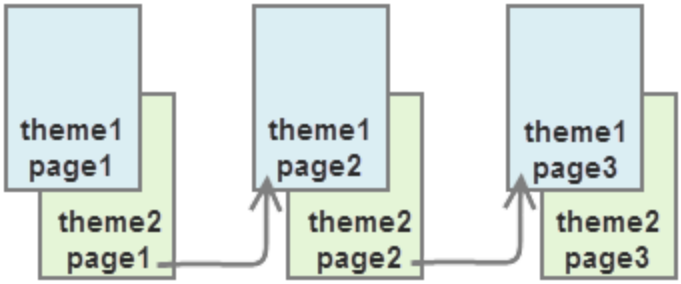
You can control the page order of a rendered report with multiple themes by selecting the collation mode in the **CollateBy** property of the report:

> **Note:** The collection of constant expressions must be the same in all themes of a report. See **Use Constant Expressions in a Theme** for further information.

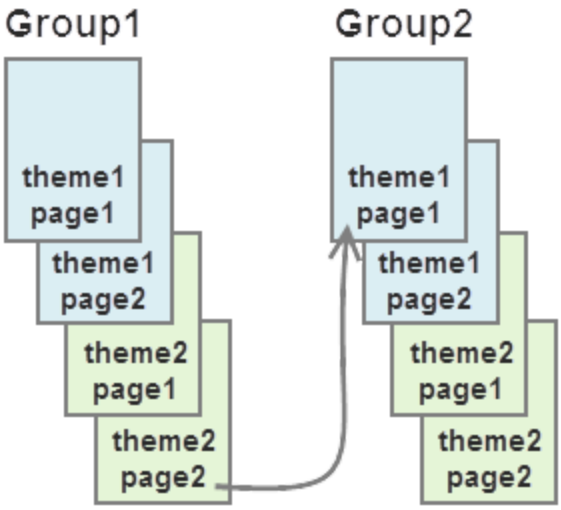
1. In the Designer, click the gray area around the report page to select the report.
2. In the Properties Panel, go to the **CollateBy** property and select one of the available options:
   - **Simple**. Renders report pages without any specific sorting. For example, if you have a report with 2 themes, the report renders all pages with theme 1, then all pages with theme 2.



   - **ValueIndex**. Sorts report pages by page number. For example, if you have a report with 2 themes, the report renders page 1 for theme 1 and 2, then page 2 for theme 1 and 2, and so on.



   - **Value**. Sorts report pages by the grouping expression that you specify in the report's FixedPage dialog. For example, if you have a report with 2 themes with grouping, the report renders group1 (pages 1 and 2 of theme1, then pages 1 and 2 of theme2), then group 2 (pages 1 and 2 of theme1, then pages 1 and 2 of theme2), and so on.



Note: In RDLX Reports, the **Value** collation mode is not available by design.

See Add Page Numbering for information on setting cumulative page count formats for $$Page Report$.

Styles

What are Styles?

Styles are a set of properties that you can apply to selected controls in your Page or RDLX reports to quickly change their appearance. A single style can define properties for font, background color, line spacing, border color, padding, and many more. You can create four different types of styles, namely **Common, Text, Table Of Contents** and **Table Of Contents Level**.

What are Style Sheets?

ActiveReports provides you with the ability to create multiple styles and store them in a style sheet. A style sheet can be understood as a collection of styles. You can add the style sheet to your Page or RDLX reports using the **StyleSheetSource ('StyleSheetSource Property' in the on-line documentation)** and **StyleSheetValue ('StyleSheetValue Property' in the on-line documentation)** properties and apply the styles to selected controls using the **StyleName ('StyleName Property' in the on-line documentation)** property. You can either embed the style sheet within your report or save it externally in the ***.rdlx-styles** format.

There are two ways to use these style sheets:

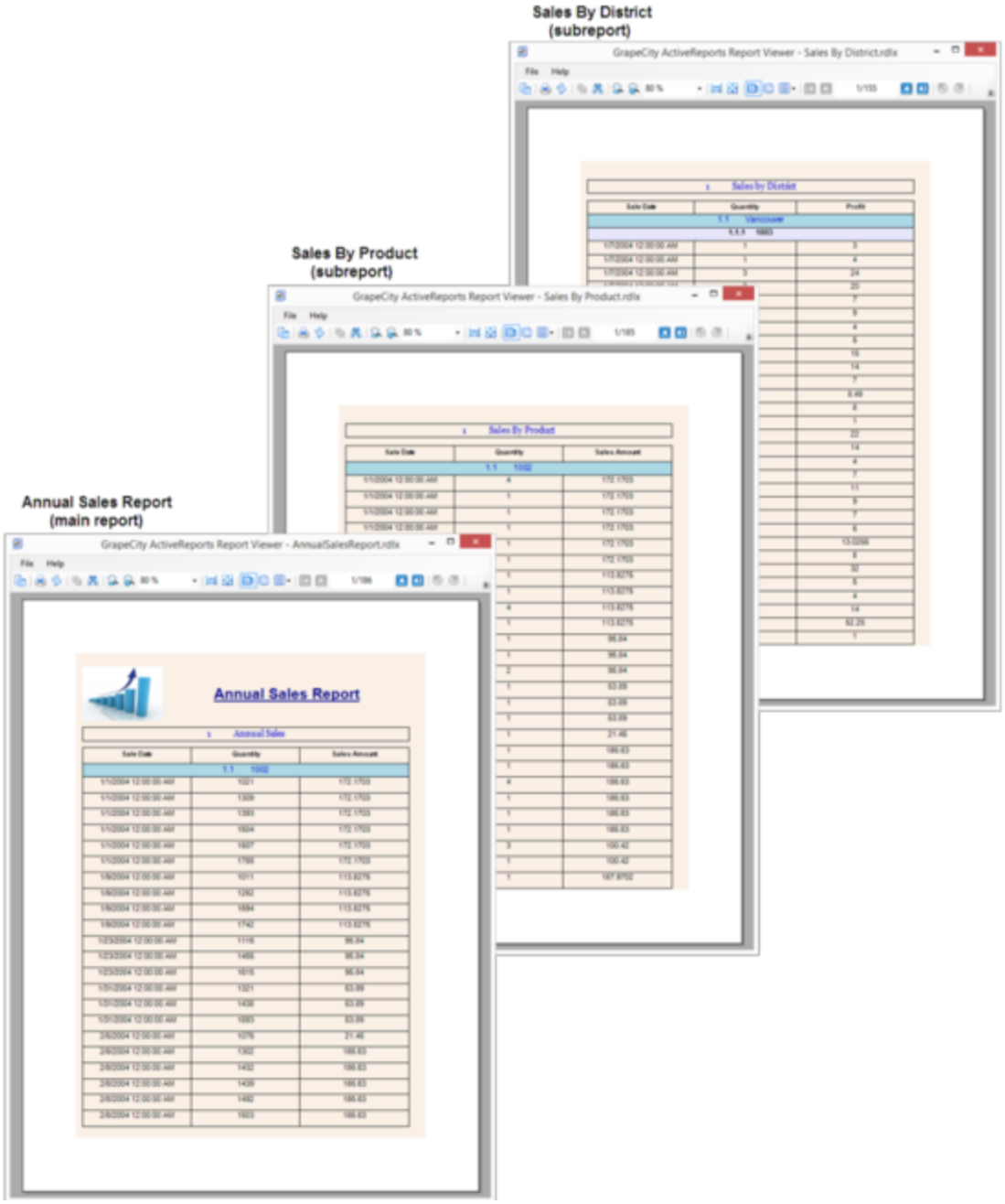
- Embed the style sheets within the report and use its styles on controls in that report
- Save the style sheets externally in ***.rdlx-styles** format and use it in multiple reports

Why use Styles?

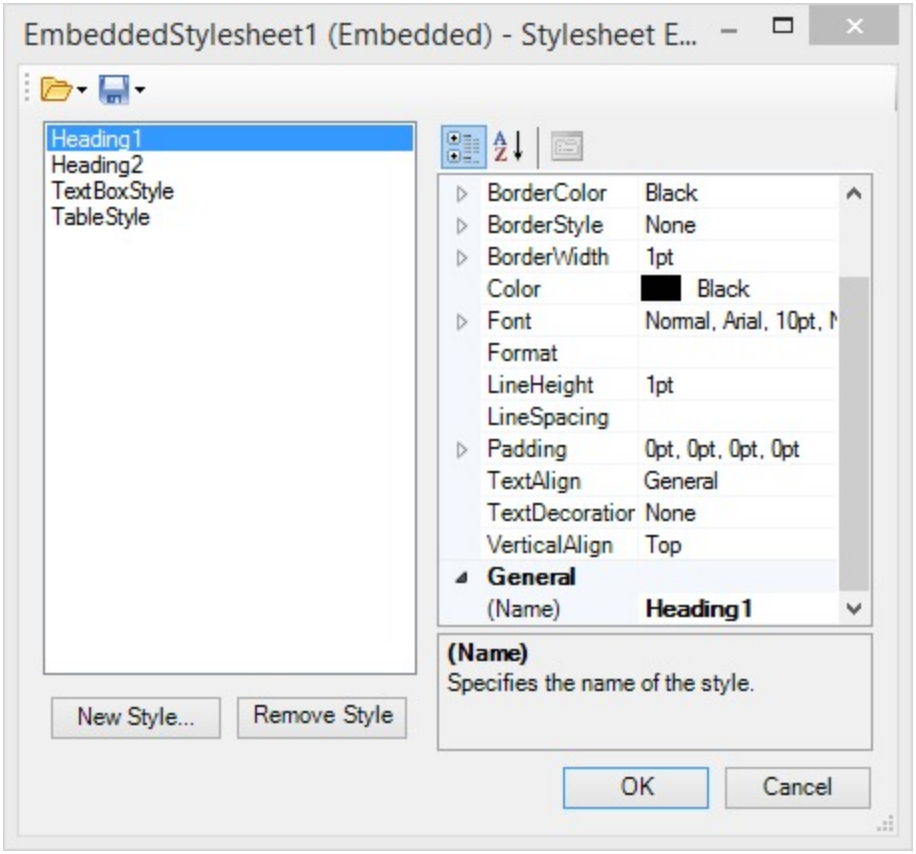
Using styles gives you more control over how you format the report. Let us look at a few scenarios to understand how styles are helpful while designing reports.

Reusing Styles

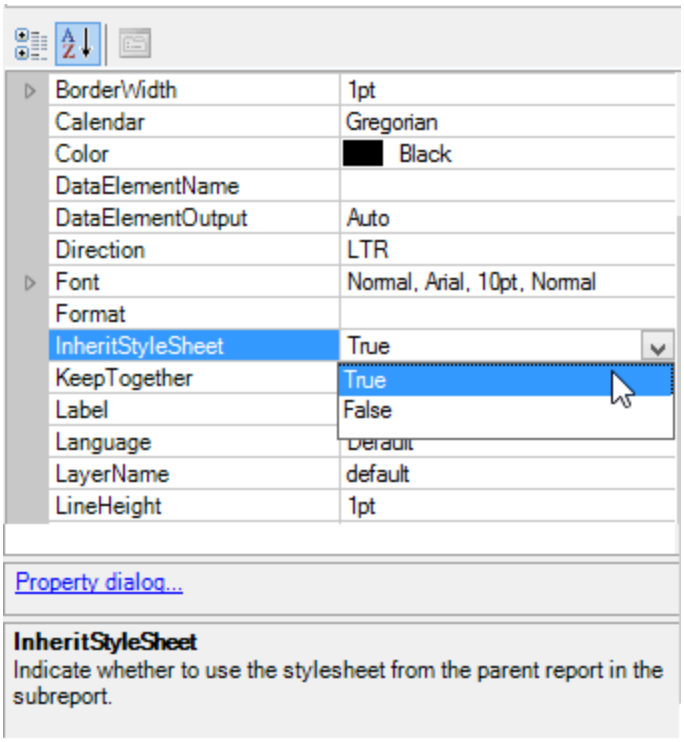
An organization wants to create an Annual Sales Report that consists of multiple subreports, representing Sales by District and Sales by Product. Since all the subreports are a part of the Annual Sales Report, the formatting needs to be consistent. You normally have to manually set properties for each control on the report to format it and then replicate those same set of properties for the two subreports. This can be time consuming and can lead to inconsistent styling in your reports. Let us see how the ActiveReports **Style** feature can help you generate consistent styles in all reports similar to the screenshot below.



Report author can create a style sheet for designing the Annual Sales Report (main report), add styles to the style sheet using the **Stylesheet Editor** dialog. After creating the styles, these styles can be applied to various controls on Annual Sales Report using the **StyleName ('StyleName Property' in the on-line documentation)** property.



Once the Annual Sales Report has been designed, the **InheritStyleSheet ('InheritStyleSheet Property' in the on-line documentation)** property can be set to True (by default) for each subreport.



By setting the **InheritStyleSheet** property to **True**, the style sheet used for the Annual Sales Report is automatically inherited in the subreports. This makes all the styles in the style sheet available to the two subreports. To apply the styles to the report controls in a subreport, you simply need to select the report control and specify the name of the style you want to use in the **StyleName** property.

In this example we can see how styles can help save time and maintain consistent formatting by giving you the flexibility to use the same style sheet in multiple subreports.

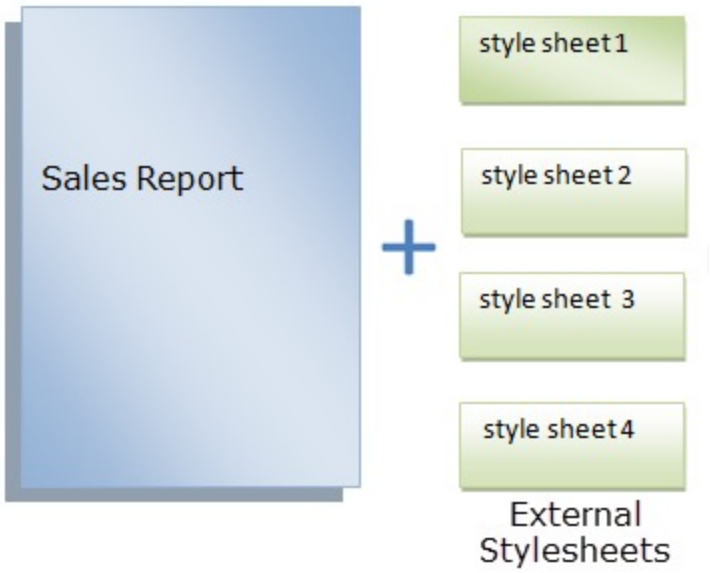
You can also use the same style sheet in multiple reports by saving the style sheets externally in ***.rdlx-styles** format.

Enhancing Report Portability

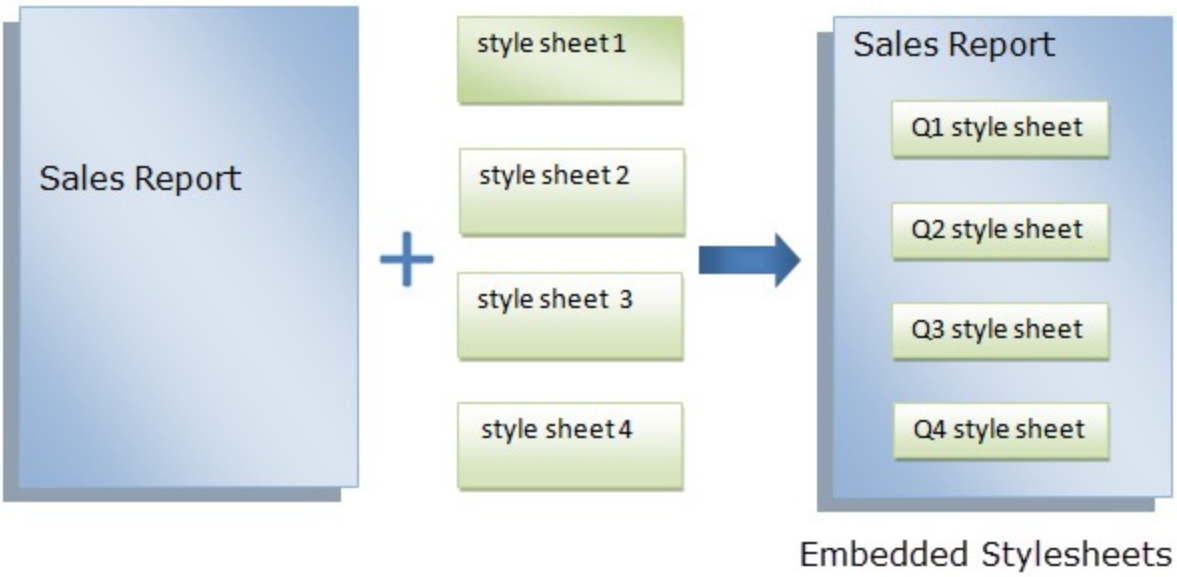
In ActiveReports, you can embed external style sheets in a report. This is particularly useful when you want to send reports that are styled using multiple style sheets. Let us take an example of a Sales Report to see how embedded style sheets can help improve the report portability.

Scenario

An organization wants to send a Sales report that is styled using 4 different external style sheets. While sending the styled report, 5 files have to be sent together, i.e. one report and 4 external style sheets. The person receiving these files needs to maintain and store 5 different files. Moreover, if the location of a style sheet is changed from the path set in the report, the style sheet will no longer be applied to the report until the path is modified in the report.



By embedding the external style sheets within the Sales report, only 1 file needs to be sent which in turn improves the portability of the report.



ActiveReports provides you the ability to create styles and store them in a style sheet. You can add these style sheets to your Page or RDLX reports and apply the styles to selected controls using the **StyleName** property. You can also save these style sheets on your system.

The styles feature consists of the following elements.

- **Style Sheet Properties**
- **Stylesheet Editor Dialog**
- **Add New Style Dialog**
- **Embed Stylesheet Dialog**
- **Open Embedded Stylesheet Dialog**

Style Sheet Properties

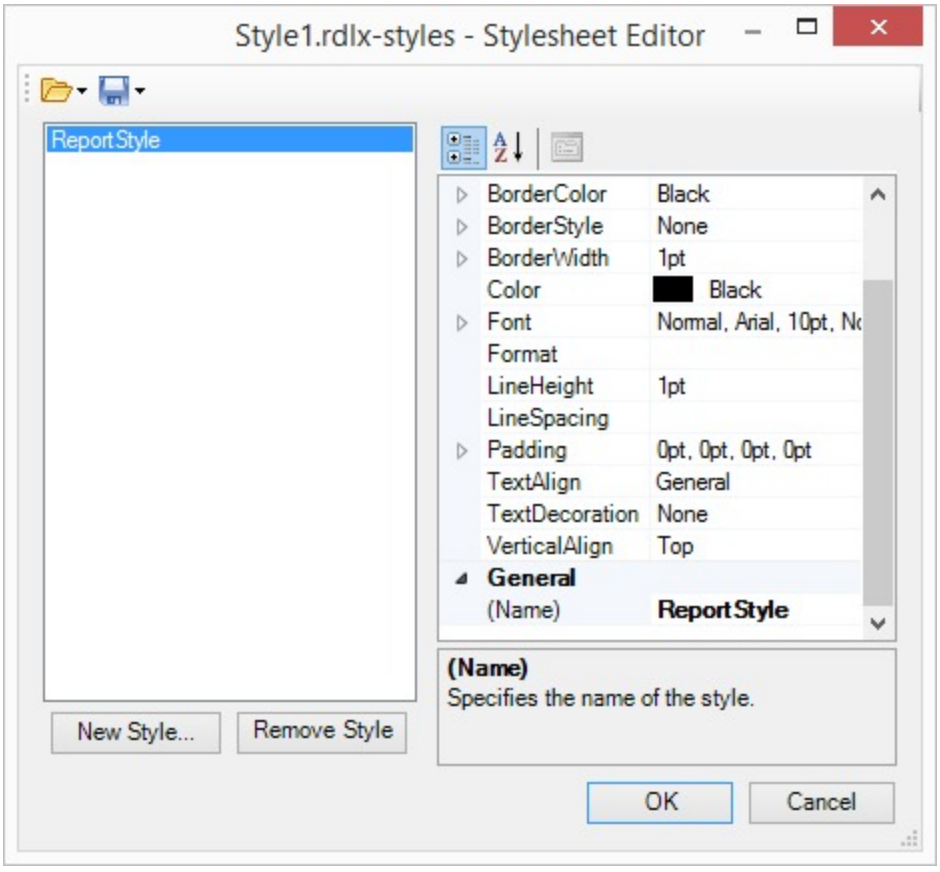
Define the style sheet of a report in the Properties panel using the **Source** property and the **Value** property. For subreports, use the InheritStyleSheet property.

Property Name	Description
Source	The source of a report's style sheet. You can choose from the following options: **External** - Choose this option if the style sheet (***.rdlx-styles** format) is located as an external source, such as a local file, an http location or a custom resource. To learn how to create external style sheets, see **Working with External Style Sheets**. **Embedded** - Choose this option if style sheet is embedded in the report. The embedded style sheets are displayed under the **Embedded StyleSheets** node of the Report Explorer.To learn how to create embedded style sheets, see **Working with Embedded Style Sheets**.
Value	The style sheet to apply to the report. You can choose from the following options:

	Expression - Opens the Expression Editor dialog to create a valid expression. **New** - Opens the **New Stylesheet Editor** dialog to create an external or embedded style sheet. **Open file** - Opens the **Open Stylesheet from file** dialog to navigate to a local style sheet file. This option is only available for external style sheets. For embedded style sheets, a list of available style sheets in the report is provided.
InheritStyleSheet	The style sheet to inherit in a subreport. Setting the **InheritStyleSheet** property to True (default value) inherits the style sheet of the main report in the subreport.

Note: Field values are not evaluated when used as an expression in Stylesheet Value, StyleName, and Styles properties.

Stylesheet Editor Dialog



You can open the **Stylesheet Editor** dialog by selecting the **Stylesheet Editor** option from the **Report** menu of the standalone designer or Visual Studio .NET designer.

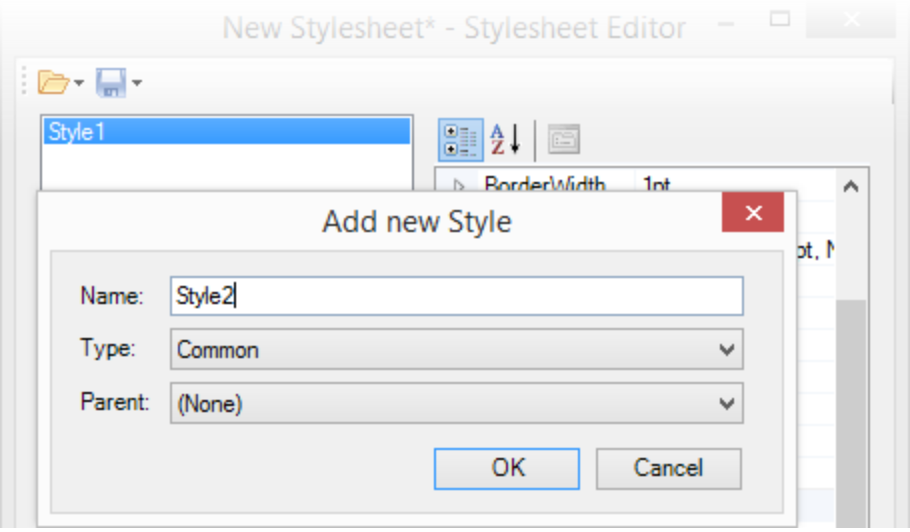
The Stylesheet Editor dialog consists of the following elements.

Elements	Description

Open Stylesheet from File	Opens a style sheet (***.rdlx-styles** format) located externally.
Open Embedded Stylesheet	Opens a style sheet embedded in the report.
Save Stylesheet to File	Saves the current style sheet as an external style sheet in ***.rdlx-styles** format.
Embed Stylesheet	Embeds the current style sheet in the report.
New Style	Creates a new style in the current style sheet.
Remove Style	Removes a style from the current style sheet.
Property panel	Modifies the properties of the selected style based on the selected style type. Available style properties change depending on the type of style selected. You set the style type when you create a new style. The style type is selected when a new style is created.
OK	Saves the current style.
Cancel	Closes the dialog without saving the changes.

Note: The values set in the Properties panel override the values defined in the report's style sheet. The overridden values are displayed in bold in the Properties panel.

Add New Style Dialog



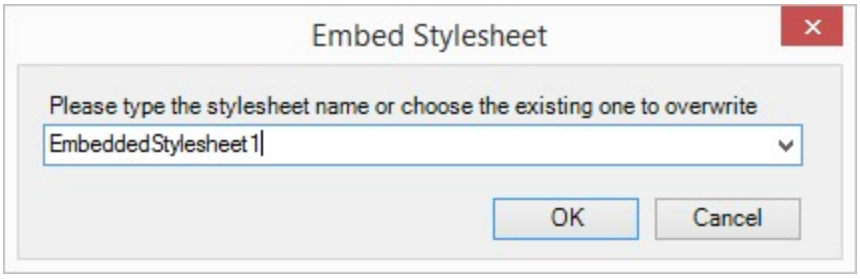
You can open the **Add New Style** dialog by clicking the **New Style** option in the **Stylesheet Editor** dialog.

The **Add New Style** dialog consists of the following elements.

Elements	Description
Name	Contains the name of the new style.
Type	Sets the type of control to which you can apply the style, which determines the options that are available in the **Properties panel** of the **Stylesheet Editor** dialog.

	Common Apply this style type to the following report controls: • CheckBox • Image • List • Tablix • Shape • Table • TableOfContents • TextBox Text Apply this style type to the **TextBox** report control. It includes all properties of the Common style type, plus it offers properties specific to the TextBox control. TOC Apply this style type to the **TableOfContents** control. TOC Level Apply this style type to the **ToC.Level** object of the TableOfContents control.
Parent	Represents the parent style of a new style. If the parent style is specified, the property values are taken from the selected parent style values. By default, the parent style is set to **None**.

Embed Stylesheet Dialog

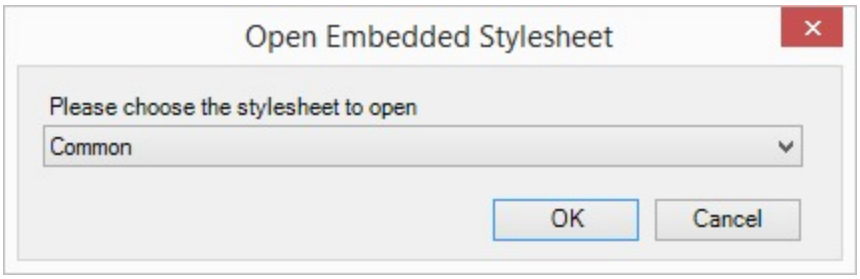


You can access the **Embed Stylesheet** dialog by selecting the **Save current Stylesheet** option and then selecting **Embed Stylesheet** in the **Stylesheet Editor** dialog.

The **Embed Stylesheet** dialog consists of the following elements.

Elements	Description
Drop-down list box for style sheet name	Enter a name for the embedded style sheet, or choose an existing style sheet from the drop-down list box to overwrite.

Open Embedded Stylesheet Dialog



You can access the **Open Embedded Stylesheet** dialog by selecting the **Open stylesheet** option and then selecting **Open embedded Stylesheet** from the **Stylesheet Editor** dialog.

The Open Embedded Stylesheet dialog consists of the following elements.

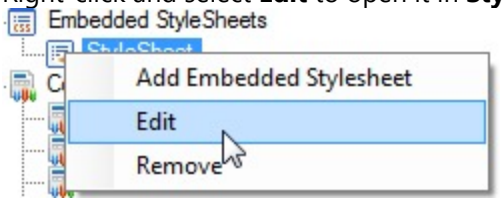
Elements	Description
Drop-down list box for opening style sheet	Provides a drop-down list box to choose an existing style sheet to overwrite. You can also enter a new name for the style sheet here.

Working with Styles within a Style Sheet

For any of these operations, you first need to open the style sheet in the editor.

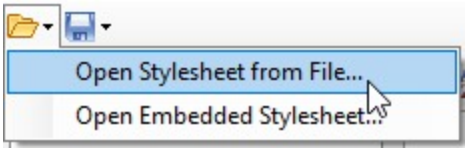
To open the editor for embedded style sheets

1. In the Report Explorer, expand the **Embedded StyleSheets** node and select the existing style sheet you want to edit.
2. Right-click and select **Edit** to open it in **Stylesheet Editor** dialog.



To open the editor for external style sheets

1. In the standalone designer or Visual Studio designer, click the Report menu and select the **Stylesheet Editor**.
2. In the Stylesheet Editor dialog, click the Open button and select the **Open Stylesheet from File** option.



3. In the **Open** dialog, navigate to the ***.rdlx-styles** file that you want to open.
4. Click **Open** to open the external stylesheet in the **Stylesheet Editor**.

To add a new style to a style sheet

1. In the **Stylesheet Editor** dialog, click the **New Style** button to add a new style.
2. In the **Add New Style dialog**, enter the **Name** of the style, and then select the **Type** and **Parent** style.

> **Tip**: For more information on the style types, see **Working with Styles**. To create style types for TableOfContents controls and heading levels, see **Apply styles to the TableOfContents control** and **Apply styles to the TableOfContents levels**.

To modify a style in a style sheet

1. In the **Stylesheet Editor**, select the existing style that you want to modify and use the property fields on the right to make the changes.
2. Click **OK** to save the changes.

To remove a style from a style sheet

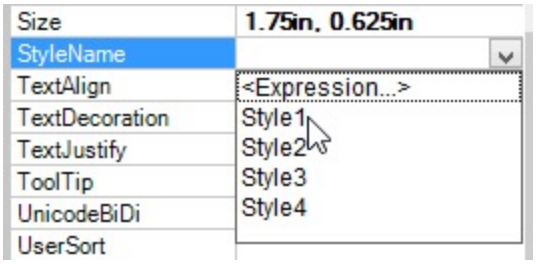
1. In the **Stylesheet Editor**, select the style that you want to remove and click **Remove Style.**
2. Click **OK** to save the changes.

To use a style from a style sheet at design time

1. Click the gray area around the report to select it, and under the Properties panel, click the **Property dialog** link in the Commands section. See Properties panel for more information on how to access commands.
2. In the Report dialog, go to the **Appearance** page.
3. In the **Appearance** page, set the Stylesheet Source to Embedded and in the Value field select an existing embedded style sheet. (Or select External and select the <Open File> option and navigate to an ***.rdlx-styles** external style sheet.)

> **Tip**: You can also access the **Source** and **Value** properties in the Properties panel by expanding the **Stylesheet** node.

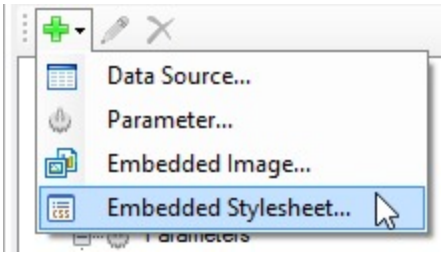
4. Click **OK** to close the dialog.
5. On the design surface, select the control you want to apply the style to.
6. In the Properties panel, from the **StyleName** property drop-down, select a style to apply to the controls.



Working with Embedded Style Sheets

Create and save a style sheet

1. In the Report Explorer, right-click the **Embedded StyleSheets** node, and select the **Add Embedded Stylesheet** option to access the **Stylesheet Editor** dialog.



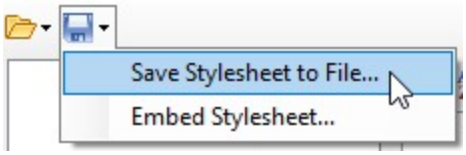
> **Tip**: You can also access the **Stylesheet Editor** dialog from the Report Explorer by clicking the Add button and selecting **Embedded Stylesheet**. In the standalone designer or Visual Studio designer, from the **Report** menu, select **Stylesheet** Editor.

2. Click the Save button and select **Embed Stylesheet** to embed the style sheet into the report.
3. Enter a name for the style sheet or choose an existing style sheet from the drop-down to overwrite, and then click **OK** to save the embedded style sheet.

All the saved style sheets embedded in the report appear under the **Embedded StyleSheets** node in the Report Explorer.

Save embedded style sheet as an external style sheet

1. In the Report Explorer, expand the **Embedded StyleSheets** node and select the embedded style sheet.
2. Right-click and select **Edit** to open the **Stylesheet Editor** dialog.
3. In the Stylesheet Editor dialog, click the Open button and select the **Save Stylesheet to file** option to save the embedded style sheet externally.



4. In the **Save As** dialog, navigate to the location where you want to save the style sheet, provide a name for the style sheet and click the **Save** button to save it as an external ***.rdlx-styles** file.

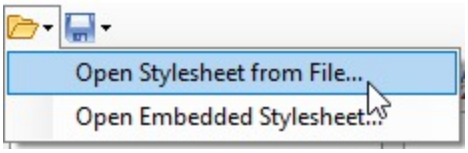
Working with External Style Sheets

Create and save a style sheet

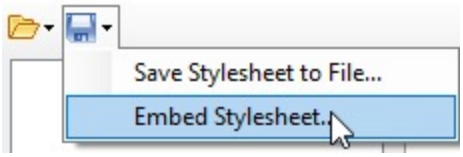
1. In the standalone designer or Visual Studio designer, click the **Report** menu and select the **Stylesheet Editor**.
2. In the Stylesheet Editor dialog, click the Open button and select the **Save Stylesheet to file** option.
3. In the **Save As** dialog, navigate to the location where you want to save the style sheet, provide a name for the style sheet and click the **Save** button to save it as an external ***.rdlx-styles** file.

Embed External style sheet into a report

1. In the standalone designer or Visual Studio .NET designer, click the Report menu and select the **Stylesheet Editor**.
2. In the Stylesheet Editor dialog, click the Open button and select the **Open Stylesheet from file** option.



3. In the Open dialog, navigate to the external style sheet **(*.rdlx-styles file)** that you want to load and click the **Open** button to load it in the Stylesheet Editor dialog.
4. In the Stylesheet Editor dialog, click the Save button and then select the **Embed Stylesheet** option.



5. In the **Embedded Stylesheet** dialog, enter a name for the style sheet and then click **OK** to embed the loaded style sheet into your report.

All the saved style sheets embedded in the report appear under the **Embedded StyleSheets** node in the Report Explorer.

Apply styles to the TableOfContents control

In the TableOfContents control, styles can be applied using the **StyleName** property.

1. Create a new style sheet and add styles that you want to apply to the TableOfContents control. For more information on creating style sheets and style types, see above sections.
2. Apply the style sheet to the report.
3. From the toolbox, drag and drop the **TableOfContents** control onto the report design surface, preferably at the start or end of the report layout to justify the significance of the control.
4. On the design surface, select the TableOfContents control.
5. In the Properties panel, from the **StyleName** property drop-down, select a style to apply to the TableOfContents control.

Apply styles to the TableOfContents levels

In the TableOfContents control, styles can be applied to each TableOfContents level using the **StyleName** property available in the **LevelDesigner Collection Editor** dialog.

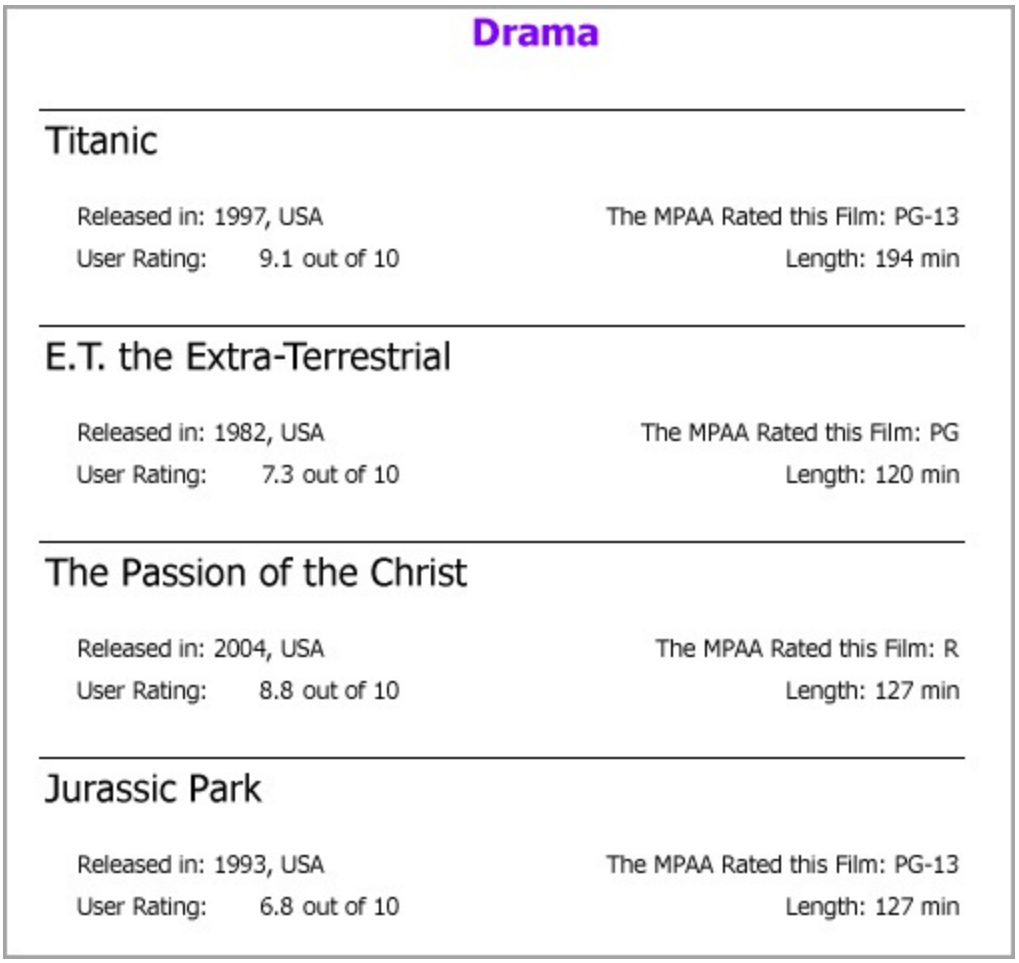
1. Create a new style sheet and add styles that you want to apply for the TableOfContents level. For more information on creating style sheets and the type of styles available for TableOfContents level, see above sections.
2. From the toolbox, drag and drop the **TableOfContents** control onto the report design surface, preferably at the start or end of the report layout to justify the significance of the control.
3. With the TableOfContents control selected, click the **Levels (Collection)** property from the Properties panel and then click the ellipsis button that appears.
4. In the **LevelDesigner Collection Edior** dialog that appears, select a TableOfContents level on which to want to apply the style.
5. From the list of properties on the right, drop-down the **StyleName** property to select a style to apply.

Tutorials: Report Controls in Page/RDLX Reports

This section takes you through the set of tutorials aimed at designing the Page and RDLX reports in the Report Designer from scratch.

Add Table of Contents to a Report with TableofContent Control

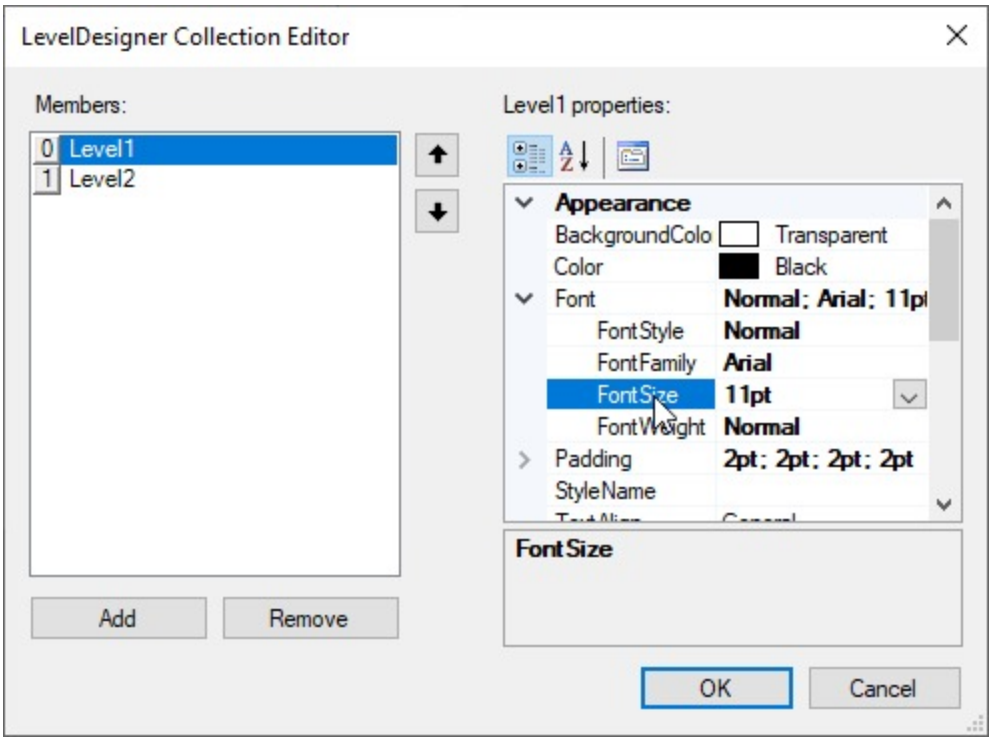
Let us add a Table of Contents to the following report. Note that this report is similar to the 'MoviesCatalog.rdlx' report available with samples on GitHub. The report contains nested List data regions to display the hierarchy in report. The parent list is grouped based on 'GenreName'. The child list is grouped based on the 'Title' and displays other information.



HeadingLevel property

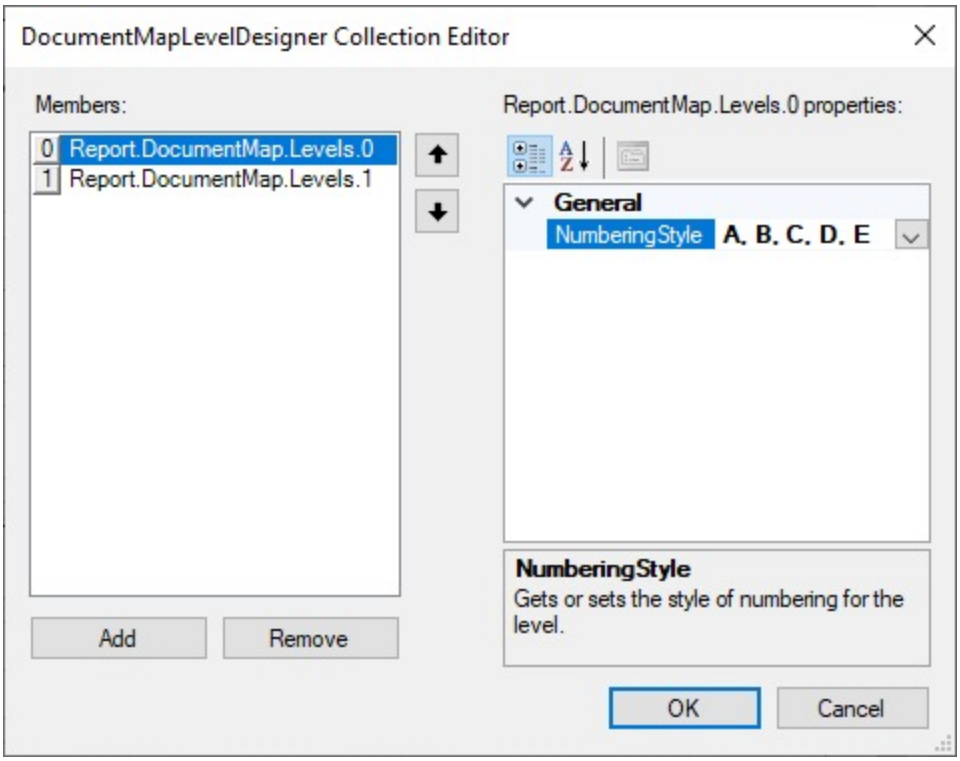
You can define a hierarchical structure for your report using the **Heading Level** property of the TextBox control. If a TextBox is set to Heading Level 1 and another TextBox at a different level to Heading Level 2, the entries in Heading Level 2 are clubbed under the Heading Level 1 entry in the Table of Contents. The headings are auto-numbered.

1. Drag and drop the TableOfContents control on top of the report, below the report title. Grab its corner and adjust its size.
2. With the TableOfContents control selected, in the Properties window, click the **Levels** property to open the **LevelDesigner Collection Editor**.
3. Select the Level1 (to display `[GenreName]`) in the Table of Contents and set the following properties:



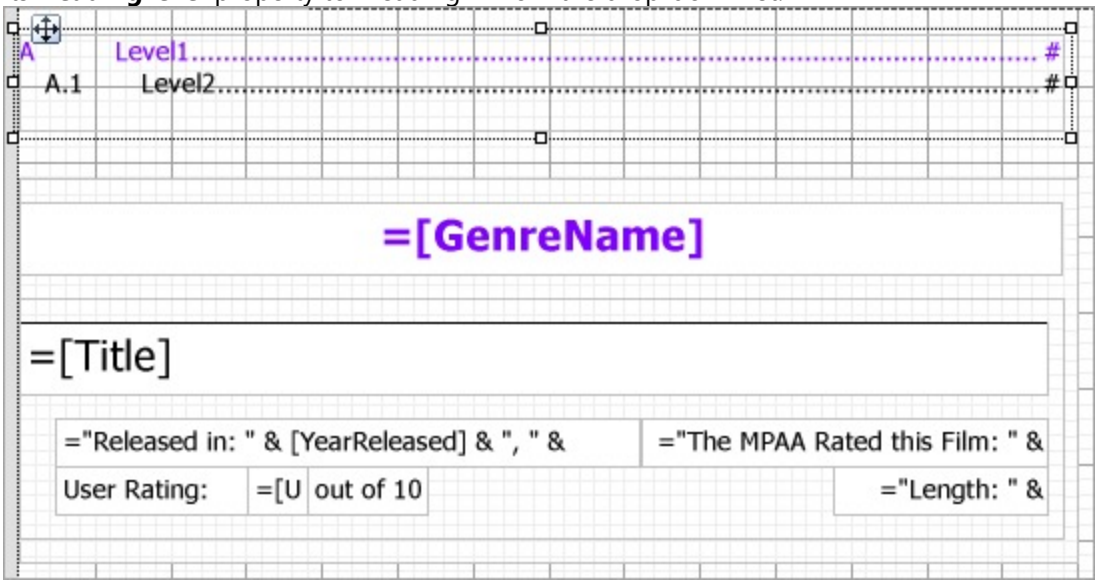
- **FontSize**: 11pt
- **FillCharacter**: "-" (a dash)
- **DisplayFillCharacter**: True

4. Click **+Add item** to add the Level2 (to display `[Title]`) to the Table of Contents and set the following properties:
   - **Padding > Left**: 5pt
   - **FillCharacter**: "." (a dot)
   - **DisplayFillCharacter**: True
5. From the Report Explorer, select Report, and from the Properties window, go to the **DocumentMap** property.
6. Expand the DocumentMap property and click the ellipses next to **Levels** (Collection) property.
7. In the **DocumentMapLevelDesigner Collection Editor,** that appears, select:

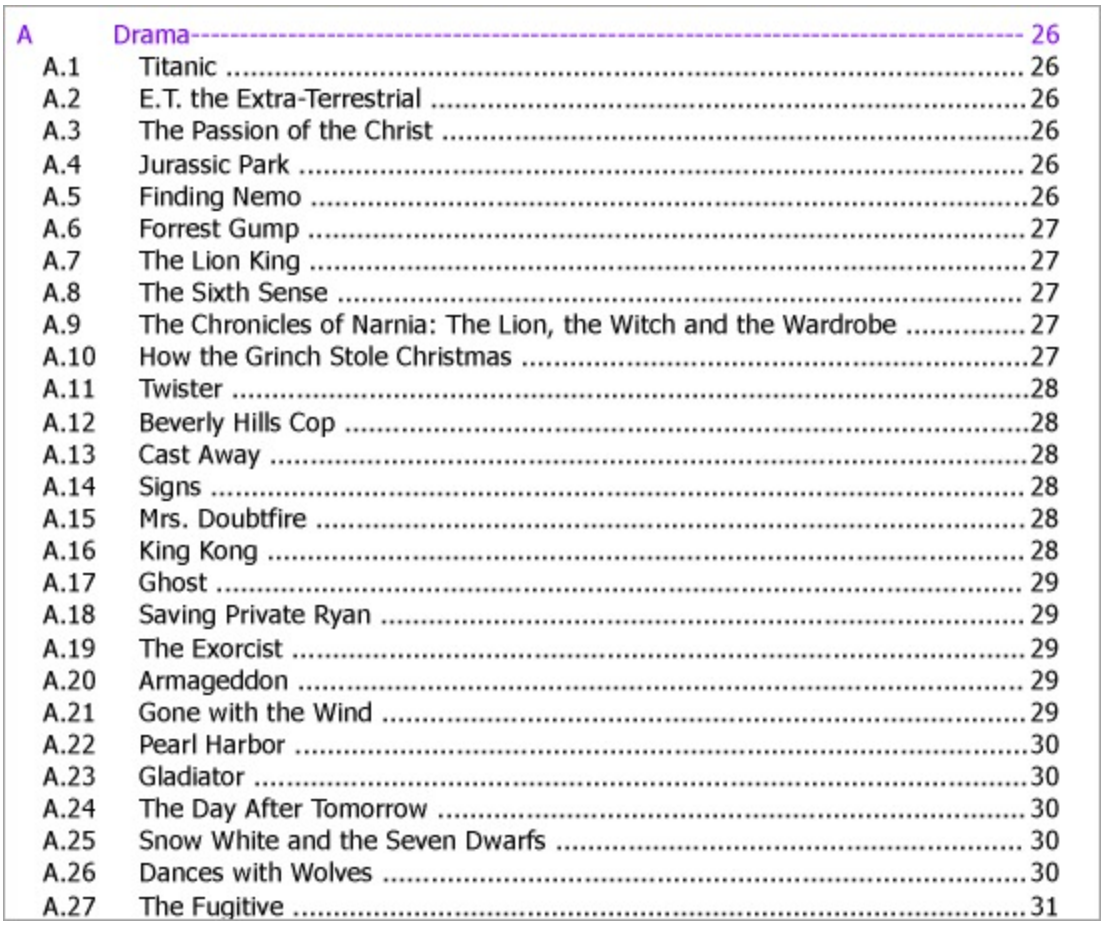


Report.DocumentMap.Levels.0 and set **NumberingStyle** to 'A, B, C, D, E'
Report.DocumentMap.Levels.1 and set **NumberingStyle** to '1, 2, 3, 4, 5'

8. Click OK to close the collection editor.
9. Set the **DocumentMap** > **Source** property to 'Headings Only'.
10. Now, select the Textbox with the **Value** property set to `=Fields!GenreName.Value` and, in the Properties window, set its **HeadingLevel** property to 'Heading 1' from the drop-down list.
11. Select the Textbox with the **Value** property set to `=Fields!Title.Value` and, in the Properties window, set its **HeadingLevel** property to 'Heading 2' from the drop-down list.



12. Preview the report. Here's how the table will look like. Click the TOC item to jump to the desired page.



Label Property

By setting the **Label** property of the controls in your report, you can show a hierarchical structure based on the parent-child relationship between the controls in the Document Map. For example, if the Label property of the List data region is set and a TextBox control is placed inside the List data region, the TextBox label appears nested inside the List data region label, thereby displaying a hierarchical structure.

1. Drag-drop the TableOfContents control on top of the report. Grab its corner and adjust its size.
2. Follow steps **2-8** in the Using HeadingLevel property section above for setting properties of levels in Table of Contents.
3. Set the **DocumentMap** > **Source** property to 'Labels Only'.
4. Select the List1 data region and set its **Label** property to `=Fields!Genre.Value` to have it displayed in the Level1 of the Table of Contents.
5. Similarly, select the List2 data region and set its **Label** property to `=Fields!Title.Value` to have it displayed in the Level2 of the Table of Contents.
6. Preview the report.

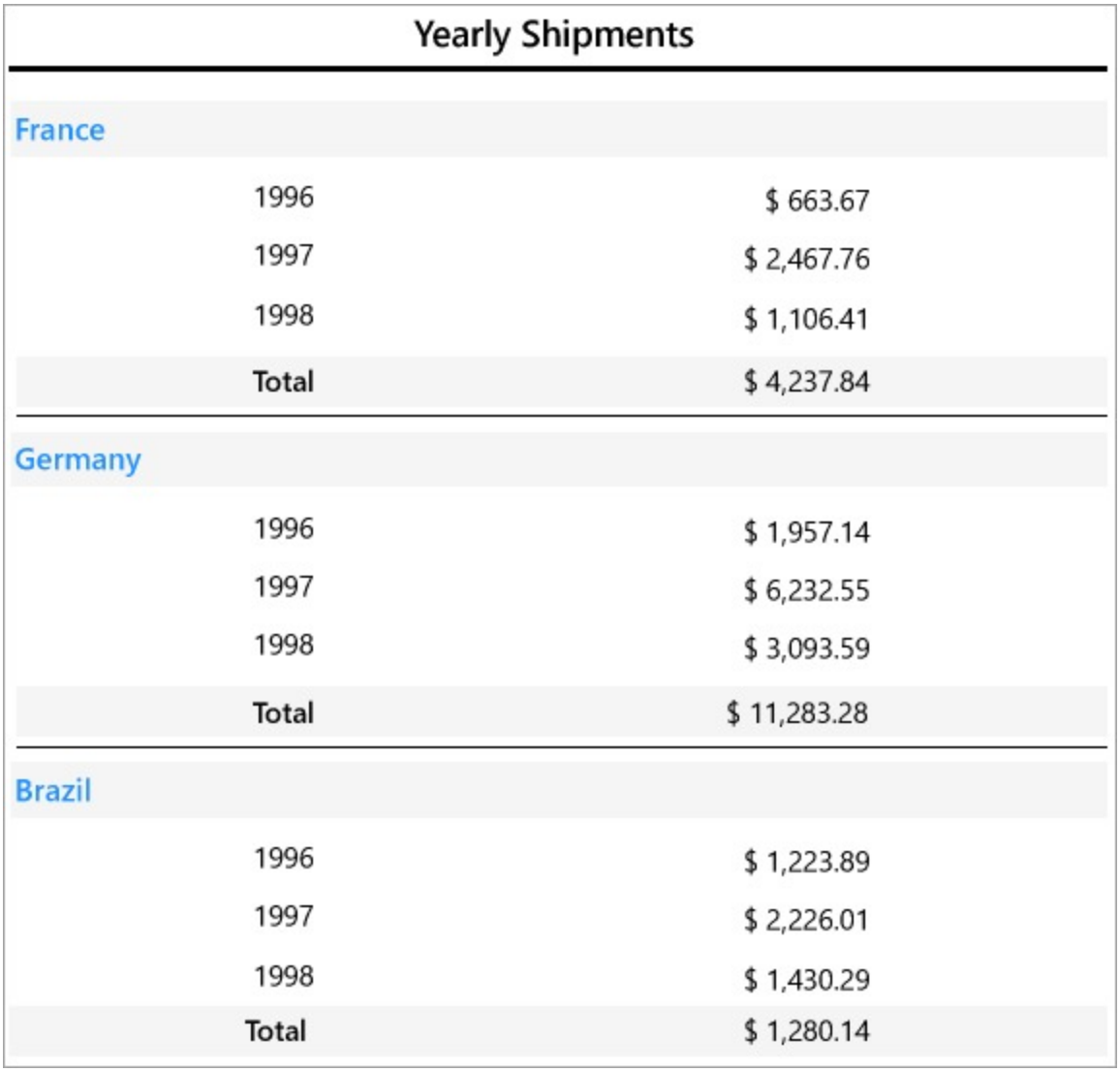
> Note: The Table of Contents mapped using labels are not numbered.

BandedList Data Region in Reports

Let us create a report that shows yearly shipments for each country. To display a summarized shipment value for each

country and year, we will use BandedList data region and add groups. The report connects to the 'Orders' JSON data available here.

The final report will look as shown.



Yearly Shipments	
France	
1996	$ 663.67
1997	$ 2,467.76
1998	$ 1,106.41
Total	$ 4,237.84
Germany	
1996	$ 1,957.14
1997	$ 6,232.55
1998	$ 3,093.59
Total	$ 11,283.28
Brazil	
1996	$ 1,223.89
1997	$ 2,226.01
1998	$ 1,430.29
Total	$ 1,280.14

Create a Report

In the ActiveReports Designer, create a new RDLX report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

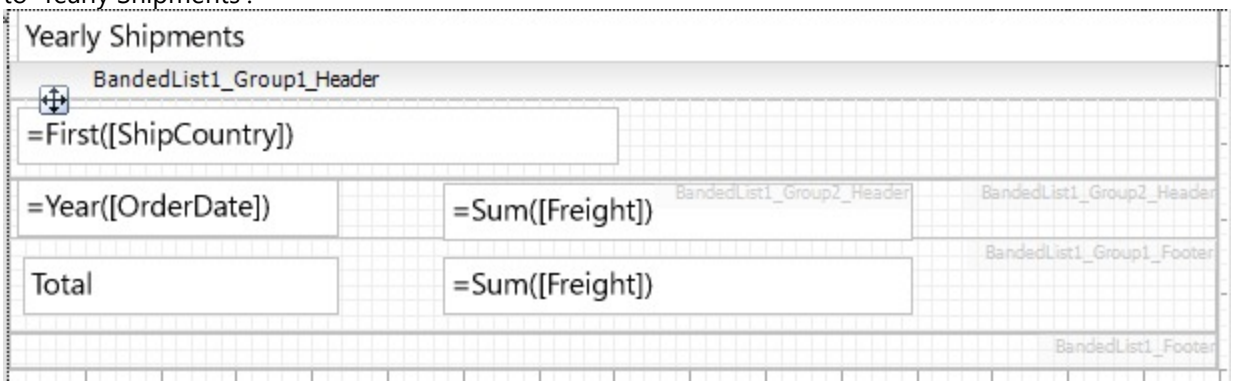
As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See JSON for details.

1. On the **Choose Data Source Type** screen of the wizard, select **JSON** and click **Next.**

2. To specify JSON **File Path**, click the **Browse** button and navigate to the desired file on your system. This report uses the 'orders.json' sample data source that can be downloaded from the following URL: https://demodata.mescius.io/northwind/odata/v1/Orders
3. Click the **Next** option and configure the dataset by adding a valid query. Enter OrderdsDataset into the **Name** field and select the node from the data tree in the **Path** section to generate the path. The resulting query is displayed in the **Query** field.
4. Click **Next** to proceed to the final screen of the Report Wizard.
5. Review the summary of the report and click **Finish** to successfully add the report with the JSON data source.

Design Report Layout

1. Drag and drop the **BandedList** control onto the report's designer.
2. With the BandedList control selected, click the **Property dialog** link to open the BandedList dialog.
3. Go to the **Groups** page and click Add.
4. Set the **Group on Expression** to `=Fields!ShipCountry.Value`.
   This will ensure that the data in the banded list is grouped based on the `[ShipCountry]` field.
5. Again, click Add to insert another group.
6. Set the **Group on Expression** to `=Year(Fields!OrderDate.Value)`.
   This will repeat the shipments by year for each country.
7. Click **OK** to close the dialog. Group Headers and a Group Footers are added.
8. From the dataset, drag the `[ShipCountry]` data field to the Group1 Header of the BandedList control.
9. Drag and drop two Textbox controls onto the Group2 Header of the control and set their **Value** property as follows:
   - TextBox1: `=Year(Fields!OrderDate.Value)`
   - TextBox2: `=Sum(Fields!Freight.Value)`
10. In the Properties pane, set the **Format** property for the `[Freight]` field to Currency.
11. Again, drag and drop two Textbox controls onto the Group1 Footer of the control and set their **Value** property as follows:
    - TextBox3: Total
    - TextBox4: `=Sum(Fields!Freight.Value)`
12. To add the report title, drag and drop the TextBox control to the PageHeader section and set its **Value** property to 'Yearly Shipments'.



13. Improve the appearance of the report and preview.

Bullet Control in Reports

Let us create a report that shows the sales relative to a target sale amount for each customer. We will use the Bullet control inside a Table data region to create this report. The report connects to the 'Sale' table from the 'Reels.db' data source available on GitHub.

Create a Report

In the ActiveReports Designer, create a new Page or an RDLX report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the **Report Explorer** and then selecting the **Add Data Source** option. See SQLite for details.

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next.**
2. To specify the **DataBase Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the Reels.db sample data source which can be downloaded from GitHub.

3. Click **Test Connection** to test the connection.
4. Then click the **Next** option and configure the dataset by adding a valid query.
5. Enter an SQL query like the following into the **Query** text box:

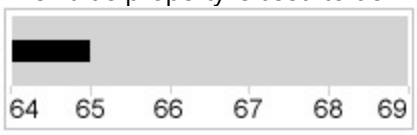
Dataset Query
`SELECT * FROM Sale ORDER BY Customer ASC`

See Query Builder in Microsoft SQL Client and OLEDB Providers for more information on building SQL queries.

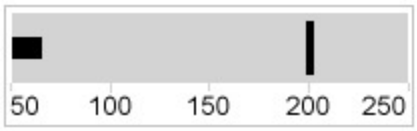
6. Click **Next** to proceed to the final screen of the Report Wizard.
7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.

Design Report Layout

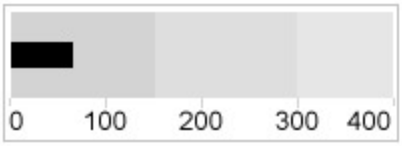
1. Drag and drop the **Table** data region onto the design area of the Report Designer.
2. Drag a **Bullet** control onto the Details row of the table and in the Properties window, set its **Value** property to `=Fields!SalesAmount.Value`
   This Value property is used to define the key measure displayed on the graph.



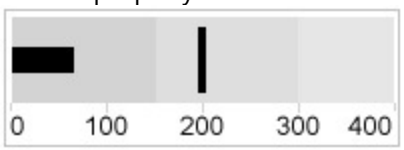
3. With the Bullet control selected on the design area:
   - Set its **Target Value** property to 200. This property defines a target for the Value to be compared to.



   - Set its **Best Value** property to 400 and the **Worst Value** property to 0. The Best Value and Worst Value properties define the value range on the graph.



   - You can also optionally encode the segments on the graph as qualitative ranges indicating bad, satisfactory, and good sections.
     - The **Range1Boundary** property defines a value for the bad/satisfactory boundary on the graph. Set this property to 150.
     - The **Range2Boundary** property defines a value for the satisfactory/good boundary on the graph. Set this property to 300.



   - You can also optionally define the **Interval** property for the graph value range. So, set this property to 100.
4. Specify the **TargetShape** as 'Line'.
5. Specify the position of **TickMarks** as 'Inside'.
6. Modify the appearance of the report. You can customize the appearance of bullets as follows:

1. Set the **TargetStyle** and the **TickStyle** properties.
2. Specify color patterns based on the relative value of the key measure. For example, set the expression like the following in the **ValueColor** property to show the color pattern based on the value of `[SalesAmount]`:

Expression for ValueColor Property
`=iif(Fields!SalesAmount.Value >= 200, "#BFECB8", "#FFB1AE")`

CheckBox Control in Reports

Let us create a report that displays information on Products that are 'Discontinued' in a Table data region. Since the value of 'Discontinued' is either True (1) or False (0), and a CheckBox control takes the Boolean value, we will use it to show the 'Tick' marks corresponding to the 'True' values. The report connects to the 'Products_header_tab.csv' data source available on GitHub.

The final report will look as shown.

Product Name	Unit Price	Product ID	Discontinued
Chai	$ 18.00	$ 1.00	☐
Chang	$ 19.00	$ 2.00	☐
Aniseed Syrup	$ 10.00	$ 3.00	☐
Chef Anton's Cajun Seasoning	$ 22.00	$ 4.00	☐
Chef Anton's Gumbo Mix	$ 21.35	$ 5.00	☑
Grandma's Boysenberry Spread	$ 25.00	$ 6.00	☐
Uncle Bob's Organic Dried Pears	$ 30.00	$ 7.00	☐
Northwoods Cranberry Sauce	$ 40.00	$ 8.00	☐
Mishi Kobe Niku	$ 97.00	$ 9.00	☑
Ikura	$ 31.00	$ 10.00	☐
Queso Cabrales	$ 21.00	$ 11.00	☐
Queso Manchego La Pastora	$ 38.00	$ 12.00	☐
Konbu	$ 6.00	$ 13.00	☐
Tofu	$ 23.25	$ 14.00	☐
Genen Shouyu	$ 15.50	$ 15.00	☐
Pavlova	$ 17.45	$ 16.00	☐
Alice Mutton	$ 39.00	$ 17.00	☑

Create a Report

In the ActiveReports Designer, create a new Page or an RDLX report.

If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access tht **Report Data Source** dialog by right-clicking the Data Sources node in the **Report Explorer** and then selecting the **Add Data Source** option. See CSV for details.

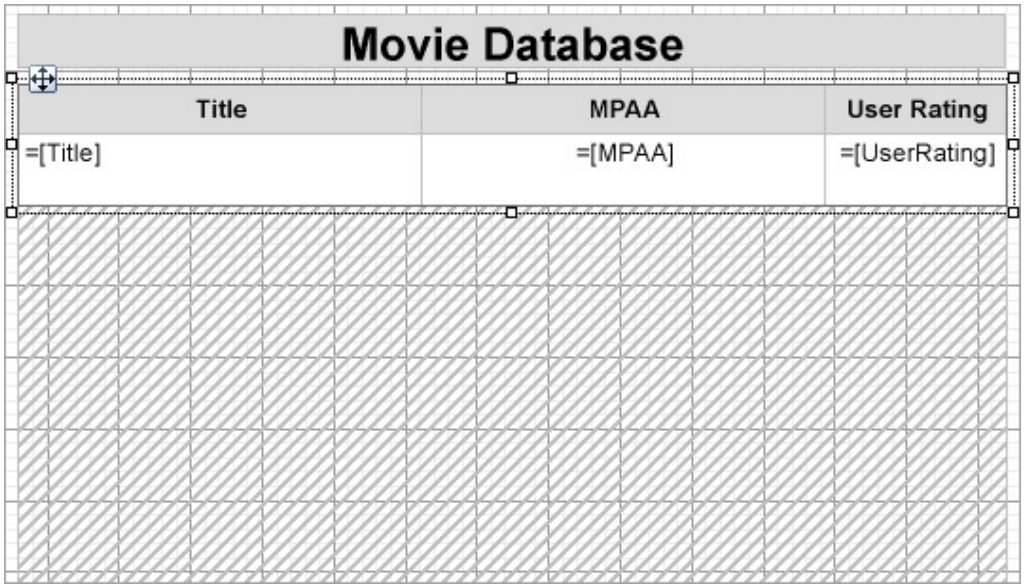
1. On the **Choose Data Source Type** screen of the wizard, select **CSV** and click **Next.**
2. To specify the **File Path**, click the **Browse** button and navigate to the desired folder on your system. For example, you can connect to the **Products_header_tab.csv** sample data source which can be downloaded from GitHub.
3. Select the **Column Delimiter** as 'Tab'.
4. Then click **Next** to check the **Fields** section, which includes field names with their corresponding data types present in the CSV file.
5. Click **Next** to proceed to the final screen of the Report Wizard.
6. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the CSV data source.
   See CSV Provider for more information on building SQL queries.

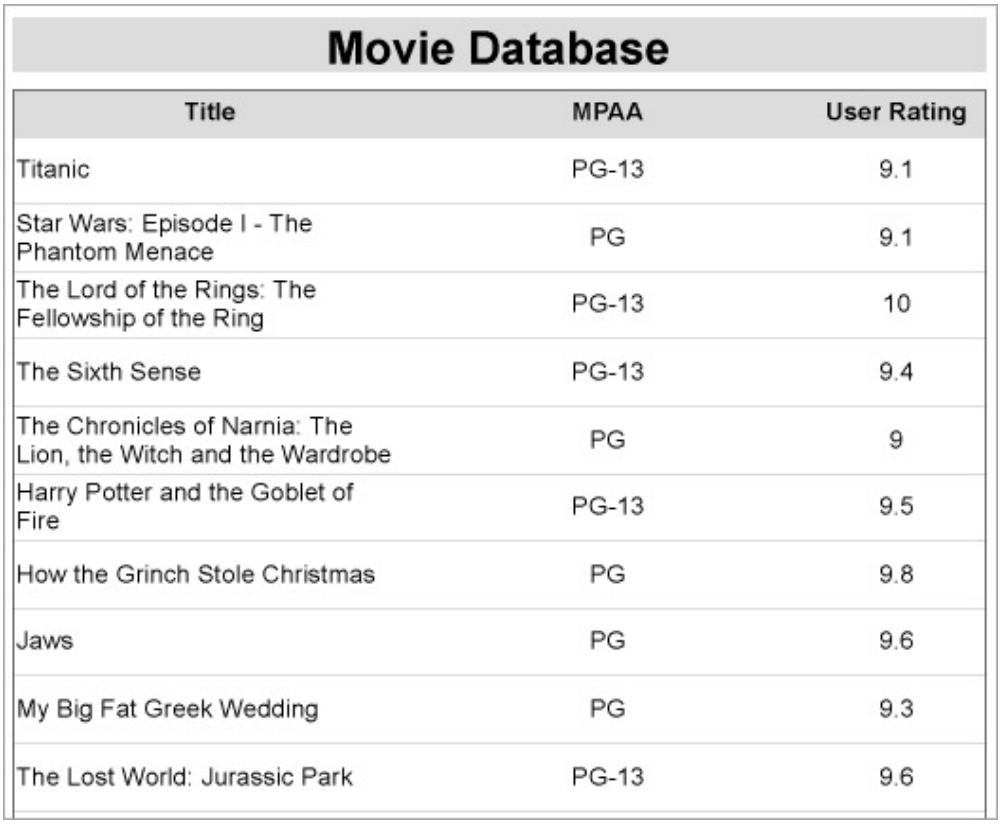
Design Report Layout

1. Drag and drop **Table** data region onto the design area of the Report Designer.
2. In the cells of the details row, drag-drop the `[ProductName]`, `[UnitPrice]`, and `[ProductID]fields`.
3. Add another column to the right of the `[ProductID]` column.
4. Drag-drop the **CheckBox** control onto the empty textbox in the details row of the added column.
5. With CheckBox selected, set the **Checked** property to the expression:
   `=ToBoolean(ToInt32(Fields!Discontinued.Value))`.
   Note that since we used the CSV data, the field `[Discontinued]`needs to be converted to boolean.
6. Preview the report.

Create Columnar Reports with OverflowPlaceholder Control

Let us create a report as shown below that uses the OverflowPlaceHolder control to create a columnar report. The report connects to the 'Movie' table from 'Reels.db' data source on GitHub.





Movie Database

Title	MPAA	User Rating
Titanic	PG-13	9.1
Star Wars: Episode I - The Phantom Menace	PG	9.1
The Lord of the Rings: The Fellowship of the Ring	PG-13	10
The Sixth Sense	PG-13	9.4
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	PG	9
Harry Potter and the Goblet of Fire	PG-13	9.5
How the Grinch Stole Christmas	PG	9.8
Jaws	PG	9.6
My Big Fat Greek Wedding	PG	9.3
The Lost World: Jurassic Park	PG-13	9.6

We will use multiple Overflow Placeholders to create the report layout with multiple columns. We will link the Table data region to an Overflow Placeholder control, followed by linking one Overflow Placeholder control to another to make data flow in case the data does not fit into these placeholders.

Create Report

In the ActiveReports Designer, create a new Page report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See SQLite for details.

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next.**
2. To specify the **DataBase Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the Reels.db sample data source which can be downloaded from GitHub.
3. Click **Test Connection** to test the connection.
4. Then click the **Next** option and configure the dataset by adding a valid query.

Dataset Query
`SELECT * FROM Movie`

5. Click **Next** to proceed to the final screen of the Report Wizard.
6. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.

Design Report Layout

1. On the Page1 of the report, drag-drop a **Table** data region.
2. In the cells of the details row, drag-drop the `[Title]`, `[MPAA]`, and the `[UserRating]` fields. Our table now displays the 'Movie' details – Title, MPAA, and User Rating fields.

Link the Table data region to an Overflow Placeholder

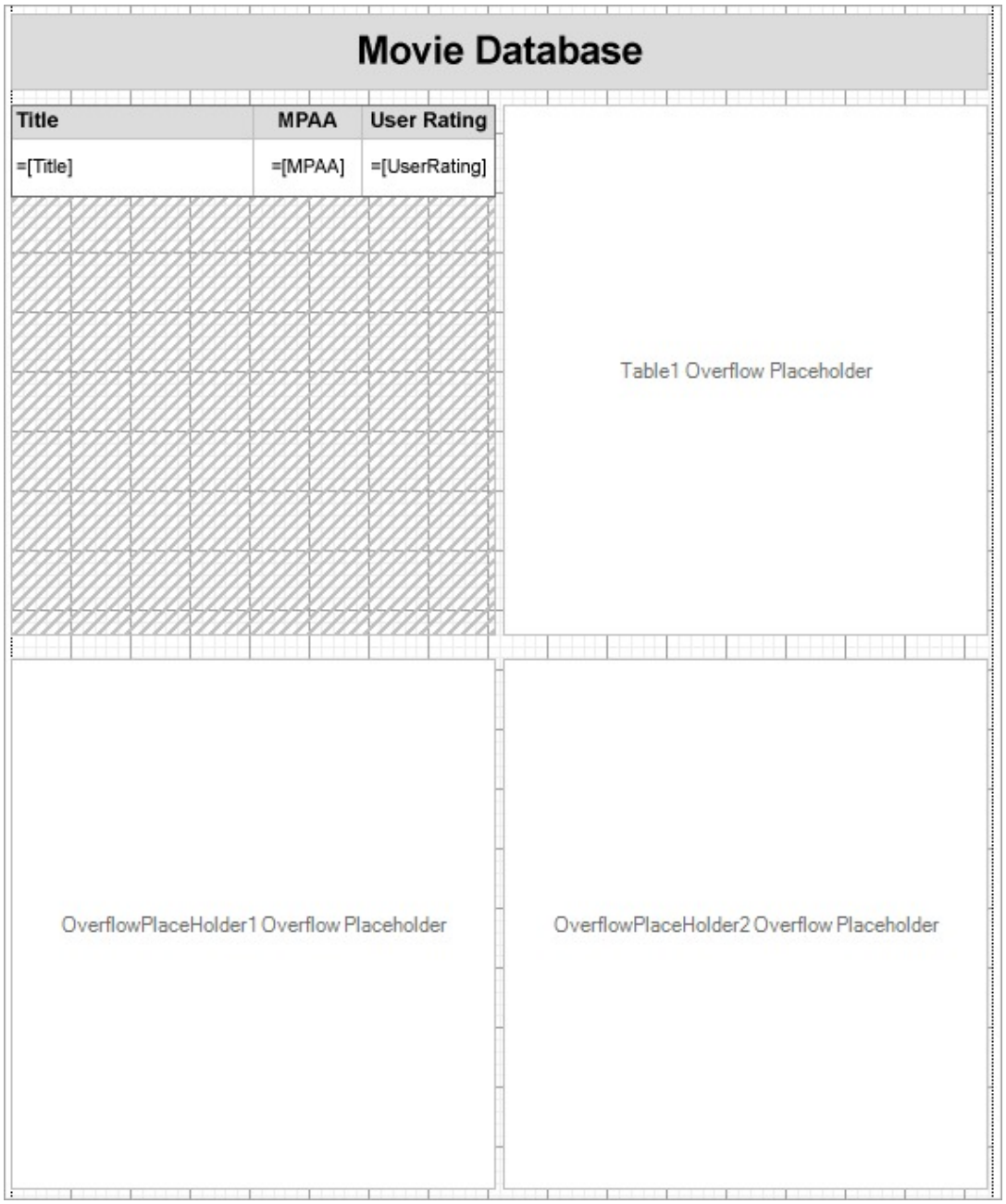
1. Select the Table data region and from the Properties window, set the following properties to some values, e.g.
   - **Location** = 0in,1in
     (we want the table to appear on the left side and make space for other Overflow Placeholder controls)
   - **RepeatHeaderOnNewPage** = True
     (for the table header to repeat on each page as well as inside the Overflow Placeholder)
   - **Size** = 3.2in, 0.6in
   - **FixedSize** = 3.2in, 3.5in
   - **OverflowName** = OverFlowPlaceHolder1 (you can set this property only after you add the OverflowPlaceHolder1 to the design surface, see **step 2**)

Link an Overflow Placeholder to another Overflow Placeholder

2. Drag and drop the **OverflowPlaceHolder1** control onto the design surface on the right side of the table and from the Properties window, set its properties as follows.
   - **Location** = 3.25in, 1in
   - **Size** = 3.2in, 3.5in (same as FixedSize of the table)
   - **OverflowName** = OverFlowPlaceHolder2 (you can set this property only after you add the OverflowPlaceHolder2 to the design surface, see **step 3**).
3. Drag and drop the **OverflowPlaceHolder2** control onto the design surface and, in the Properties window, set its properties as follows.
   - **Location** = 0in, 4.65in
   - **Size** = 3.2in, 3.5in
   - **OverflowName** = OverFlowPlaceHolder3 (you can set this property only after you add the OverflowPlaceHolder3 to the design surface, see **step 4**).
4. Drag and drop the **OverflowPlaceHolder3** control onto the design surface and, in the Properties window, set its properties as follows.
   - **Location** = 3.25in, 4.65in

- **Size** = 3.2in, 3.5in

The final report in the design view will look similar to this.



5. Preview the report. Observe how the report data is arranged.

Movie Database

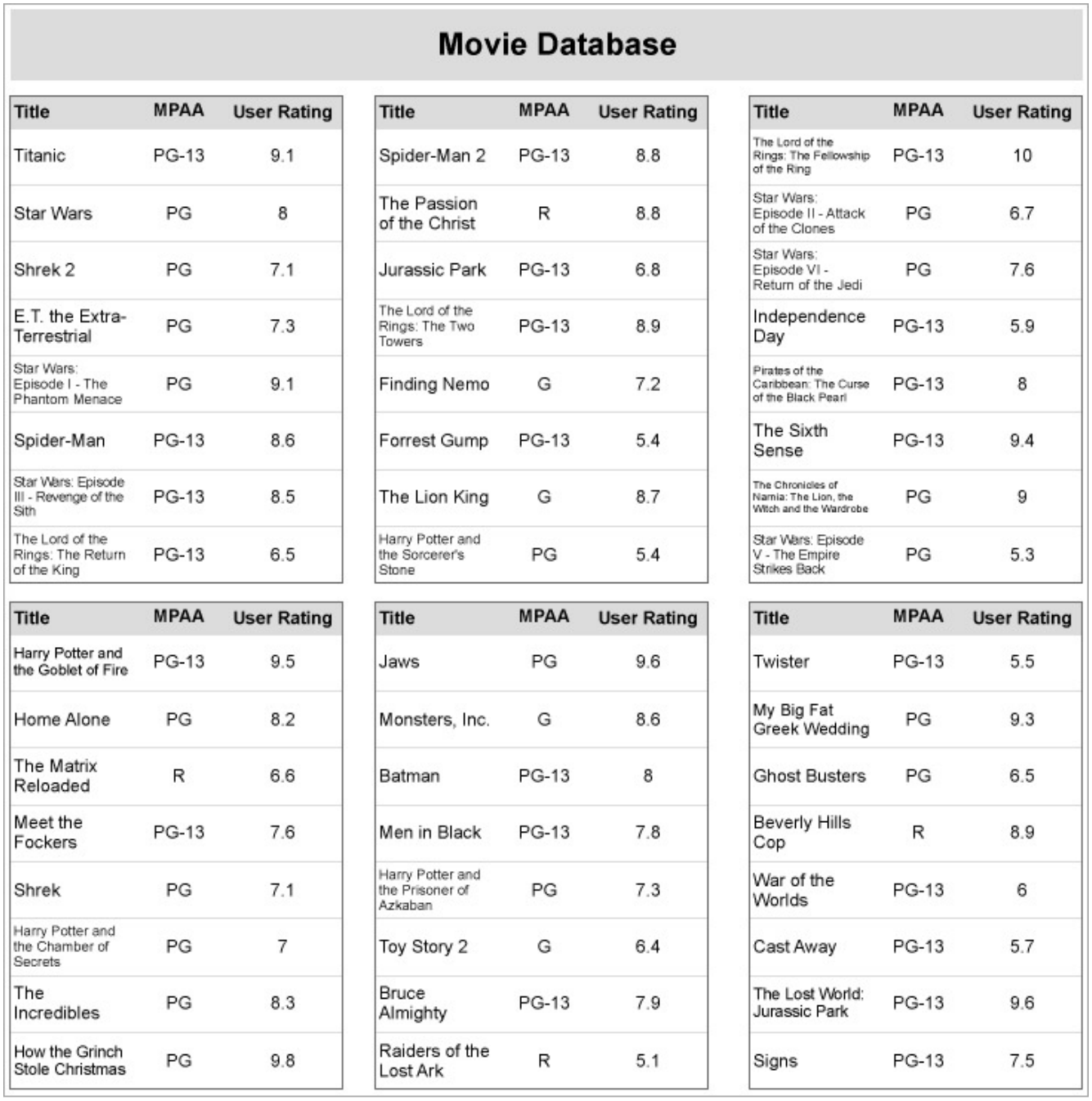
Title	MPAA	User Rating
Titanic	PG-13	9.1
Star Wars	PG	8
Shrek 2	PG	7.1
E.T. the Extra-Terrestrial	PG	7.3
Star Wars: Episode I - The Phantom Menace	PG	9.1
Spider-Man	PG-13	8.6
Star Wars: Episode III - Revenge of the Sith	PG-13	8.5
The Lord of the Rings: The Return of the King	PG-13	6.5

Title	MPAA	User Rating
Spider-Man 2	PG-13	8.8
The Passion of the Christ	R	8.8
Jurassic Park	PG-13	6.8
The Lord of the Rings: The Two Towers	PG-13	8.9
Finding Nemo	G	7.2
Forrest Gump	PG-13	5.4
The Lion King	G	8.7
Harry Potter and the Sorcerer's Stone	PG	5.4

Title	MPAA	User Rating
The Lord of the Rings: The Fellowship of the Ring	PG-13	10
Star Wars: Episode II - Attack of the Clones	PG	6.7
Star Wars: Episode VI - Return of the Jedi	PG	7.6
Independence Day	PG-13	5.9
Pirates of the Caribbean: The Curse of the Black Pearl	PG-13	8
The Sixth Sense	PG-13	9.4
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	PG	9
Star Wars: Episode V - The Empire Strikes Back	PG	5.3

Title	MPAA	User Rating
Harry Potter and the Goblet of Fire	PG-13	9.5
Home Alone	PG	8.2
The Matrix Reloaded	R	6.6
Meet the Fockers	PG-13	7.6
Shrek	PG	7.1
Harry Potter and the Chamber of Secrets	PG	7
The Incredibles	PG	8.3
How the Grinch Stole Christmas	PG	9.8

Similarly, you can add more Overflow Placeholders to create layout as shown:



Movie Database

Title	MPAA	User Rating
Titanic	PG-13	9.1
Star Wars	PG	8
Shrek 2	PG	7.1
E.T. the Extra-Terrestrial	PG	7.3
Star Wars: Episode I - The Phantom Menace	PG	9.1
Spider-Man	PG-13	8.6
Star Wars: Episode III - Revenge of the Sith	PG-13	8.5
The Lord of the Rings: The Return of the King	PG-13	6.5

Title	MPAA	User Rating
Spider-Man 2	PG-13	8.8
The Passion of the Christ	R	8.8
Jurassic Park	PG-13	6.8
The Lord of the Rings: The Two Towers	PG-13	8.9
Finding Nemo	G	7.2
Forrest Gump	PG-13	5.4
The Lion King	G	8.7
Harry Potter and the Sorcerer's Stone	PG	5.4

Title	MPAA	User Rating
The Lord of the Rings: The Fellowship of the Ring	PG-13	10
Star Wars: Episode II - Attack of the Clones	PG	6.7
Star Wars: Episode VI - Return of the Jedi	PG	7.6
Independence Day	PG-13	5.9
Pirates of the Caribbean: The Curse of the Black Pearl	PG-13	8
The Sixth Sense	PG-13	9.4
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	PG	9
Star Wars: Episode V - The Empire Strikes Back	PG	5.3

Title	MPAA	User Rating
Harry Potter and the Goblet of Fire	PG-13	9.5
Home Alone	PG	8.2
The Matrix Reloaded	R	6.6
Meet the Fockers	PG-13	7.6
Shrek	PG	7.1
Harry Potter and the Chamber of Secrets	PG	7
The Incredibles	PG	8.3
How the Grinch Stole Christmas	PG	9.8

Title	MPAA	User Rating
Jaws	PG	9.6
Monsters, Inc.	G	8.6
Batman	PG-13	8
Men in Black	PG-13	7.8
Harry Potter and the Prisoner of Azkaban	PG	7.3
Toy Story 2	G	6.4
Bruce Almighty	PG-13	7.9
Raiders of the Lost Ark	R	5.1

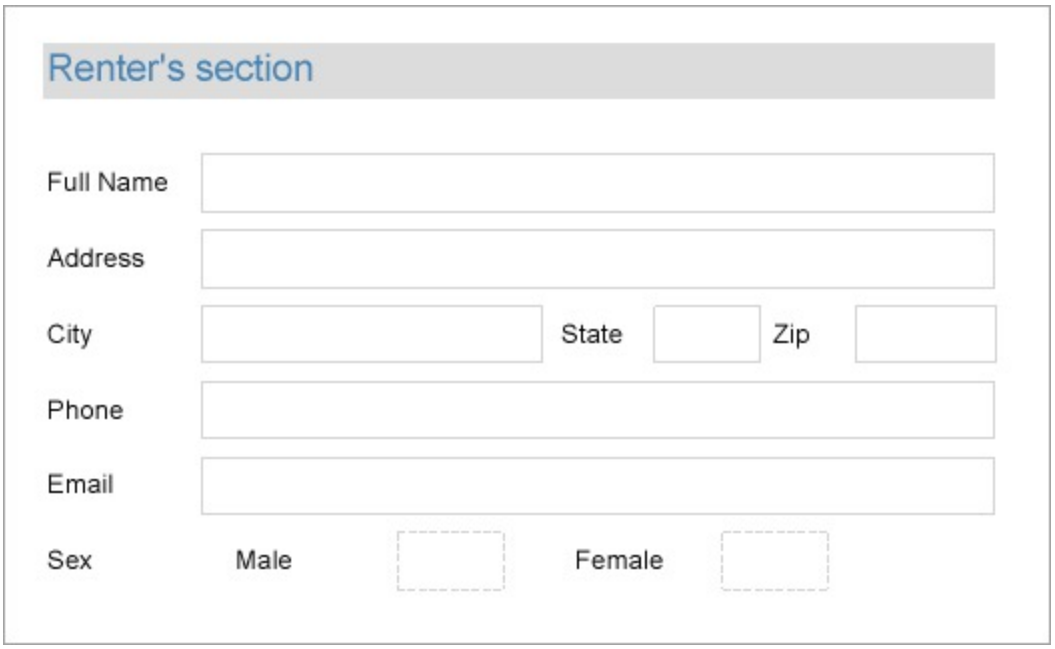
Title	MPAA	User Rating
Twister	PG-13	5.5
My Big Fat Greek Wedding	PG	9.3
Ghost Busters	PG	6.5
Beverly Hills Cop	R	8.9
War of the Worlds	PG-13	6
Cast Away	PG-13	5.7
The Lost World: Jurassic Park	PG-13	9.6
Signs	PG-13	7.5

Tip: Depending on your layout requirements, you can place the OverflowPlaceHolder control on the same page tab as the data region or a different page tab.

Create Editable PDF Forms with InputField Control

We are going to create a PDF form, using the InputField controls, where a user, besides entering text information, may also specify Yes (check) or No (uncheck) for such information like Sex, Employment Status, etc.

The final report will look as shown.



Design Report Layout

1. Drag and drop the **TextBox** control onto the report designer and set its **Value** property to Renter's section. This is the form's heading.
2. Drag and drop the TextBox controls below the Renter's section text; these controls will be the captions to the user's answers. Set the **Value** property of each of these controls to the following values:
   - TextBox2: Full Name
   - TextBox3: Address
   - TextBox4: City
   - TextBox5: State
   - TextBox6: Zip
   - TextBox7: Phone
   - TextBox8: Email
3. Now drag and drop the **InputField** controls next to the TextBox controls added at the previous step. These blank fields are required to be filled by the users filling the PDF form.
4. Set the InputField's **InputType** property to Text.
5. Drag and drop some more TextBox controls and set them to the following values in the **Value** property.
   - Value9: Sex
   - Value10: Male
   - Value11: Female
6. Now drag and drop the **InputField** controls next to the TextBox added at the previous step. These blank fields are required to be filled as yes or no using the checkboxes.
7. Set the InputField's **InputType** property to CheckBox.
8. Set **Required** property for some fields that you want to make mandatory before submitting the form.
9. Improve the appearance of the controls and preview the report.
10. Export the report to PDF format. Users can now enter the text in the blank fields and also enter their choices as yes (check) or no (uncheck) in the editable checkboxes.

Create Forms to be Filled with Freehand with Line Control

Let’s say that we want to create an order form that needs to be filled out for ordering some goods.

The final report will look like as shown.

ORDER FORM

Phone: 1-206-555-1417
Fax: 1-206-555-5938

One Portals Way
Twin Points, WA
98156

To place your order, fill out this form and return it to us at the address above. Or if you prefer, fax it to us at 1-206-555-5938.

Bill To: _____ **Ship To:** _____

Product ID	Product Name	Unit Price	Quantity
		Subtotal:	
		Total:	

This form requires the billing and shipping information, some information on the product, its price and quantity to order, and the order's subtotals and totals.

Since the information is filled out freely, by hand, or in an electronic form, you don't need to use any datasets. All you need is to create the form with a few Line controls to show blank spaces and draw a blank table - to be filled out by hand.

Design Report Layout

1. Drag and drop few TextBox controls to contain default information such as the Form name ('Order Form'), address, contact information, instructions to fill the form, etc.
2. Now, drag and drop the Line controls to draw horizontal lines that will show blank spaces that need to be filled for the 'Bill To:' and 'Ship To:' information.
3. Draw a table using Line controls. Drag and drop the Line controls one by one and draw the lines in horizontal and vertical directions.

   > **Tip:** To draw a perfect horizontal or vertical line, keep the **Shift** key pressed while drawing the line.

4. Drag and drop the TextBox controls to provide labels for table columns: 'Product ID', 'Product Name', 'Unit Price', and 'Quantity'.
5. Drag and drop the TextBox controls to provide labels for the 'Subtotal:' and 'Total:'.
6. Modify the appearance of the report and preview.

Create Hierarchical Lists with List Control

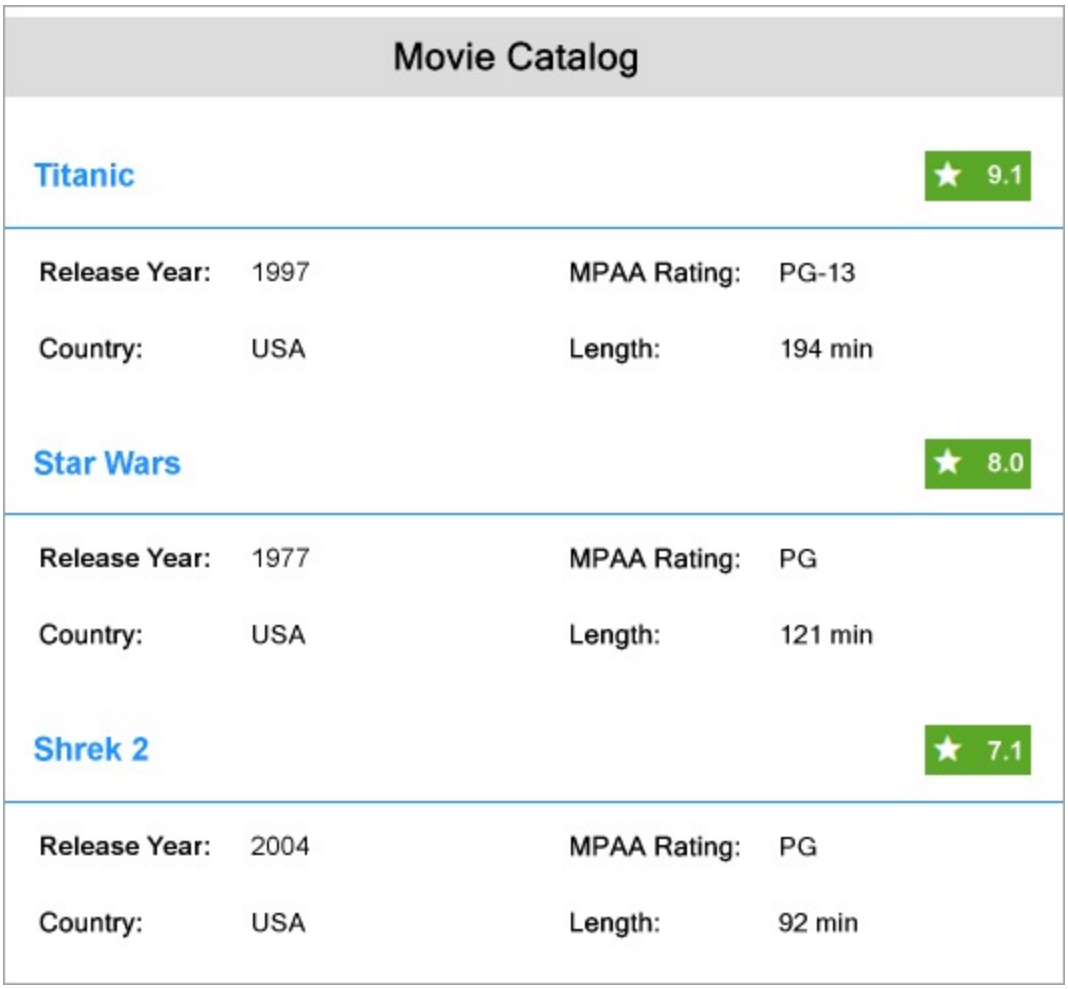
This article demonstrates the steps to create a simple list and a hierarchical list using the List control in ActiveReports.

Create Simple List

Let’s create a report that shows the products information for each category, so the product details will appear corresponding to each category, along with the category name and category description. The report uses 'NWIND.db' data source available on GitHub. It is a SQLite Provider, a custom data provider that works if System.Data.SQLite package is added and referenced in the ActiveReports.config file. See setting up the dependencies and configuration file as described in Configure ActiveReports topic.

The report uses the List data region with other controls on it so that each control repeats for every record in the dataset. The report also uses a Table to show the details corresponding to each category. The List data region uses detail grouping by the CategoryName grouping value.

The final report will look like as shown.



Create a Report

In the ActiveReports Designer, create a new RDLX report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See SQLite for details.

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next.**
2. To specify the **DataBase Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the Reels.db sample data source which can be downloaded from GitHub.
3. Click **Test Connection** to test the connection.
4. Then click the **Next** option and configure the dataset by adding a valid query.

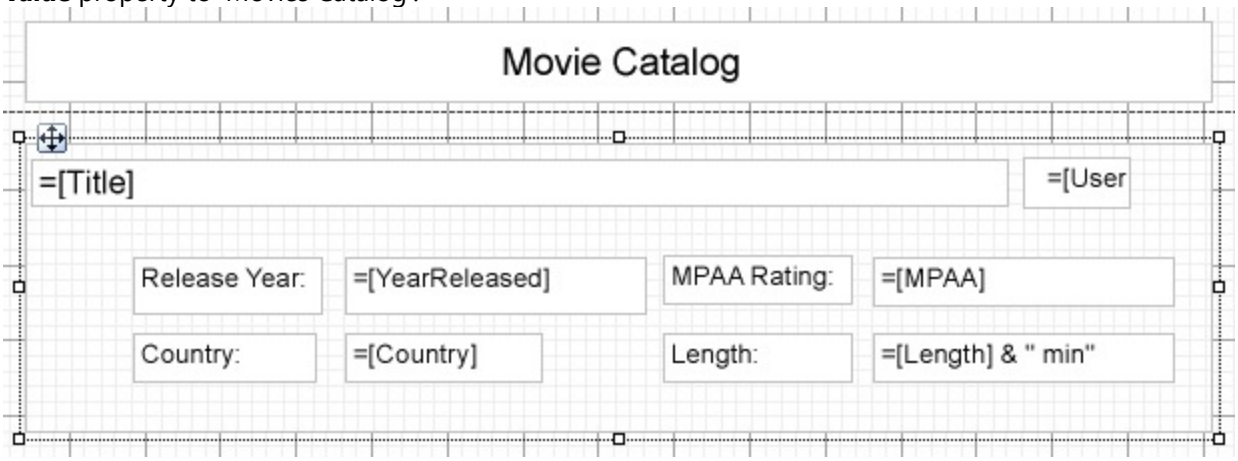
Dataset Query
`SELECT Movie.Title, Genre.GenreName, Movie.MovieID, Movie.Length, Movie.YearReleased, Movie.UserRating, Movie.MPAA, Movie.Country`

```
FROM Genre INNER JOIN (Movie INNER JOIN MovieGenres ON Movie.MovieID =
MovieGenres.MovieID) ON Genre.GenreID = MovieGenres.GenreID;
```

5. Into the Name field, enter the name of the dataset as **MoviesDataset**.
6. Click **Next** to proceed to the final screen of the Report Wizard.
7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.

Design Report Layout

1. Drag and drop the **List** data region (List1) on the report's designer.
2. With the List selected, click **Property dialog** to open the List dialog.
3. In the List dialog, go to **Detail Grouping** and set the **Group on expression** `=Fields!Title.Value`.
4. Drag and drop the **TextBox** control onto the List data region and bind it to the `[Title]` field. This will be the header of each catalog entry for a movie.
5. Next, drag and drop few **TextBox** controls below the `[Title]` textbox on the List2, and set their **Value** property as follows:
   - TextBox2: Released Year:
   - TextBox3: `=Fields!YearReleased.Value`
   - TextBox4: Country:
   - TextBox5: `=Fields!Country.Value`
   - TextBox6: MPAA Rating:
   - TextBox7: `=Fields!MPAA.Value`
   - TextBox8: Length:
   - TextBox9: `="Length: " & Fields!Length.Value & " min"`
   - TextBox10: `=Fields!UserRating.Value`
6. To add the report title, drag and drop the **TextBox** control to the PageHeader section and set its **Value** property to 'Movies Catalog'.



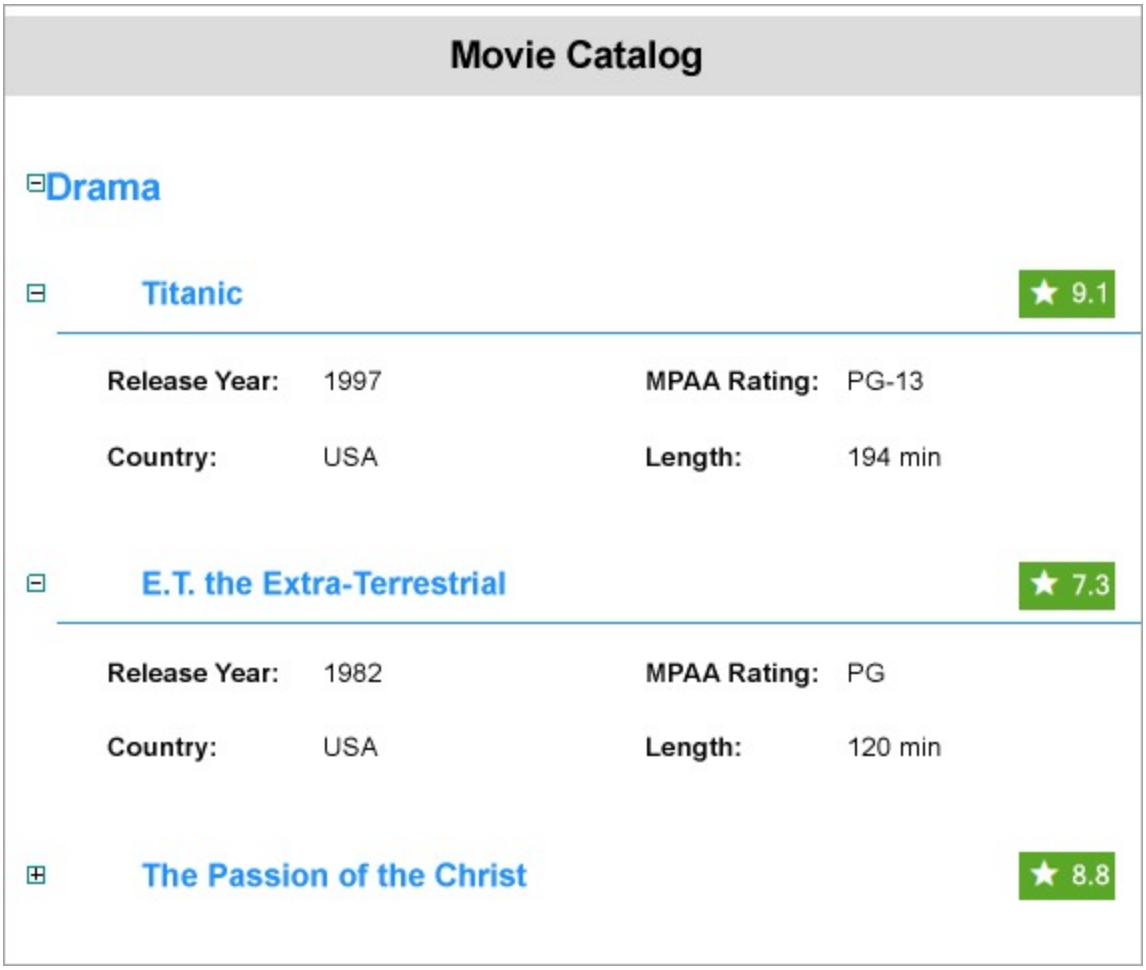
7. Improve the appearance of the report and preview.

Create Hierarchical List

You can create reports with hierarchies of expanded lists, using the List control. The report uses 'reels.db' data source available on GitHub. It is a SQLite Provider, a custom data provider that works if System.Data.SQLite package is added and referenced in the ActiveReports.config file. See setting up the dependencies and configuration file as described in Custom Data Provider topic.

The report displays the movies catalog with the list of movies along with the movies information. The report data is grouped by genre and title of the movies.

The final report will look like as shown.



Create a Report

In the ActiveReports Designer, create a new RDLX report.

Bind Report to Data

Connect to a Data Source

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option.
2. In the dialog, select the **General** page and enter the name of the data source, 'ReelsDataSource'.
3. Under **Type**, select 'SQLite Provider'.
4. In the **Connection String**, enter the path of the .db, here, 'reels.db', for example

Connection String
`Data Source=C:\Data\reels.db`

5. Click **OK** to close the dialog and complete the data source connection.

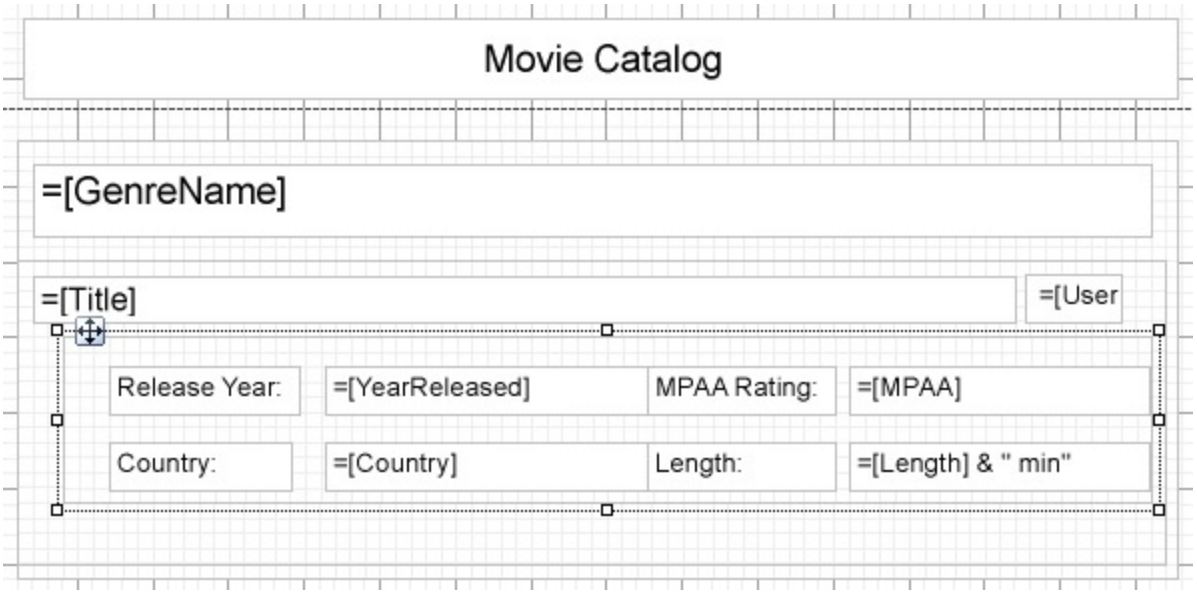
Add Dataset

1. Right-click the added data source and select **Add Dataset**.
2. In the **Dataset** dialog, select the **General** page and enter the name of the dataset, 'MoviesDataset'.
3. Go to the **Query** tab and enter the following query to fetch the required fields:

Dataset Query
`SELECT Movie.Title, Genre.GenreName, Movie.MovieID, Movie.Length, Movie.YearReleased, Movie.UserRating, Movie.MPAA, Movie.Country` `FROM Genre INNER JOIN (Movie INNER JOIN MovieGenres ON Movie.MovieID = MovieGenres.MovieID) ON Genre.GenreID = MovieGenres.GenreID;`

Design Report Layout

1. Drag and drop the **List** data region (List1) on the report's designer.
2. With the List selected, click **Property dialog** to open the List dialog.
3. In the List dialog, go to **Detail Grouping** and set the **Group on expression** to `=Fields!GenreName.Value`.
4. Drag and drop the **TextBox** control on to the List data region and bind it to the `[GenreName]` field. This will be the header of the catalog section.
5. Drag and drop another **List** data region (List2) onto the first List.
6. Click **Property dialog** to open the List dialog.
7. In the List dialog, go to **Detail Grouping** and set the **Group on expression** `=Fields!Title.Value`.
8. Drag and drop the **TextBox** control onto the List data region and bind it to the `[Title]` field. This will be the header of each catalog entry for a movie.
9. Next, drag and drop few **TextBox** controls below the `[Title]` textbox on the List2, and set their **Value** property as follows:
   - TextBox3: Release Year:
   - TextBox4: `=Fields!YearReleased.Value`
   - TextBox5: Country:
   - TextBox6: `=Fields!Country.Value`
   - TextBox7: MPAA Rating:
   - TextBox8: `=Fields!MPAA.Value`
   - TextBox9: Length:
   - TextBox10: `=Fields!Length.Value & " min"`
   - TextBox11: `=Fields!UserRating.Value`
10. To add the report title, drag and drop the **TextBox** control to the PageHeader section and set its **Value** property to 'Movies Catalog'.



11. Improve the appearance of the report and preview.

Create Tabular Report with Table Data Region

Let us create an RDLX report that uses the Table data region to display the movie details including its name, released year, country, duration, and user ratings. The report connects to the 'reels.db' data source available on GitHub.

The final report will look as shown. The data in the table is grouped by the released year of movies.

Movies List

Title	Country	Length	User Rating
2005			
Star Wars: Episode III - Revenge of the Sith	USA	140 mins	8.5
The Chronicles of Narnia: The Lion, the Witch and the Wardrobe	USA	140 mins	9
Harry Potter and the Goblet of Fire	UK	157 mins	9.5
War of the Worlds	USA	116 mins	6
King Kong	New Zealand	187 mins	6.3
Wedding Crashers	USA	119 mins	7.8
Charlie and the Chocolate Factory	USA	115 mins	5.9
Batman Begins	USA	141 mins	5.9
Madagascar	USA	86 mins	5.9
Mr. & Mrs. Smith	USA	120 mins	5.4
2004			
Shrek 2	USA	92 mins	7.1
Spider-Man 2	USA	127 mins	8.8
The Passion of the Christ	USA	127 mins	8.8
Meet the Fockers	USA	115 mins	7.6
The Incredibles	USA	115 mins	8.3
Harry Potter and the Prisoner of Azkaban	UK	141 mins	7.3

Create a Report

In the ActiveReports Designer, create a new RDLX report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access

the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See SQLite for details.

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next.**
2. To specify the **DataBase Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the Reels.db sample data source which can be downloaded from GitHub.
3. Click **Test Connection** to test the connection.
4. Then click the **Next** option and configure the dataset by adding a valid query.

Dataset Query
`SELECT * FROM Movie`

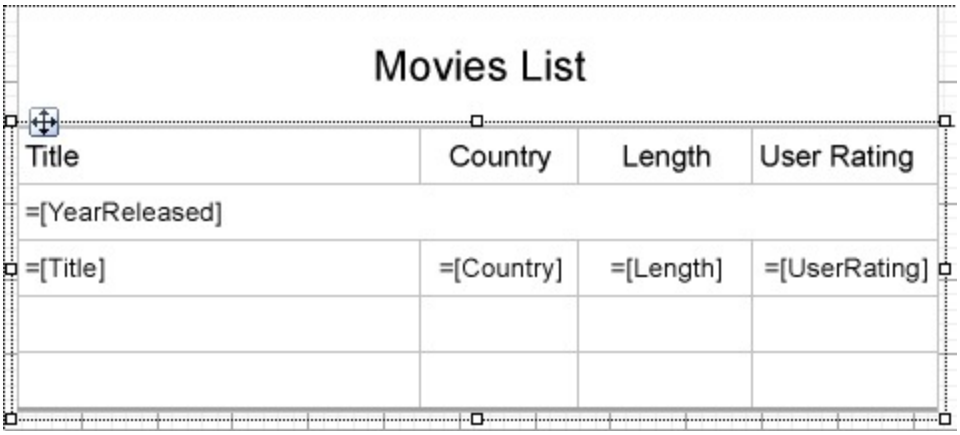
5. Into the Name field, enter the name of the dataset as **MoviesDataset**.
6. Click **Next** to proceed to the final screen of the Report Wizard.
7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.

Design Report Layout

1. Drag and drop the **Table** data region on the report's designer.
2. From the dataset, drag the following data fields to the Details row of the Table data region: `[Title]`, `[Country]`, `[Length]`, and `[User Rating]`. The **Header** row above the Details row is automatically filled with labels.
3. To display the movie length in minutes, select the `[Length]` textbox and set its **Format** property to `### mins`.
4. With the Table data region selected, click the **Property dialog** link to open the Table dialog.
5. Go to the **Groups** page and click Add.

   > **Note**: Alternatively, you can add a group by selecting **Insert Group** in the context menu of the Table data region, and adding a group in the **Table - Groups** dialog that opens.

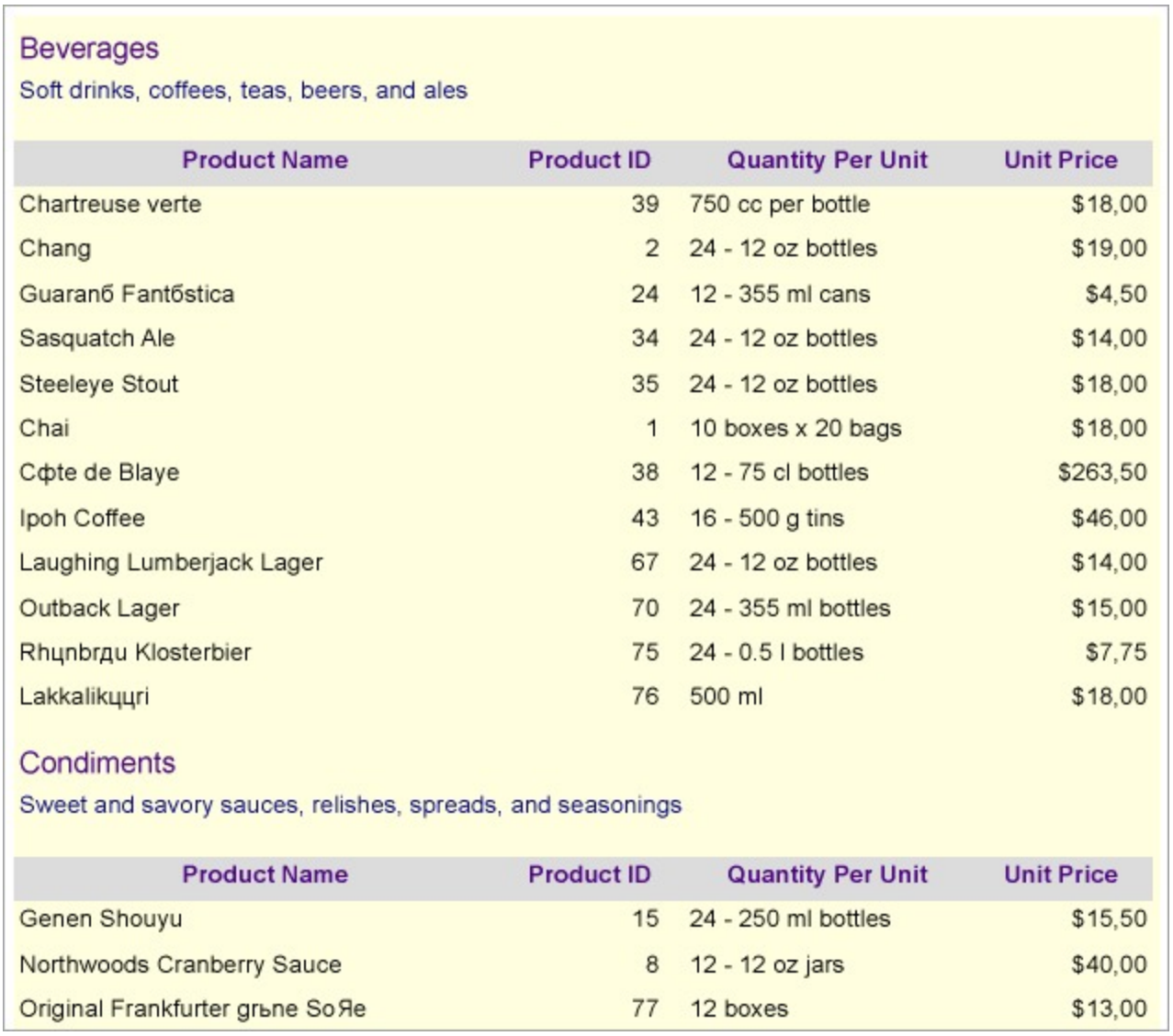
6. Set the **Group on Expression** to `=Fields!YearReleased.Value`.
   This will ensure that the data in the table is grouped based on the `[YearReleased]` field.
7. Navigate to the **Sorting** page. Set the **Sorting Expression** to `=Fields!YearReleased.Value` and **Sorting Direction** to Descending.
8. In the **Layout** tab and check the **Keep together on one page if possible** option.
   This property will try to fit the entire group on the same page; either on the current page or on the following page. If this is not possible due to a large number of records, the content will split across multiple pages.
9. Click the **OK** button to close the dialog. A Group Header and a Group Footer are added.
10. Merge the cells in the **Group Header** row and enter the expression in the **Value** property as `= [YearReleased].`
    This makes the group value appear and span above each group.
11. To add the report title, drag and drop the TextBox control to the PageHeader section and set its **Value** property to 'Movies List'.



12. Improve the appearance of the report and preview.

Enhance Report Appearance with Shape Control

Let’s say that we already have a report displaying a catalog of products grouped by categories as shown below.



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
Guaranб Fantбstica	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
Cфte de Blaye	38	12 - 75 cl bottles	$263,50
Ipoh Coffee	43	16 - 500 g tins	$46,00
Laughing Lumberjack Lager	67	24 - 12 oz bottles	$14,00
Outback Lager	70	24 - 355 ml bottles	$15,00
Rhцnbrдu Klosterbier	75	24 - 0.5 l bottles	$7,75
Lakkalikццri	76	500 ml	$18,00

Condiments

Sweet and savory sauces, relishes, spreads, and seasonings

Product Name	Product ID	Quantity Per Unit	Unit Price
Genen Shouyu	15	24 - 250 ml bottles	$15,50
Northwoods Cranberry Sauce	8	12 - 12 oz jars	$40,00
Original Frankfurter grьne SoЯe	77	12 boxes	$13,00

By using Shape controls, we will improve the appearance of this report. One shape will highlight each category of products, the other one will visually mark the boundaries of the list of products. Both shapes will use a rectangular shape with round corners for a more powerful visual effect.

Design Report Layout

1. Drag and drop the Shape control onto the report's designer, so that it surrounds text boxes `[CategoryName]` and `[Description]`.
2. Set the following properties of the Shape control to some values, for example:
   - **Border Width**: 2pt
   - **Border Style**: Solid
   - **Border Color**: Green
   - **Shape Style**: RoundRect
   - **Rounding Radius**: 5pt for each corner
3. Drag and drop another Shape control, on the List data region, and resize it to surround the List data region.
4. Set the following properties of the Shape control to some values, for example:
   - **Border Width**: 2pt

- **Border Style**: Solid
- **Border Color**: Green
- **Shape Style**: RoundRect
- **Rounding Radius**: 15pt for each corner

5. Modify the appearance of the report and preview.

Beverages

Soft drinks, coffees, teas, beers, and ales

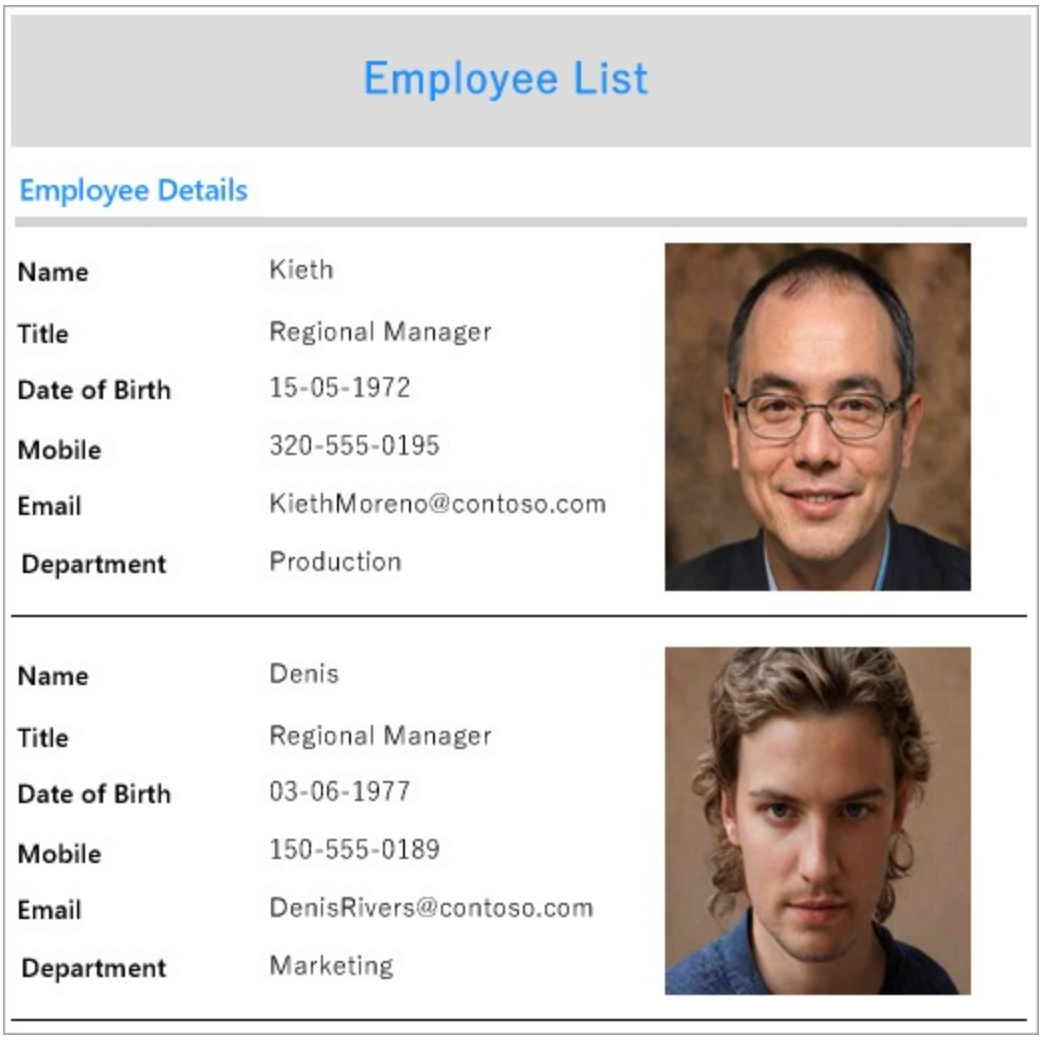
Product Name	Product ID	Quantity Per Unit	Unit Price
Chartreuse verte	39	750 cc per bottle	$18,00
Chang	2	24 - 12 oz bottles	$19,00
Guaranб Fantбstica	24	12 - 355 ml cans	$4,50
Sasquatch Ale	34	24 - 12 oz bottles	$14,00
Steeleye Stout	35	24 - 12 oz bottles	$18,00
Chai	1	10 boxes x 20 bags	$18,00
Cфte de Blaye	38	12 - 75 cl bottles	$263,50
Ipoh Coffee	43	16 - 500 g tins	$46,00
Laughing Lumberjack Lager	67	24 - 12 oz bottles	$14,00
Outback Lager	70	24 - 355 ml bottles	$15,00
Rhцnbrдu Klosterbier	75	24 - 0.5 l bottles	$7,75
Lakkalikццri	76	500 ml	$18,00

Note: Exports such as Word, Excel, and HTML do not provide good support of overflow items. In such scenarios where overflow items are used, it is better to avoid using shape as background.

Image Control in Reports

Let's say that we want to create a report that shows the photos of employees along with their details including name, job title, date of birth, mobile, email, and department. For such a report, we will use the BandedList control with Image control in the details band along with other TextBox controls. The report connects to 'DimEmployees' JSON data available here.

The final report will look like as shown.



Create a Report

In the ActiveReports Designer, create a new Page report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

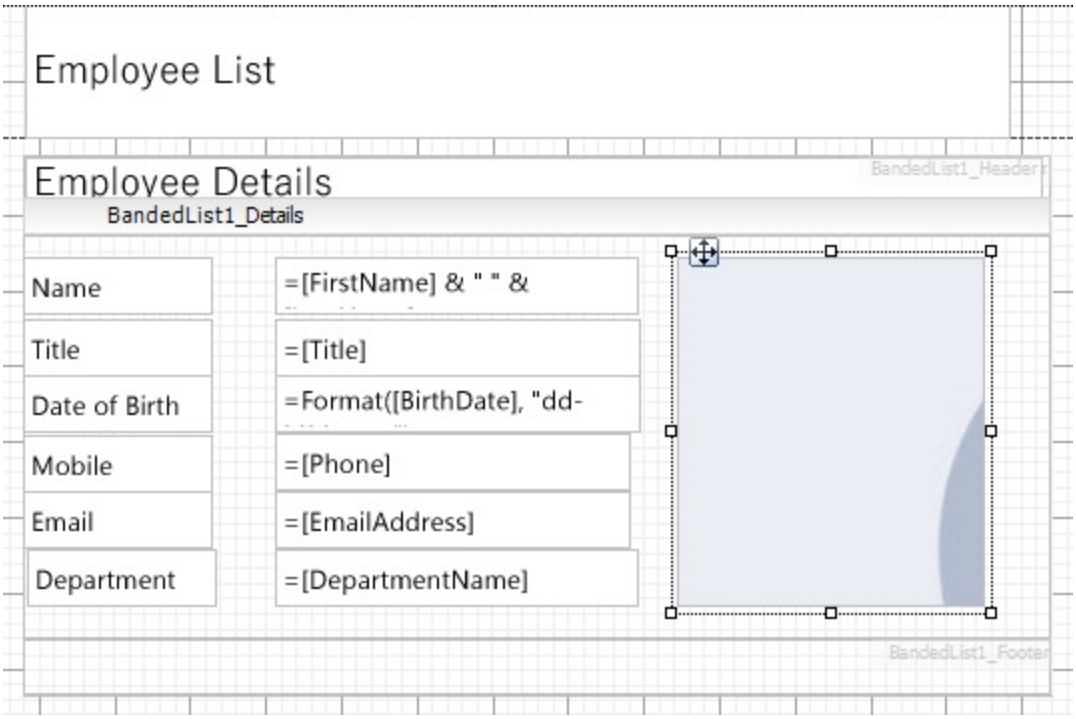
As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See JSON for details.

1. On the **Choose Data Source Type** screen of the wizard, select **JSON** and click **Next.**
2. To specify JSON **File Path**, click the **Browse** button and navigate to the desired file on your system. This report uses the 'DimEmployees.json' sample data source that can be downloaded from the following URL:
   https://demodata.mescius.io/contoso/odata/v1/DimEmployees
3. Click the **Next** option and configure the dataset by adding a valid query. Enter EmployeesDataset into the **Name** field and select the node from the data tree in the **Path** section to generate the path. The resulting query is displayed in the **Query** field.

4. Click **Next** to proceed to the final screen of the Report Wizard.
5. Review the summary of the report and click **Finish** to successfully add the report with the JSON data source.

Design Report Layout

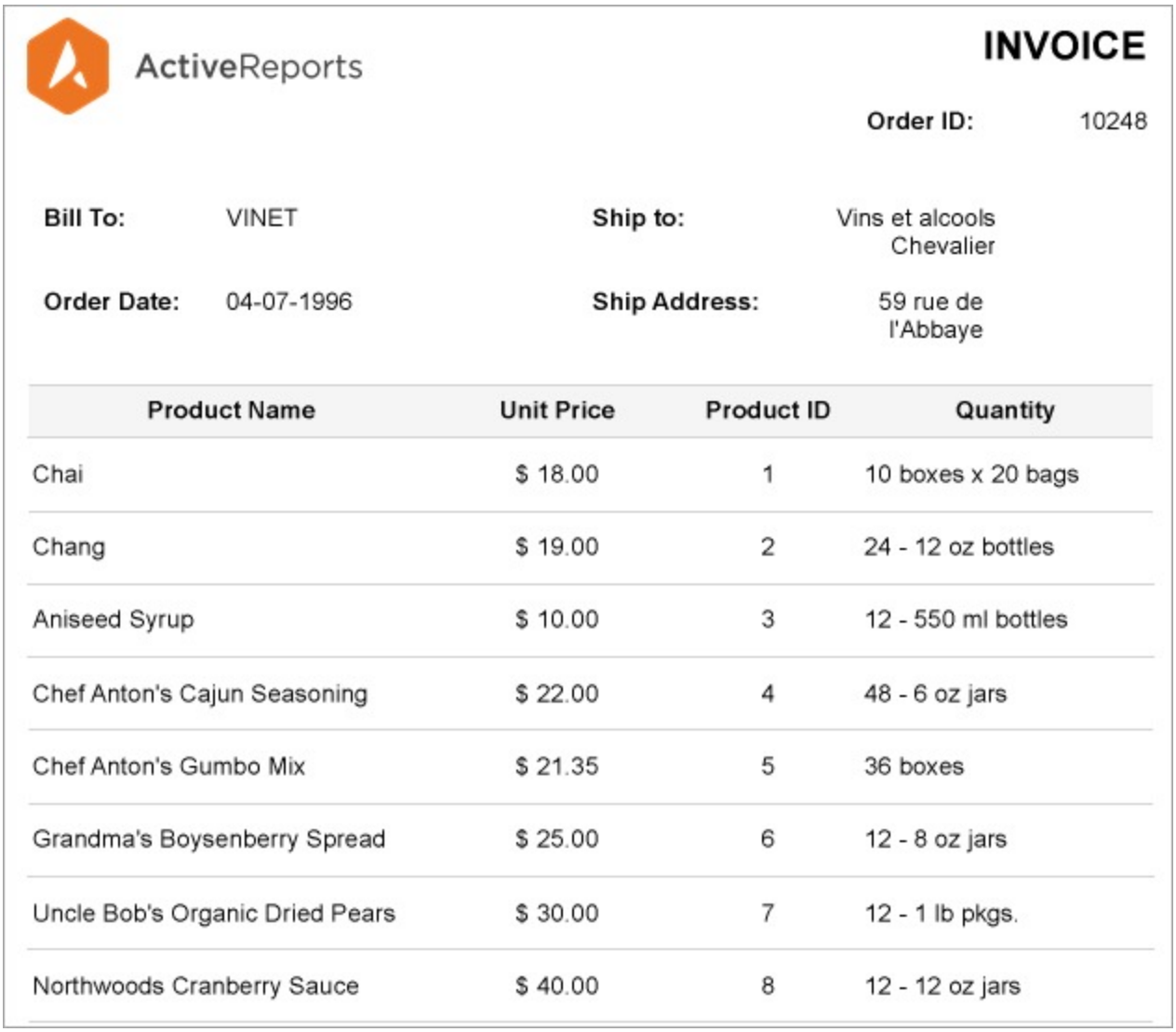
1. Drag and drop the **BandedList** control onto the report's designer.
2. Drag few TextBox controls to the **Details** band of the BandedList control, and set their **Value** property as follows:
    - TextBox1: Name
    - TextBox2: `=Fields!FirstName.Value & " " & Fields!LastName.Value`
    - TextBox3: Title
    - TextBox4: `=Fields!Title.Value`
    - TextBox5: Date of Birth
    - TextBox6: `=Format(Fields!BirthDate.Value, "dd-MM-yyyy")`
    - TextBox7: Mobile
    - TextBox8: `=Fields!Mobile.Value`
    - TextBox9: Email
    - TextBox10: `=Fields!EmailAddress.Value`
    - TextBox11: Department
    - TextBox12: `=Fields!DepartmentName.Value`
3. Drag and drop the **Image** control and set its properties as follows:
    - **Value**: `=IIF(IsNothing(Fields!AvatarUrl.Value), "https://demodata.mescius.io/images/contoso/EmployeePhotos/no-photo.jpg", "https://demodata.mescius.io" & Fields!AvatarUrl.Value)`
    - **Source**: External
    - **Sizing**: FitPropotional
    - **MIMEType**: image/bmp
4. To provide a heading (label) to the data displayed in the text boxes in Details band, drag and drop a TextBox control onto the Header band of the BandedList control and set its Value property to 'Employee Details'.
5. To add the report title, drag and drop the TextBox control to the PageHeader section and set its **Value** property to 'Employee List'.



6. Improve the appearance of the report and preview.

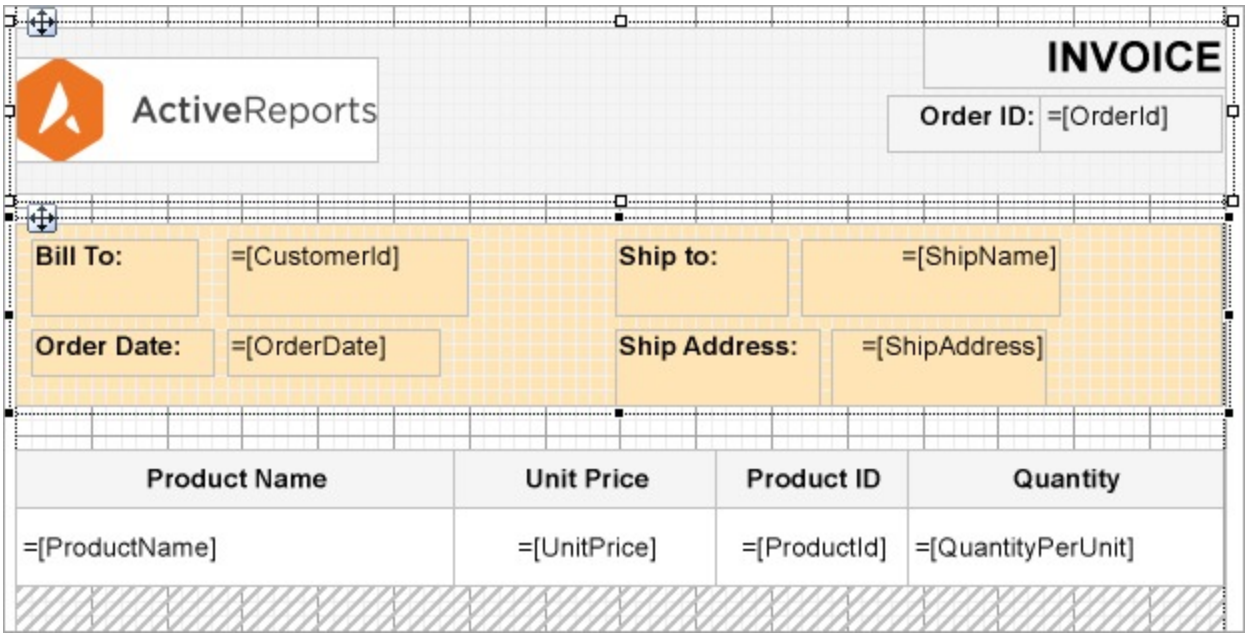
Enhance Report Appearance with Container Control

Let us enhance the appearance of the following report using Container controls.



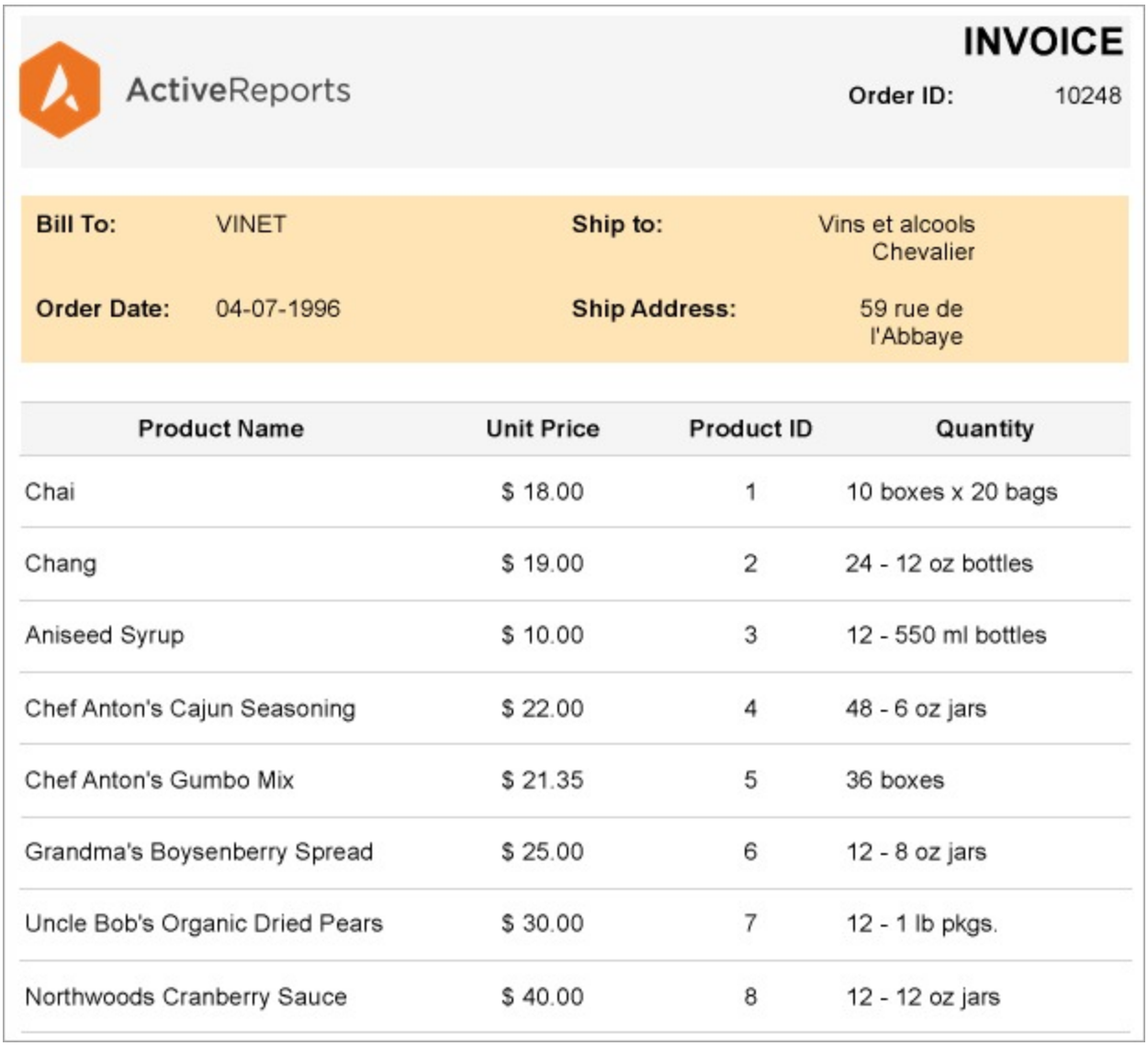
The Invoice report shown above contains the company logo and the invoice issue information, the shipping and billing information, and the products information.

We will re-create the above report to improve its appearance by using two Container controls. The containers will be placed so as to highlight the two parts of the invoice: the first Container to visually group the company logo and the invoice issue information, while the second Container to highlight the shipping and billing information.



Design Report Layout

1. Drag and drop a Container control onto the design area of the Report Designer.
2. Set the following properties to some values, for example: **Width**, **Height**, and **Color**.
3. Drag and drop Image controls into the Container and add the company logo.
4. To add information on order Id, drag-drop the `[OrderId]` field from the dataset. Also add text boxes on the container to add the labels 'Order ID:' and 'INVOICE'.
5. To highlight the shipping and billing information, drag-drop another Container control and place it below the first Container control. Set its properties to some values, for example: **Width**, **Height**, and **Color**.
6. Drag and drop the following data fields from the dataset on the second Container control: [CustomerId], [OrderDate], and [ShipName].
7. Now, add text boxes to add labels above the data fields: 'Bill To:', 'Order Date:', 'Ship To:', and 'Ship Address:'.
8. Drag-drop the Table data region onto the report designer, below the second Container control.
9. Bind the text boxes in the Table's Details row to `[ProductName]`, `[UnitPrice]`, `[ProductID]`, and `[Quantity]`.
10. Modify the appearance of the report and preview.



Mail Merge with FormattedText Control

Let's create a mail merge report that uses the **FormattedText** control to display the discount percent, start date, and end date for different sales promotions. We will use the List data region to repeat the data in the FormattedText control for each unique promotion key in the dataset.

The final report will look as shown.

Create a Report

In the ActiveReports Designer, create a new RDLX report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See JSON for details.

1. On the **Choose Data Source Type** screen of the wizard, select **JSON** and click **Next.**
2. To specify JSON **File Path**, click the **Browse** button and navigate to the desired file on your system. This report uses the 'DimPromotions.json' sample data source that can be downloaded from the following URL: https://demodata.mescius.io/contoso/odata/v1/DimPromotions
3. Click the **Next** option and configure the dataset by adding a valid query. Enter PromotionDataset into the **Name**

field and select the node from the data tree in the **Path** section to generate the path. The resulting query is displayed in the **Query** field.
4. Click **Next** to proceed to the final screen of the Report Wizard.
5. Review the summary of the report and click **Finish** to successfully add the report with the JSON data source.

Design Report

1. Drag and drop the **List** data region on the design area.
2. With the List data region selected, set the **DataSetName** as the name of the dataset, 'PromotionDataset'.
3. Go to the **Property dialog** and set the **Detail Grouping** to expression `=Fields!PromotionKey.Value`
4. Now, drag and drop the **FormattedText** control on the List data region. Check from the Report Explorer that the FormattedText is nested within the List data region.
5. To encode mail merge fields, go to the **MailMergeFields** property and click ellipses next to (Collections).
6. In the **Mail Merge** dialog, check the **Encode Mail Merge Fields** option and add the mail merge fields with the following 'Field' and 'Value' pairs:
    - Field1: PromotionName, Value: `=Fields!PromotionName.Value`
    - Field2: DiscountPercent, Value: `=Fields!DiscountPercent.Value * 100`
    - Field3: StartDate, Value: `=Format(Fields!StartDate.Value, "dd-MM-yyyy")`
    - Field4: EndDate, Value: `=Format(Fields!EndDate.Value, "dd-MM-yyyy")`
7. Set the control's **Html** property to the following text.

HTML

```
<!DOCTYPE html>

<html lang="en">

   <head>
      <title>Mail Merge Report</title>
   </head>

   <body bgcolor="#E23940">
      <h2 style="color:white; text-align:center;font-family:Segoe UI;"><% PromotionName
/%></h2>
      <h1 style="color:white;text-align:center;font-size:50px; font-family:Segoe
UI;">GET<br><big><% DiscountPercent /%>%</big> OFF</h1>
      <h2 style="background-color:white;color:#E23940;text-align:center;font-
size=35px;font-family:Segoe UI;font-weight:bold">LIMITED PERIOD OFFER</h2>

      <h4 style="color:white;text-align:center;font-size=35px;font-family:Segoe UI;">*
Valid from <%StartDate /%> to <% EndDate /%> in all our stores. Cannot be combined with
any other offers. No cash value. Northwind Traders can cancel promotion at any time.
</h4>
      <img align="center" src="https://demodata.mescius.io/images/NorthWind_large.png"
style="width:1.5inin;height:2in" />
   </body>

</html>
```

8. Improve the appearance of the controls and preview the report.

Master-Detail Report with Subreport Control

The Subreport control is embedded into the master report which acts as a placeholder for the details report. A parameter passed from the master report to the details report filters the details for each instance of master record and renders the record. Note that there should be a common field in the two reports to establish the master-detail relationship. This field is used to filter data in the details report based on the parameter passed from the master report.

Let us create Master-Detail report using subreport control. The master report will display basic information of employees - Name, Title, City, and Country, along with the list of orders fetched from the details report. Both reports use SQLite Provider, a custom data provider that works if System.Data.SQLite package is added and referenced in the ActiveReports.config file. See setting up the dependencies and configuration file as described in Custom Data Provider topic.

The master report uses the List data region that will contain TextBox controls to display the data. The List data region repeats any report control it contains for every record in the dataset. The master report uses a parameter that is passed from the master report to detail report to filter the details.

The final report will look like as shown.

Employee Orders Report (Master-Detail Report)

Name: Davolio, Nancy

Title: Sales Representative

Location: Seattle, USA

Order ID	Order Date	Shipped Date	Required Date	Ship Via	Freight
10258	17-08-1994	23-08-1994	14-09-1994	1	$ 140.51
10270	01-09-1994	02-09-1994	29-09-1994	1	$ 136.54
10275	07-09-1994	09-09-1994	05-10-1994	1	$ 26.93
10285	20-09-1994	26-09-1994	18-10-1994	2	$ 76.83
10292	28-09-1994	03-10-1994	26-10-1994	2	$ 1.35
10293	29-09-1994	12-10-1994	27-10-1994	3	$ 21.18
10304	13-10-1994	18-10-1994	10-11-1994	2	$ 63.79
10306	17-10-1994	24-10-1994	14-11-1994	3	$ 7.56
10311	21-10-1994	27-10-1994	04-11-1994	3	$ 24.69
10314	26-10-1994	04-11-1994	23-11-1994	2	$ 74.16
10316	28-10-1994	08-11-1994	25-11-1994	3	$ 150.15
10325	09-11-1994	14-11-1994	23-11-1994	3	$ 64.86
10340	29-11-1994	09-12-1994	27-12-1994	3	$ 166.31
10351	12-12-1994	21-12-1994	09-01-1995	1	$ 162.33
10357	20-12-1994	02-01-1995	17-01-1995	3	$ 34.88
10361	23-12-1994	03-01-1995	20-01-1995	2	$ 183.17

Create Details Report

In the ActiveReports Designer, create a new RDLX report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Details Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See SQLite for details.

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next.**
2. To specify the **DataBase Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the NWind.db sample data source which can be downloaded from GitHub.
3. Click **Test Connection** to test the connection.
4. Then click the **Next** option and configure the dataset by adding a valid query.

5. Enter EmployeesDataset into the **Name** and an SQL query like the following into the text box:

Dataset Query
`SELECT * FROM Employees`

   See Query Builder in Microsoft SQL Client and OLEDB Providers for more information on building SQL queries.
6. Click **Next** to proceed to the final screen of the Report Wizard.
7. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.

Design Details Report Layout

(Design view of Details Report)

1. Drag and drop the **Table** control onto the report's designer.
2. From the dataset, drag the following data fields to the **Details** row of the Table control: `[OrderId]`, `[OrderDate]`, `[ShippedDate]`, `[RequiredDate]`, `[ShipVia]`, and `[Freight]`. The **Header** row above the Details row is automatically filled with labels.
3. In the **Footer** row of the table, in the extreme right cell, set the **Value** property to the following expression `=Sum(Fields!Freight.Value)`.
4. Let us add a hidden parameter to the details report. This parameter will be passed form the master report to filter the data in the details report. To add a report parameter that filters the data based on the 'EmployeeID', go to the **Report Explorer**, right-click **Report Parameters**, and select **Add Parameters**.
5. In the **Report - Parameters** dialog that appears, click **Add**. A parameter, **ReportParameter1**, is added.
6. Select ReportParameter1 to set its properties as follows.
   - Set the **Data Type** property to Integer (since EmployeeID is integer).
   - Enable the **Hidden** property to True.
7. Save the report and enter the name of the report as 'subreport-details.rdlx'.

Create a Master Report

In the ActiveReports Designer, create a new RDLX report.

Bind Master Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See SQLite for details.

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next.**
2. To specify the **DataBase Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the NWind.db sample data source which can be downloaded from GitHub.
3. Click **Test Connection** to test the connection.
4. Then click the **Next** option and configure the dataset by adding a valid query.

5. Enter OrdersDataset into the **Name** and an SQL query like the following into the text box:

Dataset Query
`SELECT * FROM Orders`

6. Click **Next** to proceed to the final screen of the Report Wizard.
7. Review the summary of the report and click **Finish** to successfully add the report with the JSON data source.

Design Master Report Layout

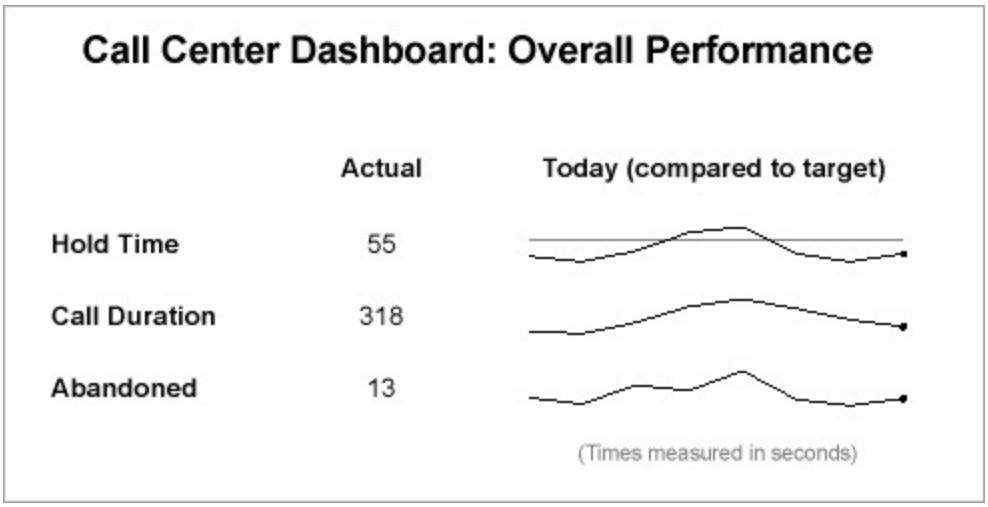
(Design view of Master Report)

1. Drag and drop the **List** data region onto the design area of the report.
2. Click Property dialog to open the List dialog.
3. In the List dialog, go to **Detail Grouping** and set the **Group on expression** to `=Fields!EmployeeId.Value.`
   This ensures the that the data in list is grouped based on [EmployeeId] field.
4. Click **OK** to close the dialog.
5. Drag and drop three **TextBox** controls onto the List and set the **Value** property of as under:
   - For TextBox1, **Value** `="Name: " & Fields!LastName.Value & ", " & Fields!FirstName.Value`
   - For TextBox2, **Value** `="Title: " & Fields!Title.Value`
   - For TextBox3, **Value** `="Location: " & Fields!City.Value & ", " & Fields!Country.Value`
6. Drag and drop the **Subreport** control onto the List data region as shown.
7. In the Properties window, set the **ReportName** property to the detail report's name (e.g., subreport-details.rdlx).
8. To pass the parameter from master report to details report, go to **Parameters** property and click the ellipses next to '(Collection)'.
9. In the **Subreport - Parameters** dialog, set the **Parameter Name** to ReportParameter1 (should be same as the report parameter name in details report) and the **Parameter Value** to `=Fields!EmployeeId.Value`.
10. Click **OK** to close the dialog.
11. To add the report title, drag and drop a **TextBox** control above the List data region.
12. Click inside the text box and enter the text 'Employee Orders Report (Master-Detail Report)'.
13. Improve the appearance of the report and preview.

Sparkline Control in Reports

Line and Area Sparklines

Let us create a report that displays the call center data trend compared to the target (times measured in seconds). Each sparkline control will display data for three criteria - call hold time, call duration, and abandoned calls.



Create a Report

In the ActiveReports Designer, create a new RDLX report.

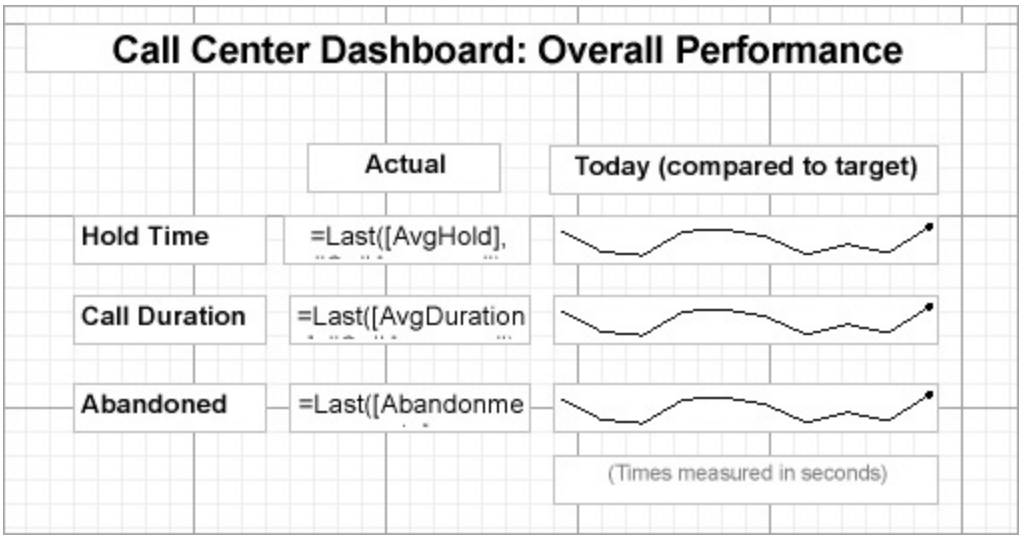
> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. This report uses the 'CallCenter.xml' data source that you can download from GitHub. See XML for details on binding to XML data.

1. On the **Choose Data Source Type** screen of the wizard, select **XML** and click **Next.**
2. To specify XML **File Path**, click the **Browse** button and navigate to the desired file on your system. This report uses the 'CallCenter.xml' sample data source that can be downloaded from GitHub.
3. Click the **Next** option and configure the following two datasets - **Targets** and **CallAverages**.
4. Enter Targets into the **Name** field and select the node from the data tree in the **Path** section to generate the path. The resulting query is displayed in the **Query** field.
5. Click Add to add the second dataset.
6. Enter CallAverages into the **Name** field and select the node from the data tree in the **Path** section to generate the path. The resulting query is displayed in the **Query** field.
7. Click **Next** to proceed to the final screen of the Report Wizard.
8. Review the summary of the report and click **Finish** to successfully add the report with the JSON data source.

Design Report Layout



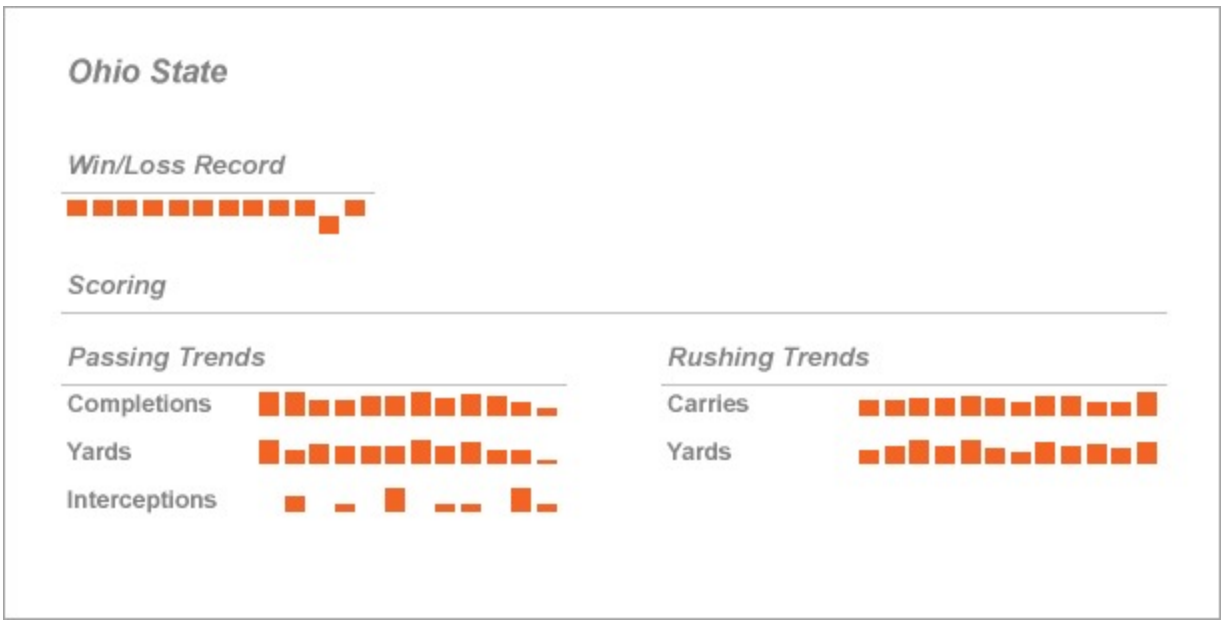
1. Drag and drop three **TextBox** controls and place them one below another. These controls will be the criterion labels. Set their **Value** property as follows:
    - TextBox2: Hold Time
    - TextBox3: Call Duration
    - TextBox4: Abandoned
2. Drag and drop a **TextBox** control near Hold Time and set its **Value** to `=Last(Fields!AvgHold.Value, "CallAverages")`.
3. Drag and drop a **TextBox** control near Call Duration and set its **Value** to `=Last(Fields!AvgDuration.Value, "CallAverages")`.
4. Drag and drop a **TextBox** control near Abandoned and set its **Value** to `=Last(Fields!Abandonments.Value, "CallAverages")`.
5. Drag and drop a **TextBox** control above the last three TextBoxes and set its **Value** to 'Actual'.
6. Drag and drop the **Sparkline** control against Hold Time and set its **Name** to HoldTimeSparkline.
7. Set the following properties of the **HoldTimeSparkline** to some suitable values, for example:
    - SeriesValue: `=Fields!AvgHold.Value / First(Fields!THoldTime.Value, "Targets")`
    - LowerBound: 1
    - UpperBound: 1.005
    - RangeVisibility: True
    - DataSetName: CallAverages
8. Drag and drop the **Sparkline** control against Call Duration and set its **Name** to CallDurationSparkline.
9. Set the following properties of the **CallDurationSparkline** to some suitable values, for example:
    - SeriesValue: `=Fields!AvgDuration.Value / First(Fields!TDuration.Value, "Targets")`
    - LowerBound: 1
    - UpperBound: 1.005
    - RangeVisibility: True
    - DataSetName: CallAverages
10. Drag and drop the **Sparkline** control against Abandoned and set its **Name** to AbandonementsSparkline.
11. Set the following properties of the **AbandonementsSparkline** to some suitable values, for example:
    - SeriesValue: `=Fields!Abandonments.Value`
    - LowerBound: 1
    - UpperBound: 1.005
    - RangeVisibility: True
    - DataSetName: CallAverages

12. Drag and drop the **TextBox** control above the last three sparklines and set its **Value** to 'Today (compared to target)'.
13. Drag-drop a **TextBox** control below the Sparkline controls and set its **Value** to '(Times measured in seconds)'.
14. To create the report's title, drag and drop the **TextBox** control onto the report's designer and set its **Value** to 'Call Center Dashboard: Overall Performance'.
15. Modify the appearance of the report and preview.

Whisker and Column Sparklines

You can use a whisker sparkline to render "win/loss/tie" scenarios (for example, sports statistics) or "true/false" scenarios (for example, was the sales goal met or the temperature above average).

Let us create a report to show the statistics for the football game using Sparkline controls of type Whisker and Column.



Create a Report

In the ActiveReports Designer, create a new RDLX report.

Bind Report to Data

Connect to a Data Source

1. As you create a new report, bind data to connect to the 'FootballStatistics.xml' data source on GitHub. See XML Provider page for details on binding with XML data. The connection string for our data is as shown:

Connection String
`xmldoc=C:\Data\FootballStatistics.xml`

Add Data Set

2. Add the following dataset with the Query as shown:

Query

```
//Game
```

Design Report Layout

1. Drag and drop the List data region inside which we will drop all the controls.
    1. Set the **DataSetName** to 'Games'.
    2. Click open the **Property dialog**.
    3. Go to **Detail Grouping** and set the **Group on** expression to `=[Team]`.
    4. Click **OK** to close the dialog.
2. Drag and drop a TextBox control inside the List data region and set its Value to 'Win/Loss Record'
3. Drag and drop the **Sparkline** control inside the List data region, below the textbox added in the previous step, and, with the sparkline selected, set the properties as follows:
    - Sparkline Type: Whiskers
    - SeriesValue: Set this property to `=Fields!WinLossTie.Value` from the connected data set.
    - FillStyle > FillColor: #f26324
    - Click open the **Properties dialog**, go to the **Sorting** page, and set the expression to `=Fields!Date.Value` and **Direction** to 'Ascending'
4. Now drag and drop a few more text boxes inside the List data region to show passing and rushing trends and some sparklines to visualize the corresponding data as shown:

5. Fill in the text in the text boxes as shown above, and set the corresponding sparkline properties as follows:
   Under 'Passing Trends',
    - Select the sparkline next to 'Completions' text and set the **SeriesValue** to `=Fields!PassCMP.Value`
    - Select the sparkline next to 'Yards' and set the **SeriesValue** to `=Fields!PassYDS.Value`
    - Select the sparkline next to 'Interceptions' and set the **SeriesValue** to `=Fields!PassINT.Value`

   Under 'Rushing Trends',

    - Select the sparkline next to 'Carries' and set the **SeriesValue** to `=Fields!RushATT.Value`
    - Select the sparkline next to 'Yards' and set the **SeriesValue** to `=Fields!RushYDS.Value`
6. To set the common properties of the sparklines added in the previous step, select the following properties:
    - SparklineType: Columns7
    - FillStyle > FillColor: #f26324
    - Click open the **Properties dialog**, go to the **Sorting** page and set the expression to `=Fields!Date.Value` and **Direction** to 'Ascending'
7. To create the report's title, drag and drop the **TextBox** control inside the List data region on the top of the

report and set its **Value** to `=[Team]`.

8. Modify the appearance of the report and preview.

Stacked Bar Sparkline

Let us create a report that shows the stacked bar sparkline use case. We will build our player performance data in the report to show the comparative scores of two players.

Create a Report

In the ActiveReports Designer, create a new RDLX report.

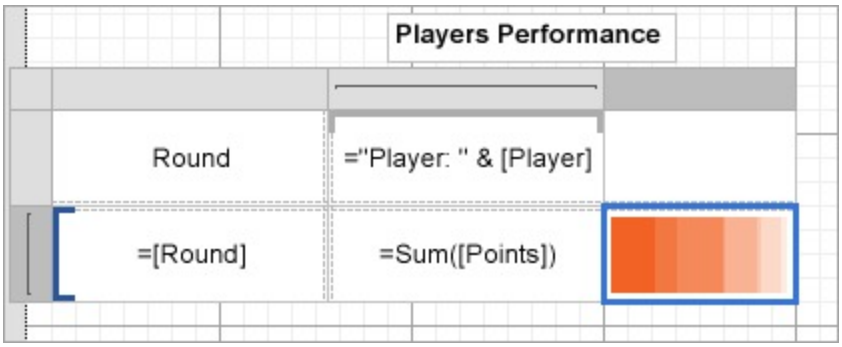
Bind Report to Data

Connect to a Data Source

1. As you create a new report, bind data to the **players.csv (on-line documentation)** file. See CSV Provider page for details on binding with CSV data. The connection string for our data is as shown:

Connection String
`Path=data\\MyOrders.csv;Locale=en-US;TextQualifier=";ColumnsSeparator=,;RowsSeparator=\r\n;HasHeaders=True`

Design Report Layout



1. Drag and drop the Tablix data region onto the report's designer.
2. Drag and drop the `[Round]` field onto the **Row Group** area.
3. In the **Values** area, write the expression `=Sum([Points])` to calculate the sum of points.
4. To add a new column, right-click anywhere on the second column of the Tablix and select **Insert Column** > **Outside Group**.
5. Drag and drop the **Sparkline** control in the third cell.
6. With the Sparkline selected, set the following properties:
    - DataSetName: Players
    - SeriesValue: `=Fields!Points.Value`
    - SparklineType: StackedBar
    - FillStyle > FillColor: Coral
7. Modify the appearance of the report and preview.

Tablix Data Region in Reports

Let us create an RDLX report that uses the Tablix data region to display shipments by country, quarter, and year. To display a summarized value for each quarter and year, we will add additional adjacent rows/columns or parent or child row/column groups to the tablix. The report connects to the 'Orders' JSON data available here.

The final report will look like as shown.

Shipments by Country, Quarter, and Year

Ship Country	Ship Name	1996			1997				
		Q3	Q4	Total	Q1	Q2	Q3	Q4	Total
France	Vins et alcools Chevalier	$ 39.54		$ 39.54				$ 18.87	$ 18.87
	Victuailles en stock	$ 41.34	$ 8.56	$ 49.90	$ 37.13	$ 194.72		$ 22.11	$ 253.96
	Blondel père et fils	$ 61.02	$ 131.70	$ 192.72	$ 209.96	$ 155.59	$ 58.30		$ 423.85
	Du monde entier	$ 24.69		$ 24.69			$ 6.25		$ 6.25
	Bon app'		$ 272.54	$ 272.54	$ 64.56	$ 361.70	$ 113.15	$ 117.00	$ 656.41
	La maison d'Asie		$ 84.28	$ 84.28	$ 106.33	$ 53.32	$ 27.65	$ 249.93	$ 437.23
	Folies gourmandes				$ 12.61		$ 487.38	$ 137.95	$ 637.94
	France restauration						$ 30.34		$ 30.34
	Spécialités du monde							$ 2.91	$ 2.91
	Alfred's Futterkiste							$ 84.96	$ 84.96
Brazil	Hanari Carnes	$ 124.00		$ 124.00		$ 68.65	$ 12.41	$ 146.10	$ 227.16
	Wellington Importadora	$ 13.97		$ 13.97	$ 44.12		$ 13.55	$ 55.23	$ 112.90
	Que Delícia	$ 9.45	$ 45.03	$ 54.48	$ 99.23		$ 108.06	$ 31.02	$ 238.31
	Ricardo Adocicados	$ 42.52		$ 42.52	$ 132.99	$ 60.43	$ 65.22		$ 258.64
	Comércio Mineiro	$ 79.70		$ 79.70	$ 11.93	$ 65.99			$ 77.92
	Tradiçao Hipermercados	$ 1.35		$ 1.35		$ 46.77	$ 79.40		$ 126.17
	Familia Arquibaldo		$ 17.09	$ 17.09	$ 21.48	$ 6.54	$ 176.81	$ 10.83	$ 215.66
	Queen Cozinha		$ 890.78	$ 890.78	$ 179.11		$ 307.10	$ 173.98	$ 660.19

Create a Report

In the ActiveReports Designer, create a new RDLX report.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

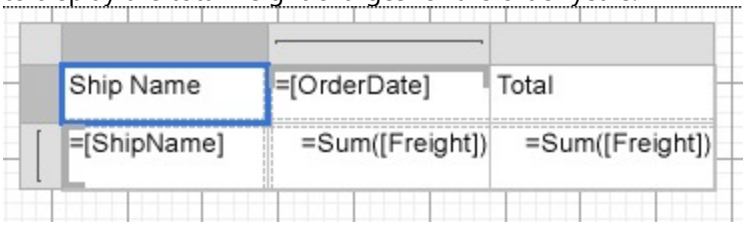
Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See JSON for details.

1. On the **Choose Data Source Type** screen of the wizard, select **JSON** and click **Next.**
2. To specify JSON **File Path**, click the **Browse** button and navigate to the desired file on your system. This report uses the 'DimEmployees.json' sample data source that can be downloaded from the following URL: https://demodata.mescius.io/northwind/odata/v1/Orders
3. Click the **Next** option and configure the dataset by adding a valid query. Enter OrdersDataset into the **Name** field and select the node from the data tree in the **Path** section to generate the path. The resulting query is displayed in the **Query** field.
4. Click **Next** to proceed to the final screen of the Report Wizard.
5. Review the summary of the report and click **Finish** to successfully add the report with the JSON data source.

Design Report Layout

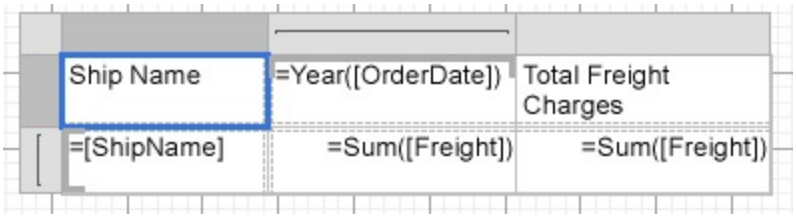
1. Drag and drop the **Tablix** data region on the report's design area.
2. From the dataset, drag and drop the `[ShipName]` field onto the **Row Group** area.
3. Select the textbox on the upper-left corner of the tablix and set its **Value** property to 'Ship Name'.
4. Then, drag and drop the `[OrderDate]` field onto the **Column Group** area.
5. Similarly, drag and drop the `[Freight]` field onto the **Values** area, which is the body of the tablix. Note that the `[Freight]` field changes to `Sum(Freight)`.
6. In the Properties pane, set the **Format** property for the `Freight` field to Currency.
7. Right-click the **OrderDate** column group area and select the **Add Total** > **After** option from the context menu to display the total freight charges for the order years.



Manage Data

In a Tablix data region, you need to manage how a grouped data is rendered across the rows and columns. This is done by using Group Expressions as explained below.

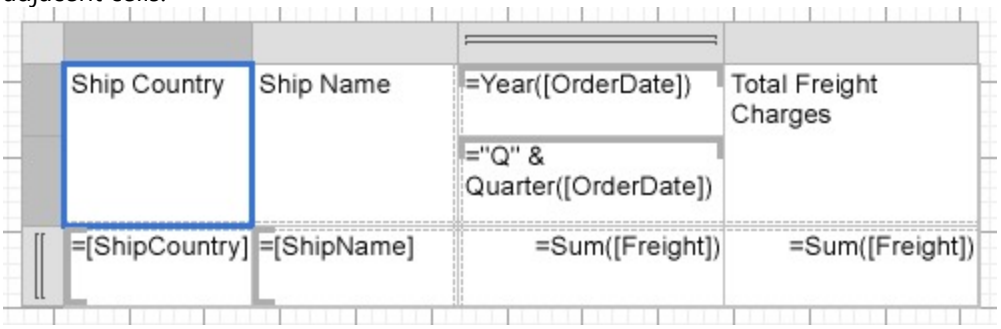
1. Modify the `=Fields!OrderDate.Value` expression to `=Year(Fields!OrderDate.Value)` to display only the Year part of the date in the textbox of the column group.
2. With the Tablix data region selected, go to the **Group Editor** to display row groups and column groups.
3. Select Tablix1_OrderDate1 column group and go to the Properties pane.
4. In the **Group Expressions** property, click **Show Items**.
5. Update the **Group Expression** to `=Year(Fields!OrderDate.Value)`.
6. Update the textbox with **Total** label to 'Total Freight Charges'.



Add Row and Column Groups

Let us add row and column groups to the tablix data region. We will add a child group to the **OrderDate** column group to display total prices for each quarter of a year. Similarly, we will add a parent group **ShipCountry** to **ShipName** row group to show countries and group the ship names according to the country.

1. Right-click the **OrderDate** column group area to view options in the context menu.
2. Go to **Column Group** and select the **Child** option.
3. From the **Group Editor**, select the newly added child group i.e. Tablix1_ColumnGroup1 and go to the Properties pane.
4. In the **Group Expressions** property, click **Show Items** and click **Add**.
5. Set the **Group Expression** to `=Quarter(Fields!OrderDate.Value)`.
6. Now, select the **OrderDate** child group area in the tablix and set its **Value** property to `="Q" & Quarter(Fields!OrderDate.Value)`.
7. Similarly, right-click the **ShipName** row group area to add a parent group from the context menu.
8. Go to **Row Group** and select **Parent** option.
9. From the **Group Editor**, select the newly added parent group i.e. Tablix1_RowGroup1 and go to the Properties pane.
10. In the **Group Expressions** property, click **Show Items** and click **Add**.
11. Set the **Group Expression** to `=Fields!ShipCountry.Value`.
12. Now, select the **ShipCountry** row group area in the tablix and set its **Value** property to `=Fields!ShipCountry.Value`.
13. Provide a label **Ship Country** to the column displaying the ship country.
14. Merge the `[Ship Country]`, `Ship Name`, and `Total Freight Charges` textboxes with the empty adjacent cells.



Add Totals to the Groups

We will add a total row to display the sum of freight charges for each country and a subtotal column to display the total freight charges for each quarter of the order year.

1. Right-click the **ShipCountry** column group area and select the **Add Total** > **After** option from the context menu.
2. Similarly, right-click the **OrderDate** child group area and select the **Add Total** > **After** option from the context

menu.

Add Report Title and Preview Report

1. To add the report title, drag and drop the TextBox control to the PageHeader section and set its **Value** property to 'Shipments by Country, Quarter, and Year'.
2. Improve the appearance of the report and preview.

TextBox Control in Reports

Let us create an RDLX report that uses TextBox controls to display the price and in stock information, and total number of items in stock. We will be using BandedList control to contain text boxes and display data for each row. The report connects to the 'Products' JSON data available here. The final report will be as shown.

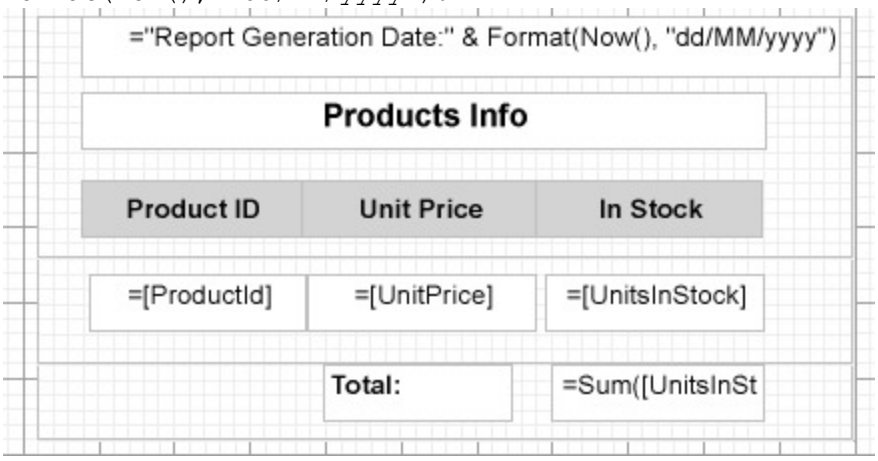
> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.

Design Report Layout

1. Drag and drop the **BandedList** control on the report's designer.
2. Drag the following data fields to the **Details** band of the BandedList control: `[ProductId]`, `[UnitPrice]`, and `[UnitsInStock]`. All these fields are bound text boxes.
3. To provide headings (labels) to the data displayed in the text boxes in Details band, drag and drop three **TextBox** controls onto the Header band of the BandedList control and set the **Value** property of the text boxes to 'Product ID', 'Unit Price', and 'In Stock', respectively.
4. To display the total number of products in stock, drag and drop two **TextBox** controls onto the Footer band of the BandedList control and set the properties as follows.
   - TextBox1: Set the **Value** property to **Total:**. This is an unbound text box.

- TextBox2: Set the **Value** property to the expression: `=Sum(Fields!UnitsInStock.Value)`. This is a calculated textbox; the expression calculates the total number of products in stock.

5. To provide the report's title, drag and drop another **TextBox** control onto the top center in the design area. Enter the text of the title into the **Value** property or directly inside the text box, e.g. Products Info.
6. To add the information on the report generation date, drag and drop a **TextBox** control on the upper right corner of the report and set the **Value** property to the expression: `="Report Generation Date:" & Format(Now(), "dd/MM/yyyy")`.

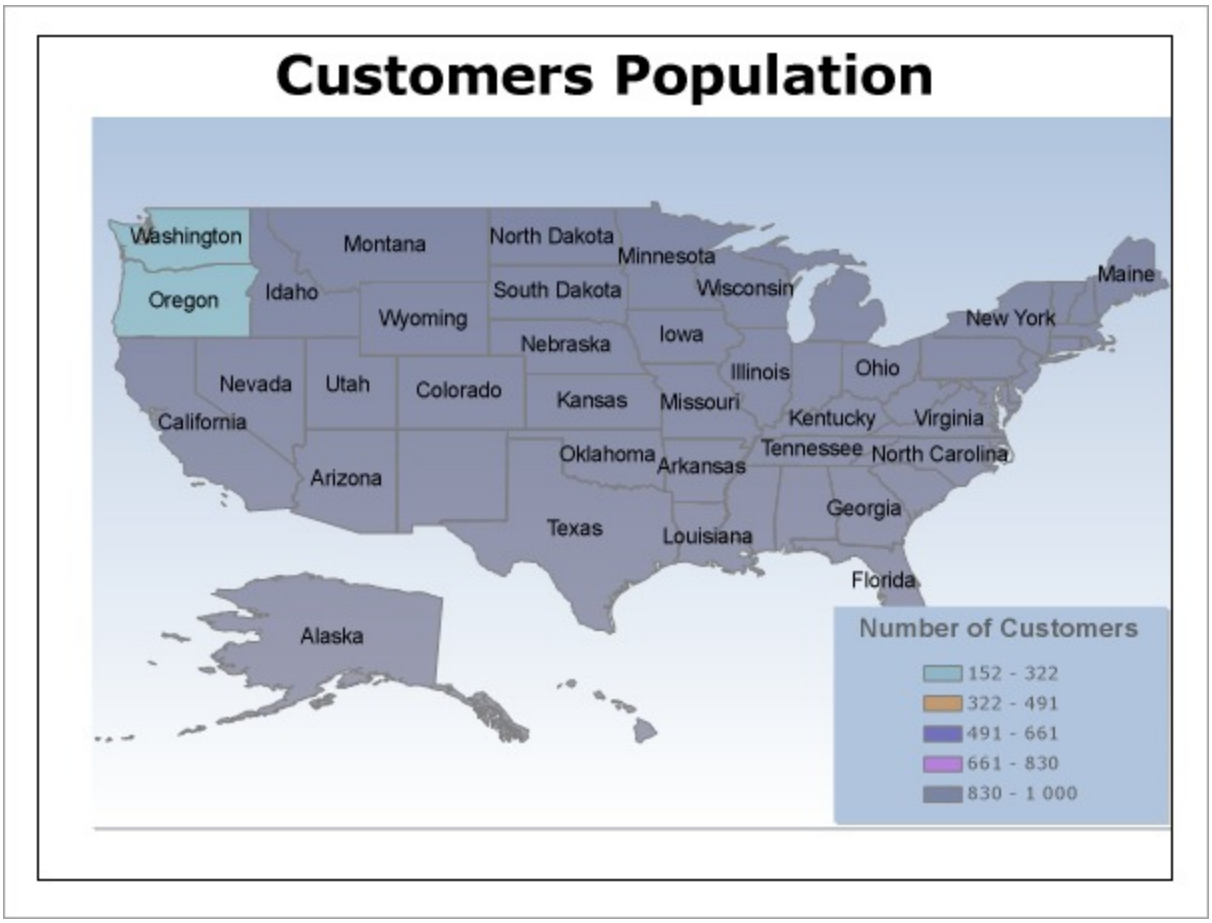


7. Modify the appearance of the report and preview.

Map Data Region in Reports

You can create a Page report that contains a map using the ActiveReports Map control. The Map data region shows your data on a geographical background. This walkthrough illustrates how to create a report that uses a Map to display data.

The final report will be as shown.



Create a Report

In the ActiveReports Designer, create a new Page report.

Bind Report to Data

As you create a new report, you can configure the report data connection in the Report Wizard. You can also access the **Report Data Source** dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option. See SQLite for details.

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next.**
2. To specify the **DataBase Path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the Reels.db sample data source which can be downloaded from GitHub.
3. Click **Test Connection** to test the connection.
4. Then click the **Next** option and configure the dataset by adding a valid query.
5. Enter Customers into the **Name** and an SQL query like the following into the text box:

Dataset Query
`SELECT Address.Region, Customer.CustomerID FROM (Address INNER JOIN Person ON Address.[AddressID] = Person.[AddressID]) INNER JOIN Customer ON Person.[PersonID] = Customer.[PersonID];`

6. Click **Next** to proceed to the final screen of the Report Wizard.
7. Review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.

Design Report

1. Drag and drop the **Map** data region on the design area.
2. In the **Select a Map Template** wizard that appears, select the **USA Map** template.
3. Click the Map to have the map panes appear.
4. In the layers pane, right-click **PolygonLayer1** and select **Layer Data** to open **Map Layer Data Properties** dialog.
5. In the **Map Layer Data Properties** dialog that appears, go to the **Analytical data** page.
6. Select **Customers** from the **Dataset** property combo box and then click the **Add** (+) button located next to the **Match** label. This creates an empty match item and enables its **Spatial** and **Analytical** fields editor.

   > **Note**: It is necessary to set match fields if you want to use a spatial data field from analytical data, or if you want to visualize analytical data on the map layer. Match fields enable the report processor to build a relationship between the analytical data and the spatial data.

7. In the **Spatial field** property, select `STATE_ABBR` from the combo box; and similarly, select `=Fields!Region.Value` in the **Analytical field** property. This builds the match field expression and relates the analytical data to map elements on a polygon layer.
8. Click **OK** to close the dialog.

   Configure the appearance of the Map
9. In the layers pane, right-click **PolygonLayer1** and select **Edit** to open **Map Polygon Layer** dialog.
10. In the **General** page of the dialog, select `#STATE_NAME` from the **Label Text** combo box to display as a label inside polygons at run time.
11. Go to the **Color Rule** page of the dialog, and select **Visualize data by using color palette** option. This activates the tabs below.
12. On the General tab, enter the following expression `=Count([CustomerID])` in the **Data field** property and set **Palette** property to SemiTransparent.
13. On the Distribution tab, set the **Method** property to EqualInterval.
14. On the Legends tab, click to select **Show in legend**.
15. In **Legend name**, select 'Legend'. This name relates to the default legend that appears in the Legend collection.
16. Click **OK** to close the dialog.
17. On the design surface, click the Map control to select it and go to the Properties Panel to set the following properties:
    - BackgroundColor: White
    - BackgroundGradientEndColor: White
    - BorderStyle: Solid
    - ColorScale > Hidden: True
    - DistanceScale > Hidden: True
    - Size: 6.5in, 4.75in
    - ViewPort > BackgroundColor: LightSteelBlue
    - ViewPort > BackgroundGradientEndColor: White
    - ViewPort > BorderStyle: None
    - ViewPort > CoordinateSystem: Planar
    - ViewPort > Meridians > Hidden: True
    - ViewPort > Parallels > Hidden: True
    - ViewPort > View > Zoom: 115
18. With the Map control selected, go to the Properties window, click the **Legends (Collection)** property and then click the ellipsis button that appears.
19. In the **LegendDesigner Collection Editor** that appears, under the **Members** list, select the existing legend and set the following properties.
    - BackgroundColor: LightSteelBlue

    - BackgroundGradientEndColor: White
    - Location > DockOutsideViewport: False
    - Location > DockPosition: RightBottom
    - Title > (Caption): Number of Customers
20. Click **OK** to close the dialog.
21. With the Map control selected, go to the Properties window, click the **Titles (Collection)** property and then click the ellipsis button that appears.
22. In the **MapTitleDesigner Collection Editor** that appears, with **Title** selected in the Members list set the following properties.
    - (Text): Customers Population
    - Color: Black
23. Click **OK** to close the dialog.
24. Preview the report.

Tutorials: Page/RDLX Report Scenarios

This section takes you through the set of tutorials aimed at covering the scenarios that you come across while designing reports.

- Create RDLX Dashboard Report
- Create Multi-Column Layout (or Columnar report)
- Create Top N Report
- Create a Red Negatives or a Green Bar Report
- Apply Theme at Runtime Using Dynamic Expression
- Add Page Numbering
- Show Row Number in Tablix and Table
- Link Multiple Datasets to Same Data Region

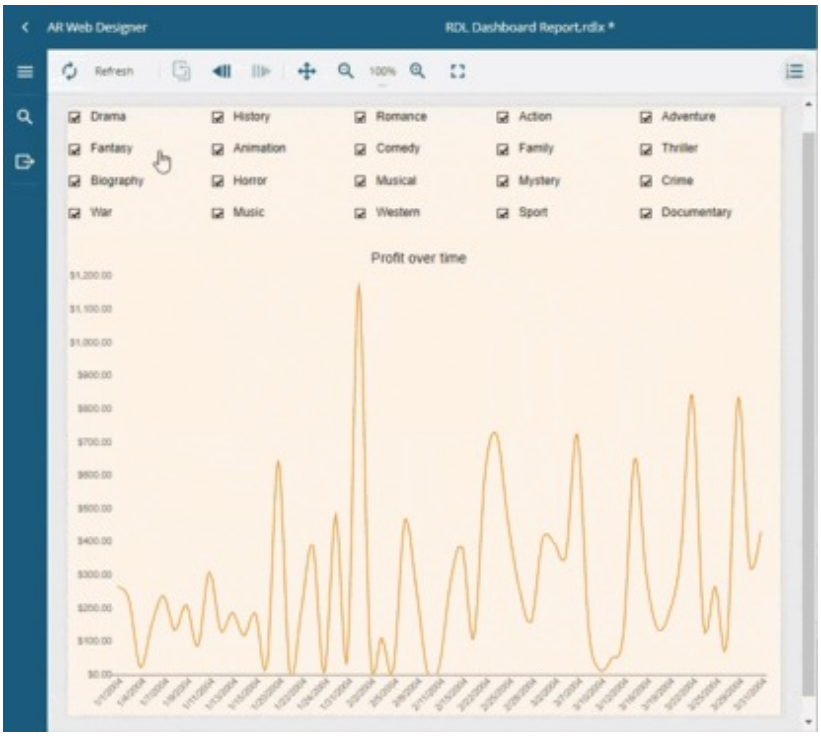
Create RDLX Dashboard Report

The RDLX Dashboard report is the latest addition to the varied reports that you can design in ActiveReports. To learn about the key features of this report type, see RDLX Dashboard Report.

Let us create an interactive RDLX Dashboard Report. The report will use the List and Chart data regions to visualize data from two datasets. In this report, you will be able to interact with the data through the **Apply Parameters** action and visualize the manipulated data. See the Actionable Parameters topic for more information.

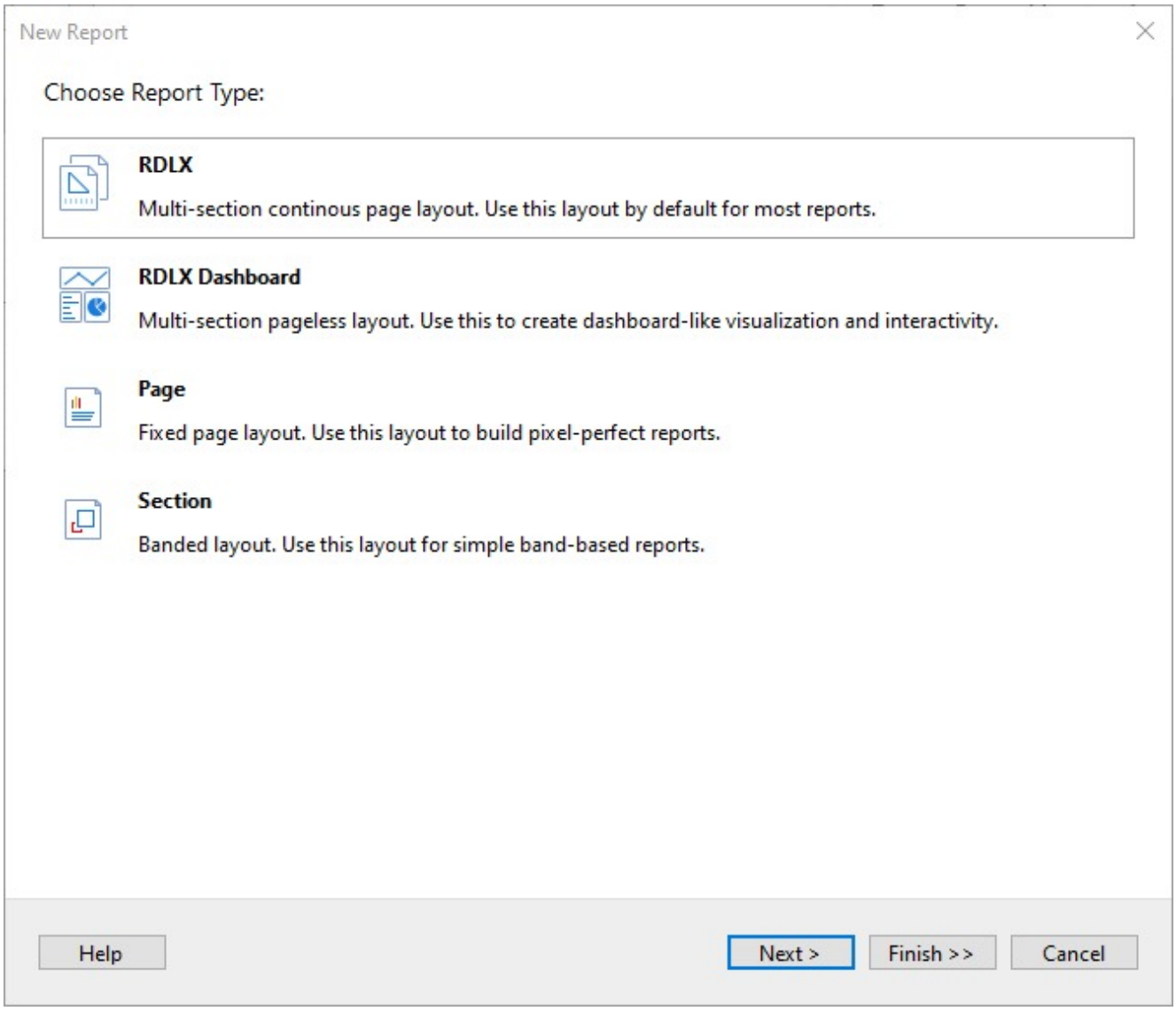
This article explains creating a new report in the ActiveReports Standalone Designer.

> If you are creating a new report in the Visual Studio Integrated Designer, see Quick Start.



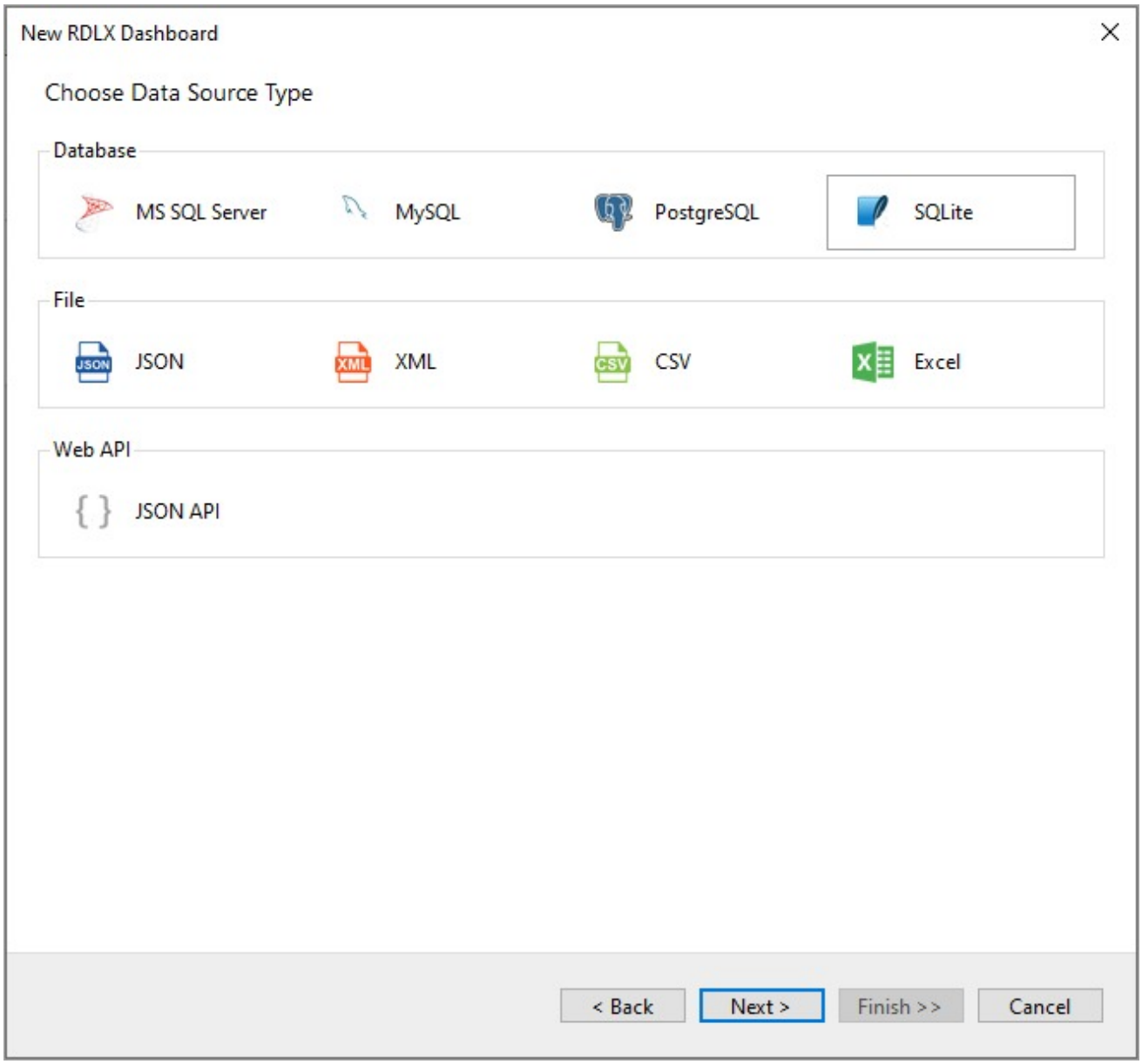
Create a Report

1. Create a New Report.
2. In the **New Report** wizard, choose the Report Type as **RDLX Dashboard** and click **Next**.

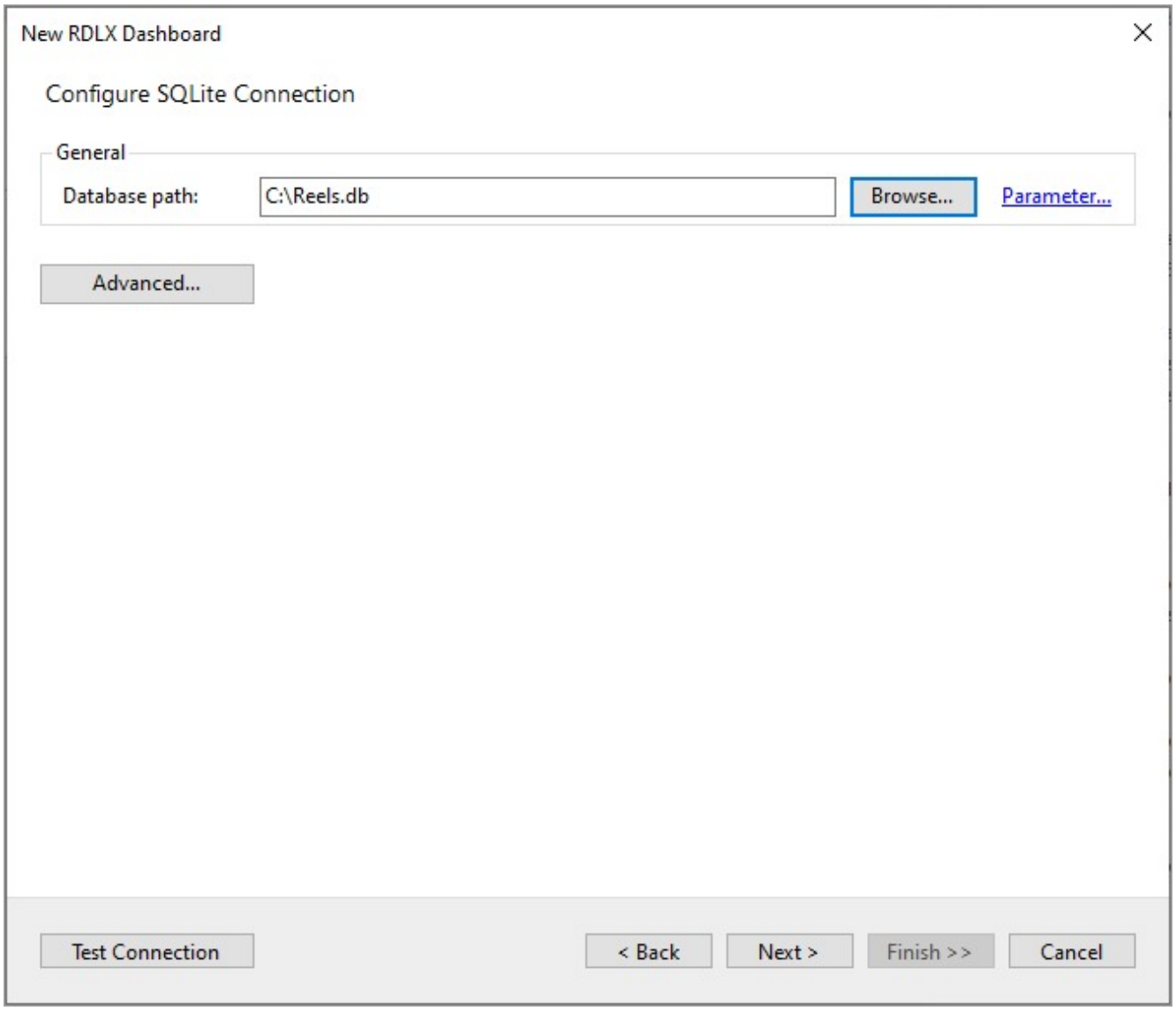


Bind Report to Data

1. On the **Choose Data Source Type** screen of the wizard, select **SQLite** and click **Next**.



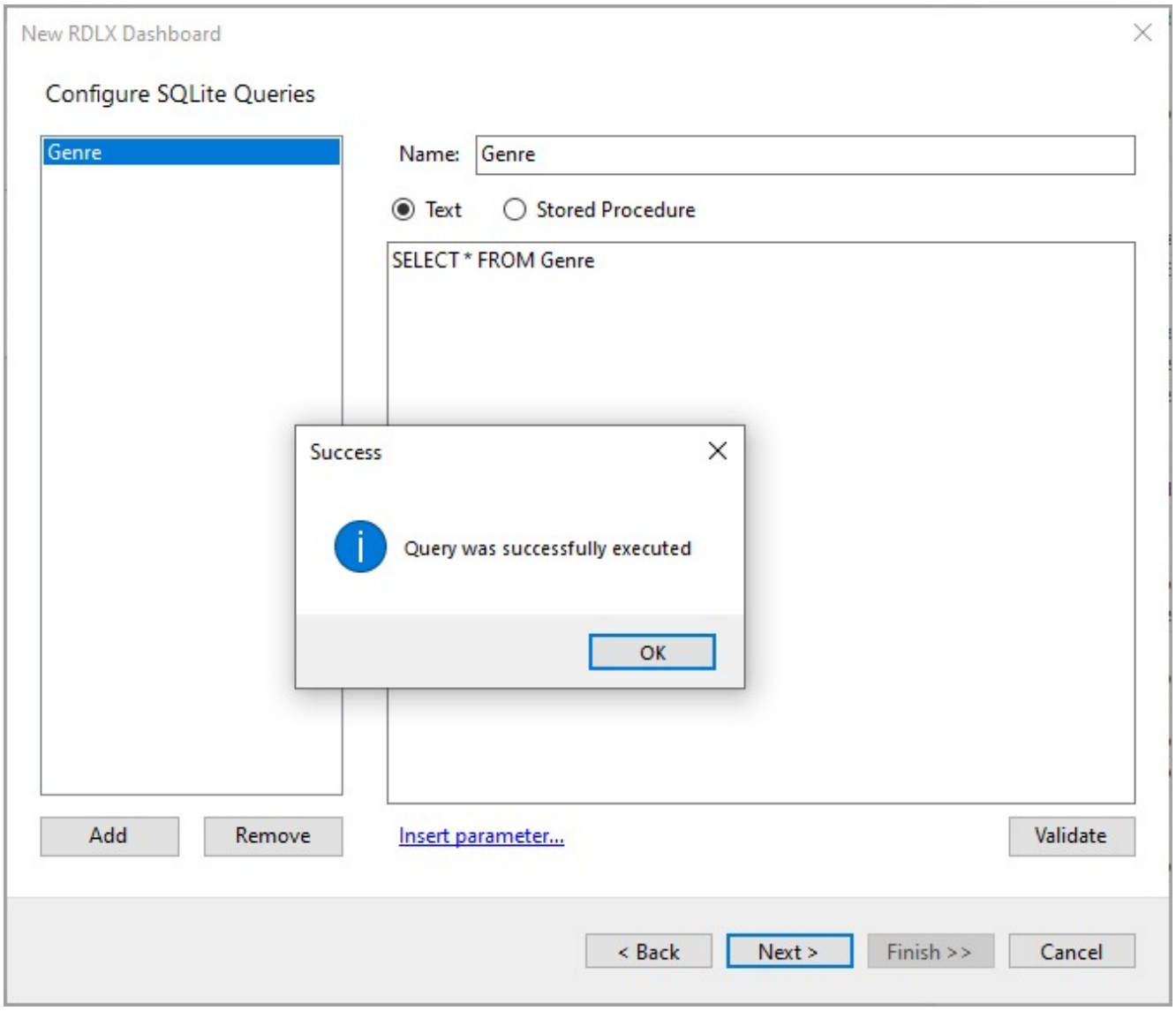
2. To specify the **Database path**, click the **Browse** button and navigate to the desired file on your system. For example, you can connect to the Reels.db sample data source which can be downloaded from GitHub. See SQLite and Configure ActiveReports topics for more information.



3. Then click the **Next** option and configure the datasets by adding valid queries.

   Add Datasets

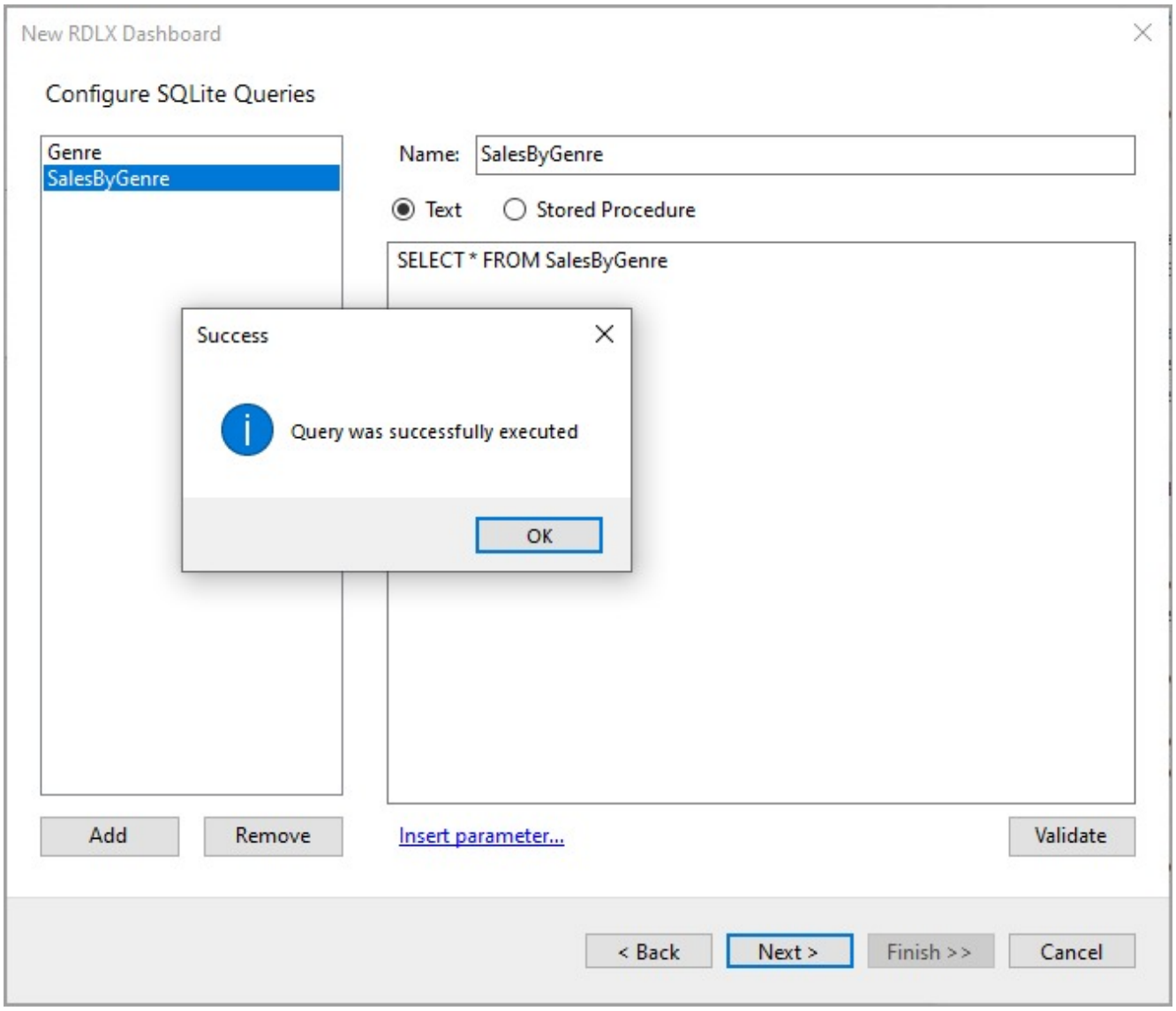
4. On the **Configure SQLIte Queries** screen, enter Genre in the **Name** field and the following SQL query: `SELECT * FROM Genre.`



5. Click **OK** to close the dialog. Your data set and queried fields appear as nodes in the **Report Explorer**.

   This dataset contains the following fields:

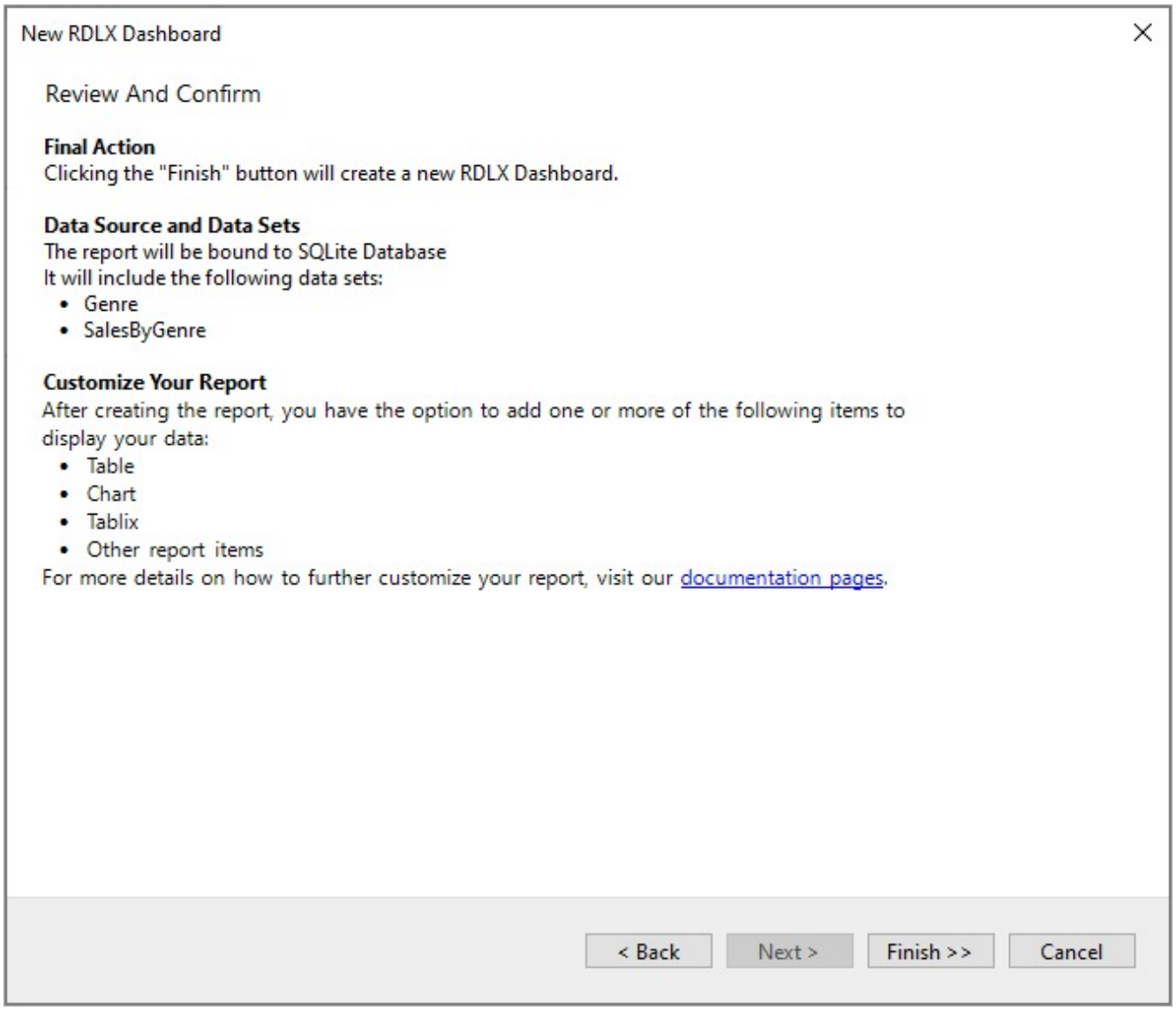
   - GenreID
   - GenreName
6. Click **Add** to add another dataset with the name **SalesByGenre.**
7. Enter the following SQL query: `SELECT * FROM SalesByGenre`.



8. Click **OK** to close the dialog. Your dataset and queried fields appear as nodes in the **Report Explorer**.

   This dataset contains the following fields:

   - GenreID
   - GenreName
   - StorePrice
   - SaleID
   - Quantity
   - SalesID
   - SaleDate
   - Profit
9. Click **Next**. On the final screen of the Report Wizard, review the summary of the report and click **Finish** to successfully add the report with the SQLite data source.



Add a Field to Dataset

10. In the Report Explorer, right-click the **Genre** dataset and select **Edit**.
11. In the **DataSet** dialog, go to the **Fields** page and add a new field with **Name** as `GenreIDInt` and **Value** as `=Convert.ToInt32([GenreID])`.
    You will see how we use the `GenreIDInt` field to avoid type mismatch.
12. Click **OK** to save your changes.

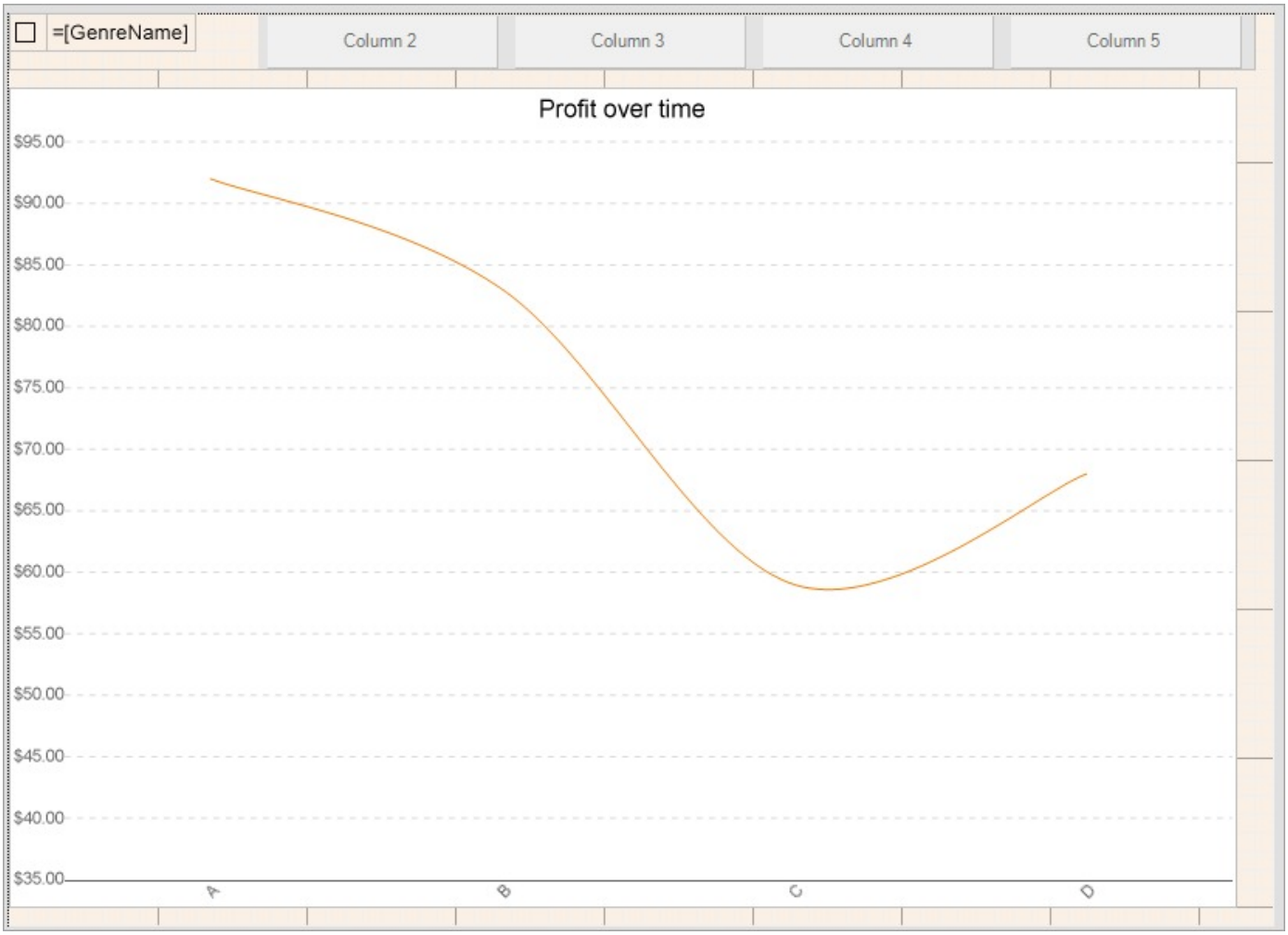
Add Report Parameter

1. In the Report Explorer, right-click the **Parameters** node and select **Add Parameter**.
2. In the **Report - Parameters** dialog that appears, add a name for the parameter, 'paramGenreID'.
3. Set the **Data Type** to 'Integer'.
4. Select the **Multivalue** and **Hidden** check boxes.
5. In the **Report - Parameters** dialog, go to the **Available Values** tab and select the **From query** radio button.
    - **Dataset**: Genre

- **Value field**: GenreIDInt
- **Label field**: GenreID

6. Go to the **Default Values** tab and select the **From query** radio button.
   - **Dataset**: Genre
   - **Value field**: GenreIDInt
7. Click **OK** to close the dialog and add the parameter to the collection.

Design Report Layout



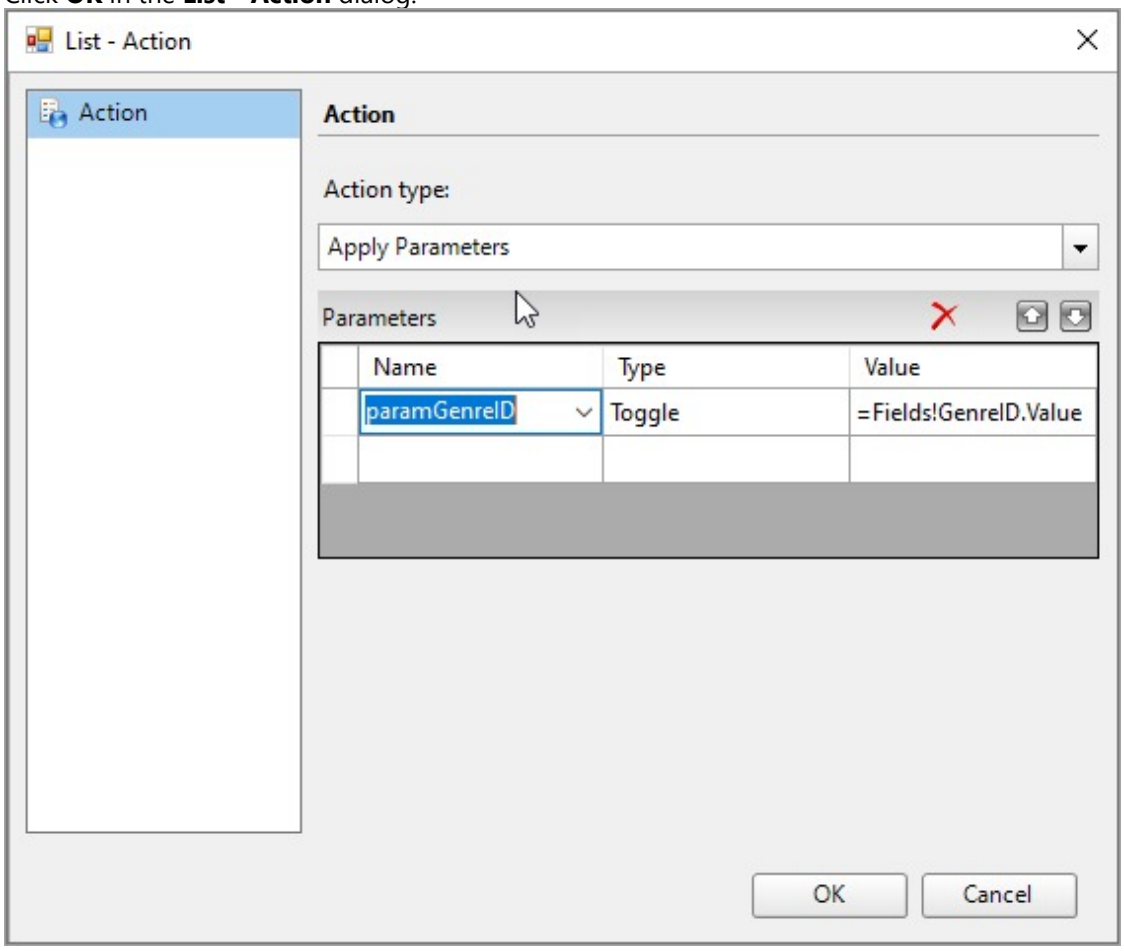
Add List data region

1. From the toolbox, drag a List data region onto the design surface of the report.
2. To display the controls placed inside the list to display in a columnar layout, go to the Properties panel to set the properties of the List data region as follows:
   - **DataSetName**: Genre
   - **RowsOrColumnsCount**: 5

Add Apply Parameters Action on the List data region

3. Go to the **Action** property and click the ellipses to open the **List - Action** dialog.

4. Select 'Apply Parameters' option and set the following values:
    - **Name**: paramGenreID
    - **Type**: Toggle
    - **Value**: `=Fields!GenreID.Value`
5. Click **OK** in the **List - Action** dialog.



Add Controls onto the List Data Region

6. Drag and drop the **TextBox** control onto the List data region and bind it to the `=Fields!GenreName.Value` field.
7. Drag and drop the **CheckBox** control onto the List data region, to the left of the TextBox and set its **Checked** property to the following expression:
`=IndexOf(Parameters!paramGenreID.Value, Fields!GenreIDInt.Value)>-1`.

Add Chart Data Region

Let us create a chart that shows the total profit over a period of time. Refer Create Single Line Chart tutorial to visualize the data in a single line chart.

8. Drag and drop the Chart data region onto the design area, below the List data region. The **Chart Wizard** dialog appears with an option to select the data and the chart type.
9. Select the **Dataset Name** as 'SalesByGenre' and the **Chart Type** as 'Line'.
10. Click **Next** to proceed. Here, you need to specify the line settings. We will define a data series value to display the profit values along the horizontal axis.

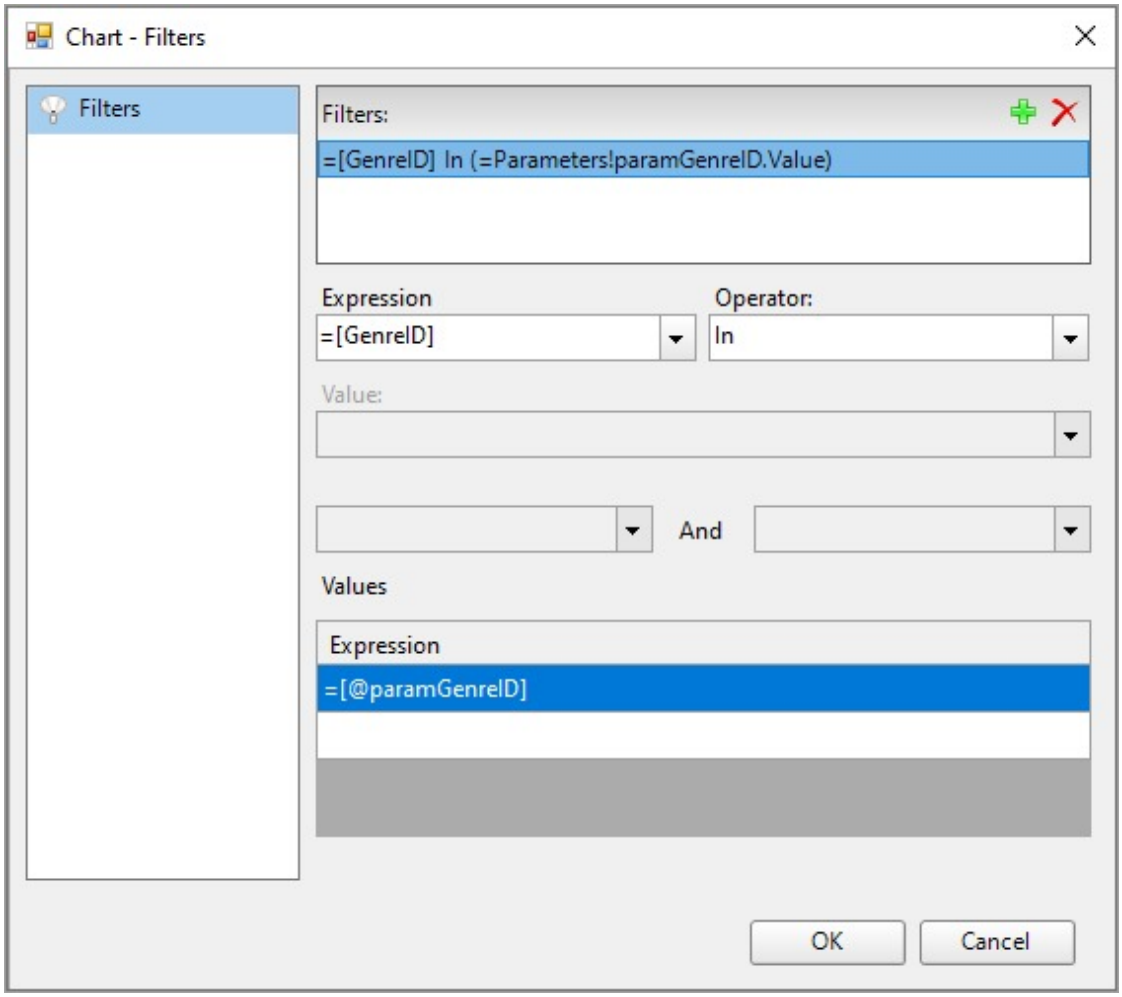
Field	Aggregate

`=[Profit]`	Sum

11. In **Choose Data Categories**, set Field to `=[SaleDate]` and **Sort Direction** to `'Ascending'`. We will add more customizations to the category in later steps.
12. Click **Next** to preview your chart.
13. Click **Finish** to complete adding the Chart.
14. Expand the width of the Chart control to the width of the List data region in the design area by dragging the side marker.
15. From the **Report Explorer**, select 'Plot - Plot1' node and set the **LineAspect** property to 'Spline'.
16. Select the chart's **X-Axis** and from the Properties Panel, set the **Labels** > **Format** to 'd' to denote date and the **Labels** > **LabelsAngle** to '-45'.
17. Select the chart's **Y-Axis** and from the Properties Panel, set the **Labels** > **Format** to 'c' to denote currency.

Add Filter to the Chart Data Region (Cross Filter)

18. From the Report Explorer, select 'Chart'.
19. Go to the **Filters** property and click the ellipses to open the **Chart - Filters** dialog.
20. In the **Chart - Filters** dialog, click the Add (+) icon to add a filter for the chart.
    - **Expression:** `=Fields!GenreID.Value`
    - **Operator: In**
    - **Values** > **Expression:** `=Parameters!paramGenreID.Value`.
    - Click **OK.**



Customize Report Appearance

You can do some customizations on the report, especially on the chart using the smart panels. For chart customization, refer the 'Set Advanced Customization' section in the Create Single Line Chart tutorial.

Preview Report

The final report is shown at the beginning of this page. Note that the preview shown is in WebDesigner preview (similar to JSViewer). Ensure that you add System.Data.SQLite or System.Data.SQLite.Core nuget package and add the ActiveReports.config file to your project. See Configure ActiveReports for more information.

Create Multi-Column Layout (or Columnar report)

This topic describes creating a columnar report layout in RDLX and Page reports.

RDLX Report

There are two ways to create a Columnar layout in an RDLX report.

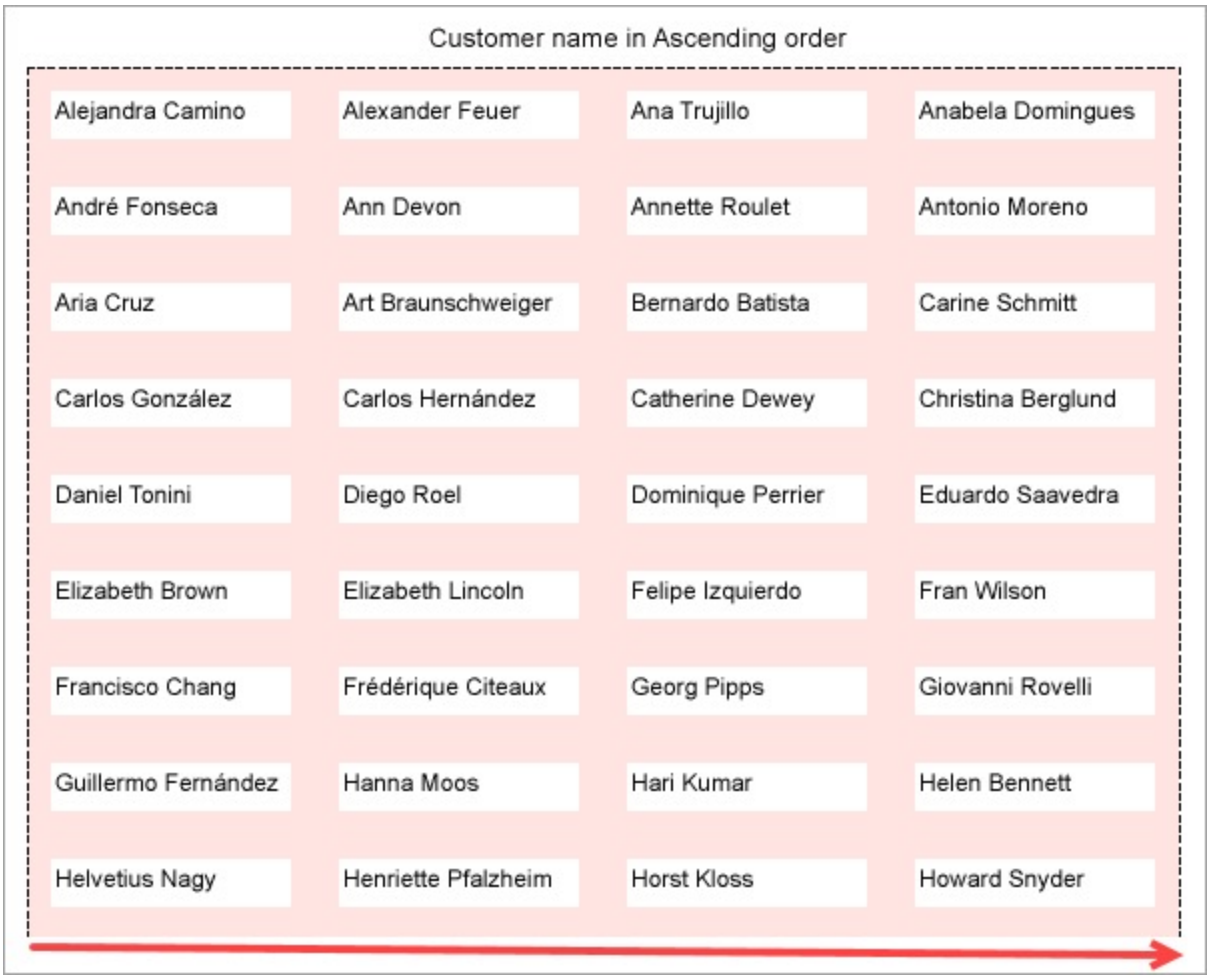
Using RowsOrColumnsCount and Count properties of the List data region

List data region available in Page and RDLX reports (RDLX and RDLX Dashboard), supports generating multiple columns or rows with the help of the following properties:

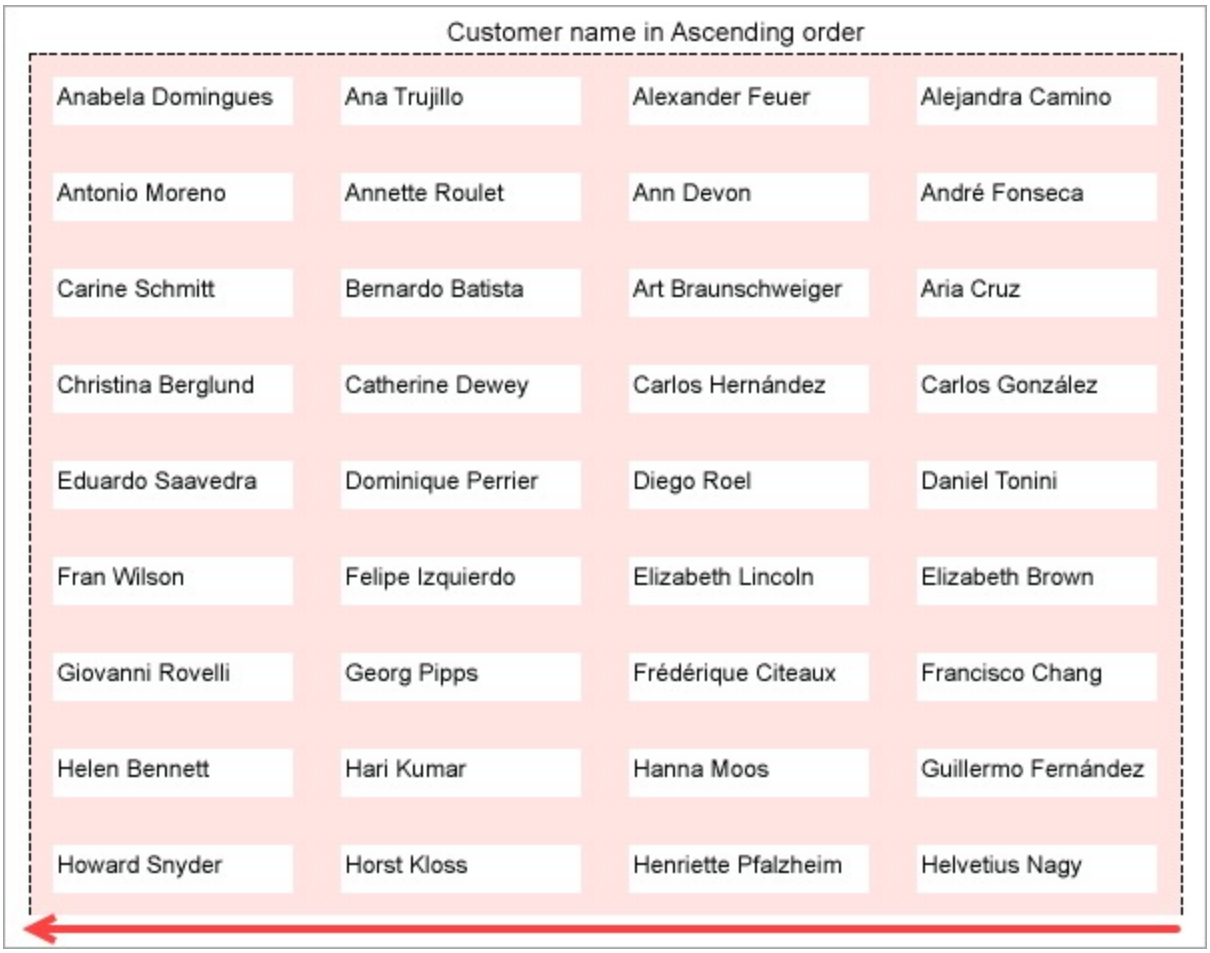
- **RowsOrColumnsCount**, which is the number of rows or columns (By default = 1)
- **GrowDirection**, which is the list layout grow direction (Row, RowReverse, Column (default), ColumnReverse)

If the **GrowDirection** = Column and the **Count** = number of columns say 4, then the data is rendered horizontally in 4 columns. The rest of the list is populated based on the data region's fixed size. Also, if the GrowDirection = Row and the Count = number of rows say 8, then the data is rendered vertically in 8 rows. The rest of the list is populated based on the data region's fixed size.

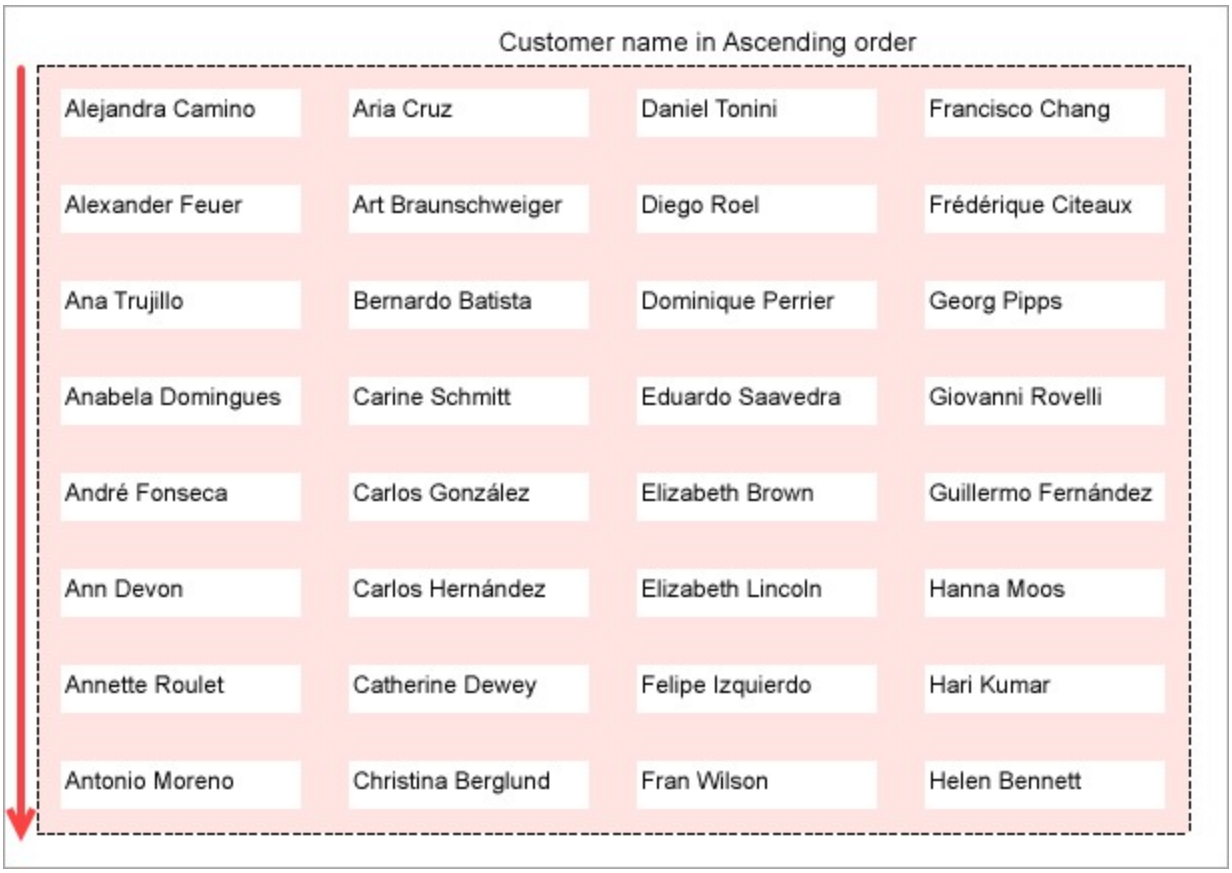
- **RowsOrColumnsCount**=4 and **GrowDirection** = Column



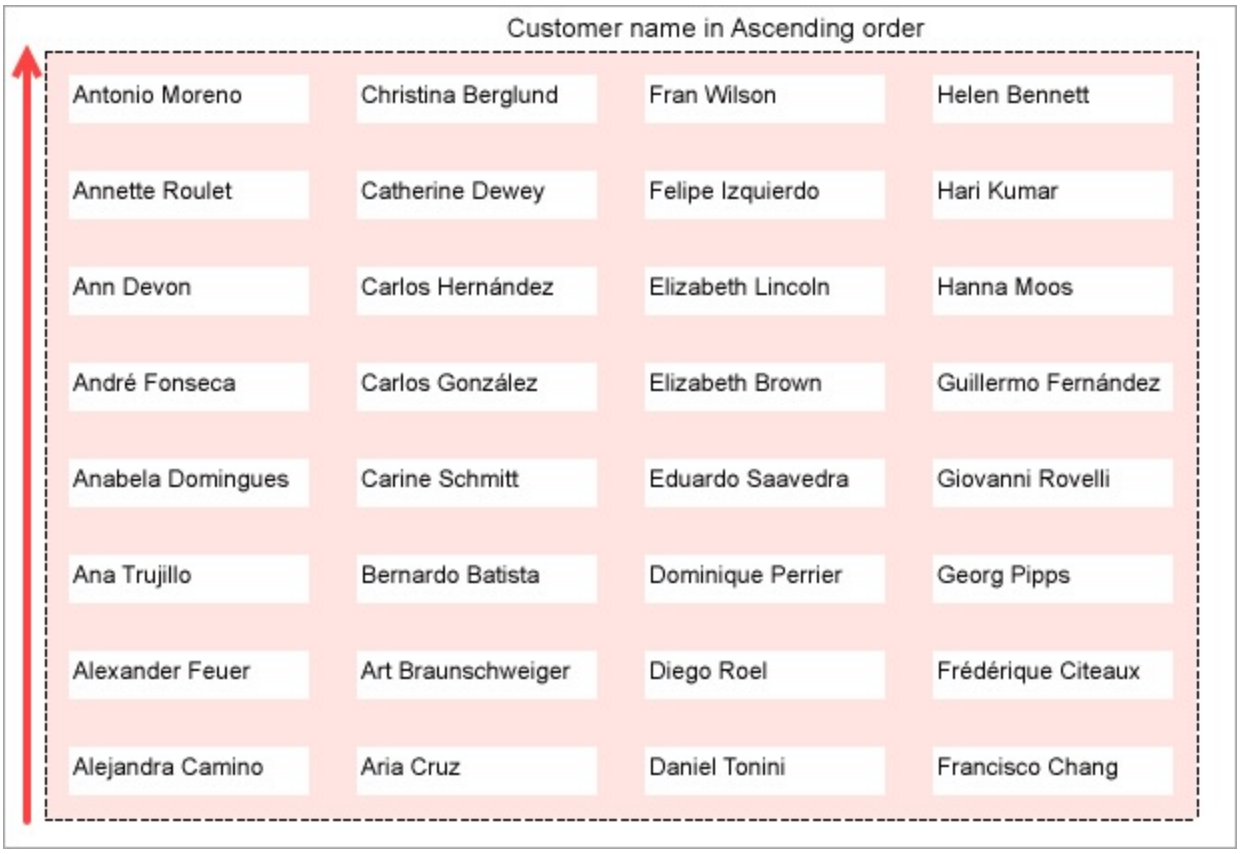
- **RowsOrColumnsCount**=4 and **GrowDirection** = ColumnReverse



- **RowsOrColumnsCount**=8 and **GrowDirection** = Row



- **RowsOrColumnsCount**=8 and **GrowDirection** = RowReverse



Go through the tutorial on Create Gauge Chart, which provides step-wise instructions on how a multi-column list layout is used for gauge charts to display the units in stock in each category of products.

Using the Columns property of the Report's Body

This section talks about creating a columnar report layout in an RDLX report by using the **Columns** property of the report's Body.

The final report will look as shown.

Alejandra Camino

Gran Vía, 1

Madrid

(91) 745 6200

Alexander Feuer

Heerstr. 22

Leipzig

0342-023176

Ana Trujillo

Avda. de la Constitución 2222

México D.F.

(5) 555-4729

Anabela Domingues

Av. Inês de Castro, 414

São Paulo

(11) 555-2167

André Fonseca

Av. Brasil, 442

Campinas

(11) 555-9482

Ann Devon

35 King George

London

(171) 555-0297

Annette Roulet

1 rue Alsace-Lorraine

Toulouse

61.77.61.10

Antonio Moreno

Mataderos 2312

México D.F.

(5) 555-3932

Aria Cruz

Rua Orós, 92

São Paulo

(11) 555-9857

Art Braunschweiger

P.O. Box 555

Lander

(307) 555-4680

Bernardo Batista

Rua da Panificadora, 12

Rio de Janeiro

(21) 555-4252

Carine Schmitt

54, rue Royale

Nantes

40.32.21.21

Create a Report

In the ActiveReports Designer, create a new RDLX Report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option.
2. In the dialog, select the **General** page and enter a name for the data source.
3. Select 'SQLite Provider'. See Custom Data Provider for more information.
4. Enter the Connection String similar to the following based on the location of nwind.db file:

Connection String
`data source= C:\data\nwind.db;`

5. Click OK.
6. In the **DataSet** dialog that appears, set the Name as 'DataSet1'.
7. Go to the **Query** page and enter a query in the Query textbox in the following format:

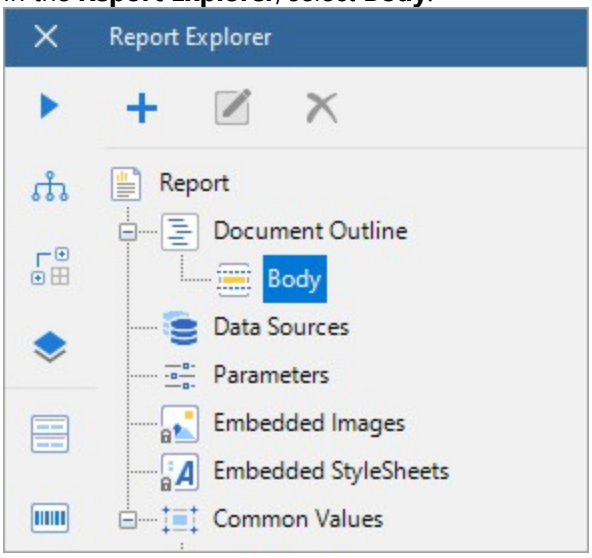
DataSet Query
`Select * from Customers order by ContactName`

8. Click the **Validate DataSet** icon to validate the query. If there is no warning, it means the query is validated. Then click **OK** to close the dialog.

Design Report Layout

Create a Column Layout for the Report

1. In the **Report Explorer**, select **Body**.



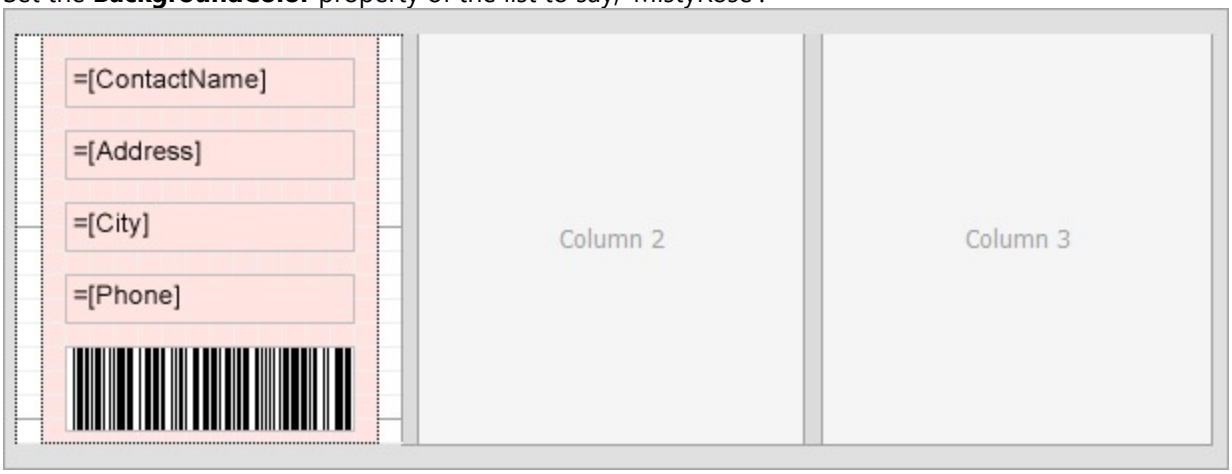
2. Set the following properties in the Properties panel:

Property Name	Property Value
Columns	3

ColumnSpacing	0.10in
Size	2in, 2.15in

Populate Data in Report

3. Drag and drop the **List** data region onto the design surface.
4. Go to Report Explorer, expand the **DataSet1** node, and drag and drop the following fields inside the List data region, one below the other, with their **Value** property set to the following fields:
   - `=Fields!ContactName.Value`
   - `=Fields!Address.Value`
   - `=Fields!City.Value`
   - `=Fields!Phone.Value`
5. From the toolbox, drag and drop the **Barcode** control below `=Fields!Phone.Value` field, inside the List data region, and set its **Value** property to `=Fields!PostalCode.Value`.
6. Ensure that the **DataSetName** property of the List data region is set to 'DataSet1'.
7. You will also need to adjust the **Size** (for example, 1.75in, 2.125in in our case) of the List data region to accommodate the data in the specified columns.
8. Set the **BackgroundColor** property of the list to say, 'MistyRose'.



9. Improve the appearance of the report.

Preview Report

The final report is shown **here**.

Page Report

There are two ways to create a Columnar layout in Page report.

Using RowsOrColumnsCount and Count properties of the List data region

List data region can be used in Page report in a similar **way** as RDLX report, to create columnar reports, such as an address labels report. See this Create Address Labels in Page Report topic for details on creating a columnar report.

Using OverflowPlaceholder control

See Create Columnar Reports with OverflowPlaceholder Control topic for details.

Create Address Labels in Page Report

With Page reports, you can create an address label report using a List data region.

> **Note:** The report connects to NWIND.db file that can be downloaded from GitHub.

The final report will look as shown.

Create a Report

In the ActiveReports Designer, create a new Page Report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option.
2. In the dialog, select the **General** page and enter a name for the data source.
3. Select 'SQLite Provider'. See Custom Data Provider for more information.
4. Enter the following Connection String similar to the following based on the location of the nwind.db file:

Connection String
`data source= C:\data\nwind.db;`

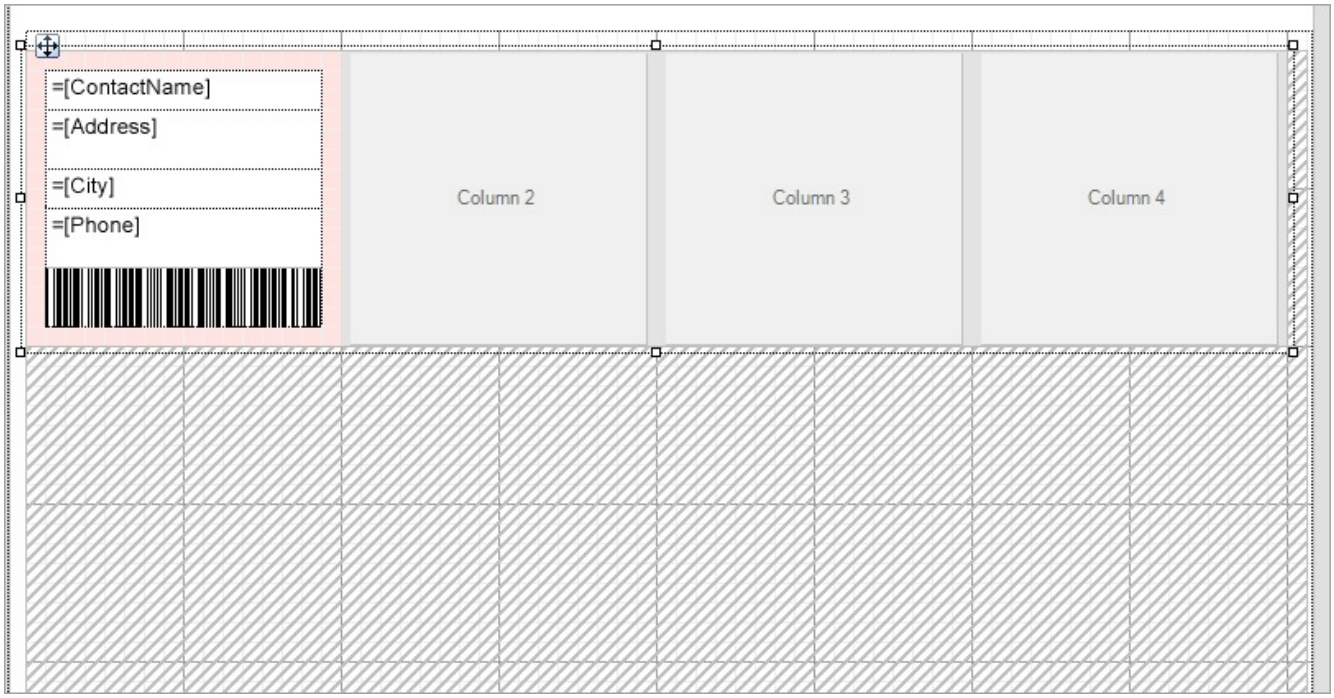
5. Click OK.
6. In the **DataSet** dialog that appears, set the Name as 'DataSet1'.
7. Go to the **Query** page and enter the following query in the Query textbox :

DataSet Query
`Select * from Customers order by ContactName`

8. Click the **Validate DataSet** icon  at the top right hand corner above the Query box to validate the query. If there is no warning, it means the query is validated. Then click **OK** to close the dialog.

Design Report Layout

1. Drag and drop **List** data region onto the design surface.
2. Go to Report Explorer, expand **DataSet1** node and drag and drop the following fields inside the List data region, one below the other, with their **Value** property set to the following fields:
   - `=Fields!ContactName.Value`
   - `=Fields!Address.Value`
   - `=Fields!City.Value`
   - `=Fields!Phone.Value`
3. From the toolbox, drag and drop the **Barcode** control below `=Fields!Phone.Value` field, inside the List data region, and set its **Value** property to `=Fields!PostalCode.Value`.
4. Select **List** data region and from the Properties panel, set the following:
   - **GrowDirection**: Column
   - **RowsorColumnsCount**: 4
5. Ensure that the **DataSetName** property of the List data region is set to 'DataSet1'.
6. You will also need to adjust the **FixedSize** and the **Size** of the List data region to accommodate the data in the specified columns and control the number of records displayed on each page.
7. Set the **BackgroundColor** property of the list to say, 'MistyRose'.



8. Improve the appearance of the report.

Preview Report

The final report is shown at the beginning of this page.

Create Top N Report

Top N reports are beneficial where there is a need to display only the specific number of top-most rows or records from a table or list. You can create this type of report by using the **TopN** operator in dataset filters or modifying the query while creating a dataset. **TopN** operator can be used in dataset filters for all data sources. Your data will be more meaningful if you also specify the order of rows.

Consider a scenario where the user wants to display the top ten movie records.

1. Create a new Page/RDLX report and bind the data to 'reels.db'. See Custom Data Provider for more information. The data source connection string in the **Report Data Source** dialog will be as follows:

Data Source Connection String
`data source = C:\Data\reels.db;`

2. In the **DataSet** dialog, go to the **Query** tab and enter the following query to fetch the data ordered by user rating from highest to lowest:

Dataset Query
`SELECT * FROM Movie ORDER BY UserRating Desc`

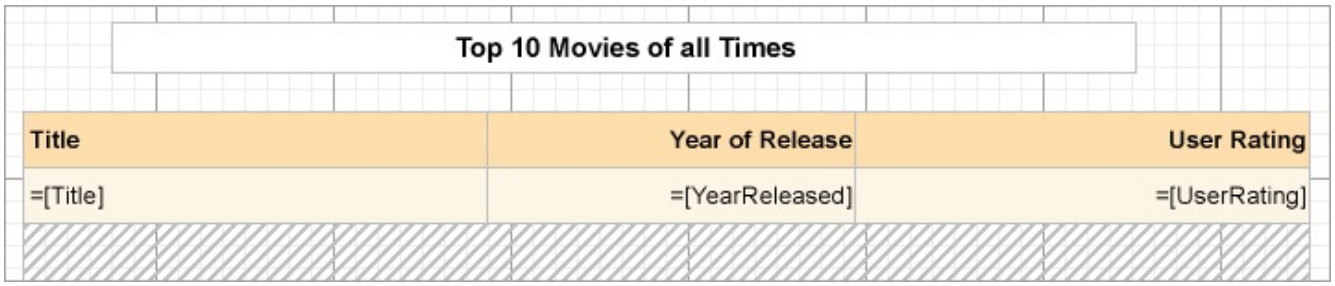
3. Navigate to the **Filters** page and fill in the following entries to fetch the top 10 movie records from the Movie table:

   Expression: `=[UserRating]`

   Operator: `TopN`

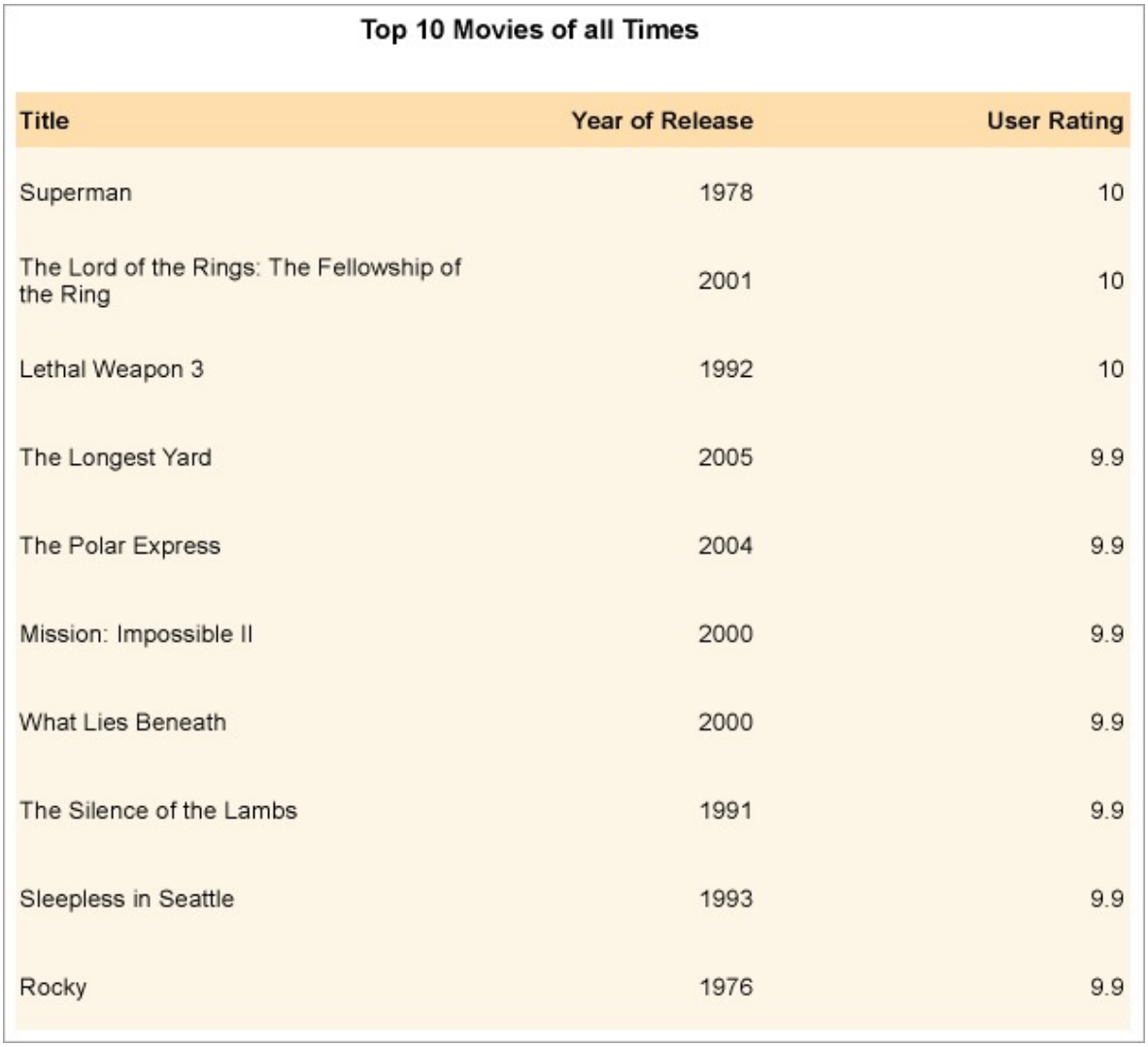
   Value: 10

4. Click **OK** to close the dialog.
5. Drag-drop the Table data region onto the design surface.
6. In the **Report Explorer**, expand the DataSet node and drag and drop the following fields inside the cells of the details row:
   `[Title]`
   `[YearReleased]`
   `[UserRating]`
7. Drag-drop a **TextBox** control for the report heading.



8. Preview the report. You'll see only **10** records in your report.

The following image illustrates Top N Report displaying top 10 movie records:



Top 10 Movies of all Times

Title	Year of Release	User Rating
Superman	1978	10
The Lord of the Rings: The Fellowship of the Ring	2001	10
Lethal Weapon 3	1992	10
The Longest Yard	2005	9.9
The Polar Express	2004	9.9
Mission: Impossible II	2000	9.9
What Lies Beneath	2000	9.9
The Silence of the Lambs	1991	9.9
Sleepless in Seattle	1993	9.9
Rocky	1976	9.9

The following sections provide sample dataset queries to obtain top N data from other data sources.

Sqlite

DataSet Query
`SELECT * FROM Movie LIMIT 10`

OLEDB, ODBC, SQL

DataSet Query
`SELECT Top 10 * FROM Movie`

XML

DataSet Query
`//countries/country[position() <= 10]`

JSON

DataSet Query
`$.Customers[:10]`

Create a Red Negatives or a Green Bar Report

A Red Negatives report is a report type that shows negative values in red color such that these values meet the requirements set in a conditional expression. A Green Bar report can be similarly created by alternating the background color of a data region like a Table using conditional formatting.

The following steps show how to generate a report with negative values in red negatives.

1. Create a new Page/RDLX report and bind the data to 'reels.db'. See Custom Data Provider for more information.
2. In the **DataSet** dialog that appears, go to the **Query** page and enter a query in the Query textbox in the following format:

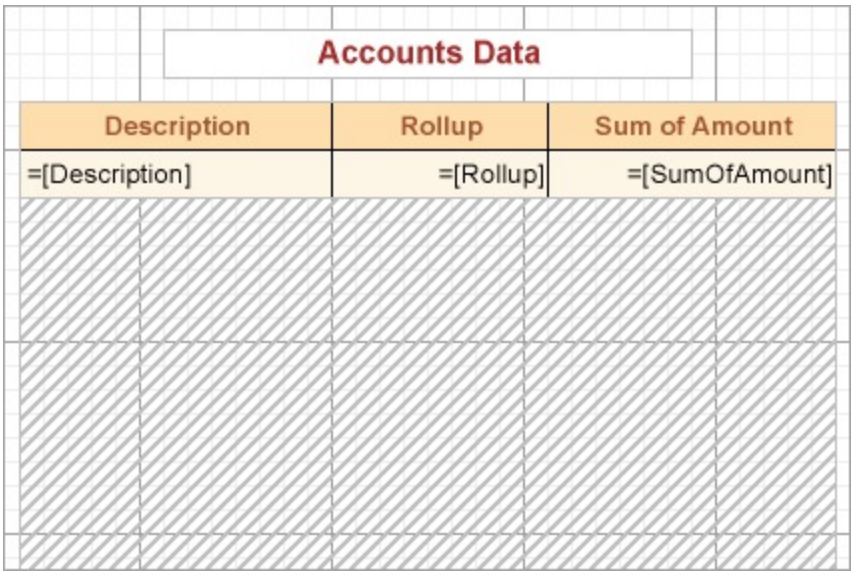
DataSet Query
`SELECT AccountsChart.AccountId, AccountsChart.ParentId, AccountsChart.Description, AccountsChart.Rollup, Expenses.ExpenseDate, Sum(Expenses.Amount) AS SumOfAmount FROM AccountsChart LEFT JOIN Expenses ON AccountsChart.AccountId=Expenses.AccountID GROUP BY AccountsChart.AccountId, AccountsChart.ParentId, AccountsChart.Description, AccountsChart.Rollup, Expenses.ExpenseDate ORDER BY Expenses.ExpenseDate, AccountsChart.AccountId;`

3. From the Toolbox, drag and drop a Table data region onto the design surface.
4. In the **Report Explorer**, expand the DataSet node and drag and drop following fields inside the cells of the details row:
   `[Description]`
   `[Rollup]`
   `[SumOfAmount]`
5. In the same table, select any cell (Textbox) that displays integer values, for example `[Rollup]`.
6. In the **Properties** panel, set the following expression in the **Color** property:

   ```
   =iif(Fields!Rollup.Value < 0, "Red", "Black")
   ```

   > **Note:** In general, the expression will be `=iif(Fields!`***`FieldName`***`.Value < 0, "Red", "Black")` where **FieldName** refers to field that the textbox contains (of integer type).

7. To avoid repetition of `[Description]` in detail row, apply detail grouping based on `[Description]`field. See Detail Grouping for more information.
8. Drag-drop a Textbox control for the report heading.



9. Preview the report.

The following image illustrates a report that contains negative values in red:

Accounts Data

Description	Rollup	Sum of Amount
Assets	0	
Liabilities	0	
Net Sales	1	
Gross Sales	1	
Cost of Goods Sold	-1	
Total Expense	-1	
Net Income	1	
General & Administration	1	$42,400.00
Information Systems	1	$6,560.00
Marketing	1	$640.00
Lease	1	$12,948.00

For a green bar report, click the row handle to the left of the detail row of a **Table** data region and set the following expression in the **BackgroundColor** property:

```
=iif(RowNumber(Nothing) Mod 2, "PaleGreen", "White")
```

On previewing the report, you will notice that every alternate detail the report displays has a green background as depicted by the following image.

DVD STOCK

Title	In Stock	Store Price
Snow White and the Seven Dwarfs	5	$16.95
Gone with the Wind	14	$18.95
Bambi	5	$19.99
One Hundred and One Dalmatians	7	$8.95
Mary Poppins	2	$11.95
Doctor Zhivago	17	$14.99
The Jungle Book	7	$9.99
Butch Cassidy and the Sundance Kid	15	$9.99
Love Story	16	$18.99
The Godfather	11	$22.99
The Sting	8	$15.95
The Towering Inferno	6	$5.00
Blazing Saddles	7	$10.99
Jaws	11	$5.00
One Flew Over the Cuckoo's Nest	3	$13.95
Rocky	9	$23.95
Close Encounters of the Third Kind	23	$15.99
Star Wars	12	$13.99
Saturday Night Fever	18	$15.99

Note that with Detail Grouping applied on a table, the background color will not appear as expected.

Apply Theme at Runtime Using Dynamic Expression

This topic explains how to apply a theme at runtime by using a dynamic expression so that the applied theme depends on the parameter's value.

Consider a sample report 'PurchaseReport.rdlx' which can be downloaded from GitHub: ..\Samples18\DesignerPro\ReportsGallery\Reports\Page Report\Other.

Add Parameter for User Input

1. Open 'PurchaseReport.rdlx' report.
2. From the **Report Explorer**, right-click the **Parameters** node and select **Add Parameter.**
3. In the General tab of the **Report - Parameters** dialog, specify the following:

- **Name**: ThemeParameter
- **Data type**: String
- **Text for prompting users for a value**: Select a Theme

4. Go to the **Available Values** tab, select **Non-queried** option and add two themes with Label and Value settings as follows:

S.no.	Label	Value
1	Theme1	**.\theme1.rdlx-theme ('theme1.rdlx-theme' in the on-line documentation)**
2	Theme2	**.\theme2.rdlx-theme ('theme2.rdlx-theme' in the on-line documentation)**

5. Click **OK**.

Add Theme to the Report

6. Select the **Report menu** and click open the **Report Properties**.
7. Go to the **Themes** page and add a theme with expression:

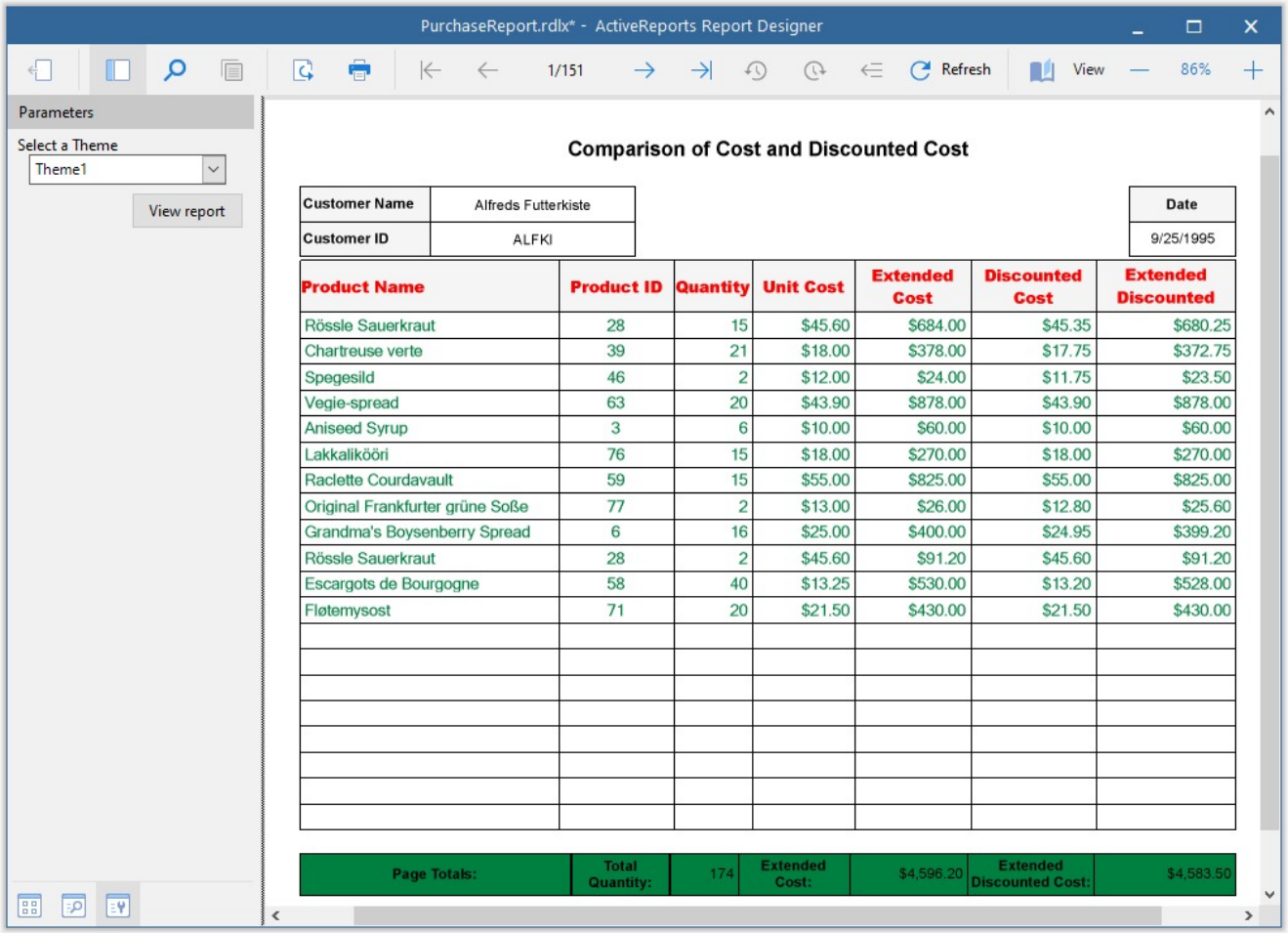
```
=Parameters!ThemeParameter.Value
```

Apply Theme to Report Controls

8. Lets us apply themes to the table header and detail row. First, select the **Header** row of the table, and set following properties:
   - Color: `=Theme.Colors!Dark1`
   - Font > FontStyle: `=Theme.Fonts!MajorFont.Style`
   - Font > FontFamily: `=Theme.Fonts!MajorFont.Family`
   - Font > FontSize: `=Theme.Fonts!MajorFont.Size`
   - Font > FontWeight: `=Theme.Fonts!MajorFont.Weight`

9. Select the **Detail** row of the table and set the following properties:
   - Color: `=Theme.Colors!Light1`
   - Font > FontStyle: `=Theme.Fonts!MinorFont.Style`
   - Font > FontFamily: `=Theme.Fonts!MinorFont.Family`
   - Font > FontSize: `=Theme.Fonts!MinorFont.Size`
   - Font > FontWeight: `=Theme.Fonts!MinorFont.Weight`

10. Select all the text boxes below the table and set the **BackgroundColor** property to `=Theme.Colors!Light1`
11. Preview the report and select any one of the two themes and click **View Report** to view the theme applied to the report.



Note: Since the report needs to be re-rendered for theme to apply, on preview, you can not change the theme of the report by changing the theme selection from the report parameter. The report prompts for the report parameter for the first time and shows the result on clicking View Report, however there is no change in the applied theme if you change the theme and click View Report, unless there are updates in report that causes report to re-render.

Add Page Numbering

You can choose the page numbering format for your Page reports/RDLX reports by selecting from a list of pre-defined formats or by creating a custom page numbering expression.

Adding page numbers to a report

There are two ways to add page numbering to a report.

- From the **Report Explorer**, under the **Common Values** node, drag a pre-defined page numbering format and drop it directly onto the report design surface.
- Or add a TextBox control to the report design surface and in the **Value** property of the control, use the Expression Editor Dialog to select a page numbering expression from the **Common Values** node.

Usually a page number is added to a report header or footer, but you can add it anywhere on the layout page.

Pre-defined page numbering formats

You can find the pre-defined page numbering formats listed in the Report Explorer under the **Common Values** node, and in the Expression Editor under the Common Values field.

Predefined Format Descriptions

Numbering Format	Description
Page N of M	This format displays the current page out of the total number of pages in the report. Here **N** signifies the current page of a report and **M** the total number of report pages. Use the following expression to set this page numbering format: `="Page " & Globals!PageNumber & " of " & Globals!TotalPages`
Page N of M (Section)	This format displays the current page out of the total number of pages of a grouped report section. Here **N** signifies the current page of a grouped report section and **M** signifies the total number of pages in a grouped report section. Use the following expression to set this page numbering format: `="Page " & Globals!PageNumberInSection & " of " & Globals!TotalPagesInSection`
Page N of M (Cumulative)	This format displays the current page out of the total number of cumulative pages in a report. Here **N** signifies the current page of the report and **M** signifies the total number of cumulative pages in a report. Use the following expression to set this page numbering format: `="Page " & Globals!CumulativePageNumber & " of " & Globals!CumulativeTotalPages`
Page Number	This format displays only the current page number of a report. Use the following expression to set this page numbering format: `=Globals!PageNumber`
Page Number (Section)	This format displays only the current page number of a specific grouped report section. Use the following expression to set this page numbering format: `=Globals!PageNumberInSection`
Total Pages	This format displays only the total number of pages in the report. Use the following expression to set this page numbering format: `=Globals!TotalPages`
Total Pages (Section)	This format displays only the total number of pages in specific grouped report section. Use the following expression to set this page numbering format: `=Globals!TotalPagesInSection`
Cumulative Page Number	This format displays only the current cumulative page number of the report. Use the following expression to set this page numbering format: `=Globals!CumulativePageNumber`
Cumulative Total Pages	This format displays only the total number of cumulative pages in a report. Use the following expression to set this page numbering format: `=Globals!CumulativeTotalPages`

Tip: In addition to modifying the page numbering expression in the Expression Editor, you can also modify the pre-defined formats directly in the control on the design surface.

Custom page numbering formats

Use the following steps to create your own page numbering format.

Create a custom page numbering format

1. From the toolbox, drag and drop a Textbox control onto the report design surface.
2. With the Textbox selected on the report, under the Properties window, click the Property dialog link. This is a command to open the TextBox dialog. See Properties Panel for more on how to access commands.
3. In the **TextBox - General** dialog that appears, in the **Value** field, enter a page numbering expression like the following: `=Globals!PageNumber & "/" & Globals!TotalPages`
4. Click **OK** to close the dialog.
5. Select the Preview tab. Page numbers appear in the expression format you set in the **Value** field above, in this case, for a one-Page Report, "1/1."

Show Row Number in Tablix and Table

In the Tablix and Table data regions, you can have displayed a number of rows by using the **GroupIndex** function. The **GroupIndex** function returns the index of a row or column member in a data region's group.

These steps assume that you have already added a Page Report/RDLX report template to your project. The reports use the Embedded JSON Data Source.

Follow these steps to use the GroupIndex function in Tablix and Table.

Show the row number in a Tablix data region with three row groups

GroupIndex	Region	GroupIndex	Country	GroupIndex	City	Population
0	East Asia	0	Japan	0	Tokyo	37,468,000
				1	Osaka	19,281,000
		1	China	0	Shanghai	25,582,000
				1	Beijing	19,618,000
				2	Chongqing	14,838,000
				3	Tianjin	13,215,000
1	South Asia	0	India	0	Bangalore	11,440,000
				1	Delhi	28,514,000
				2	Mumbai	19,980,000
				3	Kolkata	14,681,000

Create a Report

In the ActiveReports Designer, create a new RDLX Report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option.
2. In the dialog, select the **General** page and enter the name of the data source, 'PopulationData' and connect your report to the JSON data source as described here.

3. In the **Report Data Source** dialog, under Connection, select the **Embedded** option. Add the report data as follows and then click **OK**.

   Add the JSON embedded data

   JSON embedded data

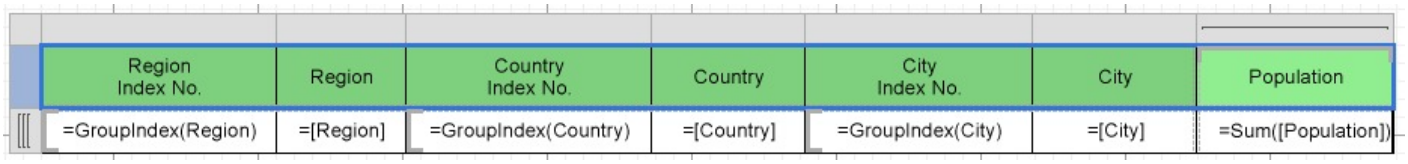
```
[{"Region":"East Asia","Country":"Japan","City":"Tokyo","Population":37468000},
{"Region":"East Asia","Country":"Japan","City":"Osaka","Population":19281000},
{"Region":"East Asia","Country":"China","City":"Shanghai","Population":25582000},
{"Region":"East Asia","Country":"China","City":"Beijing","Population":19618000},
{"Region":"East Asia","Country":"China","City":"Chongqing","Population":14838000},
{"Region":"East Asia","Country":"China","City":"Tianjin","Population":13215000},
{"Region":"South Asia","Country":"India","City":"Bangalore","Population":11440000},
{"Region":"South Asia","Country":"India","City":"Delhi","Population":28514000},
{"Region":"South Asia","Country":"India","City":"Mumbai","Population":19980000},
{"Region":"South Asia","Country":"India","City":"Kolkata","Population":14681000}]
```

4. In the Report Explorer, right-click the data source and select the **Add Data Set** option.
5. In the **DataSet** dialog that appears, go to **Query.** Enter the query as follows into the Query textbox and click **OK**.

   JSON data set query

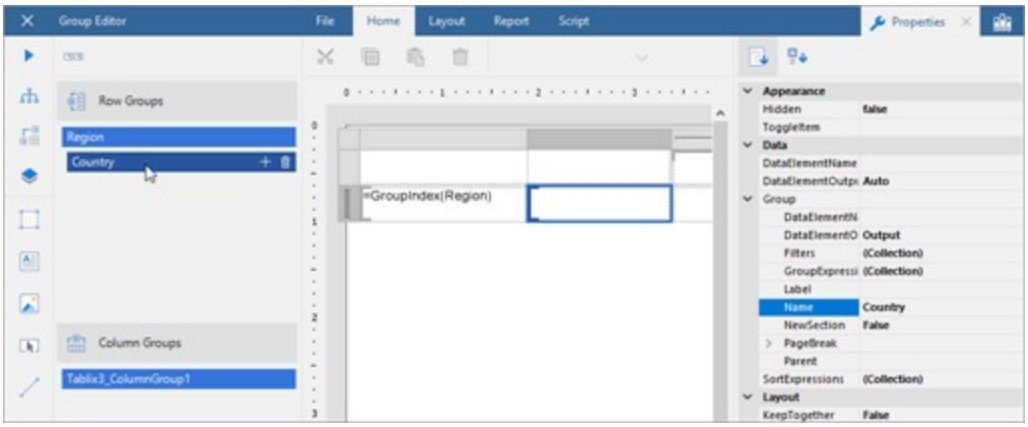
```
$.[*]
```

Design Report Layout



From the toolbox, drag a **Tablix** data region onto the report design surface and set the **DataSetName** property to the name of the dataset.

1. In the **Group Editor**, select the Tablix row group.
   1. Go to **Group** property and set the **Name** property to **Region** in the Properties grid**.**
   2. Go to **GroupExpressions** property and open the **Expression Collection Editor**.
   3. Click **Add** and set the **Expression** to `=Fields!Region.Value`.
2. Right-click TextBox3, select **Expression...** and in the **Expression Editor**, enter `=GroupIndex(Region)`.
3. In the Group Editor, go to Row Groups. Select **Region** and click **Add New Group**, then **Add Group > Child Group**.
   1. With the new group selected, go to **Group** property and set the **Name** property to **Country** in the Properties grid**.**
   2. Go to **GroupExpressions** property and open the **Expression Collection Editor**.
   3. Click **Add** and set the **Expression** to `=Fields!Country.Value`.



4. Right-click TextBox5 and select **Expression...**
5. In the Expression Editor, enter `=GroupIndex(Country)`.
6. In the Group Editor, go to Row Groups. Select **Country** and click **Add New Group**, then **Add Group > Child Group**.
    1. With the new group selected, go to **Group** property and set the **Name** property to **City** in the Properties grid**.**
    2. Go to **GroupExpressions** property and open the **Expression Collection Editor**.
    3. Click **Add** and set the **Expression** to `=Fields!City.Value`.
7. Right-click TextBox7 and select **Expression...**
8. In the Expression Editor, enter `=GroupIndex(City)`.
9. Hover over the column handler next to the Region row group and click **Insert Column**.

10. Set the **Value** of TextBox9 to =Fields!Region.Value.
11. Hover over the column handler next to the Country row group and click **Insert Column**.
12. Set the **Value** of TextBox11 to =Fields!Country.Value.
13. Hover over the column handler next to the City row group and click **Insert Column**.
14. Set the **Value** of TextBox13 to =Fields!City.Value.
15. Set the values of the last column in the Tablix as follows.

TextBox	Value
TextBox2	Population
TextBox4	`=Sum(Fields!Population.Value)`

16. Fill-in the Table Header to display the columns they represent.
17. Select the Tablix and set the **BorderStyle** property to 'Solid' to view the merged cells clearly.
18. Select the Tablix header row and set the **TextAlign** property to 'Center'. Repeat this step for the detail row.
19. Select TextBox4 and set its **Format** property to 'n0'.

Show the row number in a Table data region with two table groups

Region Index No.	Region	Country Index No.	Country	Population
East Asia				
Japan				
0	East Asia	0	Japan	37,468,000
0	East Asia	0	Japan	19,281,000
China				
0	East Asia	1	China	25,582,000
0	East Asia	1	China	19,618,000
0	East Asia	1	China	14,838,000
0	East Asia	1	China	13,215,000
South Asia				
India				
1	South Asia	0	India	11,440,000
1	South Asia	0	India	28,514,000
1	South Asia	0	India	19,980,000
1	South Asia	0	India	14,681,000

Create a Report

In the ActiveReports Designer, create a new RDLX Report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option.
2. In the dialog, select the **General** page and enter the name of the data source, 'PopulationData' and connect your report to the JSON data source as described here.
3. In the **Report Data Source** dialog, under Connection, select the **Embedded** option. Add the report data as follows and then click **OK**.

Add the JSON embedded data

JSON embedded data

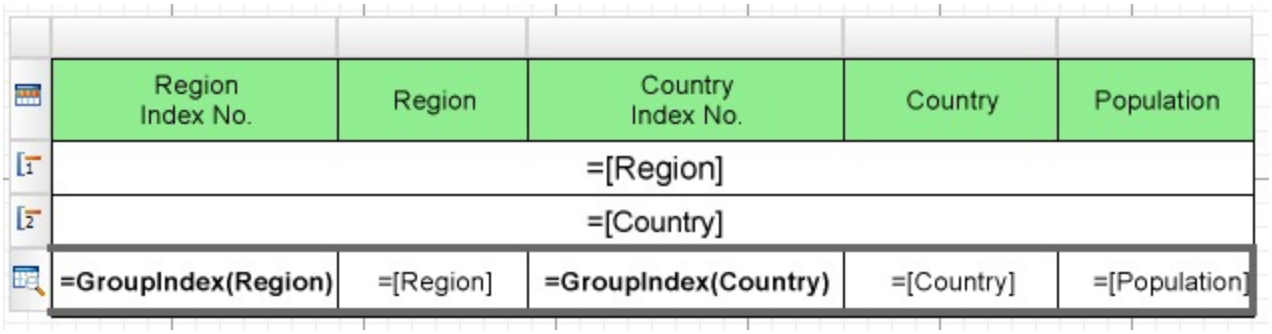
```
[{"Region":"East Asia","Country":"Japan","City":"Tokyo","Population":37468000},
{"Region":"East Asia","Country":"Japan","City":"Osaka","Population":19281000},
```

```
{"Region":"East Asia","Country":"China","City":"Shanghai","Population":25582000},
{"Region":"East Asia","Country":"China","City":"Beijing","Population":19618000},
{"Region":"East Asia","Country":"China","City":"Chongqing","Population":14838000},
{"Region":"East Asia","Country":"China","City":"Tianjin","Population":13215000},
{"Region":"South Asia","Country":"India","City":"Bangalore","Population":11440000},
{"Region":"South Asia","Country":"India","City":"Delhi","Population":28514000},
{"Region":"South Asia","Country":"India","City":"Mumbai","Population":19980000},
{"Region":"South Asia","Country":"India","City":"Kolkata","Population":14681000}]
```


4. In the Report Explorer, right-click the data source and select the **Add Data Set** option.
5. In the **DataSet** dialog that appears, go to **Query.** Enter the query as follows into the Query textbox and click **OK**.

JSON data set query
`$.[*]`

Design Report Layout



1. From the toolbox, drag a **Table** data region onto the report design surface.
2. Set the **DataSetName** property to the name of the dataset.
3. Select the **Details** row, right-click, and select **Insert Group** from the context menu.
4. In the **Table - Groups** dialog, add two groups and set the **Name** and **Group on** > **Expression** in the following sequence as follows:

S.no.	Name	Expression
1.	Region	`=Fields!Region.Value`
2.	Country	`=Fields!Country.Value`

5. Click **OK** to close the dialog.
   You will see the two groups, and respective group header and group footer rows added to the table.
6. Merge the cells of the first group header row and enter `=Fields!Region.Value.`
7. Similarly, merge the cells of the second group header and enter `=Fields!Country.Value.`
8. Delete the footer rows.
9. Insert two more columns in the table. You can insert a column by selecting the last column of the table, right-clicking, and then selecting the **Insert Column to the Right**.
10. In the first textbox of the Details row, enter `=GroupIndex(Region)` to display the index number for Region table group and in the second textbox, enter `=Fields!Region.Value` to display the region name.
11. In the third textbox of the Details row, enter`=GroupIndex(Country)` to display the index number for Country table group and in the fourth textbox, enter `=Fields!Country.Value` to display the country name.
12. In the fifth textbox of the Details row, enter `=Fields!Population.Value.` and set its **Format** property to

'n0'.
13. Fill-in the Table Header to display the columns they represent.

Link Multiple Datasets to Same Data Region

Many a time, we need to display varied data from different datasets into one data region. This is now possible by using the **Lookup** function in a data region.

The **Lookup** function returns a value corresponding to a related or a common field with the same data type in another data set. It is set as an expression in the Value property of a data region's Textbox. The Lookup function in ActiveReports is similar to the Microsoft Excel's VLOOKUP.

Category	Product ID	Quantity
Report	ARNP	10
Component Set	CDAN	1
Component Set	CDEP	6
Component Set	CDWI	7
Grid	ELST	5
Grid	FGWI	7
Input Support	IMFI	9

Create a Report

In the ActiveReports Designer, create a new Page Report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by right-clicking the Data Sources node in the Report Explorer and then selecting the **Add Data Source** option.
2. In the dialog, select the **General** page and enter a name for the data source.
3. Select 'SQLite Provider'. The report connects to the SalesResult.db file can be downloaded from GitHub: ..\Samples18\Data\SalesResult.db. See Custom Data Provider for more information.

Add Dataset1

1. In the Report Explorer, right-click the data source node and select the **Add Data Set** option or select **Data Set** from the Add button.
2. In the Add Dataset that appears, select the **General** page and let the name of the dataset be **Dataset1**. This name appears as a child node to the data source icon in the Report Explorer.
3. On the **Query** page of this dialog, in the **Query** field enter the following SQL query.

```
SQL Query
```

```
SELECT M01Product.Category, M01Product.ProductID
FROM M01Product
```

4. Click the **Validate DataSet** icon  at the top right hand corner above the Query box to validate the query.
5. Click **OK** to close the dialog. Your data set and queried fields appear as nodes in the Report Explorer.

   The Dataset1 contains following fields:

   - Category
   - ProductID

Add Dataset2

- Repeat Steps 1 and 2 to add another dataset with name **Dataset2.**
- On the **Query** page of this dialog, in the **Query** field enter the following SQL query.

  SQL Query

  ```
  SELECT * FROM T01Result
  ```

- Click the **Validate DataSet** icon  at the top right hand corner above the Query box to validate the query.
- Click **OK** to close the dialog. Your data set and queried fields appear as nodes in the Report Explorer.

The Dataset2 contains following fields:

- ID
- ProductID
- Quantity
- PDate
- FY

Design Report

1. From the toolbox, drag a Table data region onto the design surface of the report.
2. Go to the Properties panel to set the properties of Table data region as follows:

Property Name	Property Value
FixedSize	4in, 4in
Location	0in, 0in
DataSetName	Dataset1

3. Hover your mouse over the text boxes of the Table Details row to access the field selection adorner and set the following fields in the table cells along with their properties.

Cell	Field
TextBox4	Category
TextBox5	ProductID

This automatically places an expression in the details row and simultaneously places a static label in the header row of the same column.

4. Select TextBox6 of the Table data region and from the Properties pane, set the following properties:

Property Name	Property Value
Value	=Lookup(Fields!ProductID.Value, Fields!ProductID.Value, Fields!Quantity.Value, "DataSet2")
TextAlign	Left

The expression in the Value property returns the value of Quantity from Dataset2, corresponding to the related data field ProductID in Dataset1.

5. Select TextBox3 of the Table data region and from the Properties pane, set the following properties:

Property Name	Property Value
Value	Quantity
TextAlign	Left

6. Select the header row using the row handle to the left and in the Properties Panel, set the **FontWeight** property to **Bold**.

Preview Report

The final report is shown at the beginning of this page.

Design Section Reports

ActiveReports Designer supports designing successful section reports driven by data.

> **Note**:
>
> - We recommend specifying the **CrossPlatform** compatibility mode for all new reports. The CrossPlatform compatibility mode is the only available mode in the WebDesigner.
> - We also recommend specifying the virtual printer setting for all new reports by adding this script to any new Section report:
>   1. Switch to the Script tab.
>   2. Add the ReportStart event.
>   3. Add this code inside the ReportStart event:
>
> ```
> this.Document.Printer.PrinterName = "";
> ```

This section covers a wide range of topics to re-mould data into meaningful insights:

Layout

Learn about the structure of section reports.

Report Settings Dialog

Learn about setting the report margins, printer settings, styles, and design layout of a report.

Aggregates Calculations

Learn what expressions and aggregates are supported in section reports.

Display Page Numbers and Report Dates

Learn about using the ReportInfo control in section reports.

Date, Time, and Number Formatting

Learn about formatting in section reports.

Interactivity

Learn about adding interactive features in section reports.

Scripting

Learn how to use C# or VB script in section reports.

Report Events

Learn how to use events to control report behavior.

Tutorials: Report Controls in Section Reports

Learn how to design the Section reports in the Report Designer from scratch using tutorials.

Tutorials: Section Report Scenarios

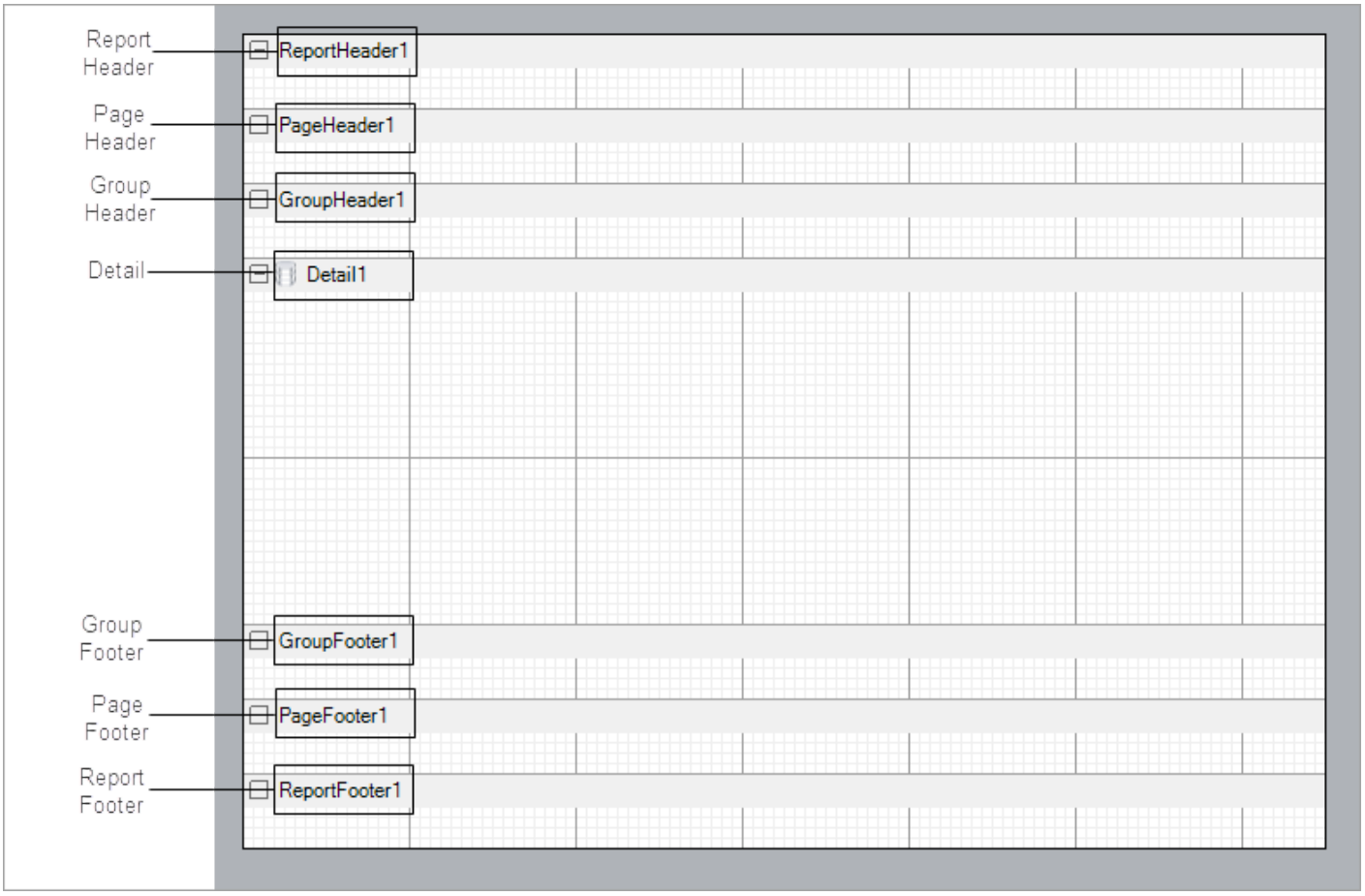
Learn how to create commonly-used section reports using tutorials.

Layout

By default, a section report is composed of three banded sections: a PageHeader, a Detail section, and a PageFooter. You can right-click the report and select **Insert** and choose other section pairs to add: ReportHeader and Footer, or GroupHeader and Footer.

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. You generally use this section 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 ('NewPage Property' in the on-line documentation)** property that you can use to add a new page before or after it renders.

The **Report Header** does not appear on a section report by default. In order to add this section, right-click the report and select **Insert > Report Header/Footer** to add a Report Header and Footer pair.

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. You can use the page header to print column headers, page numbers, a page title, or any information that needs to appear at the top of each page.

Group Header

A report can include single or nested groups, with each group having its own header and footer sections. You can insert and print the header section immediately before the detail section.

In Columnar Reports, you can 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.

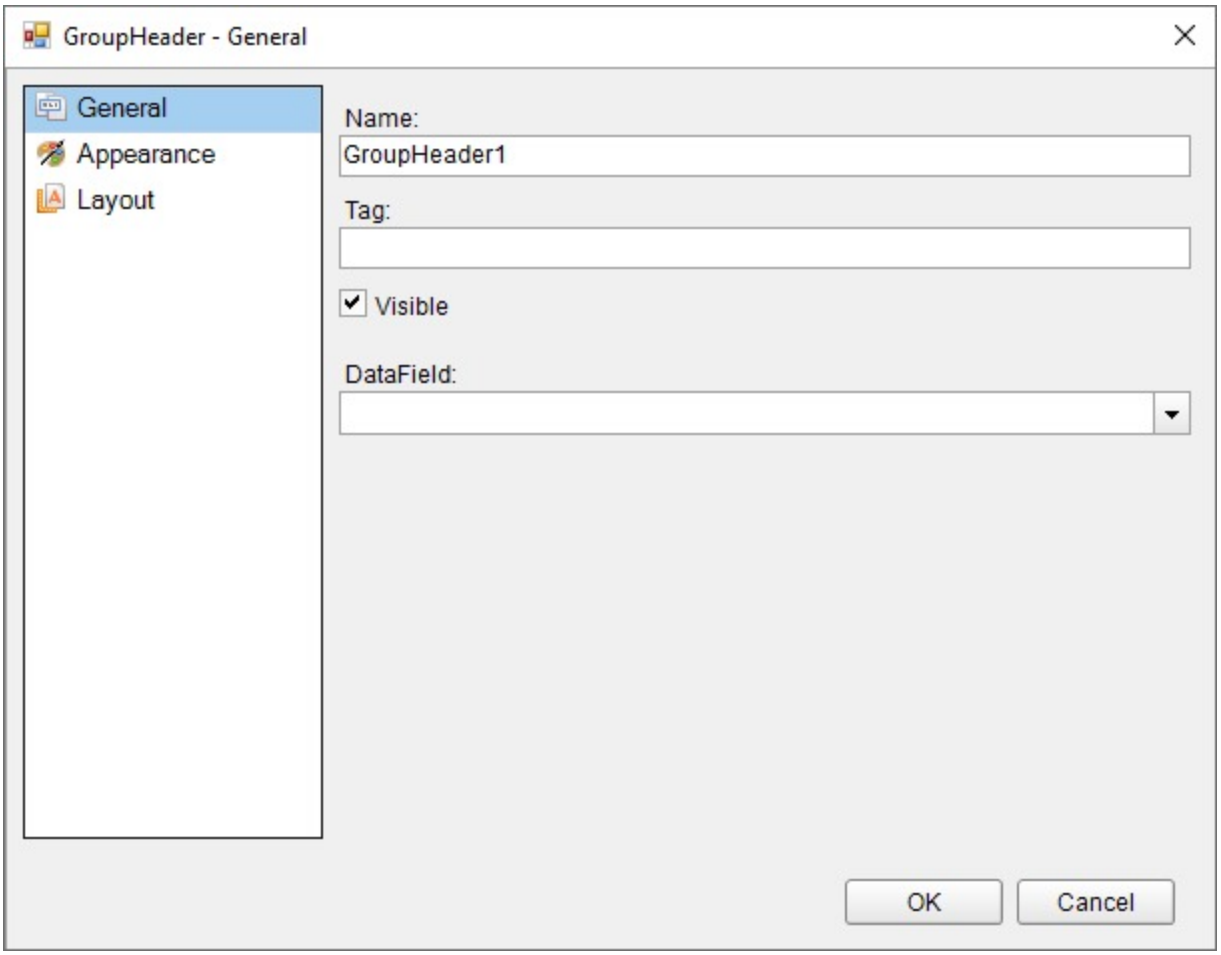
You can group data by adding a pair of group header and group footer sections to the report. These sections appear immediately above and below the Detail section.

Controls in the group header render once for each instance of the group, so you can place the column header labels to describe the data in the detail fields here.

> ⚠ **Caution**: You cannot add a header section without a corresponding footer section. If you try to do so in code, the results are unstable.

You can set the properties for the GroupHeader and GroupFooter sections in their corresponding dialogs. Following is a list of properties you can set through the options in these dialogs. Each option in the GroupHeader dialog corresponds to a property in the properties panel. To access the properties directly, select the section and open the properties window. See the associated property names in parenthesis with each dialog option below.

GroupHeader Dialog



To access the GroupHeader dialog, right click the group header and in the Properties window under the properties list where the commands are displayed, click the **Property dialog** link.

General

- **Name** (Name)**:** Indicates the name of the GroupHeader in code. It is unique for a report.
- **Tag** (Tag): Indicates the user-defined information persisted with the GroupHeader section.
- **Visible** (Visible): Checkbox to specify the visibility of the GroupHeader section.
- **DataField** (DataField): Field or expression on which you group the data.

Appearance

- **Background color** (BackColor): Dropdown list to set the background color of the GroupHeader section.

Layout

- **Insert new page** (NewPage): Dropdown list to determine whether a new page is inserted before and/or after displaying the GroupHeader section.
- **Insert new column** (NewColumn): Dropdown list to determine whether a new column (in a multi-column report) appears before and/or after displaying the GroupHeader section.
- **Repeat section** (RepeatStyle): Dropdown list to specify whether the GroupHeader section appears with every column or page that the Detail section or associated footer appears on.
- **Keep section and its footer on a single page** (GroupKeepTogether): Dropdown list to specify whether the GroupHeader section and its footer appear as a single block on the same page or not.

- **Keep section on a single page** (KeepTogether): Checkbox to specify whether the GroupHeader section appears on a single page.
- **Keep section and its footer in a single column** (ColumnGroupKeepTogether): Checkbox to specify whether the GroupHeader section and its footer appear as a single block in the same column.
- **Keep section underneath the following section** (UnderlayNext): Checkbox to specify whether the GroupHeader section appears in the following section or not. It 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.
- **Use column layout** (ColumnLayout): Checkbox to determine whether the GroupHeader section uses the same column layout as the Detail section.
- **Can increase to accommodate contents** (CanGrow): Checkbox to specify whether the height of the GroupHeader section can grow when its controls extend beyond its original height.
- **Can decrease to accommodate contents** (CanShrink): Checkbox to specify whether the height of the GroupHeader section can adjust to the total height of controls placed in it.

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.

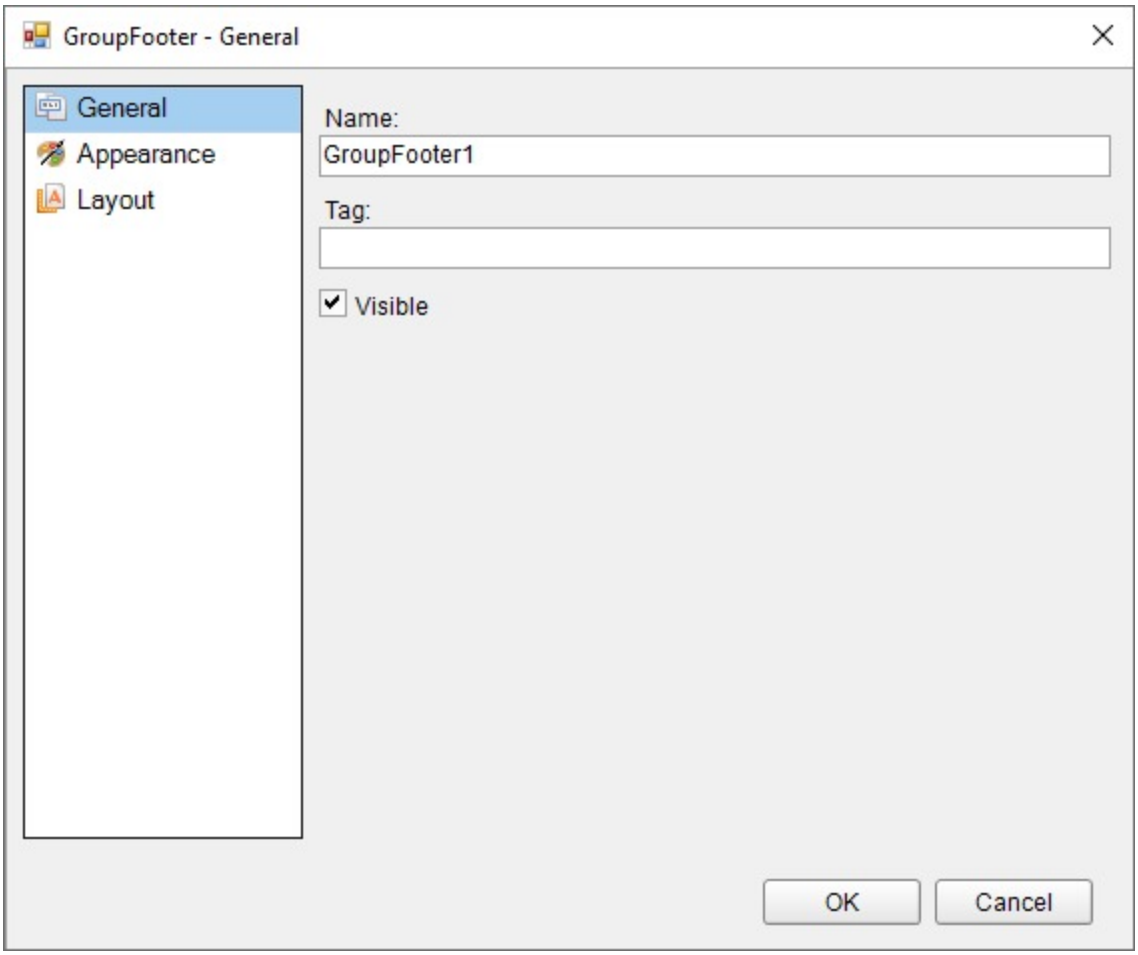
The **KeepTogether** property attempts to keep the section together on a single page, and the **RepeatToFill ('RepeatToFill Property' in the on-line documentation)** property allows you to fill each page with the same number of formatted rows, regardless of whether there is enough data to fill them. 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.

See **Detail ('Detail Class' in the on-line documentation)** for further information on properties.

> **Note:** You cannot use the **RepeatToFill** property if you are using the PageBreak or SubReport control in the Detail section, or if you have set the NewPage or NewColumn property to any value other than None. When you use this property in a report where two groups are present, the ReportFooter section prints on the next page. This property processes correctly only with single grouping.

Group Footer

A report can include single or nested groups, with each group having its own header and footer sections. You can insert and print the footer section immediately after the detail section.



To access the GroupFooter dialog, right click the group footer and in the Properties window under the properties list where the commands are displayed, click the **Property dialog** link.

General

- **Name** (Name): Indicates the name of the GroupFooter in code. It is unique for a report.
- **Tag** (Tag): Indicates the user-defined information persisted with the GroupFooter section.
- **Visible** (Visible): Checkbox to specify the visibility of the GroupFooter section.

Appearance

- **Background color** (BackColor): Dropdown list to set the background color of the GroupFooter section.

Layout

- **Insert new page** (NewPage): Dropdown list to determine whether a new page is inserted before and/or after displaying the GroupFooter section.
- **Insert new column** (NewColumn): Dropdown list to determine whether a new column (in a multi-column report) appears before and/or after displaying the GroupFooter section.
- **Keep section on a single page** (KeepTogether): Checkbox to determine whether the GroupFooter section appears on a single page.
- **Use column layout** (ColumnLayout): Checkbox to specify whether the GroupFooter section uses the same column layout as the Detail section.
- **Print at the bottom of page** (PrintAtBottom): Checkbox to specify whether the GroupFooter section is printed

at the bottom of the page immediately before the PageFooter section.
- **Can increase to accommodate contents** (CanGrow): Checkbox to specify whether the height of the GroupFooter section can grow when its controls extend beyond its original height.
- **Can decrease to accommodate contents** (CanShrink): Checkbox to specify whether the height of the GroupFooter section can adjust to the total height of controls placed in it.

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.

Page Footer

A report can have one page footer section that prints at the bottom of each page. You can use the page footer 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 end of the report.

The **Report Footer** does not appear on a section report by default. In order to add this section, right-click the report and select **Insert** > **Report Header/Footer** to add a Report Header and Footer pair.

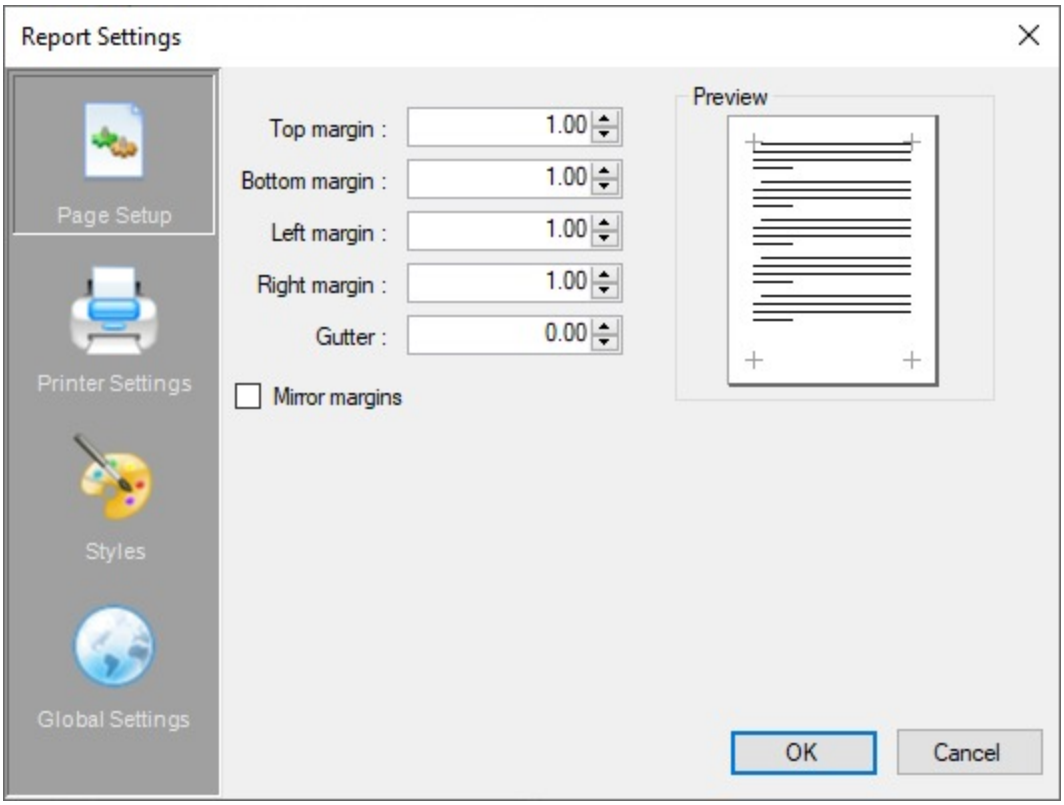
> **Note:** If the report contains a Page Footer on the last page, the Report Footer appears above the Page Footer.

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 through any of the following:

- In the Report Explorer, right-click the Settings node and select **Show** or double-click the Settings node.
- Click the gray area outside the design surface to select the report and in the Commands section at the bottom of the Properties Panel, click the **Property dialog** command.

The Report Settings dialog provides the following pages where you can set or modify various settings of your report.



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. This page also shows a preview of how each setting appears on the report page.

- **Top margin:** Set the Top margin for report pages.
- **Bottom margin:** Set the Bottom margin for report pages.
- **Left margin**: Set the Left margin for report pages.
- **Right margin**: Set the Right margin for report pages.
- **Gutter**: Set Gutter to give extra space between the edge of the page and the margins. This allows reports to be bound.
- **Mirror Margins**: Select this option to set same inner and outside margins for opposite pages in the report.

By setting a gutter and selecting Mirror margins, you can easily set up reports for publishing.

Printer Settings

On the Printer Settings page, you can make changes to the printer paper size and orientation.

- **Paper Size**: Select a paper size from the list of pre-defined paper sizes or choose Custom paper from the list to enable the Width and Height options for defining your own custom paper size.
- **Width**: Set the width of your custom paper size.
- **Height**: Set the height of your custom paper size.
- **Orientation**: Select one among Default, Portrait or Landscape as your paper orientation.
- **Collate**: Select whether to use collation or not.
- **Duplex**: Select whether the report should be printed in Simplex, Horizontal or Vertical duplex.
- **Paper Source**: Set the location of the paper source from the dropdown list.

> ⚠ **Important**: For items set to **Printer Default**, default printer settings are used from the printer installed on the environment where the report is being created. The paper size might also change based on the run time environment so in case the paper size of your report is fixed, please specify it beforehand.

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.

- **New**: Use this button to create a new style.
- **Delete**: Use this button to delete an existing style.
- **Export styles to file**: Use this button to export an existing style to an external XML *.reportstyle file.
- **Import styles from file**: Use this button to import styles a *.reportstyle file.
- **Font name**: Set or modify the font in your new or existing style.
- **Font size**: Set or modify the font size in your new or existing style.
- **Bold**: Enable or disable Bold text for your new or existing style.
- **Italic**: Enable or disable Italic text in your new or existing style.
- **Underline**: Enable or disable Underlined text in your new or existing style.
- **Strikethrough**: Enable or disable Strikethrough text in your new or existing style.
- **BackColor**: Set the Backcolor to use in your new or existing style.
- **ForeColor**: Set the Forecolor to use in your new or existing style.
- **Horizontal Alignment**: Set the horizontal alignment to Left, Center, Right, Justify for your new or existing style.
- **Vertical Alignment**: Set the vertical alignment to Top, Middle, Bottom for your new or existing style.
- **Script**: Select the script to use in your new or existing style.

Global Settings

On the Global Settings page, you can change the design layout of your report.

- **Snap Lines**: Select whether to use snap lines at design time or not.
- **Snap to Grid**: Select whether the control moves from one snap line to another at design time.
- **Show Grid**: Select whether to show or hide the grid at design time.
- **Grid Columns**: Set the count of columns in a grid.
- **Grid Rows**: Set the count of rows in a grid.
- **Dimension Lines**: Set whether to use dimension lines at design time or not.
- **Grid Mode**: Select whether to show gridlines as Dots or Lines.
- **Show Delete Prompt**: Select this option to get a warning when you try to delete a parameter or calculated field from the Report Explorer.
- **Ruler Units**: Set the ruler units in Inches or Centimeters.
- **Preview Pages**: Set the number of pages to display in the Preview tab. Minimum values is 1 and maximum is 10000 pages. By default, the Preview tab displays 10 pages.

  > **Note:** This property allows you to set number of pages to display in the Preview tab only. To set the number of pages for viewer/export, you can use **MaxPages ('MaxPages Property' in the on-line documentation)** property.

Aggregates Calculations

Section reports have a limited support for expressions and aggregates. To add an aggregation, you should specify any of these for TextBox controls in the Detail section:

Function	Description
DistinctField ('DistinctField Property' in the on-line documentation)	Gets or sets the name of the data field used in a distinct summary function.
SummaryFunc ('SummaryFunc Property' in the on-line documentation)	Determines the summary calculation function performed on the field value.
SummaryGroup ('SummaryGroup Property' in the on-line documentation)	Gets or sets the name of the group header section that is used to reset the summary value when calculating subtotals.
SummaryRunning ('SummaryRunning Property' in the on-line documentation)	Gets or sets a value that determines whether the data field summary value will be accumulated or reset for each level (detail, group or page). Possible options are None, Group, or All.
SummaryType ('SummaryType Property' in the on-line documentation)	Determines the type of field summary. Possible options are None, GrandTotal, Sub Total, PageCount, or PageTotal.

The field expressions support simple operators like the following:

- expr ? expr : expr – ternary conditional
- || - conditional or
- | - bitwise or
- ^ - xor
- & - bitwise and
- && - conditional and
- == - conditional equal
- != - conditional not equal
- < - less
- <= - less or equal
- > - greater
- >= - greater or equal
- Is – type testing
- As- type casting
- << - bitwise shift left
- >> - bitwise shift right
- + - plus
- - - Minus
- * - Multiply
- / - Divide
- % - Mod
- ~ - Complement
- . - Period
- ! - negation
- (...) and [..] brackets

The ReportInfo control also supports aggregation. For example, you can display page numbers and page count at the group level:

1. From the ActiveReports 18 Section report tab in the toolbox, drag the ReportInfo control to the **GroupHeader** or **GroupFooter** section of the report and set the **FormatString** property as above. See Display Page Numbers and Report Dates.
2. With the ReportInfo control still selected in the Properties Panel, 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**.

Display Page Numbers and Report Dates

With the ReportInfo control available in a Section Report, you can display page numbers and report dates and times by selecting a value in the **FormatString** property. This property provides the following pre-defined options for page numbering and date and time formatting.

Predefined Options and their Descriptions

Numbering Format	Description
Page {PageNumber} of {PageCount} on {RunDateTime}	Display the page numbers along with Date and Time in the following format : `Page 1 of 100 on 1/31/2012 2:45:50`

	PM
Page {PageNumber} of {PageCount}	Display the only the page numbers in the following format : Page 1 of 100
{RunDateTime:}	Display the Date and Time in the following format : 1/31/2012 2:45:50 PM
{RunDateTime: M/d}	Display the Date in the following format : 1/31
{RunDateTime: M/d/yy}	Display the Date in the following format : 1/31/12
{RunDateTime: M/d/yyyy}	Display the Date in the following format : 1/31/2012
{RunDateTime: MM/dd/yy}	Display the Date in the following format : 01/31/12
{RunDateTime: MM/dd/yyyy}	Display the Date in the following format : 01/31/2012
{RunDateTime: d-MMM}	Display the Date in the following format : 31-Jan
{RunDateTime: d-MMM-yy}	Display the Date in the following format : 31-Jan-12
{RunDateTime: d-MMM-yyyy}	Display the Date in the following format : 31-Jan-2012
{RunDateTime: dd-MMM-yy}	Display the Date in the following format : 31-Jan-12
{RunDateTime: dd-MMM-yyyy}	Display the Date in the following format : 31-Jan-2012
{RunDateTime: MMM-yy}	Display the Date in the following format : Jan-12
{RunDateTime: MMM-yyyy}	Display the Date in the following format : Jan-2012
{RunDateTime: MMMM-yy}	Display the Date in the following format : January-12
{RunDateTime: MMMM-yyyy}	Display the Date in the following format : January-2012
{RunDateTime: MMMM d,yyyy}	Display the Date in the following format : January 31, 2012
{RunDateTime: M/d/yy h:mm tt}	Display the Date and Time in the following format : 1/31/12 2:45 PM
{RunDateTime: M/d/yyyy h:mm tt}	Display the Date and Time in the following format : 1/31/2012 2:45 PM
{RunDateTime: M/d/yy h:mm}	Display the Date and Time in the following format : 1/31/12 2:45
{RunDateTime: M/d/yyyy h:mm}	Display the Date and Time in the following format : 1/31/2012 2:45

Page numbering can also be set to a group level using the **SummaryGroup** and **SummaryRunning** properties.

These steps assume that you have already added a Section Report to a project in Visual Studio.

Display page numbers and report dates on a report

1. From the ActiveReports 18 Section report tab in the toolbox, drag the **ReportInfo** control to the desired

location on the design surface.
2. With the ReportInfo control selected in the Properties Panel, drop down the **FormatString** property.
3. Select the pre-defined format that best suits your needs.

> **Tip:** You can customize the pre-defined formats in the Properties Panel. For example, if you change the **FormatString** property to **Page {PageNumber} / {PageCount}**, it shows the first page number as **Page 1/1**. For more information on creating formatting strings, see the Date, Time, and Number Formatting topic.

Display page numbers and page count at the group level

1. From the ActiveReports 18 Section report tab in the toolbox, drag the ReportInfo control to the **GroupHeader** or **GroupFooter** section of the report and set the FormatString property as above.
2. With the ReportInfo control still selected in the Properties Panel, 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**.

Date, Time, and Number Formatting

In Section Reports, you can 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 in the FormatString property for RunDateTime and Page Numbers. For more information, see ReportInfo (Section report).

> **Caution:**
>
> - The format from the OutputFormat property is applied to the value set in the DataField property or the Value property. It is not applied when a string is set in the Text property.
> - OutputFormat property settings are valid only for Double or DateTime type values. In case of a String or when no data type is set, the format is automatically applied to only those values that can be converted to Double or DateTime, otherwise no format is applied.

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.

For example: $#,#00.00; ($#,#00.00_); $0.00; #

Dates:

- dddd, MMMM d, yyyy = Saturday, December 25, 2012
- dd/MM/yyyy = 25/12/2012
- 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, 12 for 2012)
- yyyy or yyyy = year in four digit format (for example, 2012)

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

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

> **Note:** Underscore (_) keycode can be used in **OutputFormat** property to skip the width of the next character. This code is commonly used as **_)** to leave space for a closing parenthesis in a positive number format when the negative number format includes parentheses. This allows both positive and negative values to line up at the decimal point.

Interactivity

ActiveReports provides Section reports with a number of interactive features like parameters, bookmarks, and hyperlinks.

Learn about how to add these interactive features when creating Section reports.

Parameters

In a Section report, you can use the Parameters collection to pass values directly into a control, or you can also use it to display a subset of data in a particular instance of the report.

There are several ways to set up parameters in a report:

- Enter syntax like the following in your SQL query to filter the data displayed in a report:
  `<%Name|Prompt|DefaultValue|Type|PromptUser|DataFormat%>`
- Add parameters through the Report Explorer and place them on the report as TextBox controls to pass values into them.
- Add parameters through the code behind the report, inside the **ReportStart** event.

Add Parameters via the SQL Query

When you add a single parameter to a report's Parameters collection via the SQL query, the query looks like this:

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

You can also create a parameterized query from the Visual Query Designer. See Query Builder in Microsoft SQL Client and OLEDB Providers for further information on how to create a parameterized query using the interactive query designer.

There are six 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.
- **Prompt**: The string that is displayed in the user prompt to let the user know what sort of value to enter (at runtime).
- **DefaultValue**: Providing a default value to use for the parameter allows ActiveReports to populate the bound field 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, defines the 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 as explained below:
    - **S (string)** provides a textbox into which the user can enter the string.
      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 picker from which the user can select a date.
      Depending on your data source, you may need to put number signs around the parameter syntax.
      For example, `#<%MyDateParameter%>#`
    - **B (Boolean)** provides a checkbox that the user can select or clear.
      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.

  > **Note**: In the case of Microsoft Access Database, the default value for Boolean parameter is specified as -1(true) or 0(false).

- **PromptUser**: This Boolean allows you to tell ActiveReports whether to prompt the user for a value. This is set to True to use parameters at runtime. 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. The supported PromptUser values are: true/True and false/False.

- **DisplayFormat**: Choose a format for the 'Date' data type. The selected format is displayed in the preview in the Parameters pane.

When you add SQL parameters to a report, ActiveReports displays an 'Enter Report Parameters' dialog where the user can enter the values to fetch from the database. The steps to add a parameter to an SQL query are as follows:

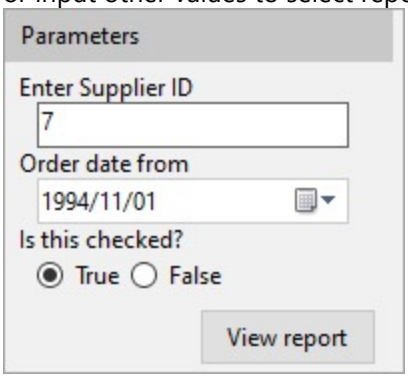
1. In the **Detail** section band, click the DataSource icon to view the **Report Data Source** dialog.
2. Connect the report to 'Nwind.mdb' data source, an OleDB data source. See Microsoft OLEDB Provider for more information.
3. In the **Query** field, enter a SQL query like the one below, which contains the parameter syntax to prompt for parameter values at runtime.

Query

```
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|true|yyyy/MM/dd%>#
AND Discontinued = <%Discontinued|Is this checked?|true|B||%>
```

4. Click **OK** to save the data source and return to the report design surface.

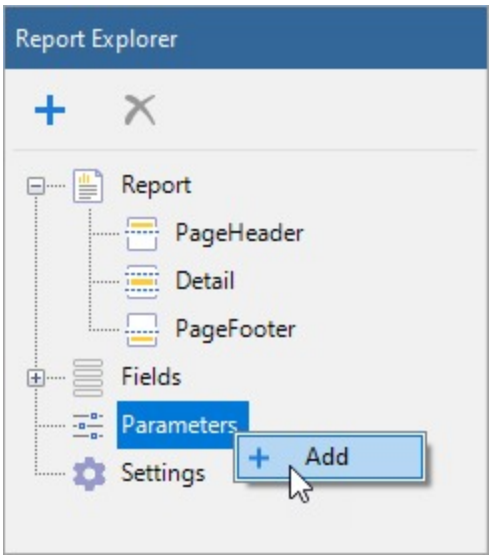
   The SQL query above causes ActiveReports to display the following dialog to the user. The user can accept these or input other values to select report data.



Add Parameter from Report Explorer

The steps to add a parameter from Report Explorer are as follows:

1. In the Report Explorer, right-click the Parameters node and select **Add**. This adds a parameter (Parameter1) as a child to the Parameters node.



2. Select the added parameter to open the Properties Panel and set values in the following properties, the description of each of which is given in the **previous section**:
    - Name
    - DefaultValue
    - Prompt
    - PromptUser
    - Type
    - DisplayFormat
3. Pass the parameter to a field on the report.

Add Parameters at Runtime

You can add, edit, and delete parameters at runtime. The following code demonstrates how to add a parameter and display its value in a Textbox control.

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 parameters at runtime.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste at beginning of code view.

```
Imports GrapeCity.ActiveReports.SectionReportModel
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Dim myParam1 As New Parameter()
myParam1.Key = "myParam1"
myParam1.Type = Parameter.DataType.String

'Set to False if you do not want input from user.
myParam1.PromptUser = True
myParam1.Prompt = "Enter Data:"
myParam1.DefaultValue = "Default Value"
Me.Parameters.Add(myParam1);
```

```
'Set to True to display parameter dialog box when report is run.
Me.ShowParameterUI = True
```

To write the code in C#

C# code. Paste at beginning of code view.

```
using GrapeCity.ActiveReports.SectionReportModel;
```

C# code. Paste INSIDE the ReportStart event.

```
Parameter myParam1 = new Parameter();
myParam1.Key = "myParam1";
myParam1.Type = Parameter.DataType.String;

//Set to false if you do not want input from user.
myParam1.PromptUser = true;
myParam1.Prompt = "Enter Data:";
myParam1.DefaultValue = "Default Value";
this.Parameters.Add(myParam1);

//Set to true to display parameter dialog box when report is run.
this.ShowParameterUI = true;
```

3. In the design view, click the gray area below the report to select it and open the Properties Panel.
4. Click the events icon in the Properties Panel 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 runtime.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the FetchData event.

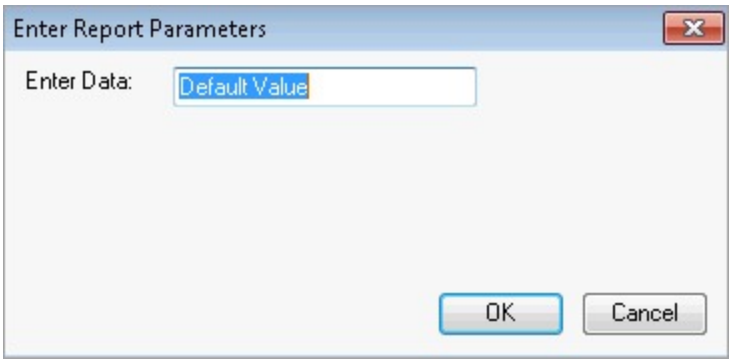
```
'Set textbox text equal to the value of the parameter.
Me.txtParam1.Text = Me.Parameters("myParam1").Value
```

To write the code in C#

C# code. Paste INSIDE the FetchData event.

```
//Set textbox text equal to the value of the parameter.
this.txtParam1.Text = this.Parameters["myParam1"].Value;
```

The run-time implementation above displays the following dialog. You can enter any text in this prompt dialog and display it on the report.



Prompt for Parameter Values

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

- At least one parameter should exist in the Parameters collection of the report.
- The **PromptUser** property for at least one parameter must be 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 set to 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 gets displayed using the specified values.

Values of a parameter added through the Report Explorer can be applied to a parameter in the SQL query by specifying the `param:` prefix for a parameter in the SQL query. This prefix relates the current parameter to the one in the Report Explorer.

For e.g., `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.

> **Note**: Within the same report, you can prompt users for some parameters and not for others by setting the **PromptUser** property to **True** for some and **False** for others. However, if the report object's **ShowParameterUI** property is set to **False**, the user prompt does not display any parameters, regardless of its **PromptUser** setting.

Bookmarks

In a Section report, bookmark links take you to a location where the bookmark is set on your report. Bookmarks and nested bookmarks appear in the document map for fields, groups, and subreports. You can also add special bookmarks at run-time.

Bookmarks are supported when you preview a report in the Viewer, or export a report in HTML and PDF formats. See Document map for more information. The following sections show setting up bookmarks in some scenarios in XML-based Section reports. Note that the same scripts are equally applicable to Code-based Section reports with cautious use of casing.

Set up basic bookmarks

1. From the toolbox, drag and drop a **TextBox** control onto the **Detail** section.

2. Double-click the **Detail** section of the report. This creates an event-handling method for the report's Detail_Format event.
3. Add the following script to set up bookmarks.

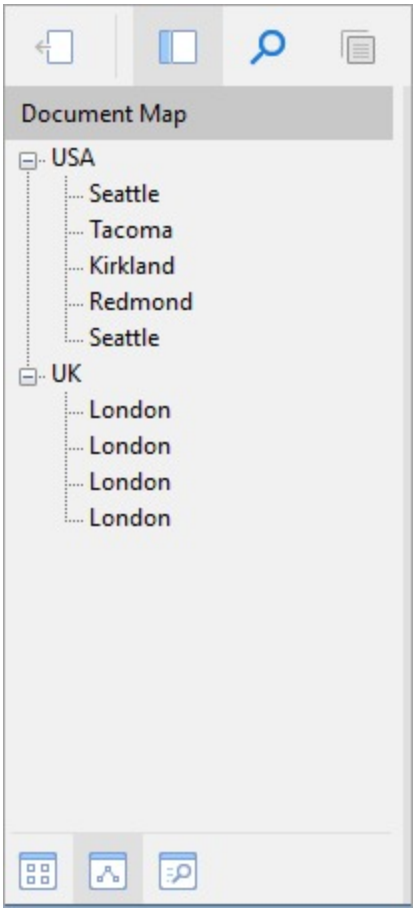
Visual Basic.NET code. Paste INSIDE the Detail Format event.

```
Detail.AddBookmark(TextBox1.text)
```

C# code. Paste INSIDE the Detail Format event.

```
Detail.AddBookmark(TextBox1.Text);
```

Set up leveled or nested bookmarks



Bookmarks can be nested to reflect a hierarchical structure; this is sometimes called a parent-child relationship.

1. Create a new section report and bind it to JSON data using the following connection string and JSON path. See JSON Provider for more information.

Connection String

```
method=POST;headers={"Content-Type":"application/json"};body={ "query": "
{employees{country, city, superior{firstName,lastName,title}}}"
};jsondoc=https://demodata.mescius.io/northwind/graphql
```

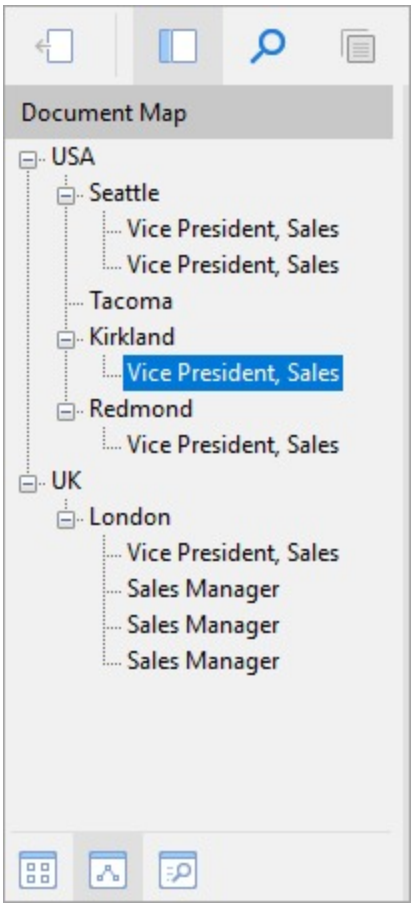
JSONPath
`$.data.employees[*]`

2. From the Report Explorer, drag and drop `country` and `city` onto the **Detail** section.
3. Double-click the **Detail** section of the report. This creates an event-handling method for the report's Detail_Format event.
4. Add the following script to set up leveled or nested bookmarks.

Visual Basic.NET code. Paste INSIDE the Detail Format event.
`Detail.AddBookmark(txtcountry1.Text + "\\" + txtcity1.Text)`

C# code. Paste INSIDE the Detail Format event.
`Detail.AddBookmark(txtcountry1.Text + "\\" + txtcity1.Text);`

Nest grandchild bookmarks and use bookmarks in grouping



1. Follow **Step 1** of the previous section to create a new section report and bind the data.
2. From the Report Explorer, drag and drop `country`, `city`, and `superior.title` fields onto the **Detail** section.
3. Double-click in the **Detail** section of the report. This creates an event-handling method for the report's Detail_Format event.
4. Add the following script to set up a bookmark for each `superior.title` and nest `superior.title`

bookmarks within each `city`, and `city` bookmarks in each `country`.

Visual Basic.NET code. Paste INSIDE the Detail_Format event.

```
Detail.AddBookmark(txtcountry1.Text + "\\" + txtcity1.Text + "\\" +
txtsuperior_title1.Text)
```

C# code. Paste INSIDE the Detail_Format event.

```
Detail.AddBookmark(txtcountry1.Text + "\\" + txtcity1.Text + "\\" +
txtsuperior_title1.Text);
```

5. Add a **Group Header** section to the layout and set its **DataField** property to `country`.
6. Double-click in the Group Header section of the report. This creates an event-handling method for the report's Group Header Format event.
7. Add the following script to set up a bookmark for each instance of the `country` group.

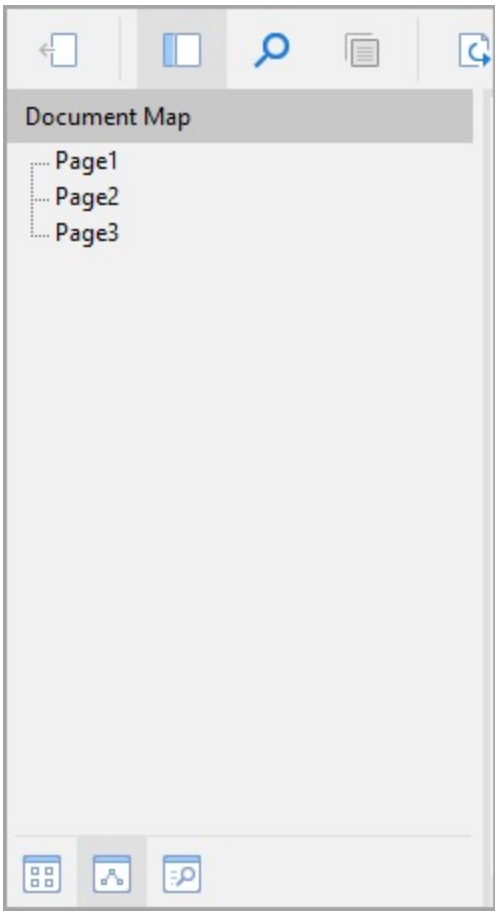
Visual Basic.NET code. Paste INSIDE the Group Header Format event.

```
GroupHeader1.AddBookmark(txtcountry1.Text)
```

C# code. Paste INSIDE the Group Header Format event.

```
GroupHeader1.AddBookmark(txtcountry1.Text);
```

Add bookmarks to a predefined page



To create and add special bookmarks to the bookmarks collection at run time, add the bookmarks to the report document's page collection.

> ⚠ **Caution**: Remember that the page collection does not exist until the report runs, so use this code in the ReportEnd event or in form code after the report has run.

1. Go to the **Script** tab of the report designer.
2. Select **ActiveReport** as **Object** and **ReportEnd** as **Event**. This creates an event-handling method for the ReportEnd event.
3. Add the following script to create bookmark for the pages.

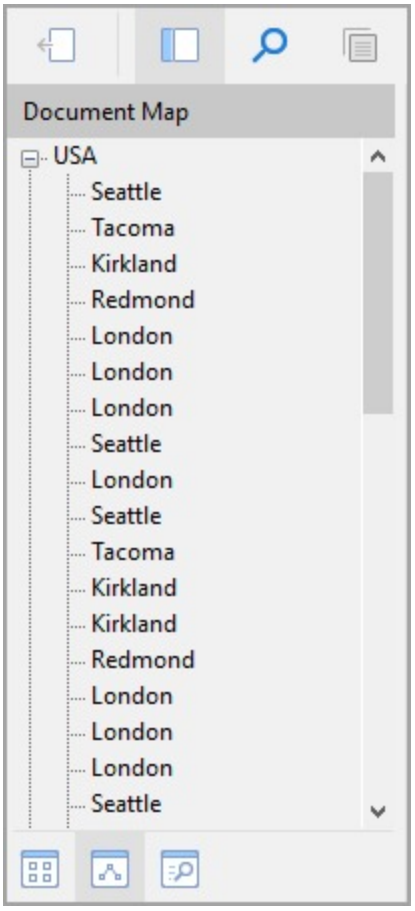
Visual Basic.NET code. Paste INSIDE the ReportEnd event.

```
rpt.Document.Pages(0).AddBookmark("Page1", 1)
rpt.Document.Pages(1).AddBookmark("Page2", 2)
rpt.Document.Pages(2).AddBookmark("Page3", 3)
```

C# code. Paste INSIDE the ReportEnd event.

```
rpt.Document.Pages[0].AddBookmark("Page1", 1);
rpt.Document.Pages[1].AddBookmark("Page2", 2);
rpt.Document.Pages[2].AddBookmark("Page3", 3);
```

Combine parent report and subreport bookmarks



1. Create a new section report and bind the data as **Step 1** of another section. This will be the parent report.
2. From the Report Explorer, drag and drop the `country` field onto the detail section of the parent report.
3. Double-click the Detail section to create an event-handling method for the report's Detail Format event.
4. Add the following script to create a bookmark for each instance of the `country` field in the parent report.

Visual Basic.NET code. Paste INSIDE the Detail Format event of the main report.

```
Private _subRpt As GrapeCity.ActiveReports.SectionReport
Public Sub Detail_Format()
    SubReport1.Report = _subRpt
    Detail.AddBookmark(txtcountry1.Text)
End Sub
Public Sub ActiveReport_ReportStart()
    _subRpt = New GrapeCity.ActiveReports.SectionReport()
    _subRpt.LoadLayout(System.IO.Path.GetFullPath("..\..\..\SubReport1.rpx"))
End Sub
```

C# code. Paste INSIDE the Detail Format event of the main report.

```
GrapeCity.ActiveReports.SectionReport _subRpt;
public void Detail_Format()
{
    SubReport1.Report = _subRpt;
    Detail.AddBookmark(txtcountry1.Text);
```

```
}
public void ActiveReport_ReportStart()
{
    _subRpt = new GrapeCity.ActiveReports.SectionReport();
    _subRpt.LoadLayout(System.IO.Path.GetFullPath(@"..\..\..\SubReport1.rpx"));
}
```

5. Drag-drop Subreport control onto the design area. By default, the **ReportName** property is set to SubReport1.
6. Create a new Section report with the name 'SubReport1', which will be a subreport to the parent report and bind it similarly as the main report.
7. From the Report Explorer, drag and drop the `city` field onto the detail section of the subreport.
8. Double-click in the Detail section to create an event-handling method for the subreport's Detail Format event.
9. Add the following script to create a bookmark for each instance of the `city` field in the subreport.

Visual Basic.NET code. Paste INSIDE the Detail Format event of the subreport.

```
Detail.AddBookmark(CType(rpt.ParentReport.Sections("Detail").Controls("txtcountry1"),
TextBox).Text + "\" + txtcity1.Text)
```

C# code. Paste INSIDE the Detail Format event of the subreport.

```
Detail.AddBookmark(((TextBox)
(rpt.ParentReport.Sections["Detail"].Controls["txtcountry1"])).Text + "\\" +
txtcity1.Text);
```

10. Preview the main report and navigate to Document Map. You will see bookmarks for city from the subreport nested into bookmarks for country from the main report.

Hyperlinks

In Section reports, you can set the **HyperLink** property available with the Label, TextBox, and Picture controls that allow you to add hyperlinks that connect to a Web page. You can use the **HyperLink** property to open an e-mail or jump to a bookmark.

You can also use any URI scheme in the **HyperLink** property of the controls mentioned above. For the full list of URI schemes, see Uniform Resource Identifier (URI) Schemes.

Hyperlinks are supported when you preview a Section report and export a report in HTML, PDF, RTF, and Excel formats.

> **Note:** We recommend that you specify the full URL address (for example, "https://developer.mescius.com") for the **Hyperlink** property to avoid broken links while viewing reports.

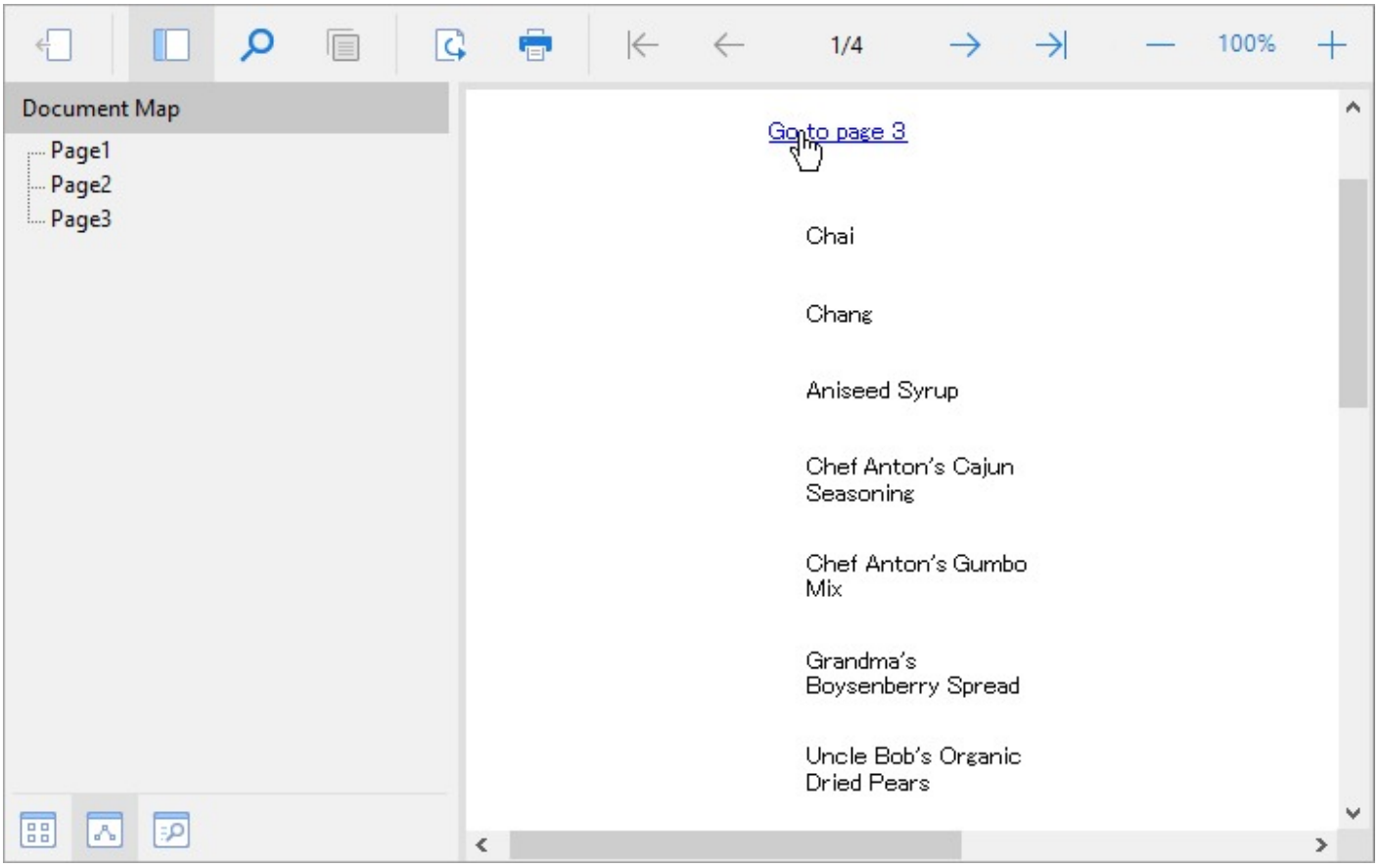
Link to a Web page

1. Select an existing control or drag and drop a control from the Toolbox onto the design surface.
2. In the Properties Panel, set the **HyperLink** property to any valid URL. For example, https://developer.mescius.com.
   Alternatively, you can set the HyperLink property in the control dialog that appears by clicking the **Property dialog** link under Properties.

Link to an e-mail address

1. Select an existing control or drag and drop a control from the Toolbox onto the design surface.
2. In the Properties Panel, set the **HyperLink** property to mailto: `any valid e-mail address`.
   Alternatively, you can set the HyperLink property in the control dialog that appears by clicking the **Property dialog** link under Properties.

Create a hyperlink that jumps to a bookmark



1. Let's first create bookmarks to predefined page. Follow the steps on Add bookmarks to a predefined page
2. Add a report header where we will place a text box to contain the hyperlink.
3. Drag-drop a **TextBox** control onto the **ReportHeader** section.
4. With the Textbox control selected, click the **Property dialog** link to open the control's properties.
5. On the **General** page, set the **Text** to 'Go to page 3' and the **Hyperlink** to 'TOC://Page3'.

Scripting

In a section report, 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 standalone 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 report files, scripting allows you to update distributed reports without recompiling your project.

Script Editor

To access the script editor, click the Script tab from the top menu in the Standalone Designer application or below the report design surface in Visual Studio Integrated Designer. The script tab contains two drop-downs (Object and Event).

- **Object**: Drop down the list and select one of the report sections, or the report itself.
- **Event**: Drop down the list and select from the list of events generated based on your selection in the Object drop down. If you select a report section as the Object, there are three events: Format, BeforePrint, and AfterPrint. If you select ActiveReport as the Object, there are seven events. See Report Events for further information.

Add script to the events in the same way that you add code to events in the code view of the report. When you select an event, the script editor generates a method stub for the event.

Using the **ScriptLanguage ('ScriptLanguage Property' in the on-line documentation)** property of the report, you can set the script language that you want to use.

Select the scripting language to use

- In design view of the report, click in the grey area below the report to select it.
- In the Properties window, drop down the ScriptLanguage property and select C# or VB.NET.

You can also add scripts at run time using the **Script ('Script Property' in the on-line documentation)** property.

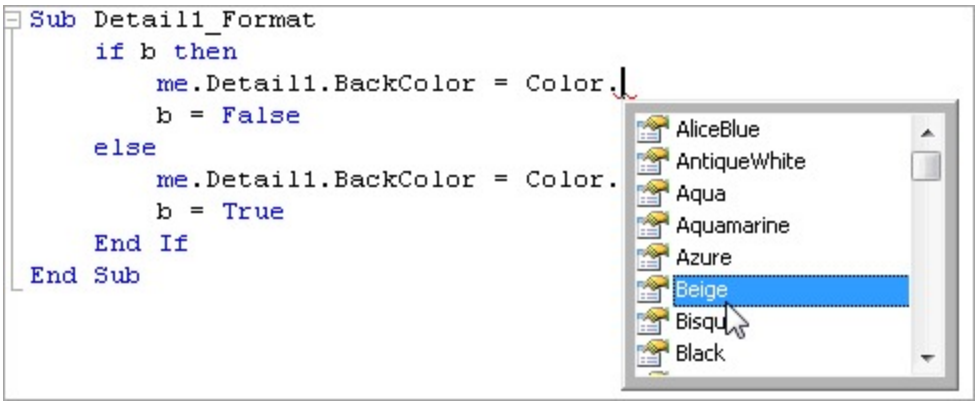
> ⚠ **Caution**: Since the RPX file can be read with any text editor, use the **AddCode ('AddCode Method' in the on-line documentation)** or **AddNamedItem ('AddNamedItem Method' in the on-line documentation)** method to add secure information such as a connection string.

Tips for Using Script

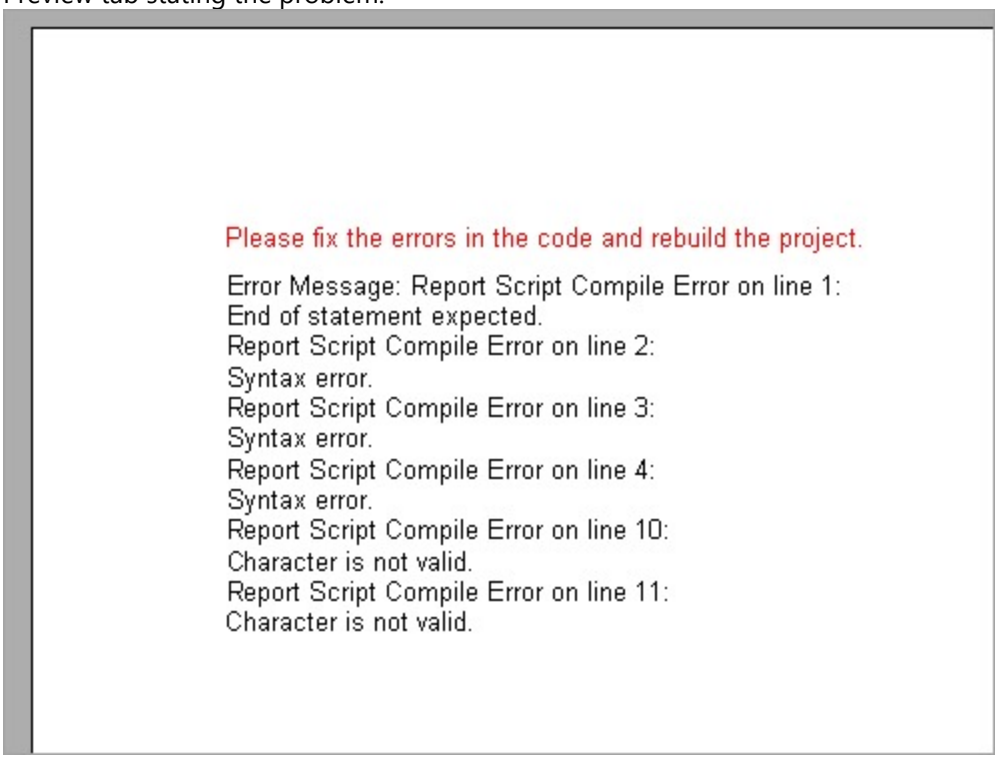
- **Keep the section report class public**: If the Section Report class is private, the script cannot recognize the items in your report. The Section Report class is public by default.
- **Set the Modifiers property of any control referenced in script to Public**: If the control's **Modifiers** property 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 Intellisense support**: The script tab supports IntelliSense that helps in inserting the language elements and provides other helpful options when adding script to a report.



- **Use Run-time error handling**: When run-time errors occur, a corresponding error message is displayed in the Preview tab stating the problem.



Note: A declared variable is not static by default. Use the **static** keyword to declare static variables.

Difference in script and code-behind event handler

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

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

Script and code-behind examples in Visual Basic

The ReportStart event handler definition in the script editor:

```
Visual Basic.NET
Sub ActiveReport_ReportStart End Sub
```

The ReportStart event handler definition in code-behind:

```
Visual Basic.NET
Private Sub SectionReport1_ReportStart(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.ReportStart End Sub
```

Script and code-behind examples in C#

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

The ReportStart event handler definition in code-behind:

```
C#
private void SectionReport1_ReportStart(object sender, EventArgs e) { }
```

Report Events

Section reports use events to allow you to control report behavior.

Single-Occurrence Events

The following events are all of the events that are raised only once during a Section report's processing. These events are raised at the beginning or at the end of the report processing cycle.

Events raised once

ReportStart

Use this event to initialize any objects or variables needed while running a report. This event is also used to set any Subreport control objects to a new instance of the report assigned to the Subreport control.

> ⚠ **Caution:** Be sure to add dynamic items to the report before this event finishes.

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 Data Control 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.

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.

Multiple-Occurrence Events

The following events are raised multiple times during a Section report's processing.

Events raised more than once

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.

Events that Occur for Each Instance of Each Section

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

Section Events

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

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

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

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.

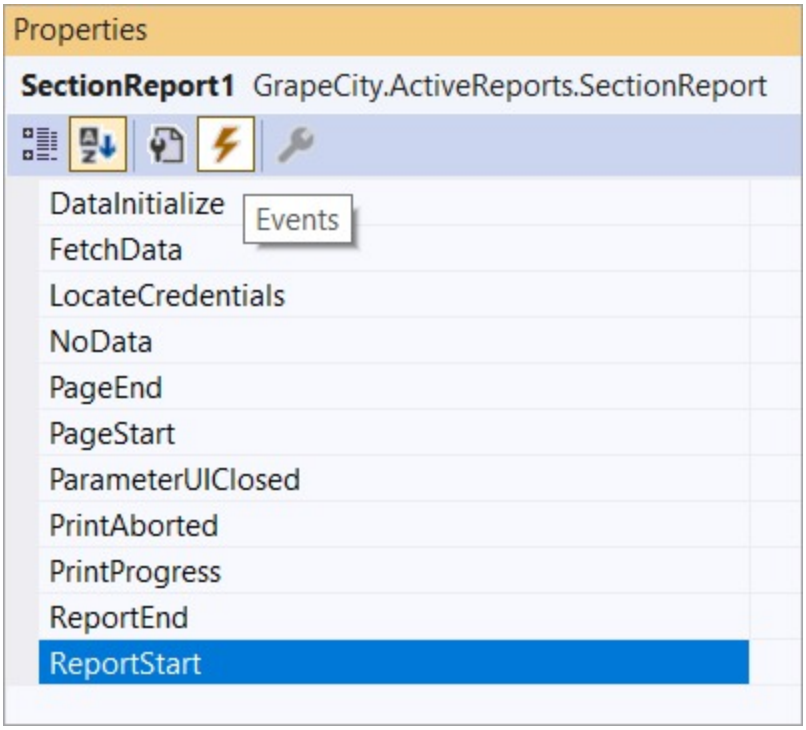
Event Sequence

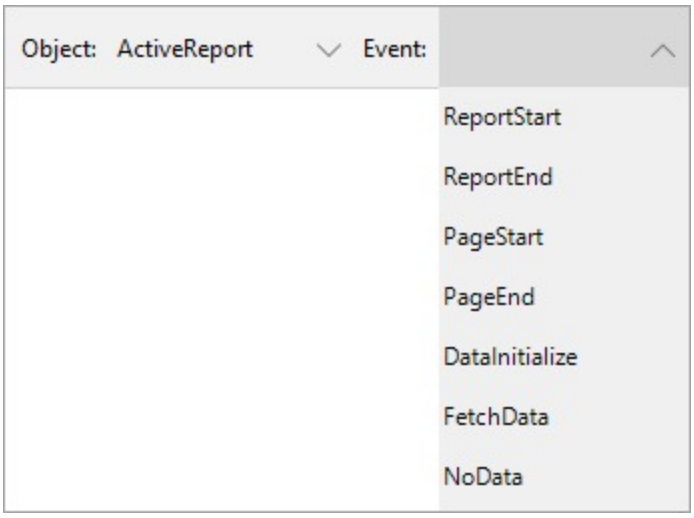
Multi-threaded, single-pass processing enables Section reports to surpass other reports in processing and output generation speed. ActiveReports processes and renders each page as soon as the page is ready. If a page has unknown data elements or its layout is not final, it places the page in cache until the data is available.

Sequence of Events

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 a Section report, or seven in an ActiveReport script.





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.

Events that May Occur

These events occur in response to user actions, or when there is no data for a report.

Other Events

DataSourceChanged

This event occurs if the report's data source is changed. This is mainly useful with the end-user designer control.

NoData

This event occurs if the report's data source returns no records.

ParameterUIClosed

This event occurs when the user closes the parameter dialog.

PrintAborted

This event occurs when the user cancels a print job.

PrintProgress

This event occurs once for each page while the report document is printing.

Tutorials: Report Controls in Section Reports

This section takes you through the set of tutorials aimed at designing the Section reports in the Report Designer from scratch.

- InputFieldCheckBox Control in Reports
- InputFieldText Control in Reports
- Picture Control in Reports
- TextBox Control in Reports

InputFieldCheckBox Control in Reports

We are going to create a PDF form, using the InputFieldCheckBox controls, where a user, besides entering text information, may also specify Yes (check) or No (uncheck) for such information like Sex, Employment Status, etc.

The final report will look as shown.

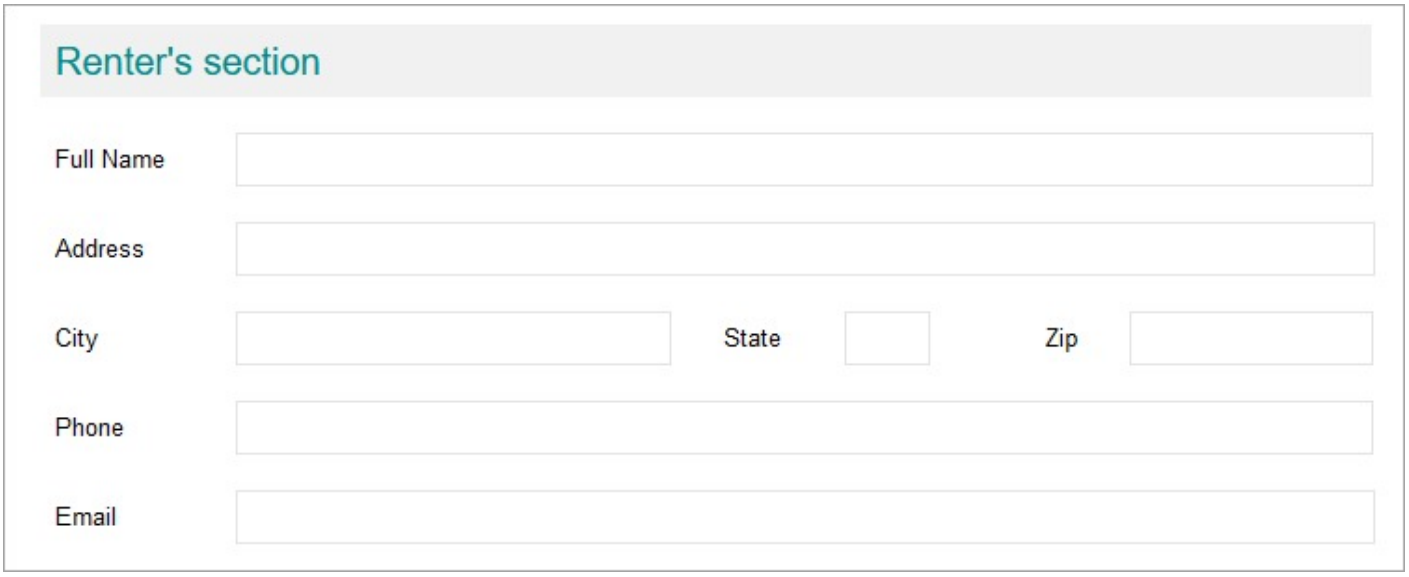
Design Report Layout

1. Drag and drop the **Label** control onto the report designer and set its **Text** property to Renter's section. This is the form's heading.
2. Drag and drop the **Label** controls below the Renter's section text; these controls will be the captions to the user's answers. Set these control's **Text** property to the following values:
    - Label2: Full Name
    - Label3: Address
    - Label4: City
    - Label5: State
    - Label6: Zip
    - Label7: Phone
    - Label8: Email
3. Now drag and drop the **InputFieldText** controls next to the Label controls added at the previous step. These blank fields are required to be filled by the users filling the form.
4. Drag and drop some more **Label** controls and set them to the following values in the **Text** property.
    - Label9: Sex
    - Label10: Male
    - Label11: Female
5. Now drag and drop the **InputFieldCheckBox** controls next to the Labels added at the previous step. These blank fields are required to be filled as yes or no using the checkboxes.
6. Set **Required** property for some fields that you want to make mandatory before submitting the form.
7. Improve the appearance of the controls and preview the report.
8. Export the report to PDF format. Users can now enter the text in the blank fields and also enter their choices as Yes (check) or No (uncheck) in the editable checkboxes.

InputFieldText Control in Reports

We will create a PDF form, using the InputFieldText controls, where a user may enter textual renter's information like Name, Address, etc. We will use Label control to write the captions for these user inputs.

The final report will look as shown.



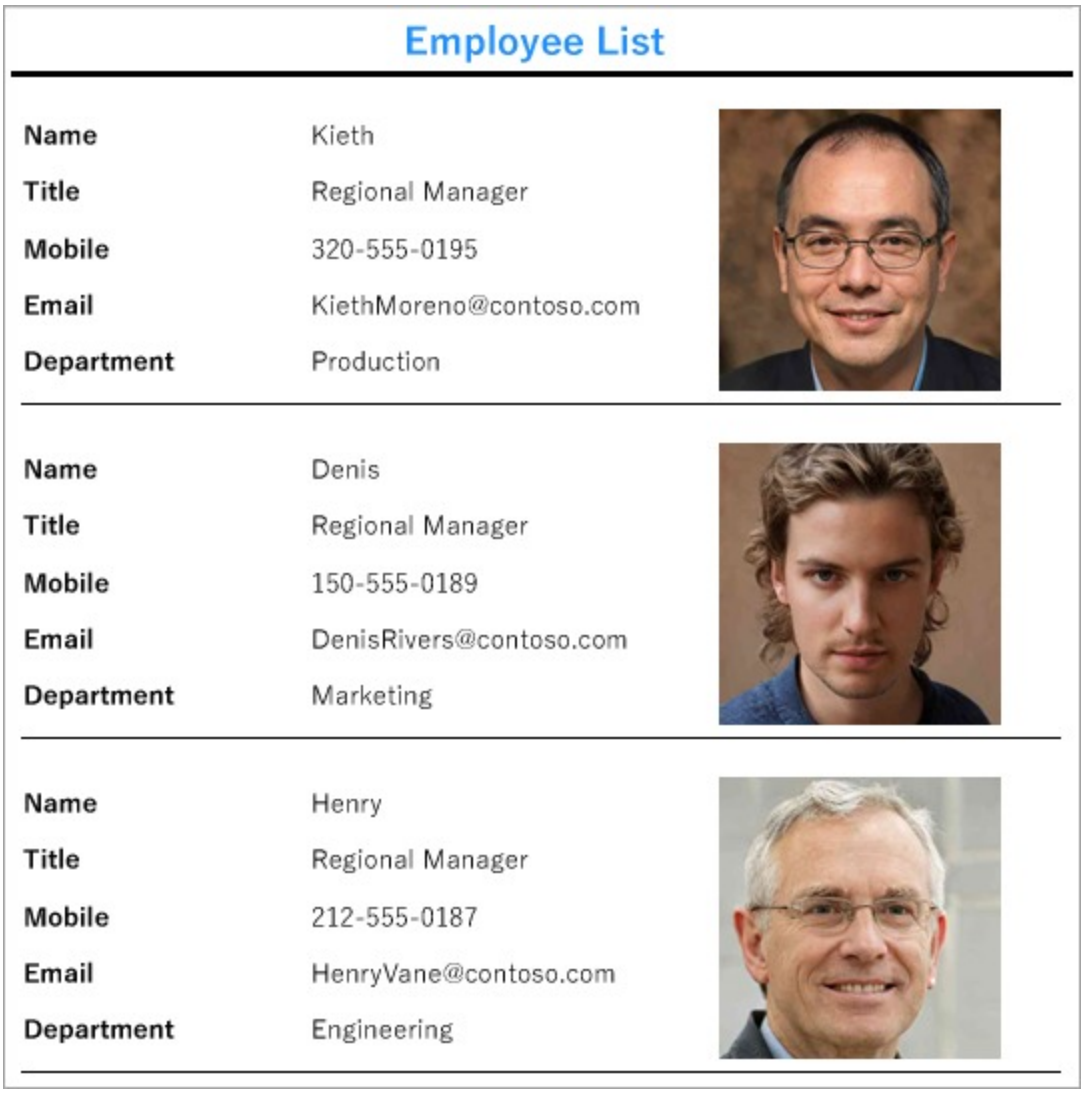
Design Report Layout

1. Drag and drop the **Label** control onto the report designer and set its **Text** property to Renter's section. This is the form's heading.
2. Drag and drop the **Label** controls below the Renter's section text; these controls will be the captions to the user's answers. Set these control's **Text** property to the following values:
    - Label2: Full Name
    - Label3: Address
    - Label4: City
    - Label5: State
    - Label6: Zip
    - Label7: Phone
    - Label8: Email
3. Now drag and drop the **InputFieldText** controls next to the Label controls added at the previous step. These blank fields are required to be filled by the users filling the form.
4. Set **Required** property for some fields that you want to make mandatory before submitting the form.
5. Improve the appearance of the controls and preview the report.
6. Export the report to PDF format. Users can now enter the text in the blank fields.

Picture Control in Reports

Let's say that we want to create a report that shows the photos of employees along with their details including name, job title, date of birth, mobile, email, and department. For such a report, we will use the **Picture** control along with other TextBox controls. The report connects to 'DimEmployees' JSON data available here.

The final report will look as shown.



Create a Report

In the ActiveReports Designer, create a new Section report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by clicking the Data Source icon on the Detail section.
2. In the dialog, go to the **JSON** tab and click **Build** to open the **Configure JSON Data Source** dialog.
3. Enter the following URL in the **Data Path** as follows:
   https://demodata.mescius.io/contoso/odata/v1/DimEmployees
4. Click **OK** to close the dialog.
   The Connection String displayed is as follows:

Connection String
`jsondoc=https://demodata.mescius.io/contoso/odata/v1/DimEmployees`

   For more information, see the JSON Provider article.

5. Click **OK** to save the changes and close the **Report Data Source** dialog.
6. To fetch the required fields, enter the following query in the **JSON Path**.

Query
`$.value[*]`

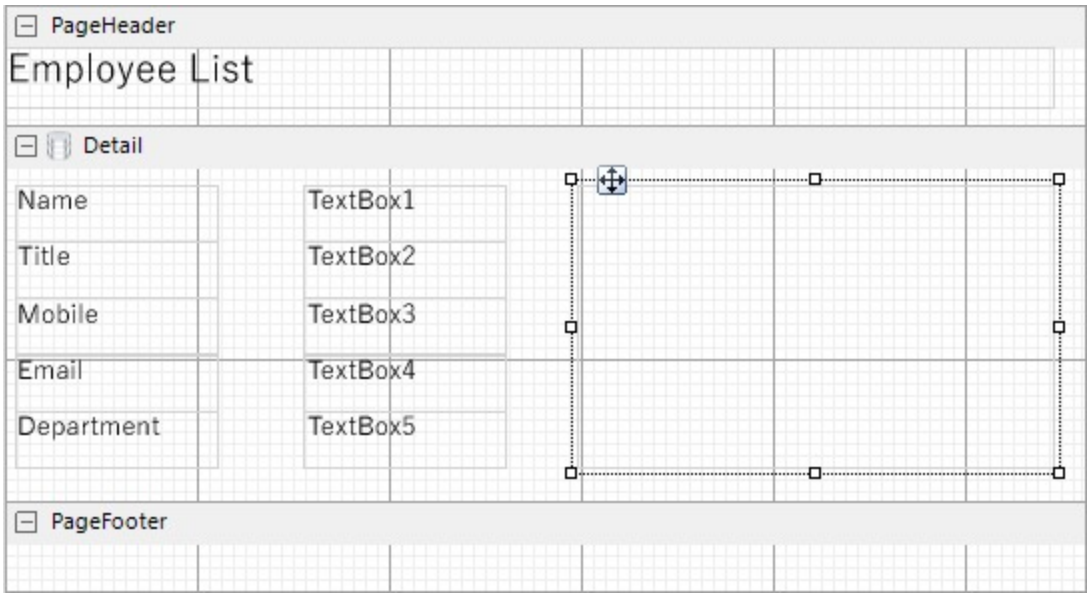
Design Report Layout

1. Drag few **Label** controls to the Details section of the report, and set their **Text** property as follows:
   - Label1: Name
   - Label2: Title
   - Label3: Mobile
   - Label4: Email
   - Label5: Email
   - Label6: Department
2. Drag and drop few **TextBox** controls next to the Labels controls, and set their **DataField** property as follows:
   - TextBox1: `FirstName`
   - TextBox2: `Title`
   - TextBox3: `Mobile`
   - TextBox4: `Email`
   - Department: `DepartmentName`
3. Drag and drop the **Picture** control in the Detail section and set the below properties as:
   - **DataField**: AvatarUrl
   - **SizeMode**: Zoom
4. In the **Script** tab, set the **Object** and **Event** to ActiveReports and ReportStart, respectively.
   For more information, see the Report Events article.
5. Then, add the below script to download the images from URL. The script will look like as shown.

Script

```
public void ActiveReport_ReportStart()
{
    rpt.ResourceLocator = new GrapeCity.ActiveReports.DefaultResourceLocator(new
System.Uri("https://demodata.mescius.io"));
}
```

6. To provide a report title, drag and drop a **Label** control onto the PageHeader section of the report and set its **Text** property to 'Employee List'.

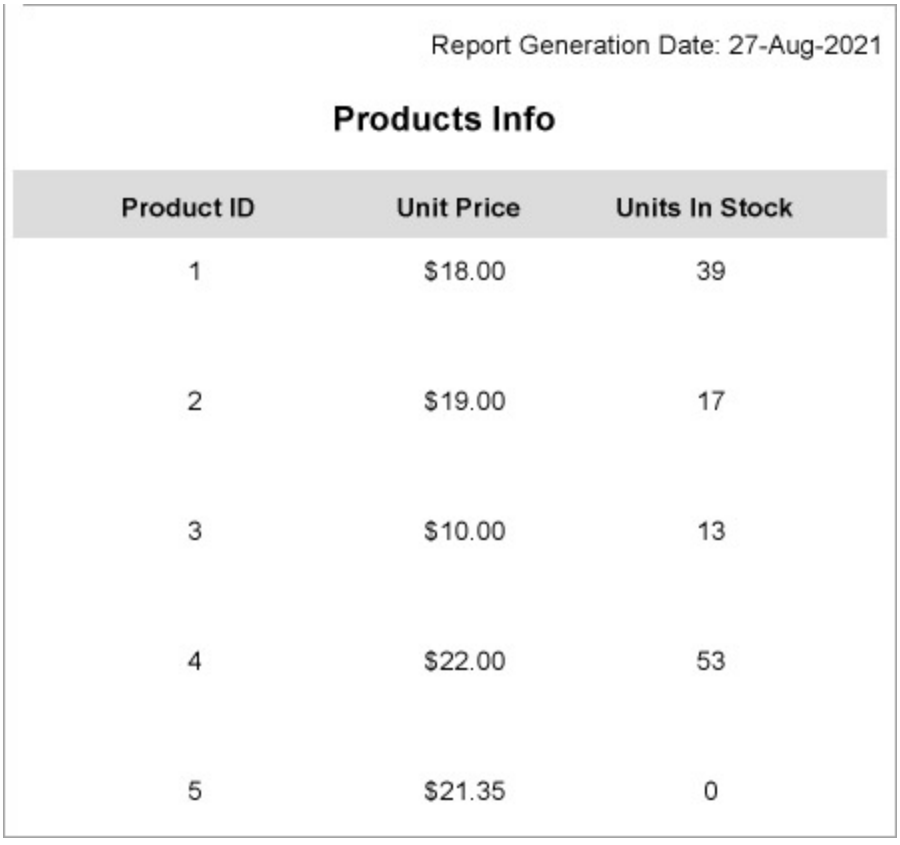


7. Improve the appearance of the report and preview.

TextBox Control in Reports

Let us create a report that uses TextBox controls to display the price and in stock information, and total number of items in stock. The report connects to the 'Products' JSON data available here.

The final report will look as shown.



Create a Report

In the ActiveReports Designer, create a new Section report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by clicking the Data Source icon on the Detail section.
2. In the dialog, go to the **JSON** page and click Build to open the **Configure JSON Data Source** dialog.
3. Enter the following URL in the **Data Path** as follows:
   https://demodata.mescius.io/northwind/odata/v1/Products
4. Click **OK** to close the dialog.
   The Connection String section displays the connection string as follows:

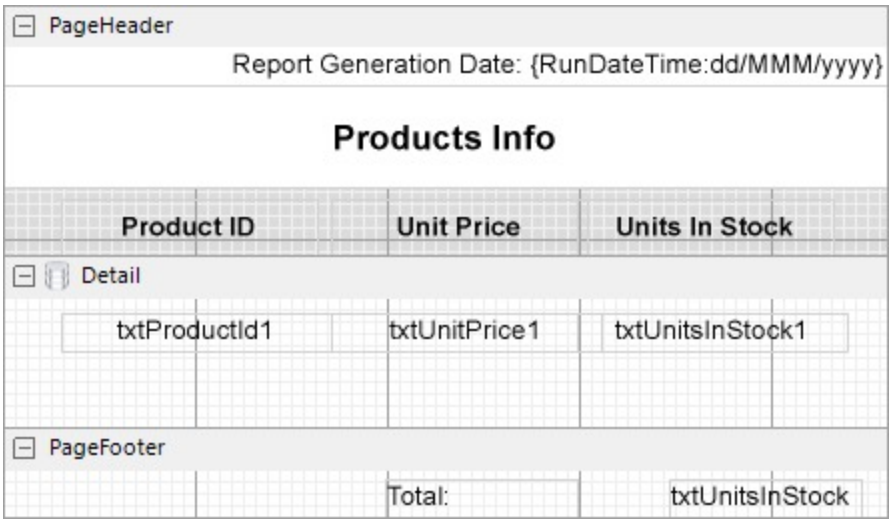
Connection String
`jsondoc=https://demodata.mescius.io/northwind/odata/v1/Products`

5. Click **OK** to save the changes and close the **Report Data Source** dialog.
6. To fetch the required fields, enter the following query in the **JSON Path**.

Query
`$.value[*]`

Design Report Layout

1. Drag the following data fields to the **Detail** section of the report: `ProductId`, `UnitPrice`, and `UnitsInStock`. All these fields are bound text boxes.
2. To provide headings (labels) to the data displayed in the text boxes in Detail section, drag and drop three **Label** controls onto the Page Header section of the report and set the **Text** property of the text boxes to 'Product ID', 'Unit Price', and 'Units In Stock', respectively.
3. To display the total number of products in stock, drag and drop two **TextBox** controls onto the Footer section of the report and set the properties as follows.
   - TextBox1: Set the **Text** property to Total:. This is an unbound text box.
   - TextBox2: Set the **DataField** property to UnitsInStock and **SummaryType** property to GrandTotal. This textbox will show the total number of products in stock.
4. To provide the report's title, drag and drop another **TextBox** control onto the top center in the PageHeader section. Enter the text of the title into the **Text** property or directly inside the text box, e.g. 'Products Info'.
5. To add the information on the report generation date, drag and drop the **ReportInfo** control on the upper right corner of the PageHeader section and set the **FormatString** property to the expression: `Report Generation Date: {RunDateTime:dd/MMM/yyyy}`.



6. Modify the appearance of the report and preview.

Tutorials: Section Report Scenarios

This section provides you with step-by-step instructions for creating commonly used section reports.

- Create Top N Report
- Create a Summary Report
- Create Green Bar Report
- Create Address Labels in Section Report

Create Top N Report

In a Section report, in order to display only the top N number of rows or records on a report, you can manipulate the data in query, or use script depending on the data source. Let's see the working example for a report bound to an Sqlite data source. Your data will be more meaningful if you also specify the order of rows.

1. Create a new Section report and bind the data to 'reels.db'. See Custom Data Provider topic for more information. The data source connection string will be as follows:

Data Source Connection String
`data source = C:\Data\reels.db`

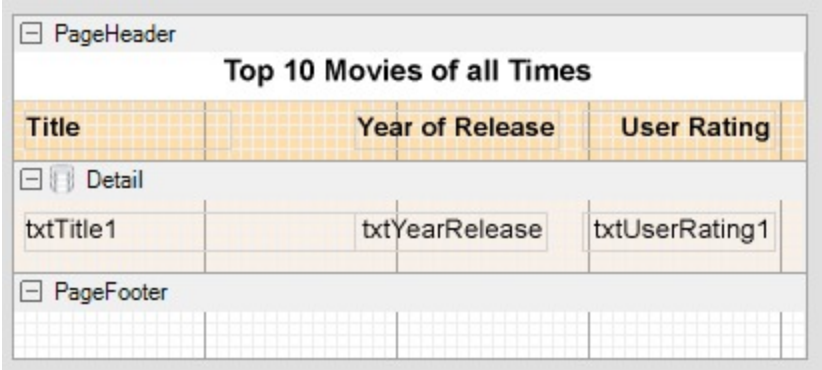
2. In the Report Data Source dialog, paste the following SQL query in the Query field to fetch **Top 10** records from the database, ordered by user rating from highest to lowest:

   ### Sqlite

Query
`SELECT * FROM Movie Order By UserRating Desc LIMIT 10`

3. Then click **OK** to close the dialog.

4. In the Report Explorer, expand the **Fields** node, then the **Bound** node.
5. Drag and drop the following fields onto the detail section and set the properties of each textbox as indicated.
   `[Title]`
   `[YearReleased]`
   `[UserRating]`
6. Drag-drop a Label controls in the PageHeader section for labeling the details in the Detail section, and for the report heading.



7. Preview the report. You will notice only **10** number of records displaying in your report. The following image illustrates Top N Report displaying top 10 movie records:

Top 10 Movies of all Times

Title	Year of Release	User Rating
The Lord of the Rings: The Fellowship of the Ring	2001	10
Lethal Weapon 3	1992	10
Superman	1978	10
Mission: Impossible II	2000	9.9
The Polar Express	2004	9.9
The Longest Yard	2005	9.9
What Lies Beneath	2000	9.9
The Silence of the Lambs	1991	9.9
Sleepless in Seattle	1993	9.9
Rocky	1976	9.9

The following sections provide sample queries or script that must be used to obtain top N data for other data sources.

OLEDB, ODBC, SQL

Query
`SELECT Top 10 * FROM Movie`

XML

Query
`//countries/country[position() <= 10]`

JSON

Query
`$.Customers[:10]`

CSV

Script

```
int i = 1;
public bool ActiveReport_FetchData(bool eof)
{
    return i++ > 10;
}
public void ActiveReport_ReportEnd()
{
    i = 0;
}
```

Create a Summary Report

In a section report, you can display totals and subtotals by modifying the summary fields of a TextBox control. Use the following steps to learn how to add totals and subtotals in a report.

> **Note**: These steps use the Products table from the NWind database. The NWIND.db file can be downloaded from GitHub.

Create a Report

In the ActiveReports Designer, create a new Section report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by clicking the **DataSource Icon** in the Detail section band.
2. Choose **Custom** tab > **SQLite Provider** in the Report Data Source dialog, and bind the report to Sqlite data using the following connection string and query.

Connection String
`data source=c:\data\NWIND.db`

DataSet Query
`Select * from Products  ORDER BY CategoryID`

3. Click **OK** to close the Report Data Source dialog and return to the report design surface.

Design Report Layout

Calculate and display subtotals in a report

UNITS IN STOCK

Category ID: 1	
Product Name	**Units In Stock**
Chartreuse verte	69
Chang	17
Guaraná Fantástica	20
Sasquatch Ale	111
Steeleye Stout	20
Chai	39
Côte de Blaye	17
Ipoh Coffee	17
Laughing Lumberjack Lager	52
Outback Lager	15
Rhönbräu Klosterbier	125
Lakkalikööri	57
Total Units:	**559**

Category ID: 2	
Product Name	**Units In Stock**
Genen Shouyu	39
Northwoods Cranberry Sauce	6
Original Frankfurter grüne Soße	32
Grandma's Boysenberry Spread	120
Gula Malacca	27
Chef Anton's Gumbo Mix	0
Chef Anton's Cajun Seasoning	53
Aniseed Syrup	13
Louisiana Fiery Hot Pepper Sauce	76
Louisiana Hot Spiced Okra	4
Vegie-spread	24
Sirop d'érable	113
Total Units:	**507**

1. Right-click the design surface and select **Insert**, then **Group Header/Footer** to add group header and group

footer sections to the layout.
2. With the GroupHeader section selected in the Properties Window, set its **DataField** property to *CategoryID*. This groups the data on the report according to the set field.
3. From the Report Explorer, drag and drop the following fields onto the corresponding sections of the report.

    - **Field Name** = CategoryID
      Section = GroupHeader

    - **Field Name** = ProductName
      Section = Detail

    - **Field Name** = UnitsInStock
      Section = Detail

    - **Field Name** = UnitsInStock
      Section = GroupFooter

4. With the UnitsInStock field in the GroupFooter selected, go to the Properties Window and set the following:
    - SummaryFunc: Sum
    - SummaryType: Sub Total
    - SummaryRunning: Group
    - SummaryGroup: GroupHeader1
5. Preview the report and see the **Sub Total** appear below each group of data similar to the following image.

Calculate and display the grand total in a report

Category ID: 8	
Product Name	**Units In Stock**
Carnarvon Tigers	42
Konbu	24
Nord-Ost Matjeshering	10
Boston Crab Meat	123
Ikura	31
Inlagd Sill	112
Gravad lax	11
Jack's New England Clam Chowder	85
Røgede sild	5
Spegesild	95
Escargots de Bourgogne	62
Röd Kaviar	101
Total Units:	**701**
Stock Grand Total:	**3119**

These steps are a continuation of the procedure above. The report generated at the end of this procedure contains

totals and subtotals.

1. Right-click the design surface and select **Insert**, then **Report Header/Footer** to add report header and footer sections to the layout.
2. From Report Explorer, drag and drop the UnitsInStock field onto the ReportFooter section go to the Properties Window and set the following:
    - SummaryFunc: Sum
    - SummaryType: GrandTotal
    - SummaryRunning: All
3. Preview the report and see the **Grand Total** appear on the last page of the report.

Create Green Bar Report

In a section report, green bar printouts can be created by setting alternate shades or background color in the report's detail section in the **Format** event. The following steps demonstrate how to create a Green Bar report in a sample report **PurchaseOrder.rpx** that can be can be downloaded from GitHub: ..\Samples18\DesignerPro\ReportsGallery\Reports\Section Report\.

Design Report Layout

Company	Store Name	Company Code	Product Class	Supplier	Date
AAA City Hall	General Affairs Division	A00001	W	Shwan Co Ltd, California	2009/06/29

Product	Product Type	Color	Size	Unit	Quantity	Cost	Total Cost	Price	Total Price
Desktop personal computer	DTW1001	Ivory	HD500	Stand	2	$55,000	$110,000	$66,800	$133,600
Desktop personal computer	DTW1002	Black	HD500	Stand	2	$55,000	$110,000	$66,800	$133,600
Laptop	NTW1001	White	HD160	Stand	3	$89,800	$269,400	$89,800	$269,400
Laptop	NTW1002	Silver	HD160	Stand	3	$89,800	$269,400	$89,800	$269,400
External HDD	HDD1001	Black	HD1000	Stand	1	$8,800	$8,800	$10,800	$10,800
Wide LCD Monitor	HD2401	Black	24inch	Stand	5	$39,800	$199,000	$45,800	$229,000
Laptop	NTW1003	White	HD500	Stand	1	$115,000	$115,000	$115,000	$115,000
Laptop	NTW1004	Pink	HD500	Stand	1	$115,000	$115,000	$115,000	$115,000
Wide LCD Monitor	HD2401	Black	24inch	Stand	1	$39,800	$39,800	$45,800	$45,800
Desktop personal computer	DTW1001	Ivory	HD500	Stand	2	$55,000	$110,000	$66,800	$133,600
Desktop personal computer	DTW1002	Black	HD500	Stand	2	$55,000	$110,000	$66,800	$133,600
Laptop	NTW1001	White	HD160	Stand	3	$89,800	$269,400	$89,800	$269,400
Wireless LAN router	WIFI1001	Black	300Mbs	Stand	2	$17,800	$35,600	$17,800	$35,600
Projector	MP555	Black	SXGA	Stand	2	$59,000	$118,000	$59,000	$118,000
UPS	UPS50BK	Black	500VA	Stand	5	$14,500	$72,500	$14,500	$72,500

1. On the design surface of the report, double-click the detail section of the report to create an event handling

method for the **Detail Format** event.

2. Add the following script to create alternate background colors.

Visual Basic.NET code. Paste JUST ABOVE the Detail Format event.

```
Dim i As Integer = 0
```

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

```
If i Mod 2 = 0 Then
    Me.detail.BackColor = System.Drawing.Color.PaleGreen
Else
    Me.detail.BackColor = System.Drawing.Color.Transparent
End If
i += 1
```

C# code. Paste JUST ABOVE the Detail Format event.

```
int i = 0;
```

C# code. Paste INSIDE the Detail Format event.

```
if (i % 2 == 0) {
  this.detail.BackColor = System.Drawing.Color.PaleGreen;
} else {
  this.detail.BackColor = System.Drawing.Color.Transparent;
}
i++;
```

3. Preview the report and see a Green Bar report alternating between Transparent and Pale Green backgrounds.

Create Address Labels in Section Report

You can use ActiveReports to print any label size by using the newspaper column layout.

These steps show 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 report connects to 'NWIND.db' data source on GitHub.

The final report will look as shown.

Create a Report

In the ActiveReports Designer, create a new Section report.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by clicking the **DataSource Icon** in the Detail section band.
2. Choose **Custom** tab > **SQLite Provider** in the Report Data Source dialog, and bind the report to SQLite data using the following connection string and query.

Connection String
`data source=c:\data\NWIND.db`

DataSet Query
`SELECT ContactName, CompanyName, Address, City, PostalCode, Country FROM Customers`

3. Click **OK** to close the Report Data Source dialog and return to the report design surface.

Design Report Layout

1. Right-click the PageHeader section and select **Delete** to remove the PageHeader and PageFooter sections from the report.
2. Go to **Report Explorer**, right-click the **Settings** option and select **Show** to view the Report Settings.

3. Change the margins as follows:
    - Top margin: 0.5
    - Bottom margin: 0.5
    - Left margin: 0.2
    - Right margin: 0.2
4. From the Report Explorer, select **Report** and in the **Properties** panel, set the **PrintWidth** property to **8.1** (the width of the label sheet less the Left and Right margins).
5. Click the Detail section of the report to select it and in the Properties window, set the properties as follows.

Property Name	Property Value
CanGrow	False
ColumnCount	3
ColumnSpacing	0.2
ColumnDirection	AcrossDown
Height	1

6. From the toolbox, drag the following fields one below the other and set the properties of each textbox as follows.

txtContactName1

Property Name	Property Value
DataField	ContactName
Location	0, 0 in
Size	2.5, 0.2 in
Font > Bold	True

txtCompanyName1

Property Name	Property Value
DataField	CompanyName
Location	0, 0.2 in
Size	2.5, 0.2 in

txtAddress1

Property Name	Property Value
DataField	Address
Location	0, 0.4 in
Size	2.5, 0.2 in

txtCity1

Property Name	Property Value
DataField	City
Location	0, 0.6 in
Size	2.5, 0.2 in

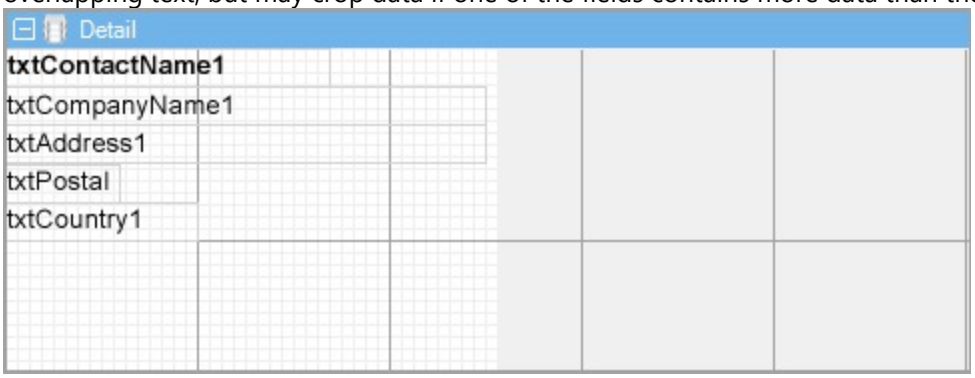
txtPostalCode1

Property Name	Property Value
DataField	PostalCode
Location	0, 0.8 in
Size	1.45, 0.2 in

txtCountry1

Property Name	Property Value
DataField	Country
Location	1.5, 0.8 in
Size	1, 0.2 in

7. Select all of the textboxes, and in the Properties Panel, 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.

Add Code to 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.

Example Title
`Imports GrapeCity.ActiveReports.SectionReportModel`

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
    rpt.LayoutAction = GrapeCity.ActiveReports.LayoutAction.MoveLayout Or
 GrapeCity.ActiveReports.LayoutAction.PrintSection
Else
    rpt.LayoutAction = GrapeCity.ActiveReports.LayoutAction.MoveLayout Or
 GrapeCity.ActiveReports.LayoutAction.NextRecord Or
 GrapeCity.ActiveReports.LayoutAction.PrintSection
    counter = 0
End If
```

C# code. Paste JUST ABOVE the Format event

```
using GrapeCity.ActiveReports.SectionReportModel;
int counter=0;
```

C# code. Paste INSIDE the Format event

```
//print each label three times
counter = counter + 1;
if (counter <= 2)
{
    rpt.LayoutAction =  GrapeCity.ActiveReports.LayoutAction.MoveLayout|
GrapeCity.ActiveReports.LayoutAction.PrintSection;
}
else
{
    rpt.LayoutAction = GrapeCity.ActiveReports.LayoutAction.MoveLayout|
GrapeCity.ActiveReports.LayoutAction.NextRecord|
GrapeCity.ActiveReports.LayoutAction.PrintSection;
    counter = 0;
}
```

3. Improve the appearance of the report and preview.

DevOps

While developers build applications and develop features for users, DevOps are IT specialists responsible for setting up processes, deploying applications, and maintaining the same.

In this section, you will find information about the ActiveReports processes that are not related directly to development practices, but to the application infrastructure:

- installation,
- configuration,
- customization,
- localization, and
- deployment.

ActiveReports Editions

Available in two editions, Standard and Professional, ActiveReports delivers outstanding reporting capabilities.

Standard Edition Features

The Standard Edition provides a report designer that is fully integrated with the Visual Studio IDE, a report viewer for Windows Forms, and export filters for generating reports in various file formats. The report designer even includes a barcode control with all of the most popular barcode styles, and its own chart control.

Designer

- Full integration with the .NET environment
- Familiar user interfaces
- Choice of section or a Page or an RDLX report types
  - C# and VB.NET support with code-based section reports
  - Script support with XML-based section reports
  - Expression support with page reports and RDLX reports
- 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

Section Reports	Page Reports/RDLX Reports
• ReportInfo • Label • Line • PageBreak • OleObject • SubReport • Shape • Picture	• Table data region • Tablix data region • Map data region • Chart data region • List data region • BandedList data region • Sparkline data region • FormattedText with mail merge capabilities and

- RichTextBox with HTML tag support
- Chart with separate data source
- Textbox
- Barcode with standard styles plus RSS and UPC styles
- Checkbox
- CrossSectionBox extends from a header section to the related footer section
- CrossSectionLine extends from a header section to the related footer section

XHTML + CSS support
- Bullet Graph
- BarCode
- CheckBox
- TextBox
- TableOfContents
- Line
- Container
- Shape
- Image
- Subreport
- Overflow Placeholder

Expressions (Page reports/RDLX reports)

- Aggregates
- Data visualization
  - Data bar
  - Icon set
  - Range bar
  - Color scale
  - Gradient
  - Hatch

Interactive Features

- Document map (table of contents)
- Bookmark links, hyperlinks, and drill through links
- Parameters
- Drill-down (Page report/RDLX reports)
- Copy, pan, and zoom
- Jump to previous, next, first, or last group or search result

Reporting Engine

- Managed code
- Binding to ADO.NET, XML, iList, and custom data sources
- Master reports, themes, and styles
- All of the features of previous versions of ActiveReports and Data Dynamics Reports

Windows Forms Report Viewer

- Managed C# code
- Very small deployment assembly, suitable for use on the Internet
- Table of contents and bookmarks
- Thumbnail view

- HyperLinking
- Annotations (section reports only)
- Configurable scroll bar jump buttons (like those found in Microsoft® Word®)
- Parameters
- Bookmark links, hyperlinks and drillthrough links
- Interactive sorting (page reports/RDLX reports)
- Touch mode

WPF Viewer

- Managed C# code
- Table of contents and bookmarks
- Thumbnail view
- Parameters
- Annotations
- Configurable scroll bar jump buttons (like those found in Microsoft® Word®)
- Bookmark links, hyperlinks and drillthrough links
- Interactive sorting

Export Filters

ActiveReports includes export filters to generate output into many popular formats.

Export formats	Section report	Page/RDLX report
Html: Export reports to HTML, DHTML, or MHT formats, all of which open in a Web browser.	✓	✓
Pdf: Export reports to PDF, a portable document format that opens in the Adobe Reader.	✓	✓
Rtf: Export reports to RTF, RichText format that opens in Microsoft Word, and is native to WordPad.	✓	✓
Word: Export reports to DOC, a format that opens in Microsoft Word.	✗	✓
Text: Export reports to TXT, plain text, a format that opens in Notepad or any text editor. Export reports to CSV, comma separated values, a format that you can open in Microsoft Excel.	✓	✓
Image: Export reports to BMP, GIF, JPEG, or PNG image format.	✗	✓
Tiff: Export reports to TIFF image format for optical archiving and faxing.	✓	✓
Excel: Export reports to formats that open in Microsoft Excel, XLS or XLSX.	✓	✓
Xml: Export reports to XML, a format that opens in a Web browser or delivers data to other applications.	✗	✓
CSV: Export reports to a CSV file, a form of structured data in plain text. The text in a CSV file is saved as series of values separated by comma.	✗	✓
JSON: Export reports to a JSON file, a text-based data format in which the data is stored in the hierarchical form.	✗	✓

Import Filters

- Access® Reports
- Crystal Reports
- SSRS Reports
- Excel file
- RPX file

Standalone Applications

- The Report Designer application lets you create report layout, define data, and add interactive features report, and lets you preview the reports and export or print them. It can be opened from the shortcut provided in the Start menu.
- The Report Viewer application contains all the functionality of the ReportPreview control. It can be opened from the shortcut provided in the Start menu.
- The WPF Viewer application contains all the functionality of the WPF Viewer control.
- The ActiveReports Import Wizard application allows importing Microsoft Access reports, Crystal Reports, SSRS Reports, Excel files and RPX files. It can be opened from the shortcut provided in the Start menu.

Professional Edition Features

The 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 RDLX) or compiled assemblies containing reports to be dropped on the server and hyperlinked.

WebViewer Control

- The WebViewer control allows quick viewing of ActiveReports on the web as well as printing capability with the AcrobatReader ViewerType enumeration.

HTTP Handlers

- The RPX and RDLX 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.

Table of Contents Control

- TableOfContents control is used to display the document map, an organized hierarchy of the report heading levels and labels along with their page numbers, in the body of a report.
- TableOfContents control allows you to quickly understand and navigate the data inside a report in all viewers that are supported in ActiveReports

Map Control

- The Map data region shows your business data against a geographical background.
- Create different types of map, depending on the type of information you want to communicate in your report.

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 or the PDF rendering extension looks for the glyphs in fonts declared in the FontFallback property.
- 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.

PDF Export

- PDF/A and PDF/UA support.
- IVS character support.
- Devanagari character support.

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 for the PDF output. The Professional Edition provides bold style emulation in the PDF export filter to eliminate this limitation.

Word Export

- Export your reports in .docx format, a format that opens in Microsoft Word application.

WebDesigner

- Create or modify reports by embedding WebDesigner into your web applications.

JSViewer and Blazor Viewer

- View reports in all modern browsers using JSViewer or Blazor Viewer.

InputField Control

- The InputField report control provides support for editable fields in an exported PDF report where the InputField's value can be modified.
- Choose one of the two report control types – Text and Checkbox. Each selected type has its own set of properties.

Comparison Between Editions

Professional Edition features are disabled or marked with an evaluation banner if you have purchased a Standard Edition license.

Features		Standard	Professional
Visual Studio Controls			
Web Forms	**WebViewer**: Use this control to display your reports on the Web. Includes viewer types HTML and PDF.	✗	✓
	HTTP Handlers: PDF and HTML (compiled report, RPX file)	✗	✓
Windows Forms	**Viewer**: Use this control to offer your users report zoom and preview, multiple tabs for hyperlinks, split-page and multi-page views, a Table of Contents pane, a Thumbnails pane, text searches, and annotations.	✓	✓
	Designer: Use this control to create a royalty-free, custom designer that your end users can use to create and modify their own reports.	✗	✓
	ReportExplorer: Use this control along with the Designer control to provide functionality to your users.	✓	✓
	ToolBox: Use this control along with the Designer control to provide report controls for your users.	✓	✓
	LayerList: Use this control along with the Designer control to provide Layers functionality to your users.	✓	✓
	ReportsLibrary: Use this control to view, add, or hide report parts.	✓	✓
WPF	**WPF Viewer**: Use this control to display your section, page and RDLX reports.	✓	✓

	The WPF Viewer offers the Thumbnails pane, the Parameters pane, the Document map pane, and the Search results pane.		
Web and Windows Forms	**HtmlExport**: Export reports to HTML, DHTML, or MHT formats that open in a Web browser.	✓	✓
	PdfExport: Export reports to PDF, a portable document format that opens in the Adobe Reader.	✓	✓
	RtfExport: Export reports to RTF, RichText format that opens in Microsoft Word, and is native to WordPad.	✓	✓
	WordExport (.doc): Export reports to DOC (Word HTML), a format that opens in Microsoft Word.	✓	✓
	WordExport (.docx): Export reports to DOCX (LibreOffice), a format that opens in any word processing software.	✗	✓
	TextExport: Export reports to TXT, plain text, a format that opens in Notepad or any text editor. This export filter can also export reports to CSV, comma separated values, a format that you can open in Microsoft Excel.	✓	✓
	ImageExport: Export reports to BMP, GIF, JPEG, TIFF, or PNG image format. Note that you can only export section reports to the TIFF image type. All other image types are for page reports and RDLX reports.	✓	✓
	XlsExport: Export reports to formats that open in Microsoft Excel, XLS or XLSX.	✓	✓
	XmlExport: Export reports to XML, a format that opens in a Web browser or delivers data to other applications.	✓	✓
Components	**WebDesigner**: Create or modify reports by embedding WebDesigner into your web applications.	✗	✓
	JSViewer: View reports in all modern browsers using JSViewer.	✗	✓
	Blazor Viewer: View reports in all modern browsers using Blazor Viewer.	✗	✓
PDF Export Advanced Features	Digital Signature	✗	✓
	Time Stamp	✗	✓
	PAdES in Digital Signature	✗	✓
	EUDC	✗	✓
	Select from Japanese embedded fonts or unembedded fonts	✗ *1	✓
	Bold	✗	✓
	Italic	✓	✓
	Multi Language	✓ *2	✓
	PDF/A and PDF/UA Support	✗	✓
	IVS Character Support	✗	✓

<table>
<tr><td></td><td>Devanagari Character Support</td><td>✗</td><td>✓</td></tr>
<tr><td></td><td>Print Presets</td><td>✗</td><td>✓</td></tr>
<tr><td colspan="4">Integrated Report Designer</td></tr>
<tr><td>Design Format</td><td>Section reports support banded layouts.
Page reports support fixed page layouts.
RDLX reports support continuous page layout.</td><td>✓</td><td>✓</td></tr>
<tr><td>Script and Code</td><td>In section reports, you can add C# or VB code to events behind your code-based reports, or add script to events in the script editor in XML-based reports. In page reports/RDLX reports, you can use regular expressions in any property, plus you can add VB.NET methods to the code tab, and call them in your expressions.</td><td>✓*3</td><td>✓*3</td></tr>
<tr><td>Report File Formats</td><td>You can save and load page reports/RDLX reports in RDLX (extended RDLX) format.
You can save and load section reports in RPX (report XML) format, and you can compile section reports in CS or VB code formats.</td><td>✓</td><td>✓</td></tr>
<tr><td>Report Controls</td><td>The BarCode control supports all of the following styles:
<table>
<tr><td>ANSI 3 of 9</td><td>ANSI Extended 3 of 9</td><td>Code 2 of 5</td><td>Interleaved 2 of 5</td></tr>
<tr><td>Code 25 Matrix</td><td>Code 39</td><td>Extended Code 39</td><td>Code 128 A</td></tr>
<tr><td>Code 128 B</td><td>Code 128 C</td><td>Code 128 Auto</td><td>Code 93</td></tr>
<tr><td>Extended Code 93</td><td>MSI</td><td>PostNet</td><td>Codabar</td></tr>
<tr><td>EAN-8</td><td>EAN-13</td><td>UPC-A</td><td>UPC-E0</td></tr>
<tr><td>UPC-E1</td><td>RoMail RM4SCC</td><td>UCC/EAN-128</td><td>QRCode</td></tr>
<tr><td>Code 49</td><td>Japanese Postal</td><td>Pdf417</td><td>EAN-128 FNC1</td></tr>
<tr><td>RSS-14</td><td>RSS-14 Truncated</td><td>RSS-14 Stacked</td><td>MicroPdf417</td></tr>
<tr><td>RSS-14 Stacked Omnidirectional</td><td>RSS Expanded</td><td>RSS Expanded Stacked</td><td>MicroQRCode</td></tr>
<tr><td>BC412</td><td>Code_11</td><td>ISBN</td><td>ISMN</td></tr>
<tr><td>ISSN</td><td>ITF14</td><td>MaxiCode</td><td>Pharmacode</td></tr>
<tr><td>Plessey</td><td>PZN</td><td>SSCC_18</td><td>Telepen</td></tr>
<tr><td>IntelligentMail</td><td>IntelligentMailPackage</td><td>HIBC Code 128</td><td>HIBC Code 39</td></tr>
</table></td><td>✓</td><td>✓</td></tr>
</table>

	Aztec2D | GS1		
The **InputField** control provides support for editable fields in an exported PDF report where the InputField's value can be modified.	✗	✓
The **Map** control allows you to display data against a geographical background on the report.	✗	✓
The **TableofContents** control allows you to display a document map in an organized hierarchy of the report heading levels and labels along with there page numbers, in the body of a report.	✗	✓

The **Chart** control supports all of the following styles:
• **Common Charts**: Area, Bar2D, Bezier, Doughnut/Pie, Line, Scatter, StackedArea, StackedBar, StackedArea100Pct, and StackedBAR110Pct
• **3D Charts**: Area3D, Bar3D, ClusteredBar, Line3D, Doughnut3D/Pie, StackedBar3D, and StackedBar3D100Pct
• **XY Charts**: Bubble, BubbleXY, LineXY, and PlotXY
• **Financial Charts**: Candle, HiLo, and HiLoOpenClose
• Composite Charts for following chart types:
  ○ Column: Plain, Stacked, Percent Stacked
  ○ Area: Plain, Stacked, Percent Stacked
  ○ Line: Plain, Smooth
	✓	✓

Other report controls include: Label | TextBox | CheckBox | Picture Line | Shape | RichText | PageBreak SubReport | ReportInfo | CrossSectionLine | CrossSectionBox	✓	✓
Styles and Report Settings	You can control page settings, printer settings, global settings such as grid display, grid size, and whether to show a verification dialog when deleting controls. You can specify row count or column count in grids, ruler units, and how many pages to display in previews.	✓	✓
External Style Sheets	You can reuse report designer styles by saving and loading style information in external files.	✓	✓
Others	The designer also offers snaplines, report preview, designer zoom, various formatting settings, control and text alignment settings, Z order settings, unbound fields, and parameters support.	✓	✓
Input and Output			
Data	Supported data includes: JSON, CSV, XML, Microsoft OLEDB, Microsoft ODBC, Microsoft SQL Client, Dataset, and Object providers, and custom data like Sqlite.	✓	✓
Print	You can control the page size, orientation, and margins, as well as specifying bound (double page spread), collating, duplex printing, and paper feed trays.	✓	✓

Import	You can import Crystal Reports, MS Access Reports, SSRS Reports, Excel files, and RPX files using the ActiveReports Import Wizard.	✓	✓

*1: Japanese fonts can only be output as embedded fonts.
*2: Cannot handle output of multiple language fonts in a single control. Please refer to Multi-Language PDF for details.
*3: See Design Code-based Section Reports in .NET Core for more information.

Open Source Software

ActiveReports supports multiple types of Open Source software (hereinafter, referred to as "OSS") programs based on the license agreement and usage rights.

The following OSS are included as a part of this software.

Development Environment

Visual Studio Integration

- **VSIXBootstrapper**
  © Microsoft Corporation. All rights reserved.
  (MIT License)
- **NuGet.VisualStudio**
  © Microsoft Corporation. All rights reserved.
  (Apache License 2.0)
- **Npgsql**
  Copyright 2021 © The Npgsql Development Team
- **System.Data.SQLite**
  Copyright © 2004-2008 Hipp, Wyrick & Company, Inc.
- **MySqlConnector**
  Copyright 2016–2023 Bradley Grainger
- **Newtonsoft.Json**
  Copyright © James Newton-King 2008

Import Wizard

- **DocumentFormat.OpenXml**
  © Microsoft Corporation. All rights reserved.
  (MIT License)
- **Rtf2Html**
  (Apache License 2.0)

Operating Environment

Report Engine

- **MartinezClipper**
  Copyright (c) 2016 BrightBit
  (MIT License)
- **Microsoft.Xml.SgmlReader**
  Copyright c 2002-2021, Microsoft Corporation; Copyright c 2007-2013, MindTouch
  (Apache License 2.0)
- **Okapi Barcode**

Copyright 2014-2015 Robin Stuart, Daniel Gredler
(Apache License 2.0)

- **ZXing-Csharp**
  Copyright 2008 ZXing authors
  (Apache License 2.0)
- **ZXing.Net**
  Copyright (C) 2010 ZXing authors
  (Apache License 2.0)
- **Knyaz.Optimus**
  Copyright (c) 2016 Konstantin Vladimirovich Petukhov
  (MIT License)

Export

- **DocumentFormat.OpenXml**
  © Microsoft Corporation. All rights reserved.
  (MIT License)
- **BouncyCastle.Cryptography**
  Copyright (c) 2000 - 2023 Legion of the Bouncy Castle Inc.
- **MemoryTributary**
  Article Copyright 2012 by sebastianfriston
  (Code Project Open License 1.02)
- **SvgNet**
  Copyright © 2003 RiskCare Ltd.
  Copyright © 2010 SvgNet & SvgGdi Bridge Project
  Copyright © 2015-2019 Rafael Teixeira, Mojmír Němeček, Benjamin Peterson and Other Contributors
  All rights reserved.
  (BSD 3-Clause “New” or “Revised” License)
- **ColorMine**
  Copyright © 2013 ColorMine.org
  (MIT License)
- **SharpDX**
  Copyright © 2010-2015 SharpDX - Alexandre Mutel
  (MIT License)
- **SixLabors.ImageSharp**
  Copyright © Six Labors
  (Apache License 2.0)
- **Cmap Resources**
  Copyright 1990-2019 Adobe. All rights reserved.
  (BSD 3-Clause “New” or “Revised” License)
- **ExCSS**
  (MIT License)
- **NPOI**
  (Apache License 2.0)

End User Designer

- **FlagEnumUIEditor**
  Article Copyright 2006 by LogicNP
  (Code Project Open License 1.02)
- **SharpExpress**

Copyright © 2014 Sergey Todyshev
(MIT License)

- **Irony**
  Copyright © 2019 Irony Project (https://github.com/IronyProject)
  (MIT License)
- **jQuery**
  Copyright OpenJS Foundation and other contributors (https://openjsf.org/)
  (MIT License)
- **Knockout**
  Copyright © 2010 Steven Sanderson, the Knockout.js team, and other contributors (https://knockoutjs.com/)
  (MIT License)
- **Bootstrap**
  Copyright © 2011-2022 Twitter, Inc.
  Copyright © 2011-2022 The Bootstrap Authors
  (MIT License)
- **spin.js**
  Copyright © 2011-2018 Felix Gnass [fgnass at gmail dot com]
  (MIT License)
- **korest**
  Copyright © 2014 Sergey Todyshev
  (MIT License)
- **Rtf2Html**
  (Apache License 2.0)

JS Viewer/Blazor Viewer/WebViewer

- **AbortController polyfill**
  Copyright (c) 2017 molsson
  (MIT License)
- **Classnames**
  Copyright (c) 2018 Jed Watson
  (MIT License)
- **core-js**
  Copyright (c) 2014-2022 Denis Pushkarev
  (MIT License)
- **downloadjs**
  Copyright (c) 2016 dandavis
  (MIT License)
- **i18next**
  Copyright (c) 2022 i18next
  (MIT License)
- **i18next-browser-languagedetector**
  Copyright (c) 2022 i18next
  (MIT License)
- **immutability-helper**
  Copyright (c) 2017 Moshe Kolodny
  (MIT License)
- **whatwg-fetch**
  Copyright (c) 2014-2016 GitHub, Inc.
  (MIT License)
- **Microsoft.Owin**

- **Microsoft.Owin.Host.SystemWeb**

- **Owin**
(Apache License 2.0)
- **React**

(MIT License)

WebDesigner

- **Webfont - Material Design Icons**
(Apache License 2.0)(MIT License)
- **Open Sans - Google Fonts**
(Apache License 2.0)
- **deep-object-diff**

(MIT License)
- **events**

(MIT License)
- **memoize-one**

(MIT License)
- **raf**

(MIT License)
- **Bowser**

(MIT License)
- **Chai**

(MIT License)
- **Color**

(MIT License)
- **Excel-Style Javascript Data Formatter**
(MIT License)
- **i18next**

(MIT License)
- **immutability-helper**

(MIT License)
- **kanjidate**

(MIT License)
- **Lodash**

Copyright OpenJS Foundation and other contributors (https://openjsf.org/)
(MIT License)

- **Moment.js**
  Copyright (c) OpenJS Foundation and other contributors
  (MIT License)
- **React**
  Copyright (c) Facebook, Inc. and its affiliates.
  (MIT License)
- **React DnD**
  Copyright (c) 2015 Dan Abramov
  (MIT License)
- **React DnD HTML5 Back End**
  Copyright (c) 2015 Dan Abramov
  (MIT License)
- **react-dom**
  Copyright (c) Facebook, Inc. and its affiliates.
  (MIT License)
- **react-onclickoutside**
  Copyright 2015-2022 Mike "Pomax" Kamermans
  (MIT License)
- **React Redux**
  Copyright (c) 2015-present Dan Abramov
  (MIT License)
- **Redux**
  Copyright (c) 2015-present Dan Abramov
  (MIT License)
- **Redux-Saga**
  Copyright (c) 2015 Yassine Elouafi
  (MIT License)
- **redux undo/redo**
  Copyright (c) 2015 Daniel Bugl
  (MIT License)
- **seedrandom.js**
  Copyright 2019 David Bau.
  (MIT License)
- **string-format**
  Copyright (c) 2018 David Chambers (email: dc@davidchambers.me)
  (MIT License)
- **tslib**
  Copyright (c) Microsoft Corporation.
  (BSD Zero Clause License)
- **typesafe-actions**
  Copyright (c) 2017 Piotr Witek (email: piotrek.witek@gmail.com) (https://piotrwitek.github.io)
  (MIT License)
- **uuid**
  Copyright (c) 2010-2020 Robert Kieffer and other contributors
  (MIT License)
- **XRegExp**
  Copyright (c) 2007-present Steven Levithan (https://xregexp.com/)
  (MIT License)

- **Microsoft.Owin**
  © Microsoft Corporation. All rights reserved.
  (Apache License 2.0)
- **Microsoft.Owin.Host.SystemWeb**
  © Microsoft Corporation. All rights reserved.
  (Apache License 2.0)
- **Owin**
  (Apache License 2.0)

Copyright Notice

End User License Agreement

The End-User license agreement is available online at https://developer.mescius.com/legal/eula.

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 Support if you have any questions about this license.

Install ActiveReports

You can download the installer from here and then follow these steps to install ActiveReports.

1. Double-click the installer.
2. In the **Welcome** screen that appears, click the Accept End User License Agreement checkbox and click **Next**.
3. In the screen that appears, select the release version to install from the drop-down list. If you want to change the default installation path, click **Change** in the Installation Directory area. Also, select the corresponding checkboxes for Visual Studio Templates installation and Customer Experience Program. After you select all necessary options, click **Next**.
4. Once the installation finishes, a screen notifying of the successful installation appears. Select **Close** to close the window and complete the installation process, or select **Activate** to continue with activating the license. See License ActiveReports for information on licensing.

> **Note**: The installer version is not identical to the ActiveReports version.

ActiveReports Utilities and Entries

When you run the installer, you get necessary utilities such as standalone designer and viewer applications, import tool, license manager, VSIX files, localization resources, etc.

- Shortcuts:
  - MESCIUS License Manager
  - ActiveReports 18 Designer
  - ActiveReports 18 Import
  - ActiveReports 18 Theme Editor
  - ActiveReports 18 User Guide(CHM)
  - ActiveReports 18 User Guide(PDF)
  - ActiveReports 18 User Guide(Web)
  - ActiveReports 18 Viewer
- MESCIUS License Manager (C:\ProgramData\Microsoft\Windows\Start Menu\Programs\MESCIUS)
- Program files (C:\Program Files\MESCIUS\ActiveReports 18 OR C:\Program Files (x86)\MESCIUS\ActiveReports 18)
  - Deployment
  - Icons
  - Localization
  - Tools
  - VisualStudio
  - ActiveReports.config
- Registry keys (Computer\HKEY_CURRENT_USER\SOFTWARE\MESCIUS\ActiveReports)
- Visual Studio integrations

About Visual Studio Integrations

The ActiveReports installation provides you with following Visual Studio Integrations that help you configure projects and design reports. Start a supported version of the Visual Studio IDE and select from the built-in project templates listed in the next section.

Project Templates

Create a report layout using built-in sample application templates:

- ActiveReports 18 Windows Forms Application (VB and C#)
- ActiveReports 18 ASP.NET Core MVC Application (VB and C#)
- ActiveReports 18 Blazor Server Application (C#)
- ActiveReports 18 Blazor WebAssembly Application (C#)

Item Templates

Add following item templates to your project:

- ActiveReports 18 Report
- ActiveReports 18 Code-Based Report

Toolbox with Report Controls

A new ActiveReports 18 tab is automatically added in the toolbox with the controls in sync with the references.

Integrated Designer

ActiveReports offers an integrated designer that lets you create report layouts in Visual Studio and edit them at design time, visually, and through code and scripts. Along with the designer, you can use:

- **Report Explorer** to view the report elements in tree view,
- **Group Editor** to manage grouping in Tablix data region,
- **Layers List** to view layers in a report,
- **Reports Library** to view list of reports and controls added in the reports.

Conversion Tool

Use the conversion or upgrade tool - **Convert to ActiveReports 18** - to easily upgrade to the latest version. This tool is available under the **Tools** menu option of Visual Studio.

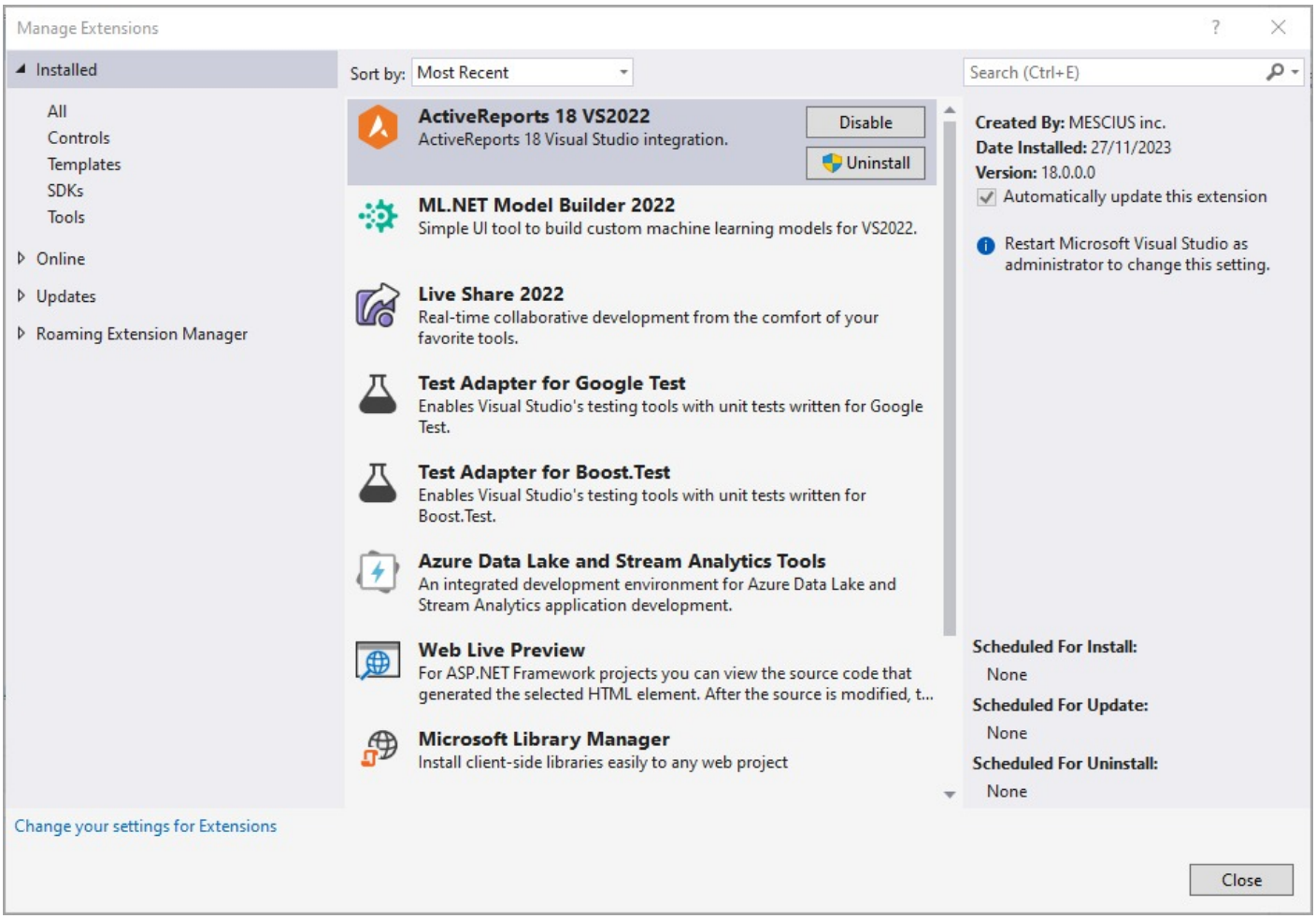
The following ActiveReports resources are available on the web:

- Samples and Web Samples
- NPM and NuGet Packages. See Available Packages.

To Remove Visual Studio Integration

In case you want to disable or uninstall ActiveReports 18 Visual Studio integration, do the following.

1. In Visual Studio 2019 and Visual Studio 2022, go to **Extensions** > **Manage Extensions**.
   OR
   In earlier supported versions of Visual Studio, go to the **Tools** menu and click **Extensions and Updates.**
2. Navigate to **ActiveReports 18 VS2022** and click Disable or Uninstall.



In case you want to install the integration again, follow these steps.

1. Go to the installation folder - C:\Program Files\MESCIUS\ActiveReports 18\VisualStudio (C:\Program Files (x86)\MESCIUS\ActiveReports 18\VisualStudio on a 64-bit Windows operating system).
2. Run the VSIX as per your Visual Studio Version.

Note:

- It is not possible to install ActiveReports 18 and ActiveReports 13 side-by-side.
- It is not possible to use different versions of ActiveReports for .NET within a single Visual Studio project.
- If you are creating an ActiveReports 18 application in Visual Studio 2017 or Visual Studio 2019 or Visual Studio 2022, and you also have ActiveReports 13 installed in the supported Visual Studio version, then you may get a few errors in the property grid such as 'Object does not match target type' or 'Object reference not set to an instance of Object'. This issue is related to Visual Studio; you just need to restart Visual Studio to resolve the issue. Before restart, save the project, if necessary.
- When using our Integrated Designer in Visual Studio 2022, you may observe invisible controls in our smart panels or tool windows (such as Report Explorer or Layers List). To solve this issue, navigate to Tools > Options > Environment > General and clear the 'Optimize rendering for screens with different pixel densities' checkbox.

Customer Experience Program

When you optionally join the Product Usage Program, your computer automatically transmits information to MESCIUS inc. about the usage of MESCIUS inc. products. This information is used to address issues within MESCIUS inc. products and improve quality/usability.

The Product Usage Program is strictly optional and you can opt-in or opt-out of the program at any time by executing the Web installer and making the appropriate selection within the installation program. The Product Usage Program does not transmit any personally identifiable information such as your name, address, or phone number. The Product Usage Program only collects information related to MESCIUS inc. controls and libraries. This information includes the usage of MESCIUS inc. controls at design time within Visual Studio. No information is collected from applications/demos you create.

Available Packages

The following packages are available for download.

NPM

Download the packages for viewer and designer from **NPM**, a JavaScript package manager .

Reference	Purpose
ActiveReports Js Viewer	Provides the report viewer based on JavaScript.
ActiveReports WebDesigner	Provides the report designer based on HTML5/JS technology stack.

NuGet

Download the following packages from **NuGet**, an open source package manager.

	Reference	Purpose
Reports		
	MESCIUS.ActiveReports	Page/RDLX and Section reports API.
	MESCIUS.ActiveReports.Chart MESCIUS.ActiveReports.Chart.Win	Chart data visualization components.
	MESCIUS.Data.DataEngine MESCIUS.Data.ExpressionInfo MESCIUS.Data.VBFunctionLib MESCIUS.ActiveReports.Core.DataProviders MESCIUS.ActiveReports.Core.Rdl MESCIUS.ActiveReports.Core.Rendering MESCIUS.ActiveReports.Core.Scripting MESCIUS.ActiveReports.Core.Drawing.Gc	Page/RDLX reports' internal implementation.

	MESCIUS.ActiveReports.Core.Drawing.Gdi	
	MESCIUS.ActiveReports.Core.Document MESCIUS.ActiveReports.Core.Document.Drawing.Gc MESCIUS.ActiveReports.Core.Document.Drawing.Gdi	ActiveReports RDF document implementation.
	MESCIUS.Documents.Imaging	See Document Solutions for Imaging.
Desktop Components		
	MESCIUS.ActiveReports.Design.Win	ActiveReports Windows Forms end-user Designer control.
	MESCIUS.ActiveReports.Viewer.Win MESCIUS.ActiveReports.Viewer.Wpf MESCIUS.ActiveReports.Viewer.Common	ActiveReports Viewer controls and printing implementations.
Web		
	MESCIUS.ActiveReports.Aspnet.Designer MESCIUS.ActiveReports.Aspnetcore.Designer	WebDesigner JS component and Blazor WebDesigner control server-side implementations for ASP.NET MVC 5 and ASP.NET Core.
	MESCIUS.ActiveReports.Aspnet.Viewer MESCIUS.ActiveReports.Aspnetcore.Viewer MESCIUS.ActiveReports.Web.Viewer	Js Viewer component and Blazor viewer control server-side implementations for ASP.NET MVC 5 and ASP.NET Core.
	MESCIUS.ActiveReports.Web MESCIUS.ActiveReports.Web.Viewer	ASP.NET WebForms control implementation.
	MESCIUS.ActiveReports.Blazor.Viewer	Blazor Viewer control implementation.
	MESCIUS.ActiveReports.Blazor.Designer	Blazor WebDesigner control implementation.
Exports		
	MESCIUS.ActiveReports.Export.Pdf MESCIUS.ActiveReports.Core.Export.Pdf.Page MESCIUS.ActiveReports.Core.Export.Pdf.Section MESCIUS.Documents.Pdf Portable.BouncyCastle	These packages include assemblies needed to export ActiveReports reports to PDF and Document Solutions for PDF.
	MESCIUS.ActiveReports.Export.Excel MESCIUS.ActiveReports.Core.Export.Excel.Page MESCIUS.ActiveReports.SpreadBuilder DocumentFormat.OpenXml	These packages include assemblies needed to export ActiveReports reports to Excel (Xls/Xlsx).

	MESCIUS.ActiveReports.Export.Word MESCIUS.ActiveReports.Core.Export.Word.Page DocumentFormat.OpenXml	These packages include assemblies needed to export ActiveReports files to Word (Doc/Docx).
	MESCIUS.ActiveReports.Export.Image MESCIUS.ActiveReports.Core.Export.Image.Page MESCIUS.ActiveReports.Core.Export.Tiff.Section	These packages include assemblies needed to export ActiveReports files to Image.
	MESCIUS.ActiveReports.Export.Xml MESCIUS.ActiveReports.Core.Export.Text.Page	These packages include assemblies needed to export ActiveReports files to text-related formats: XML, CSV, TXT, JSON.
	MESCIUS.ActiveReports.Export.Html MESCIUS.ActiveReports.Core.Export.Html.Page MESCIUS.ActiveReports.Core.Export.Svg.Page	These packages include assemblies needed to export ActiveReports files to HTML.
Design time		
	MESCIUS.ActiveReports.Serializer MESCIUS.ActiveReportsSerializer.2022	Code-based Section report CodeDOM serializers for Visual Studio 2017/2019 and Visual Studio 2022.
	MESCIUS.ActiveReports.Web.Design MESCIUS.ActiveReports.Web.Design.VS2022	ASP.NET WebViewer control design-time support for Visual Studio 2017/2019 and Visual Studio 2022.
Auxiliary		
	MESCIUS.ActiveReports.Export.Rdf	This package includes assemblies needed to export Page/RDLX reports to RDF format.
	MESCIUS.ActiveReports.Interop	The optional OleObject internal implementation.
	MESCIUS.Documents.Imaging.Windows MESCIUS.Documents.DX.Windows	Additional Document Solutions for Imaging packages, required to improve export to images.
	System.Drawing.Common	This package may require .NET desktop components and Section reports for the GDI compatibility mode. In all other situations, this package can be dropped.

You can install NuGet packages from a local folder or from web. For details, see Manage ActiveReports Dependencies.

Manage ActiveReports Dependencies

The ActiveReports dependencies in Visual Studio projects can be added, updated, or removed using **NuGet** package manager. How the dependencies are managed depends on the location of package - at a local source or public source. The ActiveReports packages are available in the local directory on installing ActiveReports using the installer. The packages are also available publicly on the **NuGet** website. For the list of packages, see Available Packages.

The following sections elaborate adding the ActiveReports assemblies in your project by either installing NuGet

packages from local source or directly from the NuGet website.

Installing Packages from Local Source

You must first run the installer to obtain the **NuGet** packages locally - **C:\Program Files (x86)\MESCIUS\ActiveReports 18\NuGet**. See Install ActiveReports for the steps on installation using MSI file. Then, create the NuGet package source to add the NuGet feed URL into your NuGet settings in Visual Studio, as follows:

1. Configure local NuGet package source

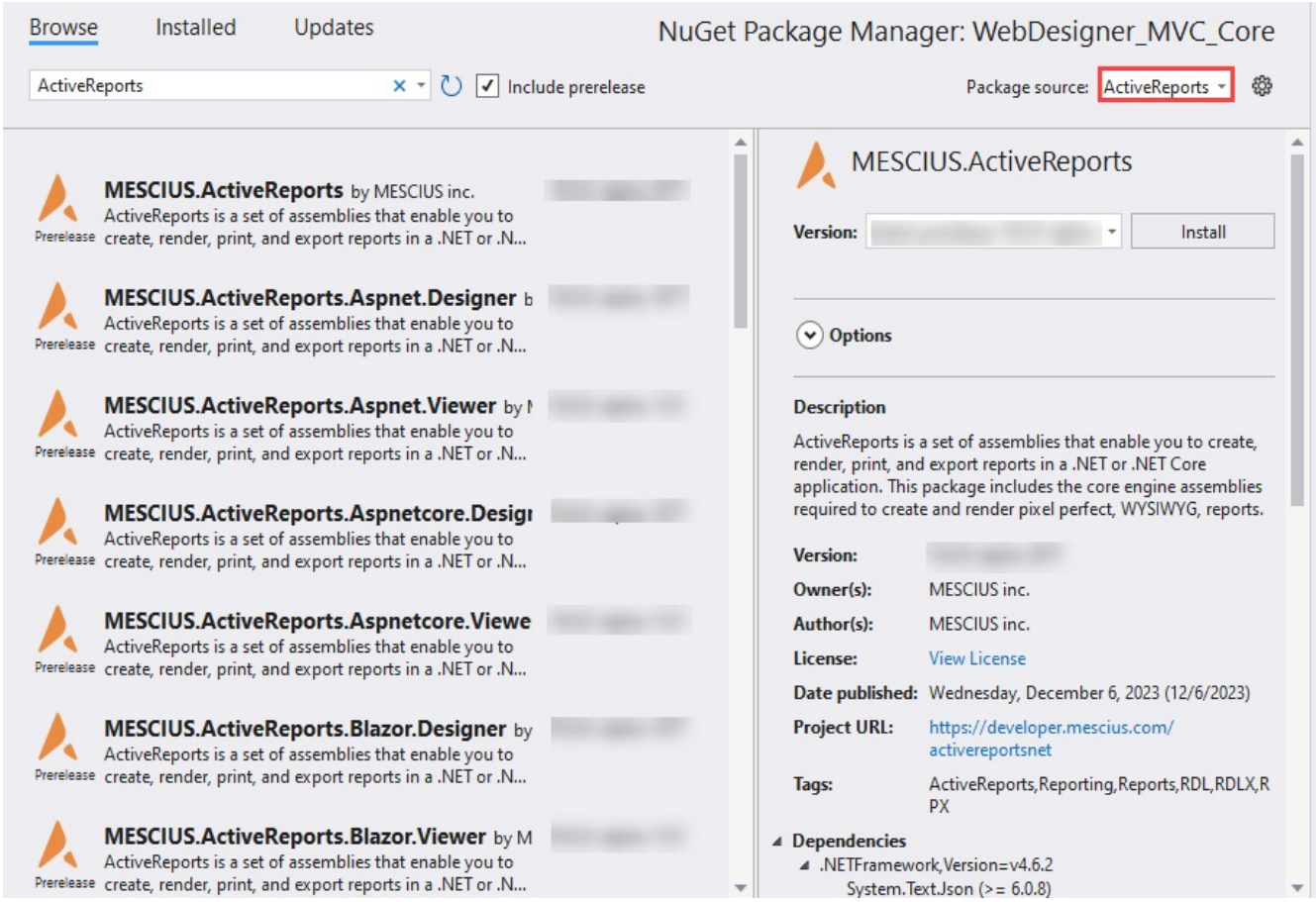
1. Open **NuGet.Config** file placed here: C:\Users\[UserName]\AppData\Roaming\NuGet.
2. Modify the content of NuGet.Config as follows. This adds a key that directs to the path where NuGet packages are available locally.

NuGet.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <packageSources>
    <add key="NuGet.org" value="https://api.NuGet.org/v3/index.json" protocolVersion="3" />
    <add key="ActiveReports" value="C:\Program Files (x86)\MESCIUS\ActiveReports 18\NuGet" />
  </packageSources>
</configuration>
```

2. Install Packages

1. Open Visual Studio.
2. Create any application (any target that supports .NET Standard 2.0).
3. Right-click the project in Solution Explorer and choose **Manage NuGet Packages**.
4. In the **Package source** on top right, select **ActiveReports** (key added in NuGet.config).
5. Click **Browse** tab on top left and search for any package, say 'MESCIUS.ActiveReports'.
6. On the left panel, select **MESCIUS.ActiveReports**.



7. On the right panel, click **Install**.
8. In the **License Acceptance** dialog, select **I Accept** to proceed the installation.

Installing Packages from NuGet

ActiveReports 18 references are available through **NuGet** and can be obtained directly from website - https://www.NuGet.org/packages?q=MESCIUS.ActiveReports. When you add reference to **MESCIUS.ActiveReports** package, a set of core engine assemblies are added to the application. Use following steps to find and install the NuGet packages in your application:

1. Open Visual Studio.
2. Create any application (any target that supports .NET Standard 2.0).
3. Right-click the project in Solution Explorer and choose **Manage NuGet Packages**.
4. In the **Package source** on top right, select **NuGet.org**.
5. Click **Browse** tab on top left and search for 'MESCIUS.ActiveReports'.
6. On the left panel, select MESCIUS.ActiveReports.
7. On the right panel, click **Install**.
8. In the **License Acceptance** dialog, select **I Accept** to proceed the installation.

For more information on NuGet configurations, please see Common NuGet configurations and NuGet Configuration Settings articles by Microsoft.

Note: The assemblies are available in the packages at the following location:

- if installer is used: C:\Program Files (x86)\MESCIUS\ActiveReports 18\NuGet\{*Package name*}\lib\net462\{*Assembly*}

- if a package is installed in an application: [*App name*]\packages\{Package name}\lib\net462\{*Assembly*}

Redistributable Files

ActiveReports is developed and published by MESCIUS inc. You may use it to develop applications in conjunction with Microsoft Visual Studio or any other programming environment that enables the user to use and integrate the control(s).

All files from the NuGet and NPM packages can be redistributed according to the License Agreement.

The following Redistributable Files require the **Professional Edition** license for redistribution:

- MESCIUS.ActiveReports.Design.Win.dll
- MESCIUS.ActiveReports.Web.dll

ActiveReports Features

ActiveReports provides a number of features, enriching you with the possibilities to develop extensive and powerful applications. Most of the features require a Standard Edition license.

You should carefully choose your ActiveReports Edition - Standard or Professional. Consider all benefits of provided features from our Standard and Professional Editions before you make a decision on which one to use.

Report Types

Choosing a Report Type

ActiveReports supports the following report types:

- **Page/RDLX reports**: xml-based formats that are similar to the MS SSRS format from Microsoft.
  - **RDLX report**: similar to the MS SSRS format with some extensions and limitations. In general, you can use Microsoft SQL Server Reporting Services (MS SSRS) reports, but there are complicated situations that require importing. In an RDLX report, controls grow vertically to accommodate data. Controls can grow and shrink, you can set up interactive sorting, you can set up drill-down reports in which detail data is initially hidden, and can be toggled by other items, and you can add drill-through links to other reports and to bookmark links within reports.
    There are a number of common concepts that apply to Page and RDLX reports. You should remember that some features require developer's experience - work with code samples, etc, so you should visit the corresponding pages.
  - **Page report**: RDLX-like pages with predefined fixed layout pages but without the same growing possibilities. In a **Page Layout**, you design reports at the page level without any banded sections. This lets you place controls anywhere on the report.
    In a Page report, controls do not change in size based on the data, but you can use an Overflow Placeholder to handle any extra data.
- **Section reports**: similar to the RDLX report type with the BandedList data region in the paginated mode.
  - **Xml-based Section report** (**RPX)**: XML-based format that uses event handlers as restricted scripts.
  - **Code-based Section Report (CodeDOM)**: the Designer is available with Visual Studio only but you can perform many report operations using event handlers.

RDLX is the default report type. If you need special fixed pages, you should choose the Page report type. We suggest using the Section report type for migration use-cases only. For XML-based RPX reports we have an import tool, which

allows migrating most of the layout.

For now, each report type has some unique feature, so it is suggested to choose the report type depending on your requirements. This table will help you choose the report type.

Report Type	End-User Designer	WebDesigner	DocX Export	Linux	Custom controls	Galley layout	Primary Target: developers or end-users	Document serialization	Different page sizes in a document	Viewer interactivity
RDLX	✓	✓	✓	✓	✓	✓	end-users	✗	✗	✓
Page	✓	✓	✓	✓	✓	✗[1]	end-users	✗	✓	✓
RDLX Dashboard	✗	✓	✗	✓	✓	✓	end-users	✗	✗	✓
XML-based Section	✓	✗	✗	✓	✓	✗[1]	developers	✓[2]	✓[3]	✗
Code-based Section	✗	✗	✗	✓	✓	✗[1]	developers only	✓[2]	✓[3]	✗

[1] Paginated only

[2] Internal RDF format

[3] Supported in RDF format, requires manual document merging using scripts/events

All report types support scripts (section reports require scripts in most situations) and the possibility to use functions from external assemblies (in expressions for Page/RDLX reports and in scripts/events for Section reports).

> **Note**: Section report requires using code/scripts for most complicated scenarios. See Scripting topic.

Report File Format Types

You can create reports in a number of file formats with a varied set of features. This section describes the use of each of these file formats.

Report Template Formats

- **RDLX:** This is an XML-based proprietary file format that provides custom extensions to the Report Definition Language (RDLX) files used by SQL Server Reporting Services. These are stand-alone files that you can process without compiling them into your application. You can customize the report through the Script Tab by embedding script in the report.
  See this msdn page for more on RDLX report.
- **VB** or **CS**: These are code-based reports, and are saved as C# or Visual Basic files that are compiled into your applications. They have corresponding code views similar to Windows forms and provide a design and coding experience in line with Visual Studio. This format is ideal for developers who are comfortable with coding in .NET programming languages and would like to use the extensive event-based API provided by ActiveReports in the code-behind rather than design view. You may also use the scripts in the Script Tab instead of the code behind.

- **RPX**: This is an XML-based proprietary file format that the ActiveReports engine can process without compiling it into an application. Instead of Visual Basic or C# code behind, you can customize the report with script embedded in the report XML using the Script Tab. You can also use an RPX file with script as a stand-alone file in a Web project.

Additional File Formats

ActiveReports also provides some additional file formats for reports. Each of these formats is used for a specific purpose as described below.

- **RDLX-master:** This is a master report file that you can reference from other RDLX report files for a standard layout, for example, you can add company logo and address sections. This file is loaded each time the report is executed, so you can change the logo on all of your reports by just changing it on the master report.
- **RDLX-theme:** This is a theme file that consists of a collection of styles that you can apply to a report. See Themes for further details.
- **RDSX:** This is a proprietary format that is created when you share a data source, making it available to multiple reports.
- **RDF:** This is the Report Document Format, in which the data is static. The RDF format is a vector-based image, this means that you can join some documents or render additional graphics. You can open it in any viewer or export it to any supported format (limited to Section reports). The RDF format doesn't require access to data sources and doesn't support parameters.
  You can save a report in this format to display the data that is retrieved. Once a report has been saved to an RDF file, it can be loaded into the viewer control. See Work with RDF Document for further details.

See the following list of file formats available in each layout.

Format	Page Layout/RDLX Layout	Section Layout
RDLX	✓	✗
VB or CS	✗	✓
RPX	✗	✓
RDLX-Master	✓	✗
RDLX-Theme	✓	✗
RDSX	✓	✗
RDF	✗	✓

Features comparison between report types

In ActiveReports, the features available in a report depend on the type of report you select. See the following comparison list of features with each report type:

Feature	Section report	Page report	RDLX report
Viewers & Editors			
Visual Studio Integrated Designer	✓	✓	✓
Expressions Editor	✗	✓	✓
Designer Script Editor	✓	✓	✓

Windows Form Viewer	✓	✓	✓
WebViewer (Pro Edition). Includes viewer types HTML, RawHTML, and PDF.	✓	✓	✓
HTTP Handlers (Pro Edition)	✓	✓	✓
Interactivity			
Hyperlinks	✓	✓	✓
Parameters	✓	✓	✓
Drill through	✗	✓	✓
Drill down	✓	✓	✓
Data Connections			
Standard Data Sources supported (e.g. SQL, OleDB, XML)	✓	✓	✓
Unbound Data Source	✓	✓	✓
Shared Data Source	✗	✓	✓
Export			
Export Filters	✓	✓	✓
Rendering Extensions	✗	✓	✓
PDF advanced export features: digital signatures, time stamp, bold font emulation (Pro Edition)	✓	✓	✓
Miscellaneous			
Master Reports	✓	✗	✓
Themes	✗	✓	✓
Collation	✗	✓	✓
Styles (through Report Settings dialog)	✓	✗	✗
Printing	✓	✓	✓
Filtering	✗	✓	✓
Grouping	✓	✓	✓
Sorting	✗	✓	✓
Standalone Applications			
ActiveReports Viewer	✓	✓	✓
ActiveReports Theme Editor	✗	✓	✓
ActiveReports Designer	✓	✓	✓

For comparison in available report controls or data regions, see Report Controls page.

ActiveReports Components

ActiveReports is a tool for developers that provides possibilities to work with reports in the most productive manner.

- **Distributable items**
  - **Controls**
    - WinForms Designer control (Professional Edition feature)
    - WinForms Viewer control
    - WPF Viewer control
    - WebDesigner JS component and 2 backends: ASP.NET MVC 5 and ASP.NET Core (Professional Edition feature)
    - JS Viewer component, Blazor component and 2 backends: ASP.NET MVC 5 and ASP.NET Core (Professional Edition feature)
    - ASP.NET WebViewer control
  - **Exports**
    - Export to Image (several formats)
    - Export to PDF (some PDF-related features are part of Professional Edition)
    - Export to Excel (XLS/XLSX). Export to Excel Data (XSLX) for Page/RDLX reports is the Professional Edition feature
    - Export to Word (DOC/DOCX for Page/RDLX and RTF for Section). The DOCX option is part of the Professional Edition.
    - Export to HTML/MHT
    - Export to XML (only for Page/RDLX reports)
    - Export to JSON (only for Page/RDLX reports)
    - Export to Text (CSV/TXT for Page/RDLX reports and TXT for Section reports)
  - **Imports**
    - Import from the MS SSRS RDLX format (see MS SSRS RDLX)
- **For developers only**
  - ActiveReports Installer
  - License Manager
  - Visual Studio Integrated Designer
  - Standalone Designer/Viewer applications
  - Theme Editor application
  - Import Wizard and Console tools from Crystal/Access/Excel/RDLX (for details, see Report Import and Migration)
  - Import options from the RPX to RDLX formats.
  - Samples on GitHub

Standalone Designer and Viewer Applications

Standalone ActiveReports Designer

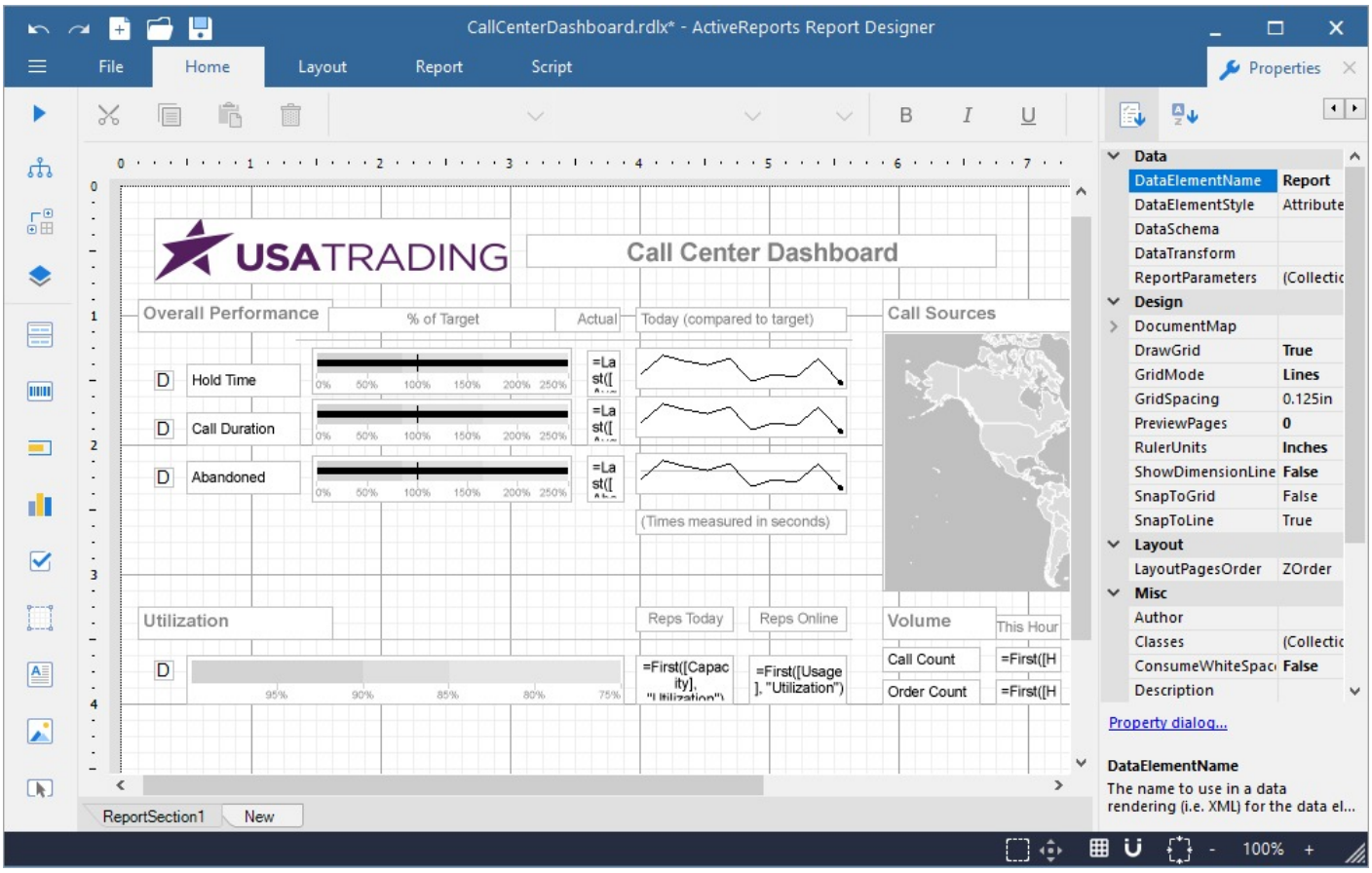
To access the standalone report designer application:

From the Start Menu, select **ActiveReports 18 Designer**.

OR

Select the ActiveReports.Designer.exe application located in the installation folder: *C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools*.

Design area



For details, see Standalone Designer page.

Standalone ActiveReports Viewer

To access the Standalone Viewer application:

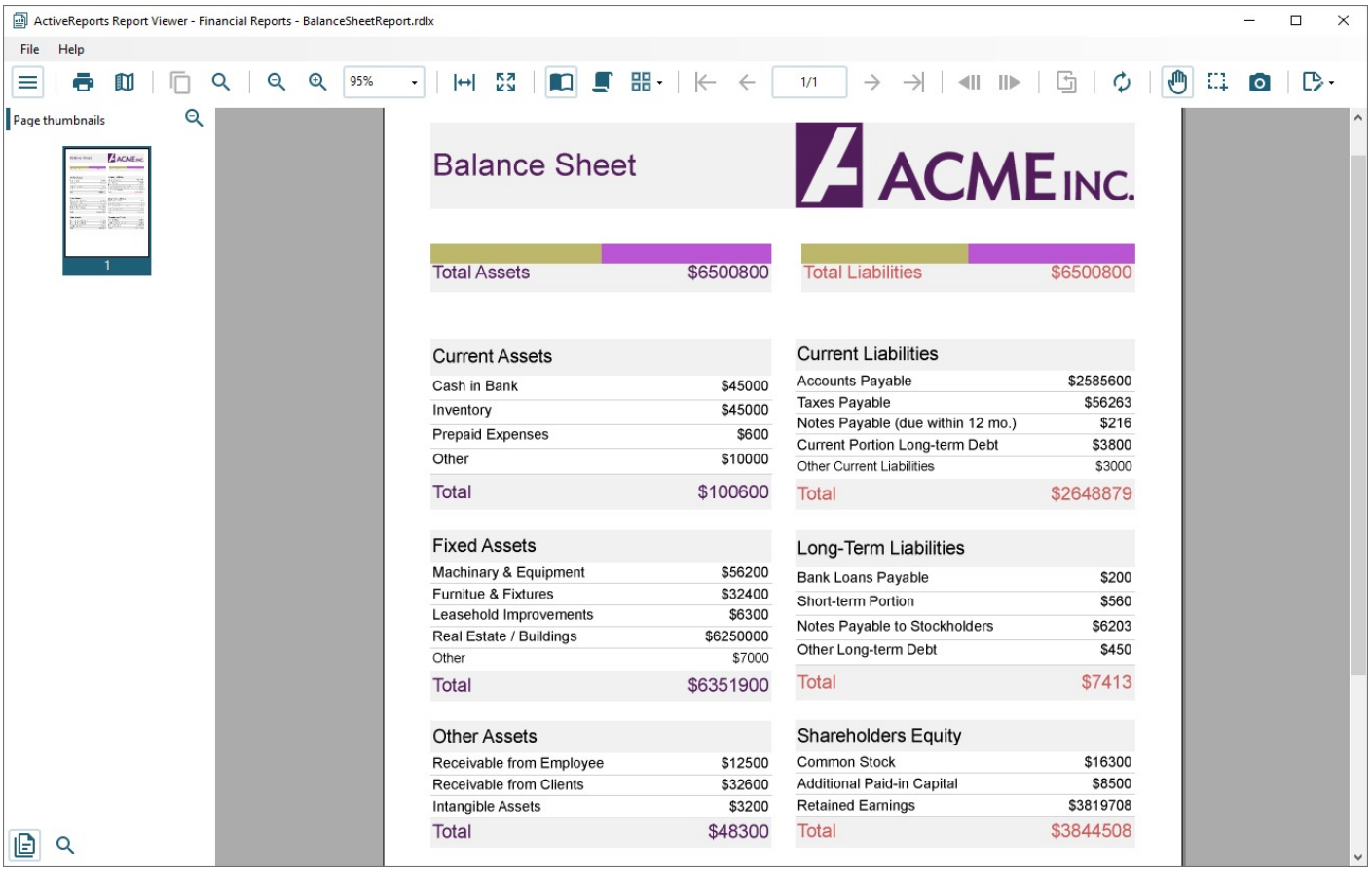
From the Start Menu, select **ActiveReports 18 Viewer**.

OR

Select the application located under: *C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools*.

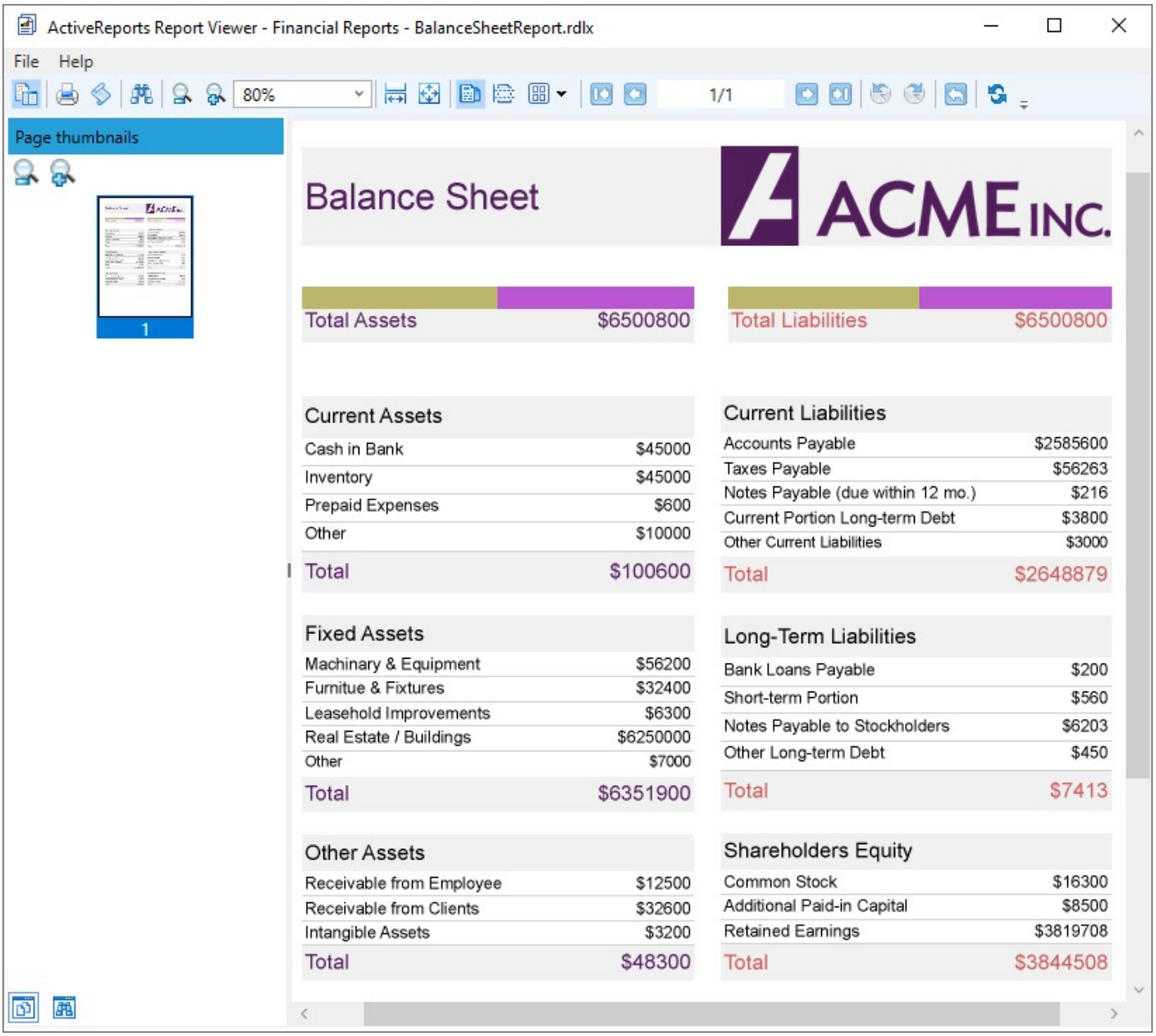
- ActiveReports.Viewer.exe
- ActiveReports.WpfViewer.exe

These executable files function as standalone applications to help view a report quickly. Use the standalone designer application to create a report layout, save it in .rpx or .rdlx format and then load it in the standalone viewer application to view the report.



See Windows Forms Viewer for more information on how to implement these features in the Viewer control.

Standalone WPF Viewer



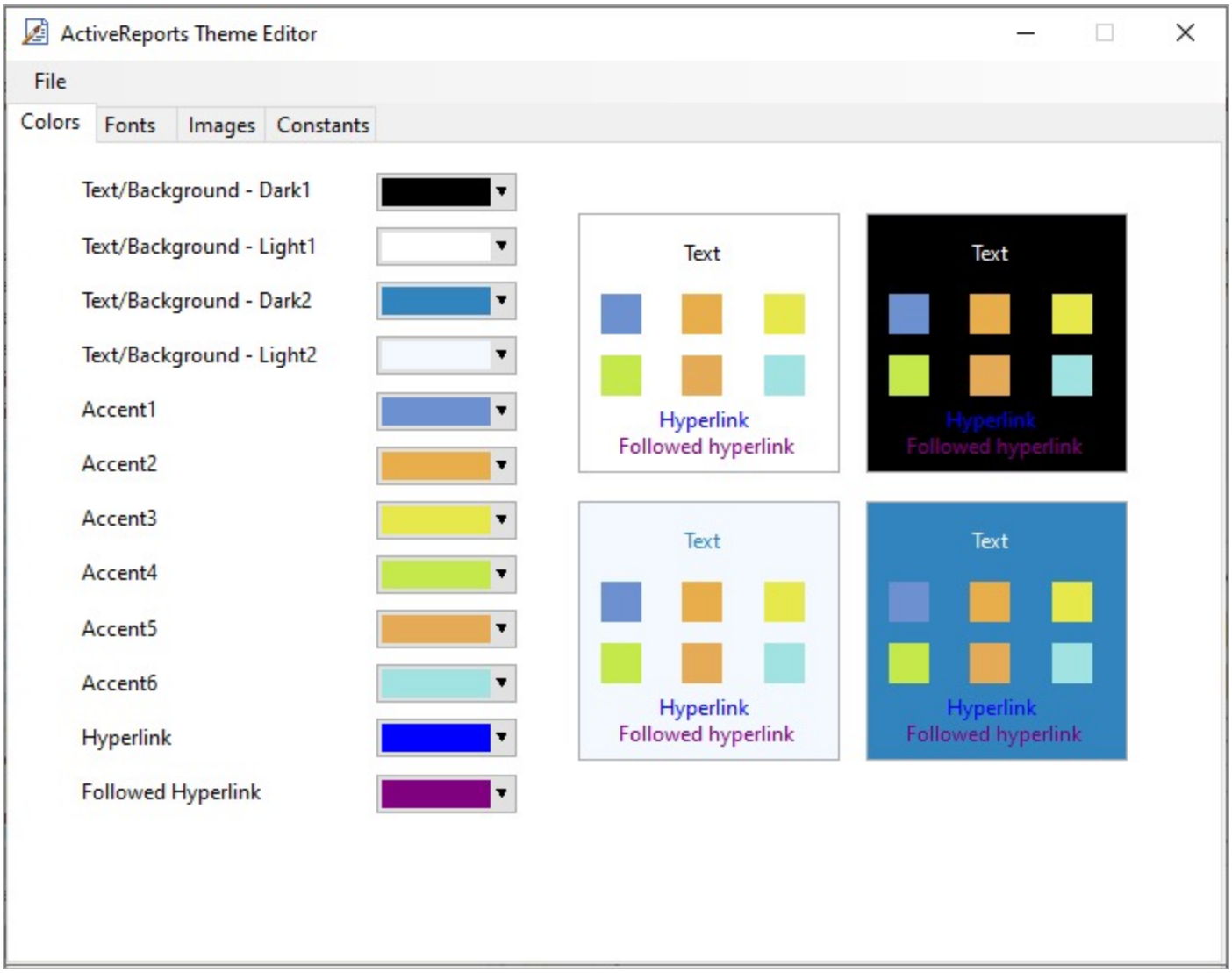
See Windows Presentation Foundation Viewer for more information on how to implement these features in the Viewer control.

Theme Editor

The Theme Editor is an application that you can use to create a new theme by setting colors, fonts, images, and use constant expressions in a theme and then saving a new theme as an .rdlx-theme file on your local machine.

To access the Theme Editor

You can run the application by selecting the **ActiveReports 18 Theme Editor** from the Start menu, or by running the **ActiveReports.ThemeEditor.exe** from the *C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools* location.



Import Wizard

You can import a Crystal Reports report, a Microsoft Access report, an Excel file, or an ActiveReports section report in ActiveReports by running the ActiveReports Import Wizard.

You can run the ActiveReports Import Wizard from the start menu or by executing ActiveReports.Imports.Win.exe from **C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools** location.

For details on each import options, see Import Reports and Migration.

Designers

ActiveReports Designers support these report types and items:

Section Reports	Page Reports/RDLX Reports
• ReportInfo	• Table data region

- Label
- Line
- PageBreak
- OleObject
- SubReport
- Shape
- Picture
- RichTextBox with HTML tag support
- Chart with separate data source
- Textbox
- Barcode with standard styles plus RSS and UPC styles
- Checkbox
- CrossSectionBox extends from a header section to the related footer section
- CrossSectionLine extends from a header section to the related footer section

- Tablix data region
- Map data region
- Chart data region
- List data region
- BandedList data region
- Sparkline data region
- FormattedText with mail merge capabilities and XHTML + CSS support
- Bullet Graph
- BarCode
- CheckBox
- TextBox
- TableOfContents
- Line
- Container
- Shape
- Image
- Subreport
- Overflow Placeholder
- ContentControl (RDLX Master only)
- CustomReportItem

Viewers

ActiveReports Viewers support these interactive features.

	Section Reports	Page/RDLX Reports
Pan/Zoom/Copy	+	+
Bookmarks	+	+
Hyperlinks	+	+
Drill-through links	-	+
Search	+	+
Parameters	+	+
Dependant parameters	-	+
Annotations	+ (Windows Forms Viewer only)	-
Sorting	-	+
Toggling	-	+
TOC	+	+
Thumbnails (WinForms/WPF)	+	+

Touch mode	+	+
Page navigation	+	+
Printing	+	+

Note: The JS Viewer component, the Blazor Viewer control, and the WebDesigner JS component require back end (available backends for ASP.NET MVC 5 and ASP.NET Core).

Exports

Export formats	Section report	Page/RDLX report
Html: Export reports to HTML, DHTML, or MHT formats, all of which open in a Web browser.	✓	✓
Pdf: Export reports to PDF, a portable document format that opens in the Adobe Reader.	✓	✓
Rtf: Export reports to RTF, RichText format that opens in Microsoft Word, and is native to WordPad.	✓	✗
Word: Export reports to DOC, a format that opens in Microsoft Word.	✗	✓
Text: Export reports to TXT, plain text, a format that opens in Notepad or any text editor. Export reports to CSV, comma separated values, a format that you can open in Microsoft Excel.	✓	✓
Image: Export reports to BMP, GIF, JPEG, or PNG image format.	✗	✓
Tiff: Export reports to TIFF image format for optical archiving and faxing.	✓	✓
Excel: Export reports to formats that open in Microsoft Excel, XLS or XLSX.	✓	✓
Xml: Export reports to XML, a format that opens in a Web browser or delivers data to other applications.	✗	✓
CSV: Export reports to a CSV file, a form of structured data in plain text. The text in a CSV file is saved as series of values separated by comma.	✗	✓
JSON: Export reports to a JSON file, a text-based data format in which the data is stored in the hierarchical form.	✗	✓

Note: If you don't find an export you need among the provided exports, you can write it manually. See our CustomPdfExport sample for details.

You can use the section-only exports for Page/RDLX reports too, using the MESCIUS.ActiveReports.Export.Rdf package. Although, Page/RDLX reports have a more powerful set of exports, using this package is helpful for some rare situations like:

- An RTF output is needed (available only for Section report).
- Manual rendering of graphic elements is required.
- Need to save an RDF document and export it later without access to data.

Professional Edition

You need to have the Professional Edition if you are going to use these features.

ActiveReports Components

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

WebDesigner

Create or modify reports by embedding WebDesigner into your web applications.

JS Viewer

View reports in all modern browsers using JSViewer.

Blazor Viewer

It uses the same client-server architecture as JS viewer (works with the same back end as JS Viewer).

ActiveReports Controls

These report items are included into the Professional Edition features.

Table of Contents

The TableOfContents control is used to display the document map, an organized hierarchy of the report heading levels and labels along with their page numbers, in the body of a report. It allows you to quickly understand and navigate the data inside a report in all viewers that are supported in ActiveReports.

Map

The Map data region shows your business data against a geographical background. You can create different types of map, depending on the type of information you want to communicate in your report.

InputField

The InputField report control provides support for editable fields in an exported PDF report where the InputField’s value can be modified. !!You can choose one of the two report contorl types – Text and Checkbox. Each selected type has its own set of properties.

ActiveReports Exports

Word export

Export your reports in .docx format, a format that opens in Microsoft Word application.

Excel Data export

Excel Data exports only data from Tablix, Table, and Matrix data regions, preserving the data region structure and ignoring layout-related features (page break, cumulative total, etc). Other controls and data regions of the original report are ignored at this export.

PDF export

- PDF/A and PDF/UA support: "Section 508" accesability requirements.
- IVS character support - if you are going to use IVS or EUDC characters in your reports.
- Devanagari character support.
- Bold font emulation - it is required to obtain WYSIWYG output in case of font fallback.
- PdfSignature and TimeStamp: The PdfSignature class allows you to provide PDF document digital signatures and certification. The TimeStamp class allows you to add a TSA (Time Stamping Authority) stamp to your digital signatures.
- Section report-related PDF export in legacy GDI mode requires a Professional license for Font Fallback and Font Linking.

Report Import and Migration

You can import a Crystal Reports report, a Microsoft Access report, an Excel file, or an ActiveReports section report in ActiveReports by running the ActiveReports Import Wizard.

Importing MS Access

Learn about importing MS Access reports.

Importing Crystal

Learn about importing Crystal reports.

Importing Excel

Learn about importing Excel files.

Importing RPX

Learn about importing ActiveReports section reports.

Importing MS SSRS RDLX

Learn about importing SSRS reports.

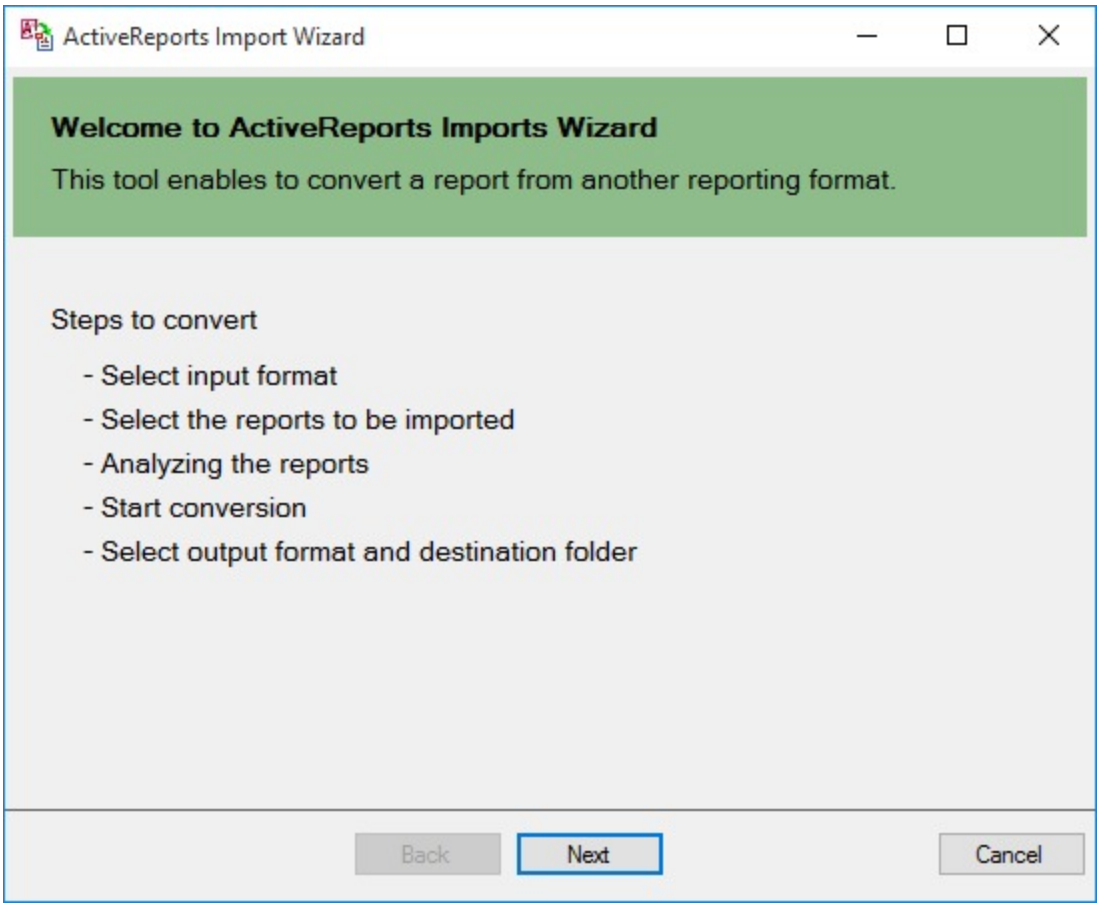
Other Migration Options

Please contact our support team to learn if we can provide more import possibilities for your requirements.

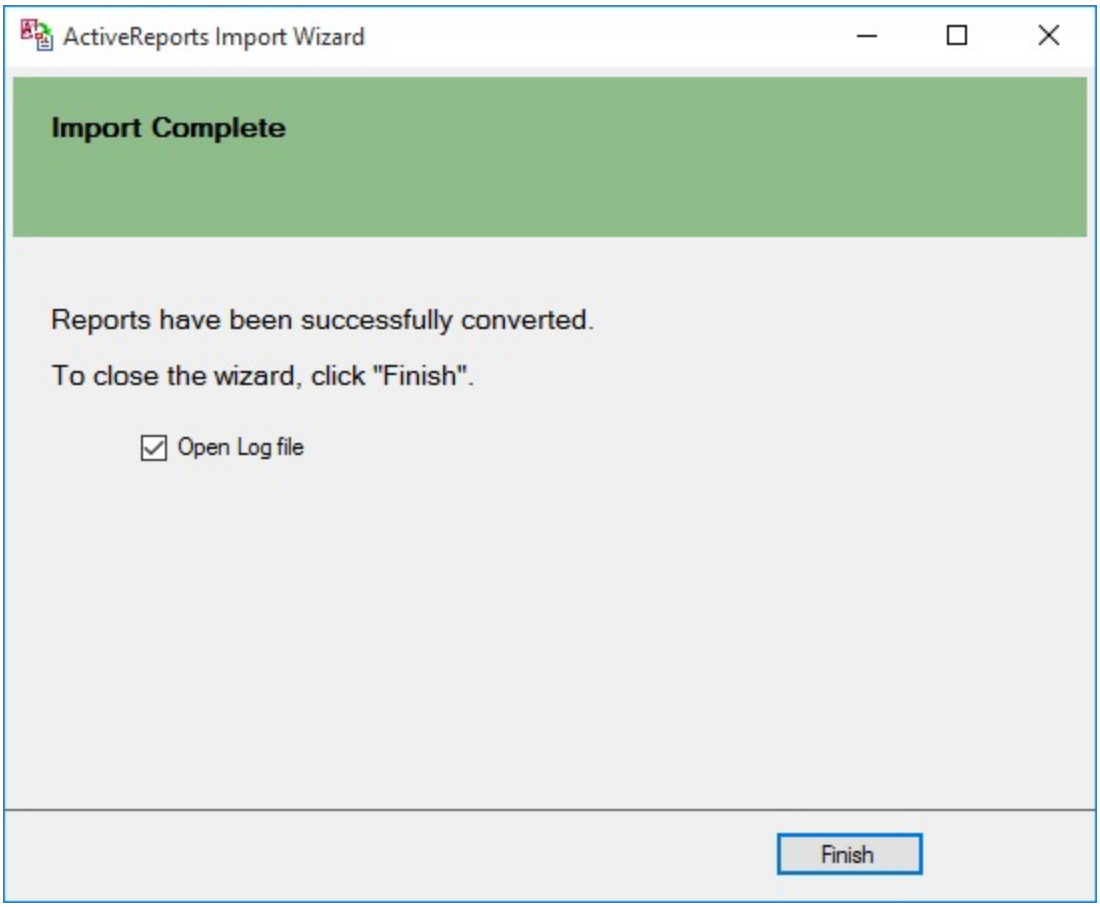
MS Access

Importing MS Access Reports in the ActiveReports Import Wizard

1. Run the ActiveReports Import Wizard. The wizard can be run from the start menu or by executing ActiveReports.Imports.Win.exe from **C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools** location.
2. In the ActiveReports Import Wizard that appears, click **Next**.



3. Choose **Microsoft Access (mdb or accdb)** as the input format and click **Next**.
4. Click the ellipsis button to browse to the location that contains the files that you want to import. A list of files that you can import appears.
5. Select the reports to import, click **Open,** and then click **Next** to analyze them.
6. Use the ellipsis button to select a destination folder to store the converted reports. Also select an output format (Section Report, Page report or RDLX report or Both) for each report in the Output Format column.
7. Click **Next** to start the conversion.
8. Once the conversion process is complete, click **Finish** to close the wizard and go the destination folder to view the converted reports. You may optionally leave the check on for the **Open Log file** checkbox to see the results log.



The import wizard converts reports to the closest possible ActiveReports format, but due to differences between products and versions, the extent to which your reports are converted depends on your specific report layout. You may have to partially redesign the report and add script or code to get the same output as Microsoft Access Reports.

When converting to Page Reports or RDLX Reports, whether a report is imported as a Page report or RDLX report, depends on the following factors:

- If a report has a single detail section it is imported as a Page report.
- If a report has a CrossTab control and its layout is composed of multiple sections it is imported as an RDLX report.

Note: Sections in a report appear as BandedList.

Please refer to the additional information below, to understand the conversion process in detail.

Importing Microsoft Access Reports

To import Microsoft® Access® reports in ActiveReports, you must have Access 97, 2000, 2002, 2003, 2007, 2010, or 2013 installed on your system.

Microsoft Access report controls are converted in ActiveReports as follows:

Microsoft Access Report	Section Report	Page report/RDLX report	Note
Rectangle	Shape	Container	Controls placed inside the Rectangle control are also imported along with the parent control.
CheckBox	Label	Textbox	...
Image	...	Image	Image control is not converted while converting to a Section Report.
Label	Label	Textbox	...
Textbox	TextBox	Textbox	...
Line	Line	Line	...
Page Break	PageBreak	Container	In Page Reports and RDLX Reports, the **PageBreakAtEnd** property is automatically set to True on importing a Page Break control.
Subform/Subreport	SubReport	Subreport	...

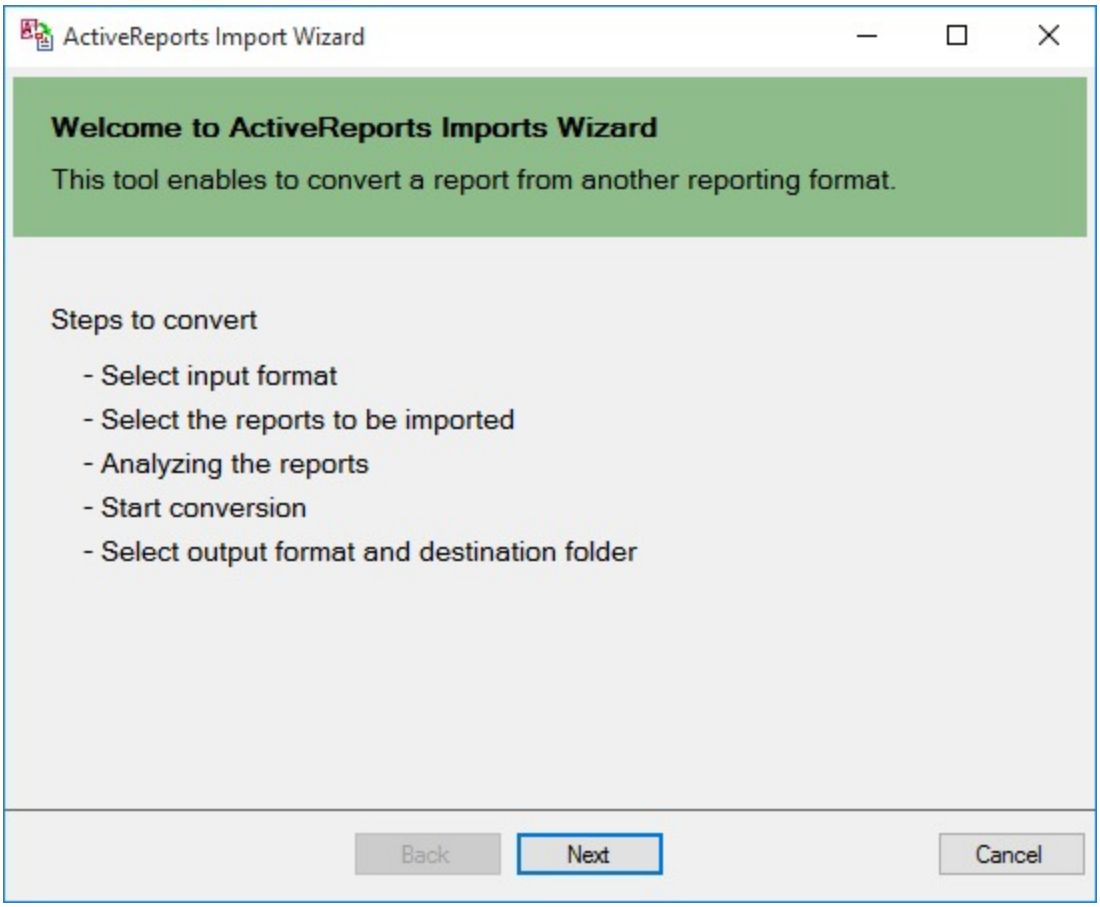
Limitations in MS Access conversion

- Any controls, functions, and text formats which are not supported by ActiveReports are not imported.
- The shadow property of a control is not imported while converting a report.
- In Microsoft Access reports, VBA code appears in as commented statements in script. You have to modify the code after importing.

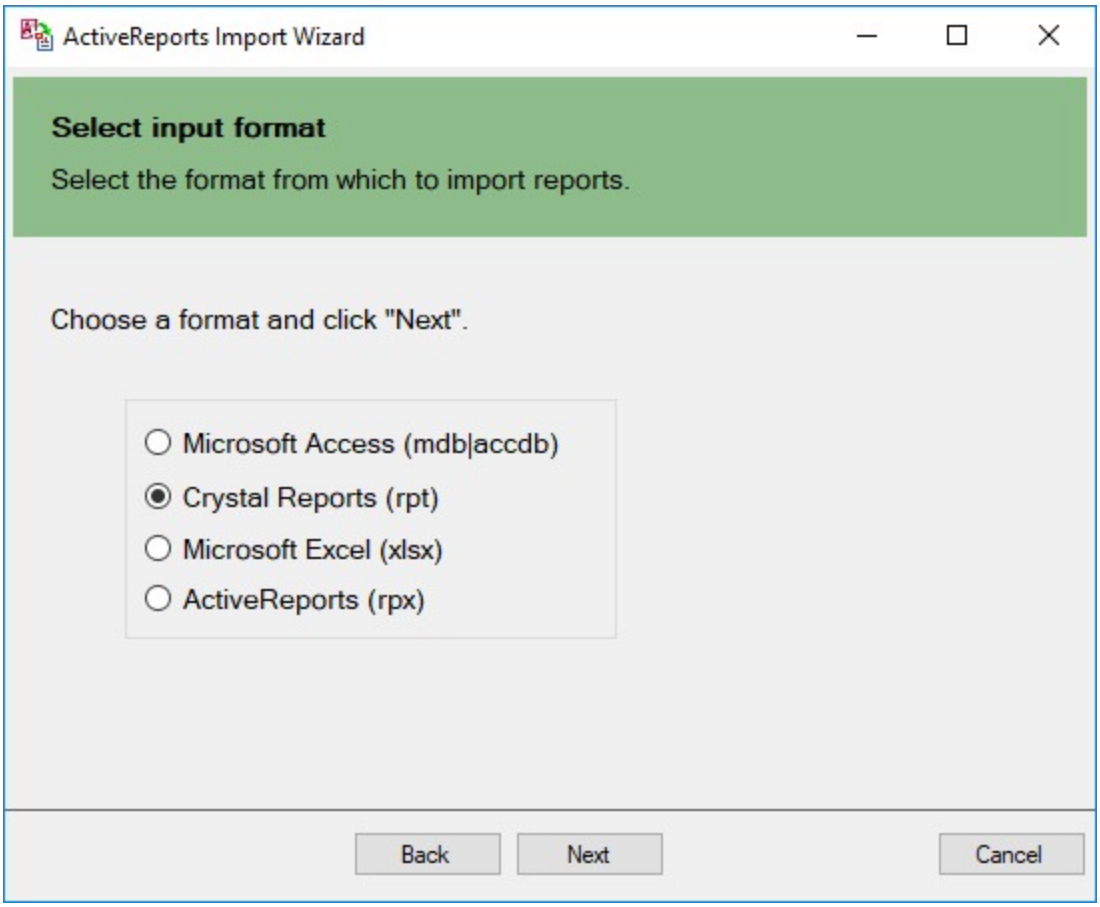
Crystal

Importing Crystal reports in the ActiveReports Import Wizard

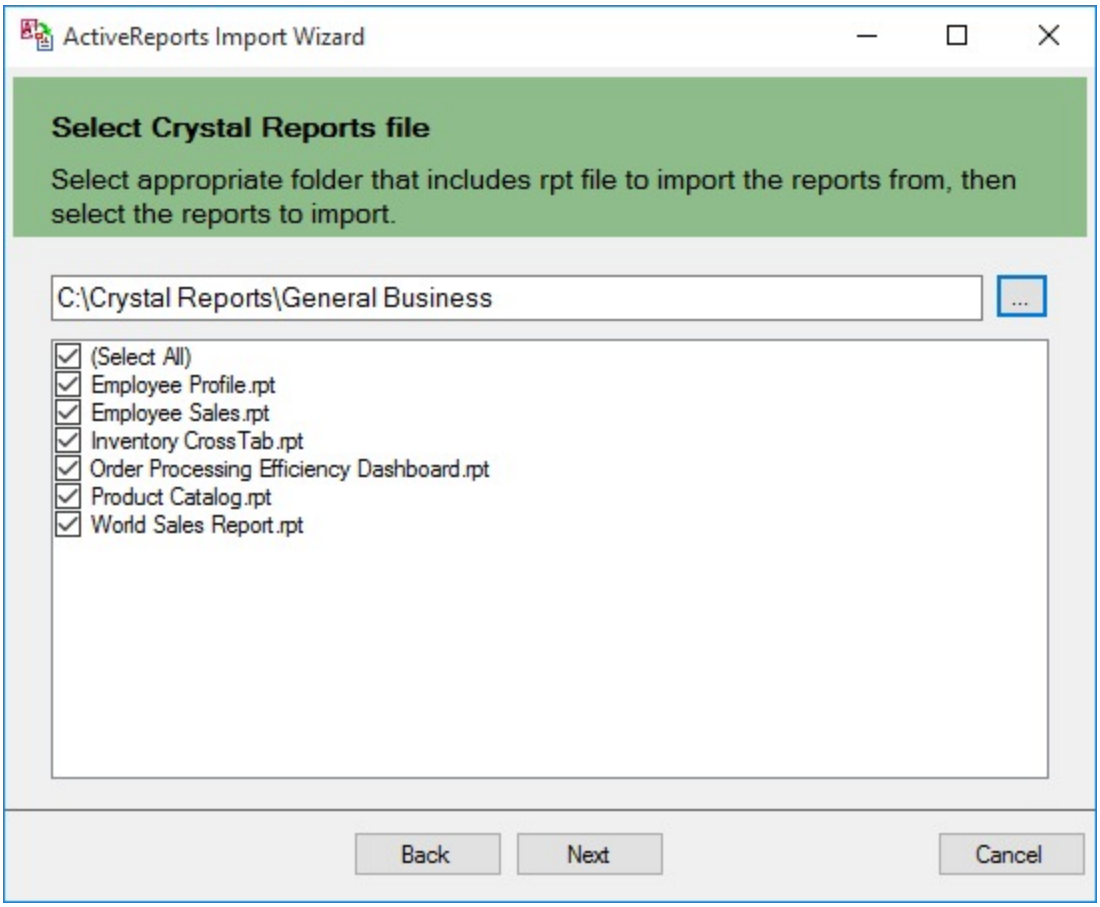
1. Run the ActiveReports Import Wizard. The wizard can be run from the start menu or by executing ActiveReports.Imports.Win.exe from **C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools** location.
2. In the ActiveReports Import Wizard that appears, click **Next**.



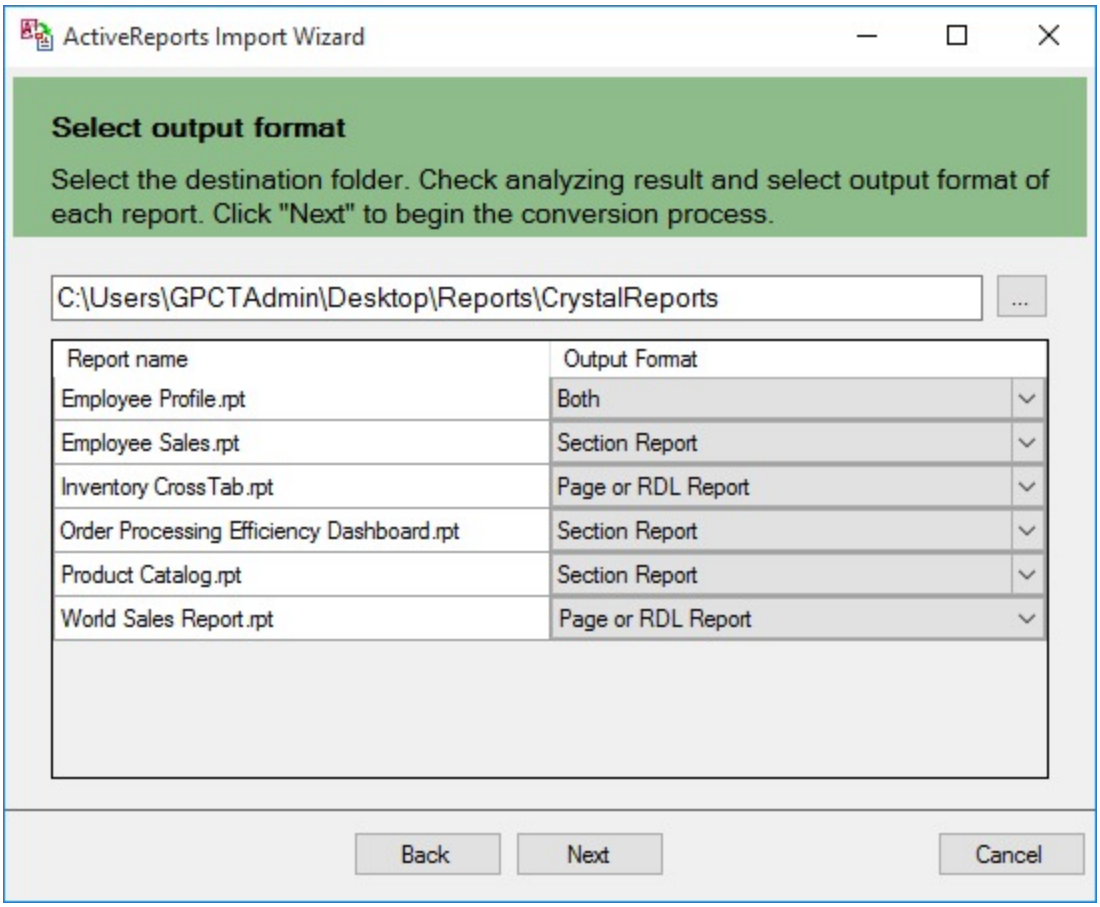
3. Choose **Crystal Reports (rpt)** as the input format and click **Next**.



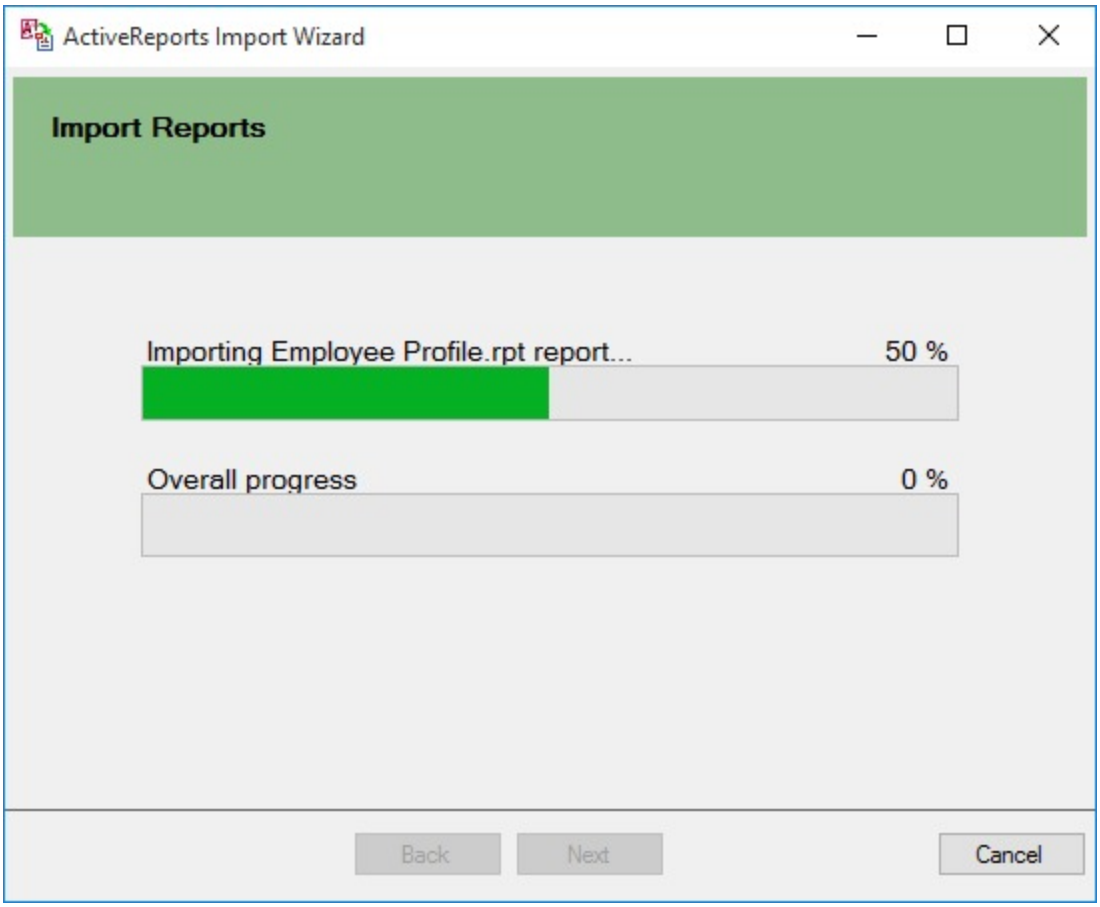
4. Click the ellipsis button to browse to the location that contains the files that you want to import. A list of files that you can import appears.
5. Select the reports to import, click **Open,** and then click **Next** to analyze them.



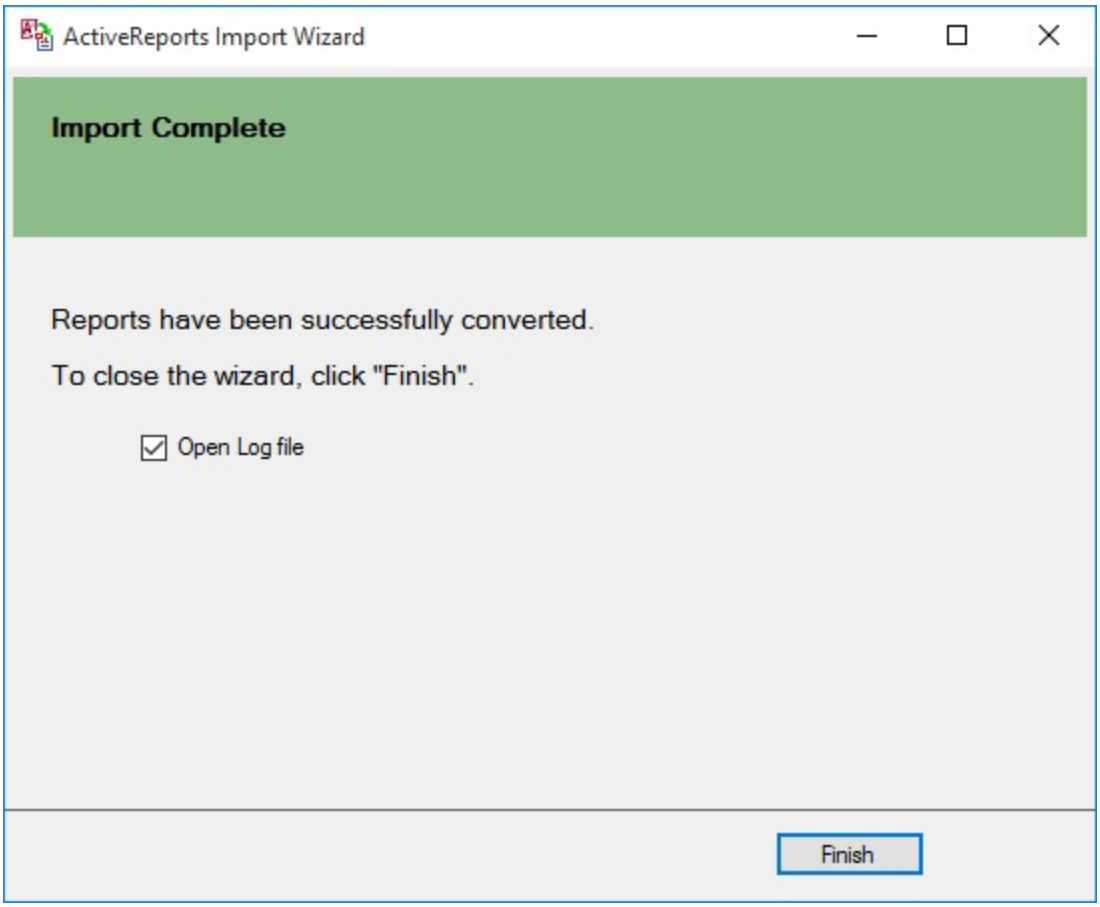
6. Use the ellipsis button to select a destination folder to store the converted reports. Also select an output format (Section Report, Page report or RDLX report or Both) for each report in the Output Format column.



7. Click **Next** to start the conversion.



8. Once the conversion process is complete, click **Finish** to close the wizard and go the destination folder to view the converted reports. You may optionally leave the check on for the **Open Log file** checkbox to see the results log.



The import wizard converts reports to the closest possible ActiveReports format, but due to differences between products and versions, the extent to which your reports are converted depends on your specific report layout. You may have to partially redesign the report and add script or code to get the same output as Crystal Reports.

When converting to Page Reports or RDLX Reports, whether a report is imported as a Page report or RDLX report, depends on the following factors:

- If a report has a single detail section it is imported as a Page report.
- If a report has a CrossTab control and its layout is composed of multiple sections it is imported as an RDLX report.

Note: Sections in a report appear as BandedList.

Please refer to the additional information below, to understand the conversion process in detail.

Importing Crystal Reports

To import Crystal Reports in ActiveReports, you need to install Visual Studio and Crystal Reports for Visual Studio on your machine. The supported versions of Visual Studio and corresponding Crystal Reports are as follows:

Visual Studio version for Crystal Reports	Editions	Crystal Reports	Assembly Version
2008	Professional, Team	Crystal Reports for Visual Studio 2008	10.5.3700.0

	System		
2010, 2012, 2013, 2015, 2017, 2019	...	SAP Crystal Reports, developer version for Microsoft Visual Studio	13.x.x.x

Crystal Report controls are converted in ActiveReports as follows:

Crystal Report	Section Report	Page report/RDLX report	Note
Box	Shape	Container	The LineWidth property and rounded boxes are not imported. If the Box control extends to multiple sections, the box is imported as line controls.
CrossTab	Empty SubReport	Empty Tablix	CrossTab control is not imported as it is.
Line	Line	Line	The size of Dot and Dash (the LineStyle property) is not the same as the original report.
Subreport	SubReport	Subreport	Set the subreport in code after conversion.
TextObject	TextBox	Textbox	Only page number, total page, page n of m in Special Fields are imported.
FieldObject	TextBox	Textbox	Only page number, total page, page n of m in Special Fields are imported.
Picture	Picture	Image	Picture object is not converted.

Limitations in Crystal Report conversion

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

Excel

The migration from Microsoft Excel file to ActiveReports can now be accomplished by using the **ActiveReports Import Wizard**. The ActiveReports Import Wizard is particularly useful when you want to convert multiple sheets of an Excel file to ActiveReports. It saves the time and effort of a developer to manually replicate the layout of each sheet of an Excel file in ActiveReports.

You can import a single sheet or multiple sheets of an Excel file to a Page or an RDLX report with just a few clicks. A single Excel sheet is imported as a report file, and the report name is the name of the sheet. An Excel file with multiple sheets is by default imported as separate report files, and the report names are the name of the corresponding sheets in the Excel file. You can also set **Merge all sheets into a single report file** option in the ActiveReports Import Wizard to import multiple sheets of Excel file as different pages of the report.

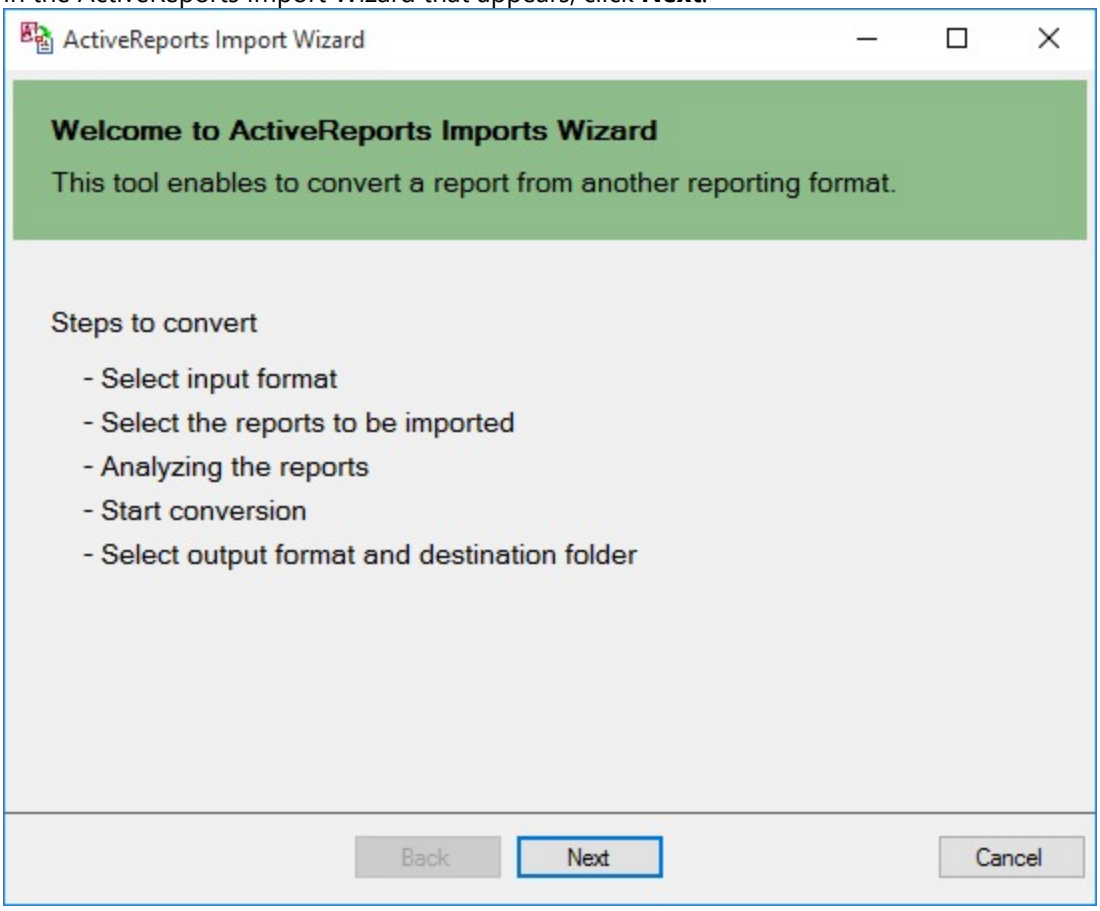
- **Importing Excel files in the ActiveReports Import Wizard**
- **Defining Table area in an Excel file**
- **Naming Rules for defining a Table area in Excel**
- **Conversion Rules for Table area in Excel**
- **Supported Objects and Properties**

- **Limitations**

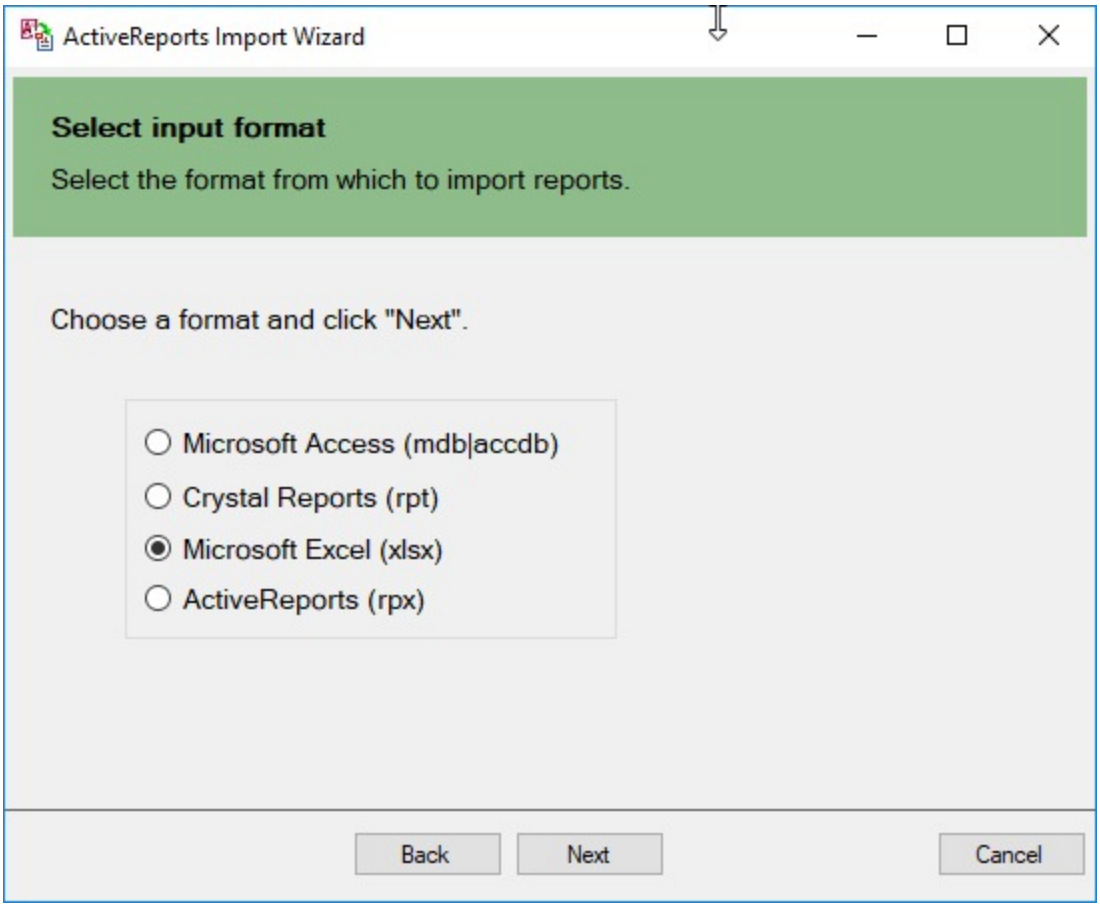
Note: The import formats that are not supported are .xls (Excel 97-2003) and .xlsm (Open XML with macro).

Importing Excel files in the ActiveReports Import Wizard

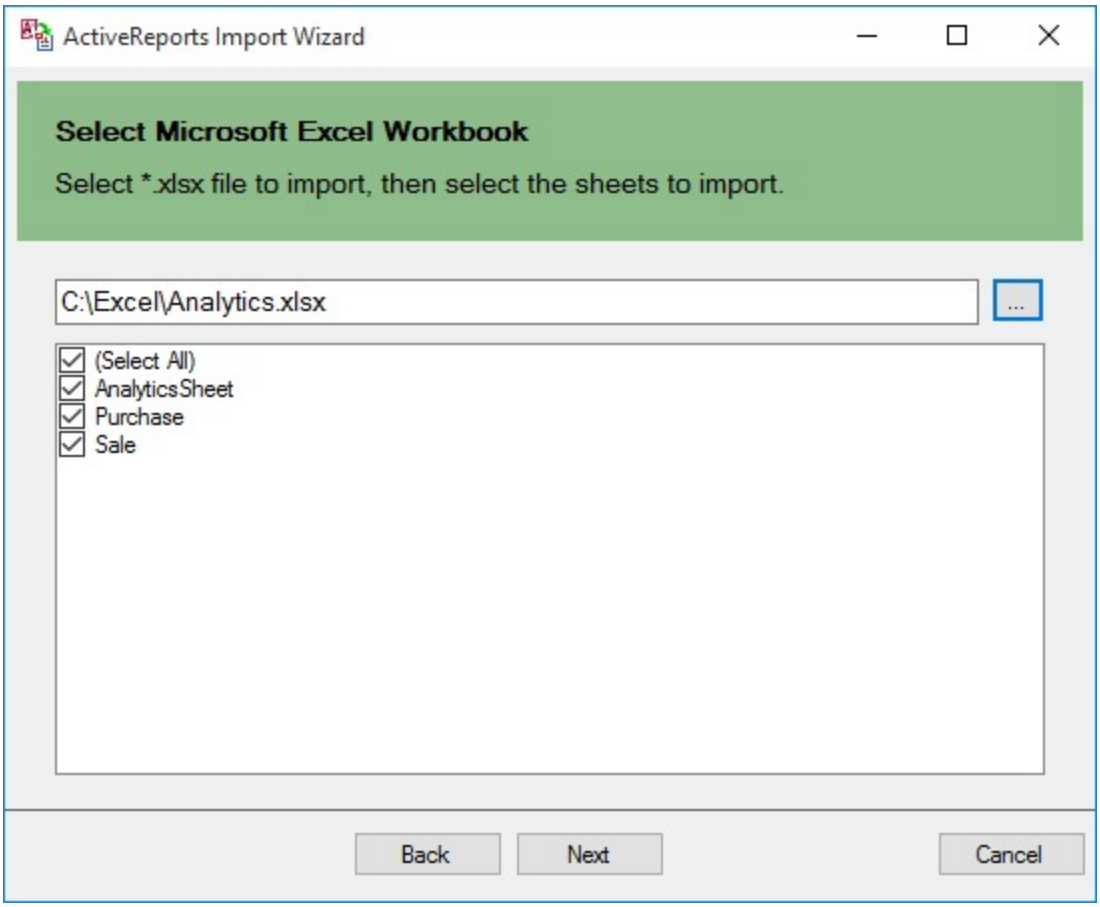
1. Run the ActiveReports Import Wizard. The wizard can be run from the start menu or by executing ActiveReports.Imports.Win.exe from **C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools** location.
2. In the ActiveReports Import Wizard that appears, click **Next**.



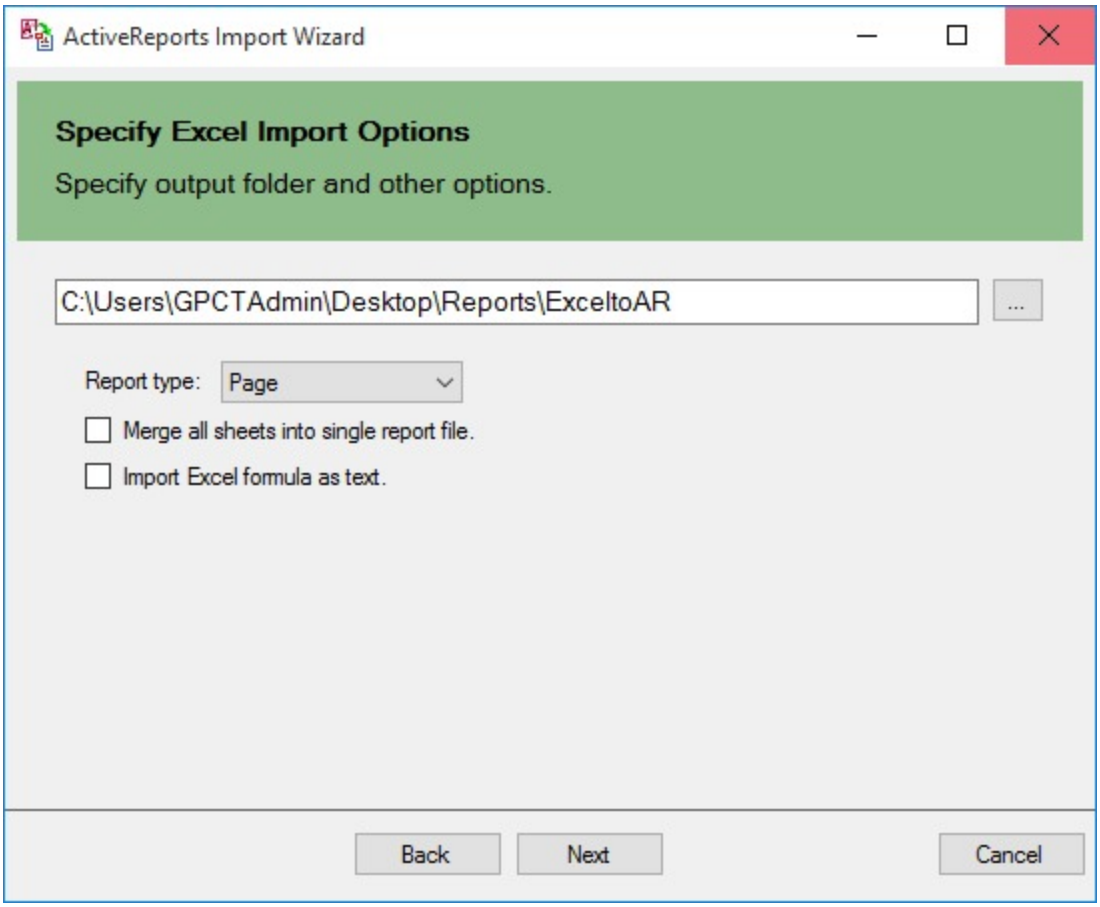
3. Choose **Microsoft Excel (xlsx)** as the input format and click **Next**.



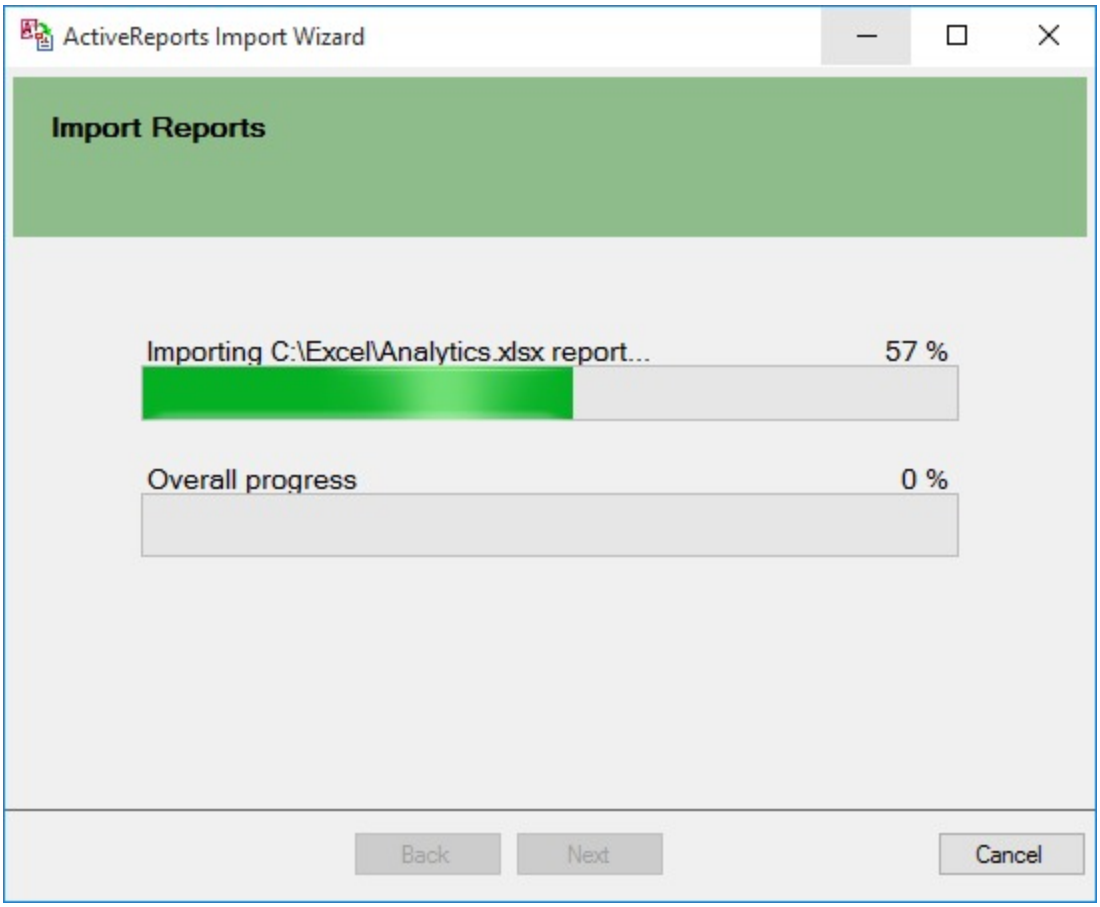
4. Click the ellipsis button to browse to the location that contains the files that you want to import. A list of files that you can import appears.
5. Select the sheets to import, click **Open,** and then click **Next** to analyze them.



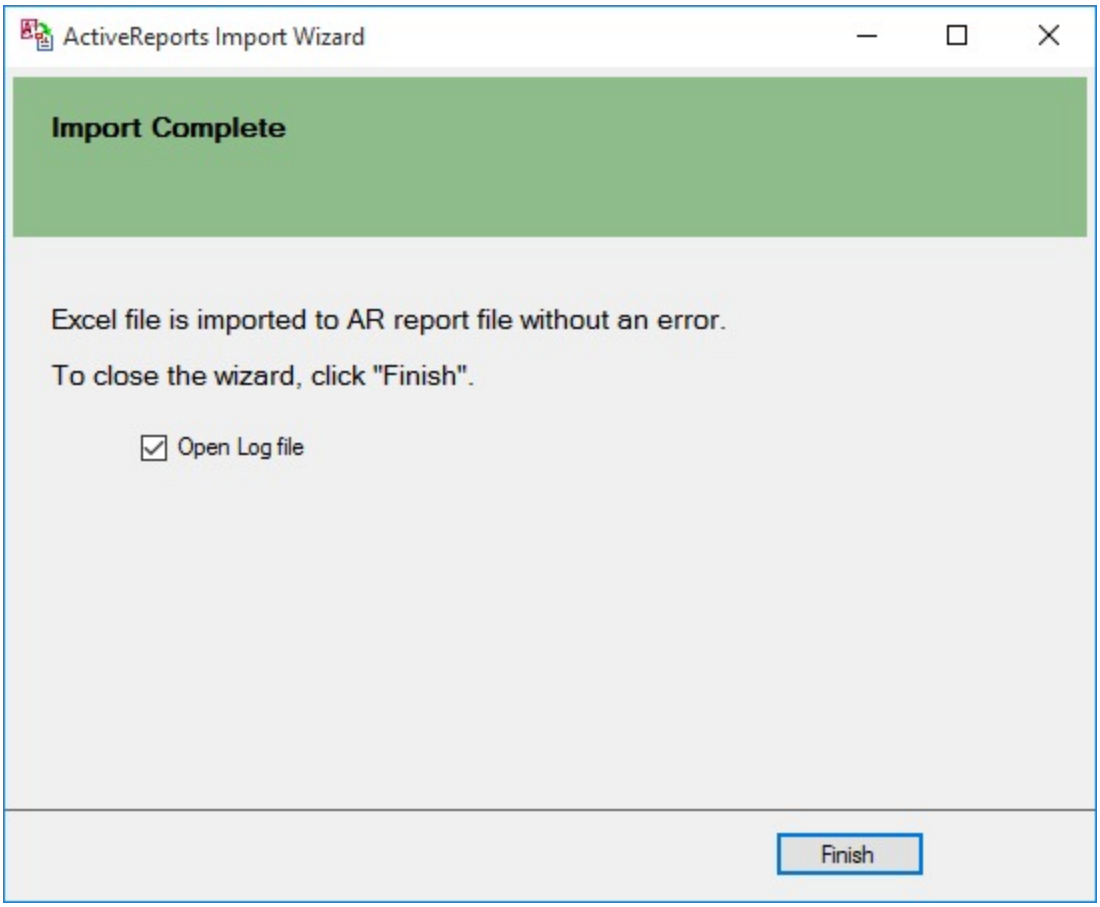
6. Use the ellipsis button to select a destination folder to store the converted reports. You can set the following options:
    - **Report Type:** Choose from Page report or RDLX report formats to import the Excel file. Note that Page report does not support multiple data sources. You should select RDLX report type if you want to add multiple data sources to the report.
    - **Merge all sheets into single report file:** Choose this option to import sheets of the Excel file as separate pages of a Page report. The report name is the name of the first sheet of the Excel file.
    - **Import Excel formula as text:** Choose this option to import Excel formula as text. If you keep the option unchecked, the Excel formula is imported as a calculated result.



7. Click **Next** to start the conversion.



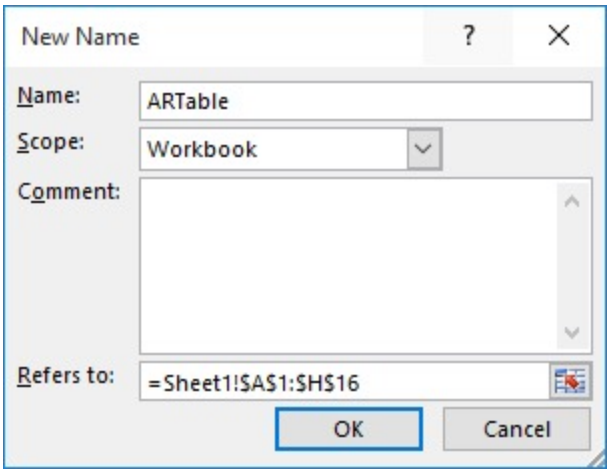
8. Once the conversion process is complete, click **Finish** to close the wizard and go the destination folder to view the converted reports. You may optionally leave the check on for the **Open Log file** checkbox to see the results log.



Defining Table area in an Excel file

Table area in Excel is the range of cells representing a Tabular data in the Excel.
If an Excel file has the table area that you want to import into ActiveReports as the Table data region, you must define the Table area first and then run the ActiveReports Import Wizard. Otherwise, defining the table area is not required.

1. Open the Excel file and select the table area.
2. Right-click to view the context menu.
3. Select the **Define Name** option.
4. In the **New Name** dialog box, define the table area and the rows based on **Naming Rules**. These naming rules must be followed for defining table areas in Excel.



5. Click **OK**.

Naming Rules for defining a Table area in Excel

To obtain the required table sections in ActiveReports' Table data region, you need to define the table area and its rows in the Excel file. In general, the table area is defined as **ARTable#.******, where:

- **#** is used to define more than one table areas. It can be any character, except symbol or character restricted by Excel. For example, ARTable, ARTable_1, or ARTableAbc.
- ****** is the name of the row (section): Detail, TableHeader, TableFooter, GroupHeader, or GroupFooter.
  In case of multiple rows (Table Header/Footer, Detail, Group Header/Footer), you need to set "#" for each row as well (for example, GroupHeader1, GroupHeader2, etc.)

Example 1: **To define a single table area**

Action	Naming Rule
Define whole table area	ARTable
Define each row	ARTable.Detail ARTable.TableHeader ARTable.TableFooter ARTable.GroupHeader1 ARTable.GroupFooter1

Example 2: To define a multiple table area

Action	Naming Rule
Define whole table area	ARTable1 ARTable2
Define each row	ARTable1.Detail ARTable1.TableHeader ARTable1.TableFooter ARTable2.Detail ARTable2.TableHeader

	ARTable2.TableFooter

Conversion Rules for Table area in Excel

The table area of Excel is imported as a Table data region in ActiveReports based on the following conversion rules.

- If the defined table area of Excel has three or more rows, the file data is converted to Table Header, Detail, and Table Footer as:

Excel Table	ActiveReports Table
Top Row	Table Header
Bottom Row	Table Footer
Other Rows	Detail

Note: For the Table Detail row, values for properties such as Value, Location, Size, etc. are imported from the cells of the first row.

- If the defined table area of Excel has two rows, the file data is converted as follows.

Excel Table	ActiveReports Table
First Row	Table Header
Second Row	Detail

- If the defined table area of Excel has only one row, the file data is converted as follows.

Excel Table	ActiveReports Table
First Row	Detail

Supported Objects and Properties

Excel		Page/RDLX report	
Item	Property	Item	Property
Page	Page setting	**Report**	-
	Size		PaperSize
	Orientation: Portrait		PaperOrientation: Portrait
	Orientation: Landscape		PaperOrientation: Landscape
	Margins (Top, Bottom, Left, Right)		Margins (Top, Bottom, Left, Right)
Cell	Value		Value

	Location	Location (Left, Top)
	Size	Size (Width, Height)
	Alignment	-
	Horizontal alignment: General	TextAlign: General
	Horizontal alignment: Left (Indent)	TextAlign: Left
	Horizontal alignment: Center	TextAlign: Center
	Horizontal alignment: Right (Indent)	TextAlign: Right
	Horizontal alignment: Justify	TextAlign: Justify
	Horizontal alignment: Distributed (Indent)	TextJustify: DistributeAllLines
	Vertical alignment: Top	VerticalAlign: Top
	Vertical alignment: Center	VerticalAlign: Middle
	Vertical alignment: Bottom	VerticalAlign: Bottom
	Text control: Wrap text	WrapMode: WordWrap
	Text control: Shrink to fit	ShrinkToFit: True
	Text direction: Left-to-Right	Direction: LTR
	Text direction: Right-to-Left	Direction: RTL
	Font	-
	Name	FontFamily
	Style: Regular	FontStyle: Normal
	Style: Italic	FontStyle: Italic
	Style: Bold	FontWeight: Bold
	Style: Bold Italic	FontWeight: Bold
	Size	FontSize
	Color	Color
	Underline: None	TextDecoration: None
	Underline: Single	TextDecoration: Single
	Border	-
	Line style (Top, Bottom, Left, Right): xlLineStyleNone	BorderStyle: None
	Line style (Top, Bottom, Left, Right): xlContinuous	BorderStyle: Solid
	Line style (Top, Bottom, Left, Right): xlDot	BorderStyle: Dotted
	Line style (Top, Bottom, Left, Right): xlDash	BorderStyle: Dashed

	Line style (Top, Bottom, Left, Right): xlDouble		BorderStyle: Double
	Line color		BorderColor
	Line weight: xlThin		BorderWidth: 1pt
	Line weight: xlMedium		BorderWidth: 2pt
	Line weight: xlThick		BorderWidth: 3pt
	Fill		-
	Background color		BackgroundColor
Table area	Each cell in a table area is converted to TextBox report item. **Note:** Whole table area is imported in ActiveReports even if table data is filtered.	**Table**	Location (Left, Top) Size (Width, Height) FixedSize (Width, Height)
Picture	Picture object is converted to Image report item.	**Image**	Value Source: Embedded Sizing: FitProportional Location (Left, Top) Size (Width, Height)

Limitations

- An Excel file that contains merged cells and table areas that are partially out of bounds, is not imported.
  - **Merged cell** - If merged cell starts within the page bounds and ends outside of them, the cell is not imported.
  - **Table area** - If table area starts within the page bounds and ends outside of them, the table area is not imported.
- ActiveReports Import Wizard does not support conversion of a **password protected Excel** file.
- The layout of only the first page of an Excel, as shown in the Page Break Preview option, is imported.
- The following Excel items are not imported to ActiveReports.
  1. **Page**
     - Header/Footer
  2. **Cell**
     - Number format (all categories, like Number, Date, Currency, etc.)
     - Strikethrough
     - Border (diagonal)
     - Fill effect
     - Fill pattern
     - Comment
     - Hyperlink
     - Table styles
     - Conditional formatting
  3. **Object**
     - Table
     - Shape
     - Chart
     - Pivot Table

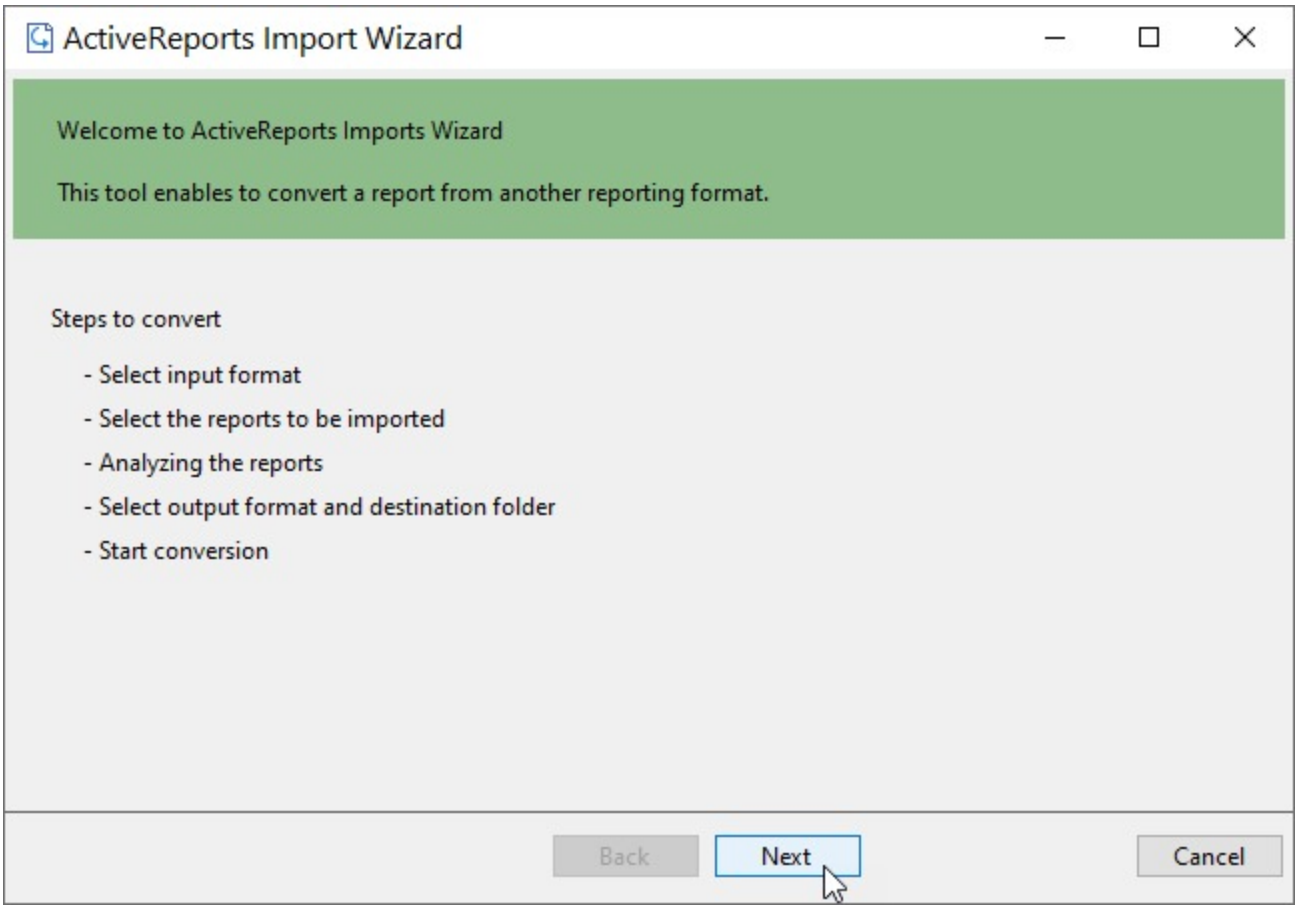
- WordArt
- ClipArt

MS SSRS RDLX

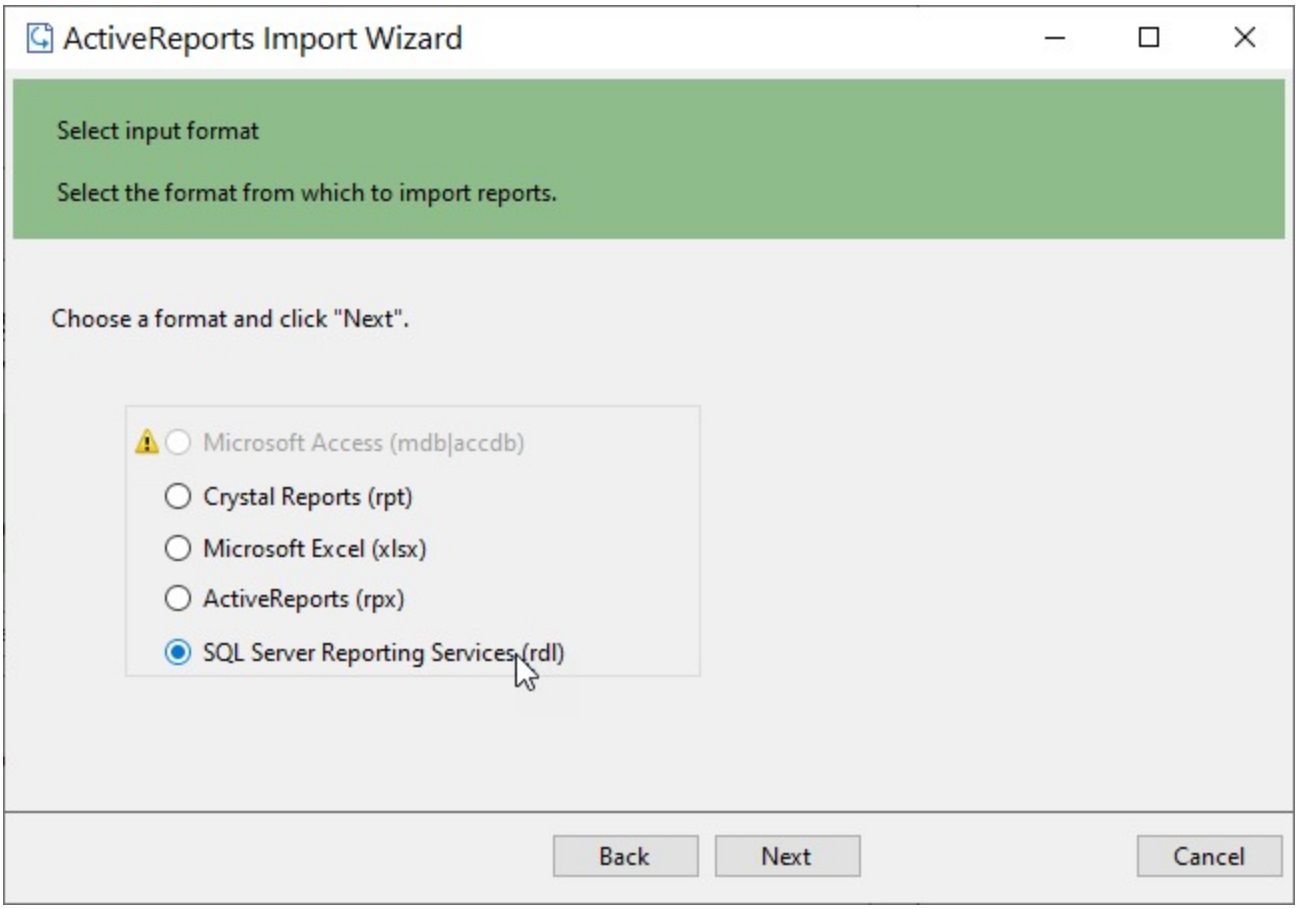
After conversion, most of the SSRS report controls are opened as is, with some limitations. Major differences in conversion are related to the TextBox and Chart controls.

Importing SSRS Reports in the ActiveReports Import Wizard

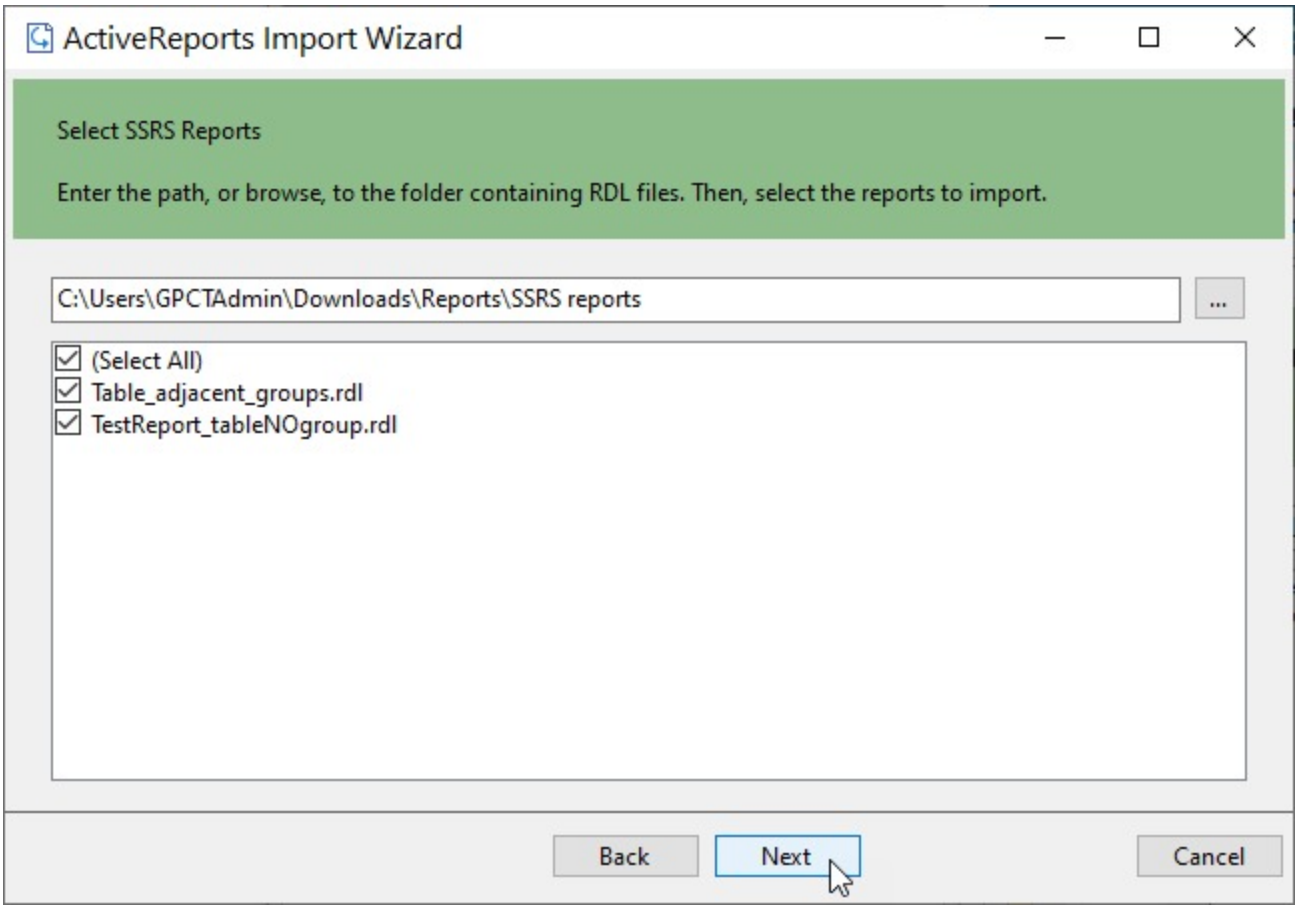
1. Run the ActiveReports Import Wizard. The wizard can be run from the start menu or by executing ActiveReports.Imports.Win.exe from **%Program Files (x86)%\MESCIUS\ActiveReports 18\Tools** location.
2. In the ActiveReports Import Wizard that appears, click **Next**.



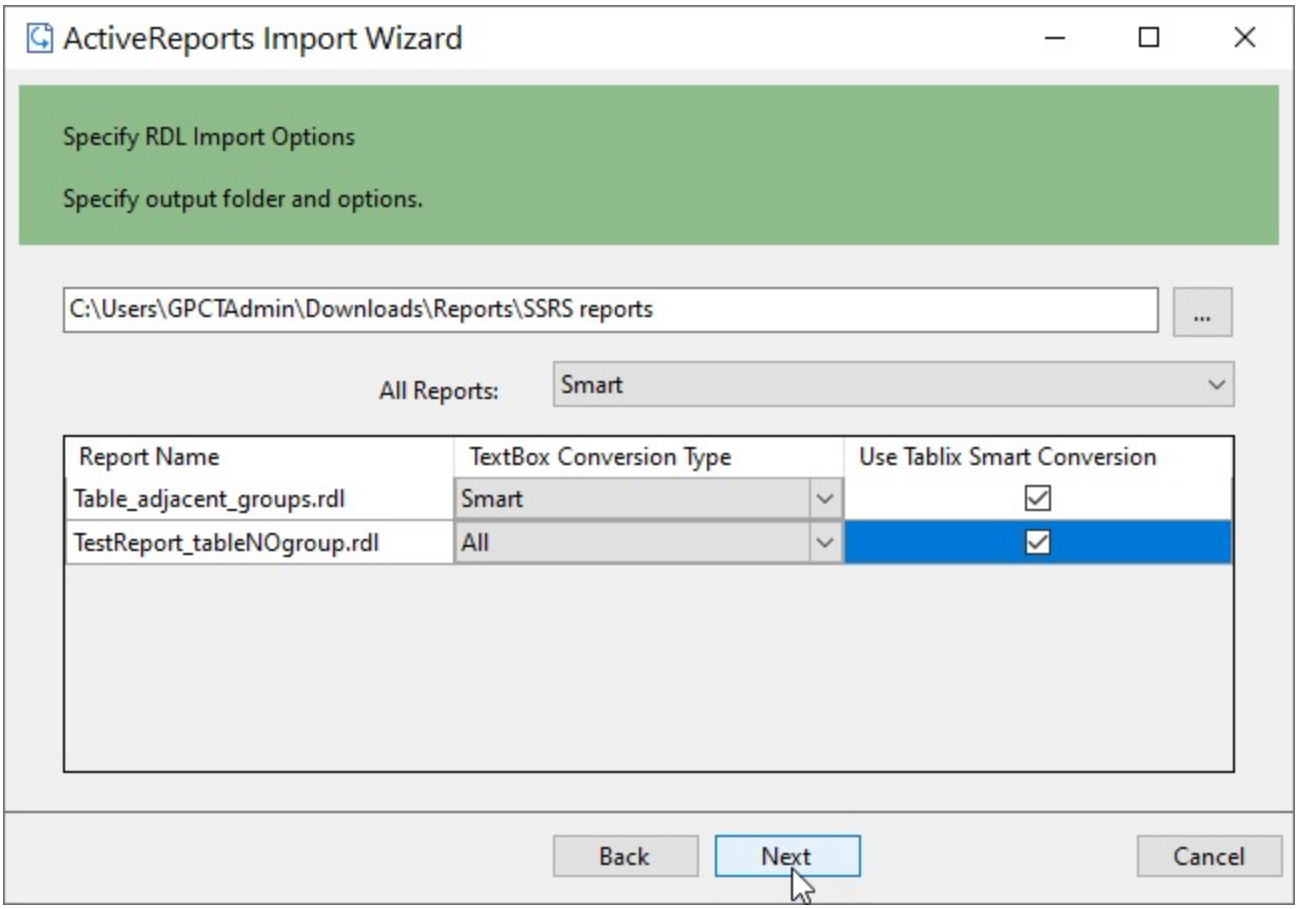
3. Choose **SQL Server Reporting Services (rdl)** as the input format and click **Next**.



4. Click the ellipsis button to browse to the location that contains the files that you want to import. A list of files that you can import appears.
5. Select the reports to import, click **Open,** and then click **Next** to analyze them.



6. Use the ellipsis button to select a destination folder to store the converted reports.

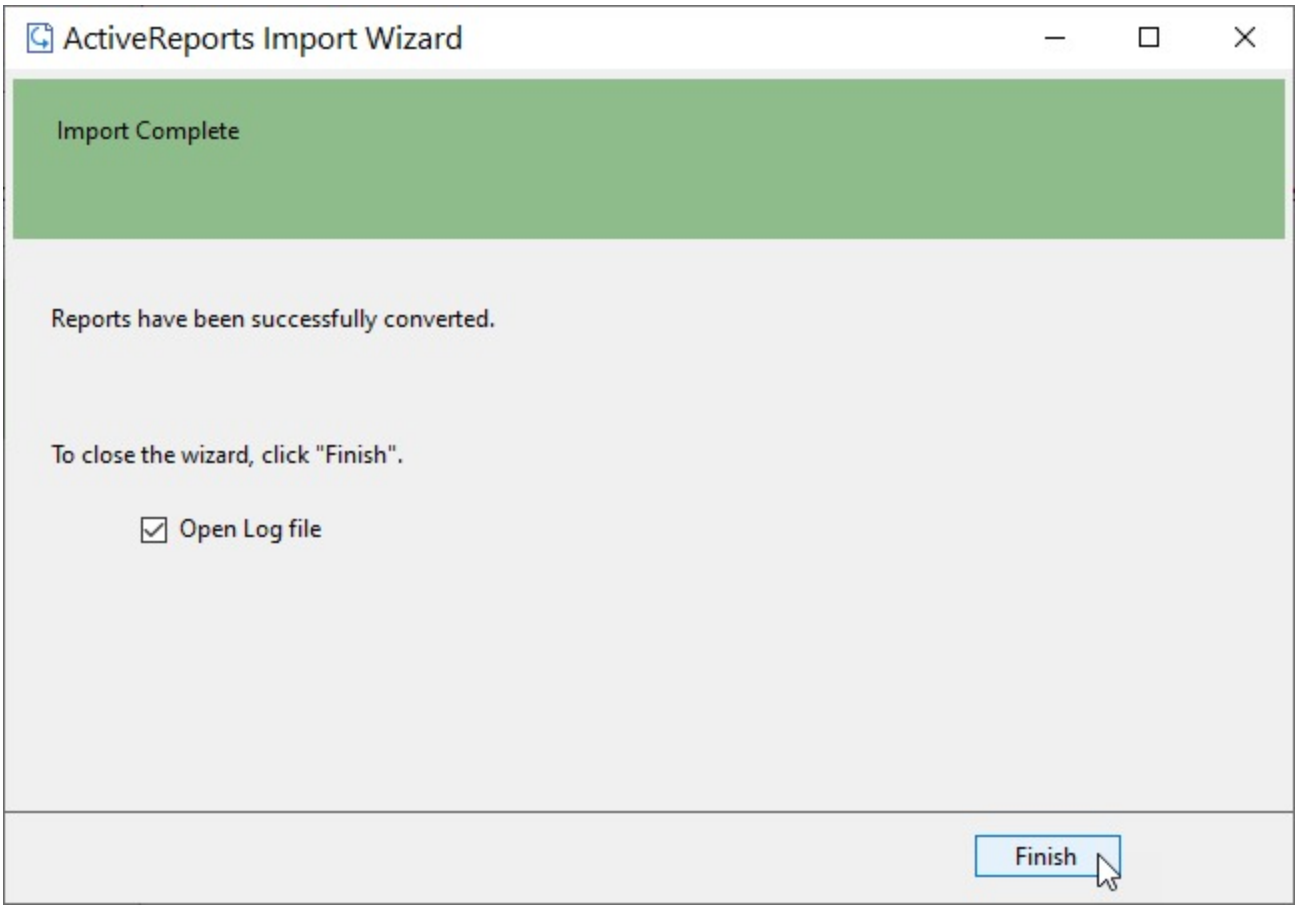


Select a **TextBox Conversion Type** for all reports in the drop-down list. The available options are as follows.

- **All** - All textbox controls are converted to FormattedText.
- **None** - All textbox controls are converted to TextBox, paragraph settings are lost.
- **Smart** (default) - Textbox controls with more than one paragraph or TextRun is converted to FormattedText.

Select the **Use Tablix Smart Conversion** option to convert an SSRS 'Tablix' to an ActiveReports 'Table' or 'List' data region, if possible. Check the Tablix Conversion limitation below.

7. Click **Next** to start the conversion.
8. Once the conversion process is complete, click **Finish** to close the wizard and go the destination folder to view the converted reports. You may optionally leave the check on for the **Open Log file** checkbox to see the results log.



Limitations

Tablix Conversion

The ActiveReports model is different from latest SSRS model. ActiveReports does not fully support latest RDLX specification. So, even when you check the **Use Tablix Smart Conversion**, some properties may be ignored for the closest conversion.

Chart Conversion

The ActiveReports chart model and its implementation is different from the latest RDLX specification, so you may need to manually adjust the conversion to achieve a similar behavior.

These charts are not supported and ignored at conversion to ActiveReports.

1. Sunburst
2. Treemap
3. ErrorBar
4. Map
5. Range Column
6. Range Smooth
7. Range Plain
8. Gauges

Multi-Section Reports Conversion

SSRS reports with multiple sections are converted to RDLX Report.

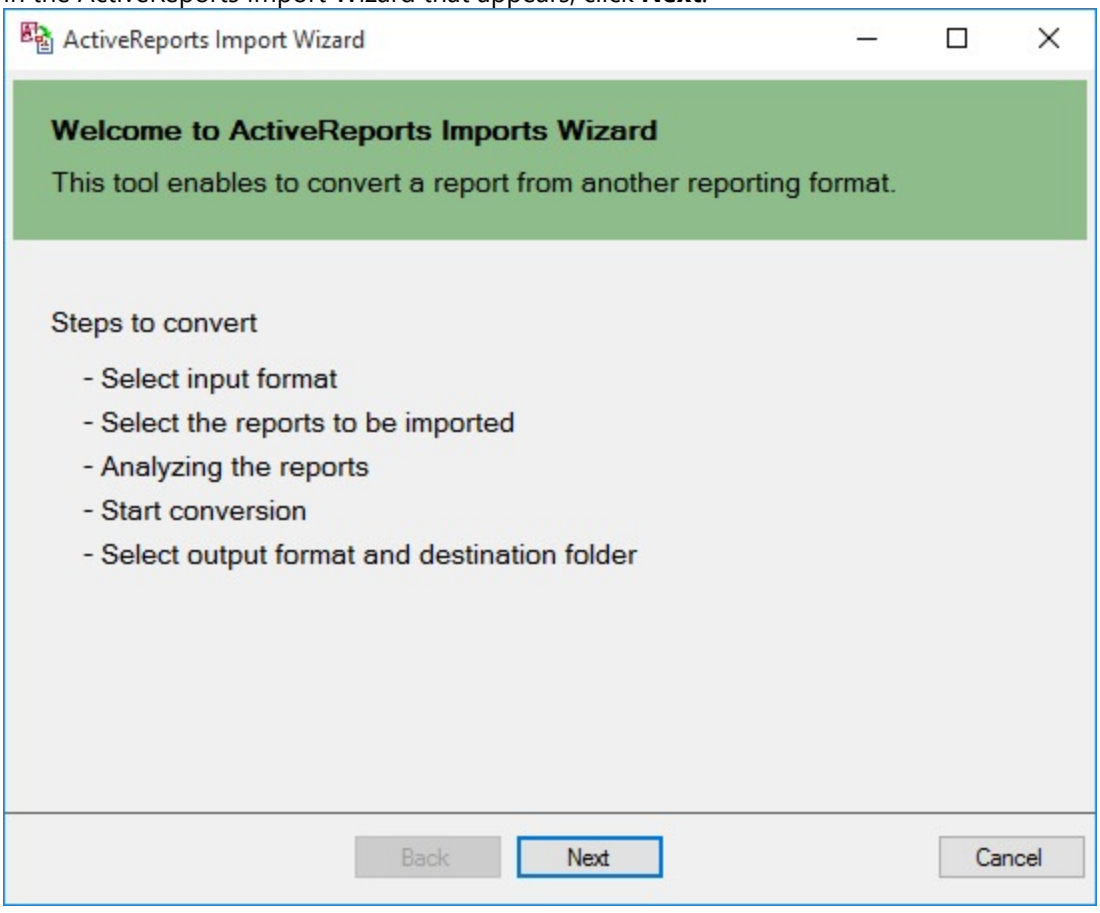
Migration between Supported Report Types

Migration from Section reports to Page/RDLX is also an import possibility.

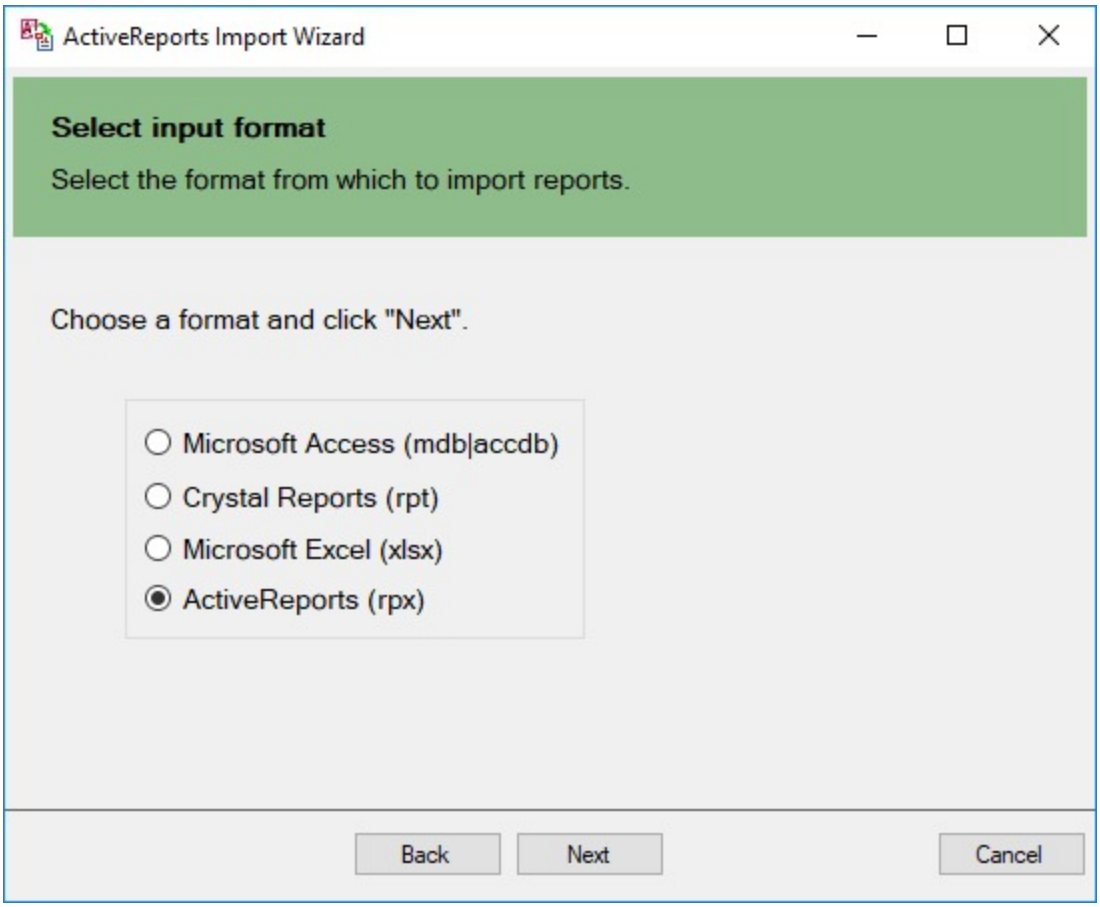
The ActiveReports Import Wizard helps you to convert existing ActiveReports Section report(s) (rpx) to Page and RDLX reports.

Importing Section Report(s) (rpx) the ActiveReports Import Wizard

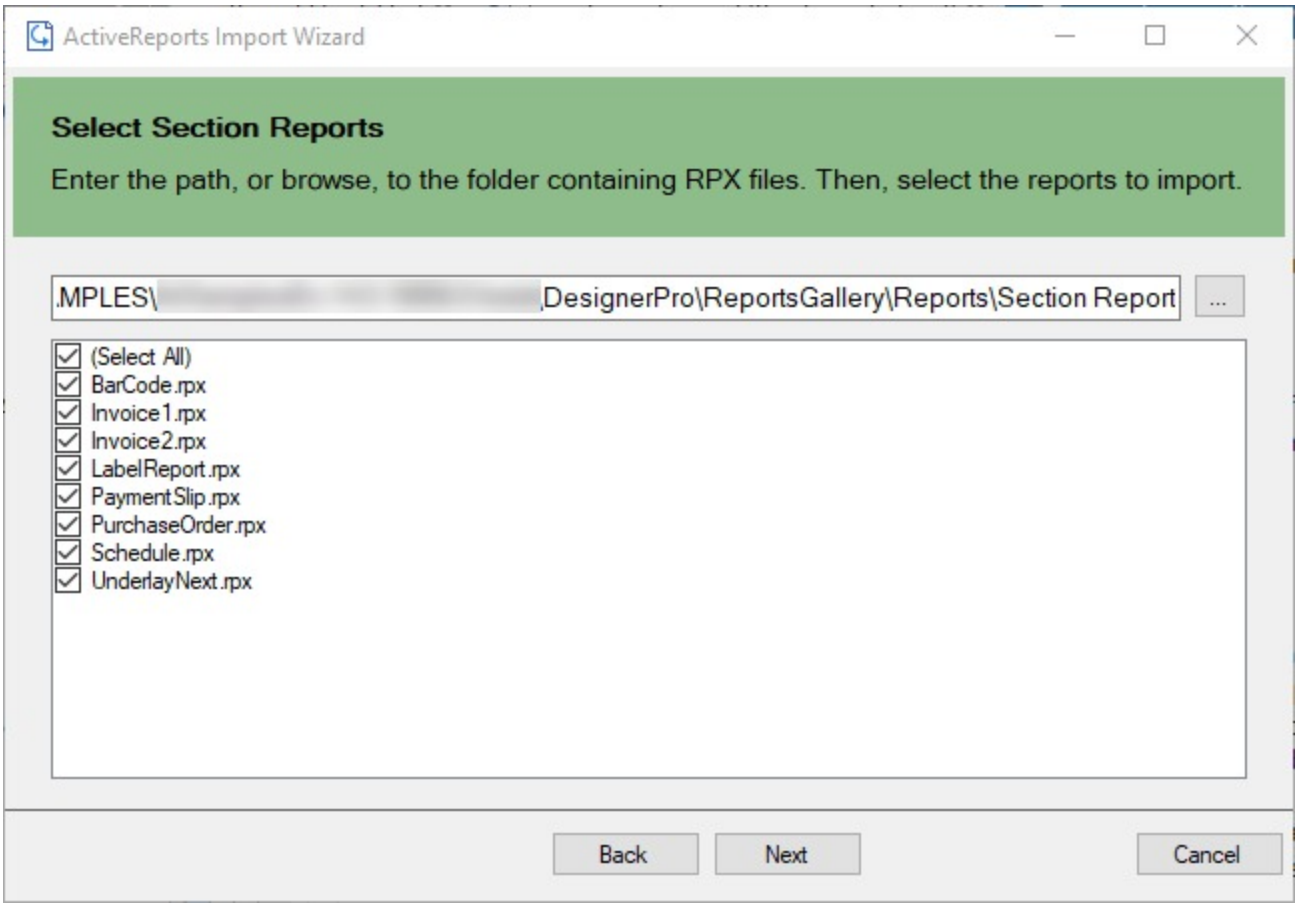
1. Run the ActiveReports Import Wizard. The wizard can be run from the start menu or by executing ActiveReports.Imports.Win.exe from **C:\Program Files (x86)\MESCIUS\ActiveReports 18\Tools** location.
2. In the ActiveReports Import Wizard that appears, click **Next**.



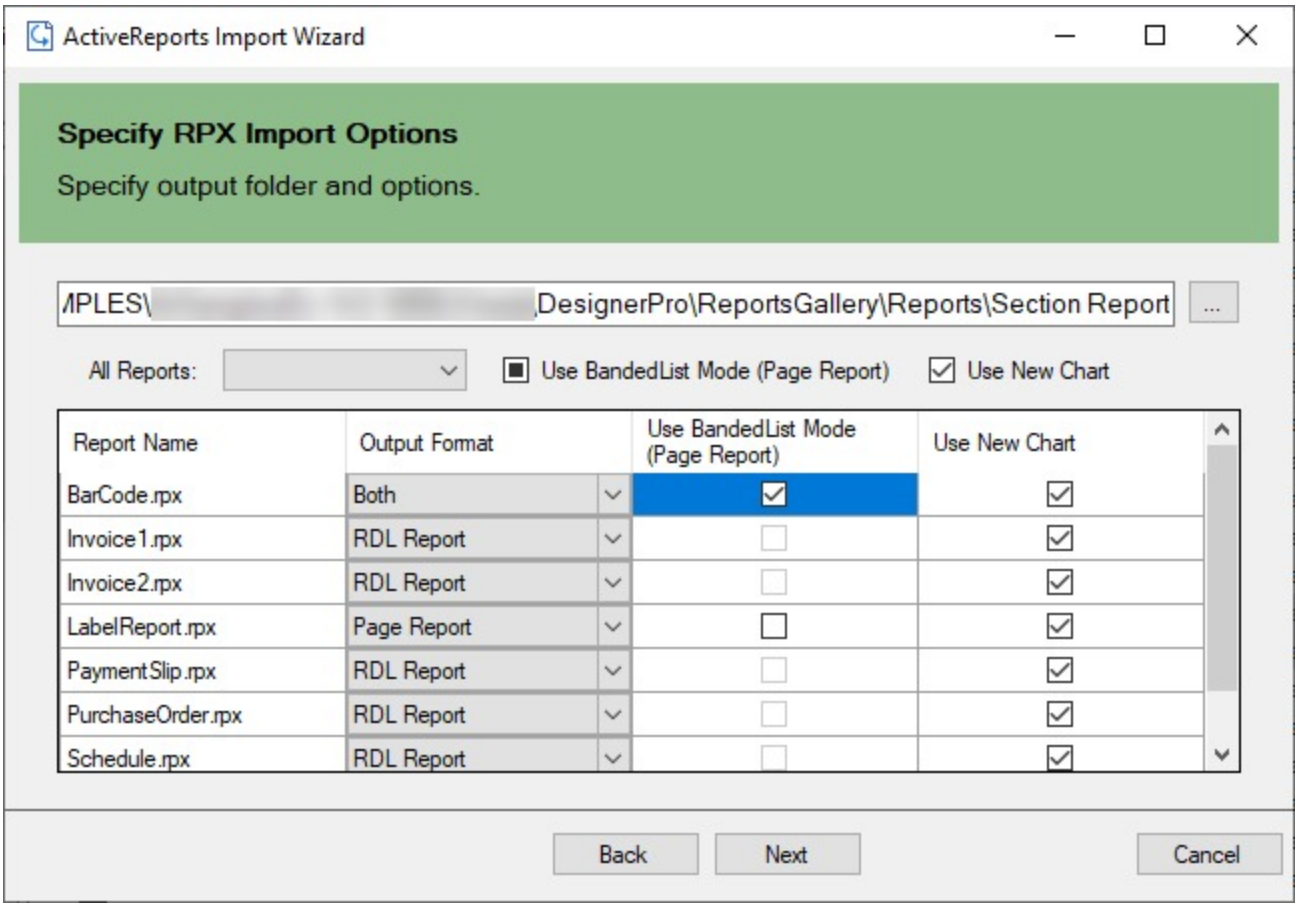
3. Choose **ActiveReports (rpx)** as the input format and click **Next**.



4. Click the ellipsis button to browse to the location that contains section report(s).
5. Select the reports to import and then click **Next** to analyze them.



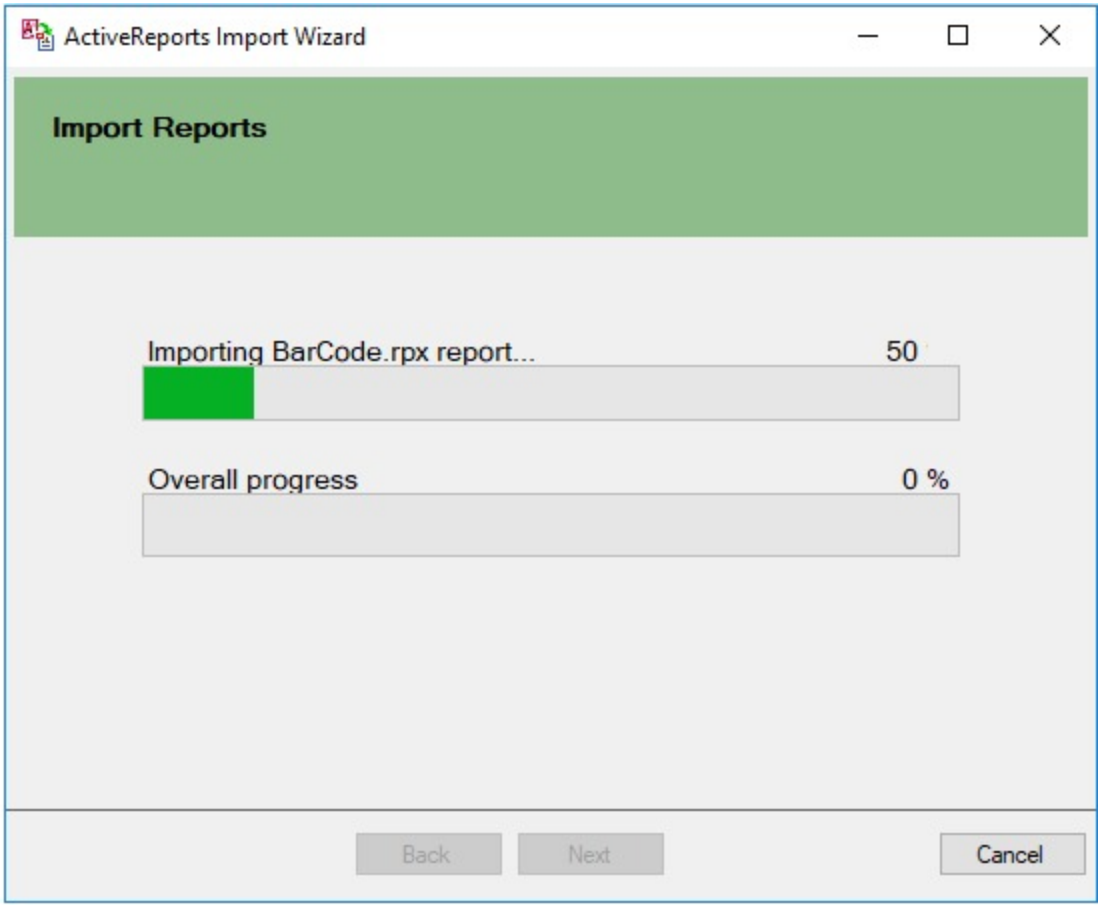
6. Specify the import options.



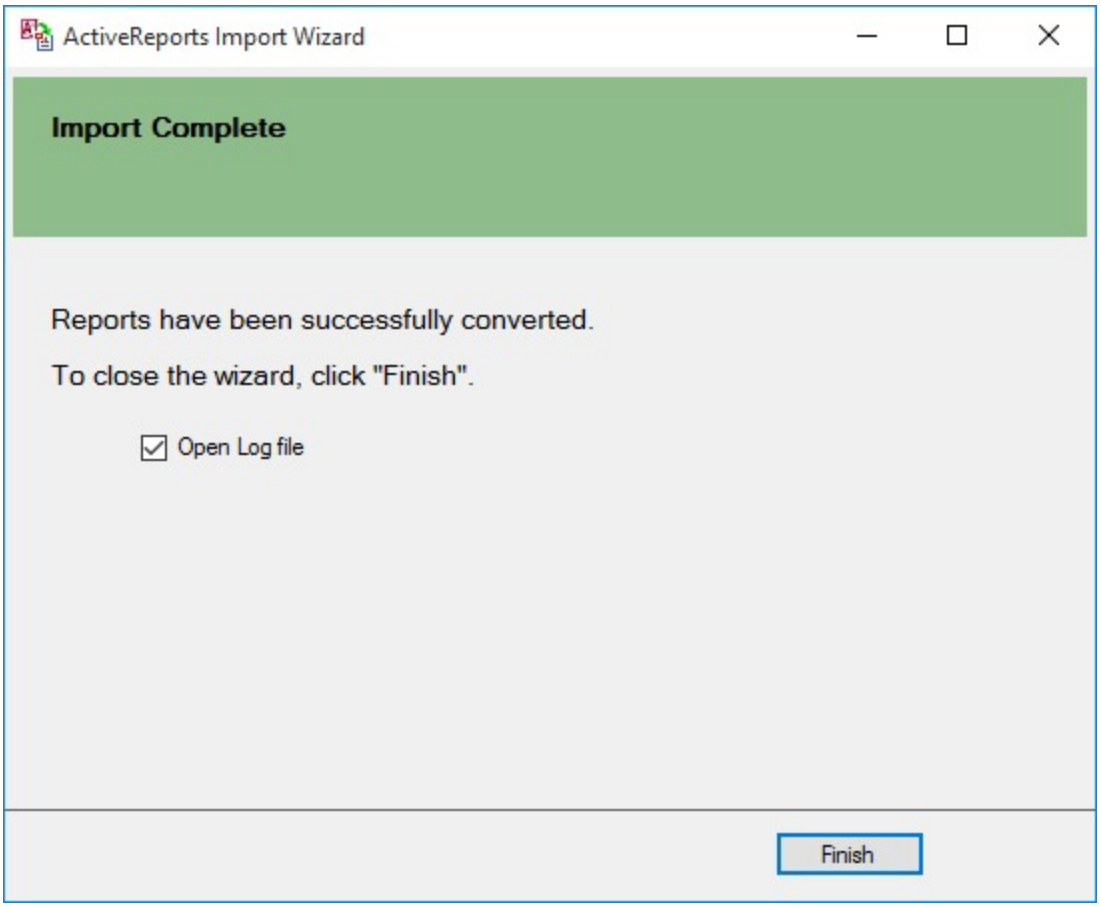
1. Use the ellipsis button to select a destination folder to store the converted reports. Specify the path to the output folder where the converted reports are saved.
2. Select the output format of the reports.
   From the drop-down, select output format as Page, RDLX, or Both. If 'Both' is selected, then the converted Page and RDLX reports are added with postfixes _page and _rdl, respectively in the destination folder.
3. Select whether to use Banded List for Page report.
   The option to select **Use Banded List Mode** is available if the output format of a report is Page report.
   - If you want all controls to be placed on one BandedList in the same way as in RDLX report, you should use banded list mode. In this case, the Fixed size for Page report needs to be set manually.
   - If you want to convert a Section report that has only one section (that is detail), you should not use banded list mode. With 'Use Banded List Mode' not selected (default):
     - controls are not placed on BandedList
     - sections such as Report Header/Report Footer and Page Header/Page Footer are ignored
     - all converted controls are placed on one Page and treated as they have been on one section

   The option to select **Use New Chart** lets you convert an existing chart to a new chart. By default, this option is selected; if unselected, the chart converts to classic chart.

7. Click **Next** to start the conversion.



8. Once the conversion process is complete, click **Finish** to close the wizard and go the destination folder to view the converted reports. You may optionally leave the check on for the **Open Log file** checkbox to see the results log.



⚠ **Important:** At conversion, any script from the script editor is ignored.

Section report controls are replaced with the RDLX or Page report controls as follows.

Section Report Control	RDLX/Page report Control
Label	TextBox
TextBox	TextBox
CheckBox	CheckBox
RichTextBox	FormattedText
Shape	Shape
Picture	Image
Line	Line
Barcode	Barcode
SubReport	SubReport
Chart	Chart (New or Classic)
ReportInfo	TextBox

CrossSectionLine	Line (in Page report when BandedList is not used)
CrossSectionBox	Container (in Page report when BandedList is not used)
PageBreak	Container in RDLX report with 'Height=0in' and 'PageBreakAtEnd=True' Not supported in Page report

Conversion rules for Section report to Page report

- The converted report inherits these properties of Section report - Paper Size, Orientation, and the Margins.
- If controls are placed outside the paper size (red line appears), they are not converted.
- If the sum of height of all sections (PageHeader, Detail, etc.) in a Section report is more than its paper height, controls above or below the paper height are ignored.

Conversion rules for Section report to RDLX report

- The entire Section report is converted to an RDLX report as the BandedList control.
- PageHeader and PageFooter sections are automatically created at conversion. You must not remove these sections even if no content is available for them.

Conversion Limitations (both Page and RDLX)

- The support of **Chart**, **Subreport**, and **RichTextBox** controls is limited to the basic functionality.
- Unused database fields are not imported.
- Calculated fields are converted into simple expressions.
- **PageTotal** and **PageCount** summary functions are not supported.
- **Visual Basic functions** are not supported. Expressions with function calls are imported as is.
- Following properties of Section report, on conversion to Page/RDLX report, are not supported.
  - Culture
  - DataMember
  - ExpressionErrorMessage
  - MaxPages
  - TrayHeight
  - TrayLargeIcon
  - UserData
  - Watermark (for RDLX and Page (using banded list mode))
- MaxLength property for RichTextBox control on conversion to Page/RDLX report is not supported.

Conversion Limitations for Chart control to New Chart

- **BezierXY**, **Bubble**, **HiLo**, **HiLo Open Close**, and **LineXY** chart types are not properly implemented.
- **Axis labels** may be duplicated when using several **Series** with the same category grouping.
- **Colors** may differ when set to default.
- **Marker labels** may disappear when Field type doesn't match Format type and implicit type conversion is required.
- **Custom angle** for labels are not supported (can be only fixed 0, 90, 270).
- **Multiple chart titles** are not supported (can be only one title).
- **Stacked** option for Charts is not supported.

Section 508

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.

Summary

Guideline		Applicable	ActiveReports Area
Section 1194.21	Software Applications and Operating Systems	Applicable	End User Designer, Windows Viewer, WPF Viewer, Web controls (HTML Viewer).
Section 1194.22	Web-Based Intranet and Internet Information and Applications	Applicable	Web controls (HTML Viewer).
Section 1194.23	Telecommunications Products	Not applicable	none
Section 1194.24	Video and Multimedia Products	Not applicable	none
Section 1194.25	Self-Contained, Closed Products	Not applicable	none
Section 1194.26	Desktop and Portable Computers	Not applicable	none
Section 1194.31	Functional Performance Criteria	Applicable	End User Designer, Windows Viewer, WPF Viewer, Web controls (HTML Viewer).
Section 1194.41	Information, Documentation and Support	Applicable	Documentation and Support.

Accessibility Summary:

All major features of ActiveReports software are accessible via keyboard navigation.

DISCLAIMER:

MESCIUS inc. MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. The following information reflects the general accessibility features of MESCIUS inc. 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.

Software Applications and Operating Systems

Section 1194.21

Criteria	Status	Remarks
(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.	Supported with exceptions	ActiveReports partially supports keyboard navigation. Users cannot execute report functions completely using the keyboard.
(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 application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer.	Supported	The controls do not disrupt or disable industry standard accessibility features of other products.
(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 with exceptions	Some elements in ActiveReports cannot track focus. Focus is not exposed to screen readers like NVDA.
(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 with exceptions	The images representing program elements do not support alternative text. Partial support for Assistive Technology is provided for some interface elements.
(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 image used to identify a programmatic element has a consistent meaning throughout an application's performance.
(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 such as text content, caret location, and any text attribute (i.e. bold, italic, size, color, etc.) is available for all the viewer elements.

(g) Applications shall not override user selected contrast and color selections and other individual display attributes.	Supported with exceptions	User selected contrast and color settings are not overridden, except for some minor exceptions in End User Designer.
(h) When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.	Supported	Animated image is only used when a report is being loaded in the viewer. The users can provide a static text for the animation using the loadCompleted event of the Viewer.
(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	ActiveReports supports a variety of colors, themes and styles that can be applied to controls in a report.
(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.	Supported	The controls do not use flashing or blinking text or objects.
(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	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-based Internet Information and Applications

Section 1194.22

Criteria	Status	Remarks
(a) A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).	Supported	Each non-text element has a text equivalent.
(b) Equivalent alternatives for any multimedia presentation	Not applicable	ActiveReports web controls do

shall be synchronized with the presentation.		not include multimedia.
(c) Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.	Not applicable	ActiveReports web controls do not provide any information using color.
(d) Documents shall be organized so they are readable without requiring an associated style sheet.	Supported with exceptions	Toolbar icons in some web controls are not visible when the associated styles are disabled.
(e) Redundant text links shall be provided for each active region of a server-side image map.	Not applicable	There are no server-side image maps in ActiveReports web controls.
(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.	Not applicable	ActiveReports does not use client-side image maps in ActiveReports web controls.
(g) Row and column headers shall be identified for data tables.	Not applicable	By default, there are no data tables associated with the ActiveReports web controls.
(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.	Not applicable	By default, there are no data tables associated with the ActiveReports web controls.
(i) Frames shall be titled with text that facilitates frame identification and navigation.	Not applicable	By default, there are no frames associated with ActiveReports web controls.
(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	ActiveReports Web controls do not contain any feature that causes page flickering.
(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.	Not applicable	Text-only page is not available in ActiveReports WebViewer or Web controls.
(l) 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).	Not applicable	Not applicable
(m) 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.	Supported with exceptions	The components used in ActiveReports have been configured for maximum compliance. Some form controls may not support Assistive Technology.
(n) A method shall be provided that permits users to skip	Not applicable	ActiveReports does not support

repetitive navigation links.		repetitive navigation links.
(o) When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.	Not applicable	Timed response is not required.

Performance Criteria

Section 1194.31

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.	Supported with exceptions	Partial support is provided for assistive technologies like screen readers.
(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 visually impaired shall be provided.	Supported	ActiveReports does not require visual acuity greater than 20/70. It is exposed to magnification tools like the Screen Magnifier that can be used for magnifying the screen.
(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	ActiveReports does not use any sound output. Hearing is not necessary to use the product.
(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	ActiveReports does not provide functionalities that require audio or video aids.
(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	ActiveReports does not provide any functionality that make use of user speech.
(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.	Supported	ActiveReports does not provide any functionality that requires fine motor control or simultaneous action.

Information, Documentation and Support

Section 1194.41

Criteria	Status	Remarks
(a) Product support documentation provided to end-users shall be made available in alternate formats upon request, at	Supported	Documentation is available in four formats: HTML(Web), *.chm, *.PDF, and

no additional charge.		Help3(Visual Studio help content)
(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	ActiveReports provides information about accessibility and compatibility features in the documentation. The documentation is exposed to screen readers.
(c) Support services for products shall accommodate the communication needs of end-users with disabilities.	Supported	Support services such as telephone, email and forum support are provided to the customers.

Azure and Medium Trust

ActiveReports supports Azure. We recommended using Page/RDLX reports or SectionReports in CrossPlatform compatibility mode.

All features of ActiveReports are available without restrictions in a Full trust environment. You can also use ActiveReports under Medium trust, but with limitations on some of the features.

> ⚠ **Caution**: Medium trust does not adequately protect your application and should not be used. For more information see MSDN link.

> **Note**:
>
> - 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.
> - For information on licensing a class library project, see the article on Licensing Compiled Code.

Feature Limitations

1. Exporting
   - RTF, Text, TIFF and Excel filters are not supported in Medium trust.
   - Digital signatures cannot be used in case of PDF rendering extension and PDF export.
   - Chart and Tablix control of Page Reports and RDLX Reports are not displayed properly in case of PDF rendering extension and PDF export.
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 control must be set to **PNG**.
5. OleObject, Formatted Text 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.
7. WebViewer
   - Bullet and Sparkline controls of Page Reports and RDLX Reports are not displayed in all the Viewer types.
   - Report containing the Image control of Page Reports and RDLX Reports is not displayed properly in case of HTML type.

Recommended Development Environment for Medium Trust Tests

To set up a Medium trust environment

Open the Web.config file and 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\v4.0.30319\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.

Customization

ActiveReports provides a wide variety of customization and localization options.

This section covers the following customization aspects:

Configure ActiveReports

Localization

Additional Customization Options

Configure ActiveReports

The default location of the ActiveReports configuration file (ActiveReports.config) is the installer location: C:\Program Files (x86)\MESCIUS\ActiveReports 18. Otherwise, the file may be placed beside the application or the plugin assembly. See the Custom Tile Provider sample description as an example.

You can control various settings from the configuration file, such as:

- print dialog settings,
- enable classic charts,
- disable new charts in Page/RDLX reports,
- enable Report Library
- enable Matrix control (required for some legacy Page/RDLX reports),
- enable OleObject control (required for some legacy Section reports),
- custom report items and designers,
- specify custom fonts,
- map control and map tile providers
- custom map tile providers,
- custom data providers (including ADO.NET),
- control the directory for Report Parts, and
- prevent report crashing.

Note: For now, the WebDesigner and Js Viewer components have only limited support of the configuration file. It is planned to extend this support in future versions. Please check with our support for the details.

Control Print Dialog Settings

Set **PrintDialogStyle** key to the values shown below to change the default print dialog for the designer.

ActiveReports.config file

```
<Configuration>
  <appSettings>

          <!-- For OS native print dialog -->
                <add key="PrintDialogStyle" value="Standard"/>

          <!-- For OS native print dialog shown in Windows XP style
                <add key="PrintDialogStyle" value="XpStyle"/>

               For Print dialog with advanced printing settings, used by default
                <add key="PrintDialogStyle" value="Advanced"/> -->
  </appSettings>
</Configuration>
```

Enable Classic Charts

Set **EnableChartClassic** key to the value 'true' to enable the classic chart in the designer.

ActiveReports.config file

```
<Configuration>
  <appSettings>
    <add key="EnableChartClassic" value="true"/>
  </appSettings>
</Configuration>
```

See the Classic Chart topic.

Disable New Charts in Page/RDLX Reports

Set the **EnableChart** key to the value 'false' to disable the classic chart in the designer.

ActiveReports.config file

```
<Configuration>
  <appSettings>
    <add key="EnableChart" value="false"/>
  </appSettings>
</Configuration>
```

See the Chart topic.

Enable Report Library

Set the **ShowReportsLibrary** key to the value 'true' to show the Library panel in the designer.

ActiveReports.config file

```
<Configuration>
   <appSettings>
     <add key="ShowReportsLibrary" value="true"/>
   </appSettings>
 </Configuration>
```

Control the directory for Report Parts

Set the **ReportPartsDirectory** key to some folder path to specify the reports folder for the designer. This is valid only if Report Library is enabled.

ActiveReports.config file

```
<Configuration>
  <appSettings>
    <add key="ReportPartsDirectory" value="path to folder"/>
  </appSettings>
</Configuration>
```

Enable the OleObject Control

Set the **EnableOleObject** key to the value 'true' to enable OLEObject control in Visual Studio Designer for Section report.

ActiveReports.config file

```
<Configuration>
  <appSettings>
    <add key="EnableOleObject" value="true"/>
  </appSettings>
</Configuration>
```

Enable Matrix Control

Set the **EnableMatrix** key to the value 'true' to enable the Matrix control in the designer.

ActiveReports.config file

```
<Configuration>
  <appSettings>
    <add key="EnableMatrix" value="true"/>
  </appSettings>
</Configuration>
```

Specify Line Breaking Algorithm

Set the **LineBreakingAlgorithm** key to 'Legacy' to enable legacy mode for wrapping lines when preparing text with numerical values for display. By default, 'Unicode' line breaking algorithm is used.

ActiveReports.config file

```
<Configuration>
    <appSettings>
        <add key="LineBreakingAlgorithm" value="Legacy"/>
    </appSettings>
</Configuration>
```

Custom Report Items

Page and RDLX reports allow extending the functionality of existing controls in the ActiveReports Designer. The config file should be similar to the following.

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
```

```
        <ReportItems>
            <ReportItem Name="RadarChart" Type="GrapeCity.ActiveReports.Samples.Radar.RadarChart, RadarChart" />
        </ReportItems>
        <ReportItemDesigner>
            <ReportItem Name="RadarChart" Type="GrapeCity.ActiveReports.Samples.Radar.RadarDesigner, RadarDesigner"
BitmapResource="GrapeCity.ActiveReports.Samples.Radar.RadarIcon.png" />
        </ReportItemDesigner>
    </Extensions>
</Configuration>
```

See the Custom Chart sample.

Specify Custom Fonts

In the ActiveReports.config file, you can also specify **custom fonts** to be used in Page, RDLX, and Section reports (in the CrossPlatform compatibility mode) as in the following example.

Note: Custom font settings have higher priority over default font settings.

ActiveReports.config file

```
<Configuration>
    <Extensions>
        <FontFactory>
            <AddFolder Path="%WINDIR%/Fonts" Recurse="true" />
            <AddFolder Path="%USERPROFILE%/AppData/Local/Microsoft/Windows/Fonts" Recurse="true" />
            <Substitute Font="Arabic Transparent" To="Arial" />
            <Substitute Font="Coutier" To="Coutier New" />
            <Substitute Font="Helv" To="Arial" />
            <Substitute Font="Helvetica" To="Arial" />
            <Substitute Font="MS Sans Serif" To="Microsoft Sans Serif" />
            <Substitute Font="MS Serif" To="Times New Roman" />
            <Substitute Font="Times" To="Times New Roman" />
            <Substitute Font="Tms Rmn" To="Times New Roman" />
            <SetFallbackFont Font="Microsoft Sans Serif" />
            <SetFallbackFont Font="MS UI Gothic" />
            <SetFallbackFont Font="Arial" />
            <AddFontLink Font="Lucida Sans Unicode" List="MS UI Gothic,PMingLiU,SimSun,Gulim,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI,Malgun Gothic" />
            <AddFontLink Font="Microsoft Sans Serif" List="MS UI Gothic,Yu Gothic
UI,PMingLiU,SimSun,Gulim,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun Gothic" />
            <AddFontLink Font="Tahoma" List="MS UI Gothic,PMingLiU,SimSun,Gulim,Yu Gothic UI,Microsoft JhengHei
UI,Microsoft YaHei UI,Malgun Gothic" />
            <AddFontLink Font="Segoe UI" List="Tahoma,Meiryo UI,MS UI Gothic,Microsoft JhengHei UI,Microsoft YaHei
UI,Malgun Gothic,PMingLiU,SimSun,Gulim,Yu Gothic UI" />
            <AddFontLink Font="Ebrima" List="Segoe UI,Meiryo UI,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="Gadugi" List="Segoe UI,Meiryo UI,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="Khmer UI" List="Segoe UI,Meiryo UI,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="Lao UI" List="Segoe UI,Meiryo UI,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="Leelawadee" List="Segoe UI,Meiryo UI,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="Leelawadee UI" List="Segoe UI,Meiryo UI,Microsoft JhengHei UI,Microsoft YaHei
UI,Malgun Gothic,Yu Gothic UI" />
            <AddFontLink Font="Nirmala UI" List="Segoe UI,Meiryo UI,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun
Gothic,Yu Gothic UI" />
```

```
            <AddFontLink Font="MingLiU" List="Microsoft Sans Serif,SimSun,MS Mincho,BatangChe,Microsoft JhengHei
UI,Microsoft YaHei UI,Yu Gothic UI,Malgun Gothic" />
            <AddFontLink Font="PMingLiU" List="Microsoft Sans Serif,SimSun,MS PMincho,Batang,Microsoft JhengHei
UI,Microsoft YaHei UI,Yu Gothic UI,Malgun Gothic" />
            <AddFontLink Font="Microsoft JhengHei" List="Segoe UI,MingLiU,Microsoft YaHei,Meiryo,Malgun Gothic,Yu
Gothic UI" />
            <AddFontLink Font="Microsoft JhengHei UI" List="Segoe UI,MingLiU,Microsoft YaHei UI,Meiryo UI,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="SimSun" List="Microsoft Sans Serif,PMingLiU,MS PMincho,Batang,Microsoft YaHei
UI,Microsoft JhengHei UI,Yu Gothic UI,Malgun Gothic" />
            <AddFontLink Font="NSimSun" List="PMingLiU,MS Mincho,BatangChe,Microsoft YaHei UI,Microsoft JhengHei
UI,Yu Gothic UI,Malgun Gothic" />
            <AddFontLink Font="Microsoft YaHei" List="Segoe UI,Segoe UI,SimSun,Microsoft JhengHei,Meiryo,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="Microsoft YaHei UI" List="Segoe UI,SimSun,Microsoft Jhenghei UI,Meiryo UI,Malgun
Gothic,Yu Gothic UI" />
            <AddFontLink Font="Yu Gothic UI" List="Segoe UI,Microsoft JhengHei,Microsoft YaHei,Malgun Gothic" />
            <AddFontLink Font="Meiryo" List="Segoe UI,Yu Gothic UI,MS UI Gothic,Microsoft Jhenghei,Microsoft
YaHei,Malgun Gothic" />
            <AddFontLink Font="Meiryo UI" List="Segoe UI,Yu Gothic UI,MS UI Gothic,Microsoft Jhenghei UI,Microsoft
YaHei UI,Malgun Gothic" />
            <AddFontLink Font="MS Gothic" List="MingLiU,SimSun,GulimChe,Yu Gothic UI,Microsoft JhengHei
UI,Microsoft YaHei UI,Malgun Gothic" />
            <AddFontLink Font="MS PGothic" List="PMingLiU,SimSun,Gulim,Yu Gothic UI,Microsoft JhengHei UI,Microsoft
YaHei UI,Malgun Gothic" />
            <AddFontLink Font="MS UI Gothic" List="Microsoft Sans Serif,PMingLiU,SimSun,Gulim,Yu Gothic
UI,Microsoft JhengHei UI,Microsoft YaHei UI,Malgun Gothic" />
            <AddFontLink Font="MS Mincho" List="MingLiU,SimSun,Batang,Yu Gothic UI,Microsoft JhengHei UI,Microsoft
YaHei UI,Malgun Gothic" />
            <AddFontLink Font="MS PMincho" List="PMingLiU,SimSun,Batang,Yu Gothic UI,Microsoft JhengHei
UI,Microsoft YaHei UI,Malgun Gothic" />
            <AddFontLink Font="Batang" List="MS PMincho,PMingLiU,SimSun,Malgun Gothic,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="BatangChe" List="MS Mincho,MingLiU,SimSun,Malgun Gothic,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="Dotum" List="MS UI Gothic,PMingLiU,SimSun,Malgun Gothic,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="DotumChe" List="MS Gothic,MingLiU,SimSun,Malgun Gothic,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="Gulim" List="Microsoft Sans Serif,MS UI Gothic,PMingLiU,SimSun,Malgun Gothic,Yu
Gothic UI,Microsoft JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="GulimChe" List="MS Gothic,MingLiU,SimSun,Malgun Gothic,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="Gungsuh" List="MS PMincho,PMingLiU,SimSun,Malgun Gothic,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="GungsuhChe" List="MS Mincho,MingLiU,SimSun,Malgun Gothic,Yu Gothic UI,Microsoft
JhengHei UI,Microsoft YaHei UI" />
            <AddFontLink Font="Malgun Gothic" List="Segoe UI,Gulim,Meiryo UI,Microsoft Jhenghei UI,Microsoft YaHei
UI,Yu Gothic UI" />
            <DefaultEudcFont File="EUDC.tte" />
            <AddEudcFont Font="MS UI Gothic" File="myEUDC1.tte" />
            <AddEudcFont Font="Meiryo" File="myEUDC2.tte" />
        </FontFactory>
    </Extensions>
</Configuration>
```

ActiveReports provides a powerful Font Resolver feature for all report types. It allows running ActiveReports on any environment without fonts

(like Red Hat OpenShift) and obtain WYSIWYG output on different environments. See the topic Custom Font Resolver and the sample FontResolver for the details on the implementation.

See PDF Font Settings for Section report in GDI topic for information on default and custom font settings required in PDF in the Medium trust environment.

Map Report Item and Map Tile Providers

The Map report item allows configuring built-in tile providers and extensions with custom report items. External providers in general have limited support to the resources. They require something like an external key to access without limits. ActiveReports allows configuring the following keys:

- Bing provider requires Bing account: https://www.microsoft.com/en-us/maps
- Google provider requires Google Developer account: https://developers.google.com/maps/get-started
- MapQuest provider requires MapQuest Developer Network account: https://developer.mapquest.com/documentation/open/

See the example below.

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
        <MapTileProviders>
            <MapTileProvider Name="MapQuest" DisplayName="MapQuest Tiles">
                <Settings>
                    <add key="ApiKey" value="Fmjtd%7Cluur21ua21%2C2x%3Do5-90t5h6" />
                    <add key="Timeout" value="5000" />
                </Settings>
            </MapTileProvider>
            <MapTileProvider Name="Google" DisplayName="Google Tiles">
                <Settings>
                    <add key="ApiKey" value="AIzaSyBdJ88HN7LTGkHHK5whfaVv8a5ozlx2E_k" />
                    <add key="Timeout" value="5000" />
                </Settings>
            </MapTileProvider>
            <MapTileProvider Name="Bing" DisplayName="Bing Tiles" >
                <Settings>
                    <add key="ApiKey" value="Al3SxHf0LOva-5GI5fOajO0LCuKjYKLarjWQFcdGiKfaJNRrE6SBbH9o3-HJDOX7" />
                    <add key="Timeout" value="5000" />
                </Settings>
            </MapTileProvider>
        </MapTileProviders>
    </Extensions>
</Configuration>
```

Custom Map Tile Providers

You can implement your own tile provider and register it in ActiveReports.

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
        <MapTileProviders>
            <MapTileProvider Name="MapQuest-Sample" DisplayName="MapQuest-Sample"
Type="GrapeCity.ActiveReports.Samples.CustomTileProviders.MapQuestTileProvider, CustomTileProviders,
Version=1.0.0.0">
                <Settings>
                    <add key="ApiKey" value="Fmjtd%7Cluur21ua21%2C2x%3Do5-90t5h6" />
                    <add key="Timeout" value="3000" />
```

```
            </Settings>
        </MapTileProvider>
    </MapTileProviders>
</Extensions>
</Configuration>
```

See the Custom Tile Provider sample.

Custom Data Providers

You can configure custom data providers by adding just a few lines in the configuration file.

SQLite Provider

The following example shows configuring SQLite data provider:

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
        <Data>
            <Extension Name="SQLITE" Type="System.Data.SQLite.SQLiteFactory, System.Data.SQLite" DisplayName="SQLite
Provider" />
        </Data>
    </Extensions>
</Configuration>
```

⚠ **Note:** System.Data.SQLite or System.Data.SQLite.Core NuGet package must be installed to be able to use Sqlite data.

CSV Provider

If you require custom UI, use **CommandTextEditorType** key as shown. The following example shows configuring a custom CSV data provider:

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
        <Data>
            <Extension Name="CustomCSV"
             DisplayName="CSV Data Provider"

Type="GrapeCity.ActiveReports.Samples.CustomDataProvider.CSVDataProvider.CsvDataProviderFactory,CustomDataProvider,
version=0.0.0.0, Culture=neutral"

CommandTextEditorType="GrapeCity.ActiveReports.Samples.CustomDataProviderUI.QueryEditor,CustomDataProviderUI,
Version=1.0.0.0, Culture=neutral"
            />
        </Data>
    </Extensions>
</Configuration>
```

Custom Data Providers with ADO.NET

To bind reports to ADO.NET Providers like SQLite, Oracle, and PostgreSQL, you need to add the following lines in configuration file.

MS SQLite Provider

The following example shows configuring SQLite data provider:

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
        <Data>
            <Extension Name="MSSQLITE" Type="Microsoft.Data.Sqlite.SqliteFactory, Microsoft.Data.Sqlite”
DisplayName="MS SQLite Provider" />
        </Data>
    </Extensions>
</Configuration>
```

⚠ **Note:** Following NuGet packages must be installed to be able to use MS SQLite data:

- System.Data.SQLite or System.Data.SQLite.Core
  and
- Microsoft.Data.Sqlite or Microsoft.Data.Sqlite.Core

After updating the configuration file like above, you can easily use SQLite through UI in the designer.

Note: The WebDesigner component requires some extra steps to specify the data providers on client-side. See the WebDesigner Custom Data Providers sample for details.

Oracle Provider

The following example shows configuring Oracle data provider:

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
        <Data>
            <Extension Name="ORACLE" Type="Oracle.ManagedDataAccess.Client.OracleClientFactory,
Oracle.ManagedDataAccess" DisplayName="Oracle Provider" />
        </Data>
    </Extensions>
</Configuration>
```

⚠ **Note:** Oracle.ManagedDataAccess or Oracle.ManagedDataAccess.Core NuGet package must be installed to be able to use Oracle data.

See the Oracle Data Provider sample for details.

PostgreSQL Provider

The following example shows configuring PostgreSQL data provider:

ActiveReports.config file

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
    <Extensions>
        <Data>
            <Extension Name="POSTGRESQL" Type="Npgsql.NpgsqlFactory, Npgsql" DisplayName="PostgreSQL Provider" />
        </Data>
    </Extensions>
</Configuration>
```

⚠ **Note:** Npgsql (last for .NET 4) or Npgsql NuGet package must be installed to be able to use PostgreSQL data.

Prevent Report Crashing

You can disable the report validation in WinViewer and End User Designer, which is performed to prevent report crashing. By default, the report validation is enabled.

ActiveReports.config file

```
<add key="SkipReportValidation" value="true"/>
```

To disable the report validation in WebViewer, the configuration file has to be specified in the settings in **UseReportViewer ('UseReportViewer Method' in the on-line documentation)** method.

ActiveReports.config file

```
app.UseReportViewer(settings =>
{
//...
settings.UseConfig("C:\\Program Files (x86)\\MESCIUS\\ActiveReports 18\\ActiveReports.config");
});
```

PDF Font Settings for Section report in GDI

Learn about the PDF export behavior for fonts in the legacy GDI compatibility mode.

> **Note**: The section report PDF export does not support font factory settings. Instead, you can configure fonts using APIs in DsPdf and DsImaging.

Default PDF Font Settings

Font linking helps resolve the situation when fonts on a deployment machine do not have the glyphs that are used in a development environment. When you find such a glyph mismatch, there is a possibility that the PDF output on development and deployment machines may be different.

In order to resolve this issue, the PDF export filter or the PDF rendering extension looks for the missing glyphs in the installed fonts as follows:

- Checks the system font link settings for each font that is used in the report. 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, searches for the glyphs in the fonts declared in the **FontFallback Property (on-line documentation)**.
- Uses glyphs from the font links collection to replace fonts that do not have their own declared linked fonts.
- If necessary glyphs are not found by font links or in the fonts declared in the FontFallback property, glyphs from Microsoft Sans Serif font are taken as the predefined font.

Custom PDF Font Settings

Typically, when using PDF export filters and drawing extensions in a Medium trust level environment or Azure web application, ActiveReports does not have access to the System Fonts folder due to security restrictions. Therefore, 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.

The ActiveReports custom font factory allows you to embed any fonts you need in PDF in the Medium trust

environment.

ActiveReports looks for the custom fonts in the order as follows:

1. A font specified in the control.
2. A mapped specified font in the **Substitute** setting.
3. A font specified in the the **AddFontLink** setting (Pro Edition only).
4. A font specified in the **SetFallbackFont** setting (Pro Edition only).

Notice that you need to copy the required font files (.ttc, .ttf) manually into the font folder you are accessing.

Custom Font Factory in Windows Azure

For your Azure project, you need to 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**.

Custom Font Configuration Settings

EUDC configuration settings

Element	Description
DefaultEUDCFont	This is the node, like the SystemDefaultEudcFont entry in the registry file, that contains the default EUDC font settings. It specifies a default unique EUDC font for all fonts. It is an absolute path or a relative to the first one, found in the AddFolder setting. The same role as the "SystemDefaultEudcFont" entry in the registry - see https://learn.microsoft.com/en-us/windows/win32/intl/eudc.
DefaultEUDCFont File	Specifies the file name of the default EUDC file.
AddEudcFont	This is the node that associates the EUDC file and the fontname. This node can be added more than once. See https://learn.microsoft.com/en-us/windows/win32/intl/eudc.
AddEudcFont Font	Specifies the font name.
AddEudcFont File	Specifies the file name of the EUDC file to associate the above font.

FontFactory

This is the main font factory node to which you can add fonts.

Attributes

Element	Description
Mode	Setting the **Mode** attribute to **File** allows to use a file based factory, or remove the attribute for a Windows GDI factory.

Child Elements

None.

Parent Elements

Element	Description
ActiveReports.PdfExport	The assembly that contains the PdfExport namespace (PDF export, document options, and security classes).

Example

```
<FontFactory Mode="File">
```

AddFolder

Adds all TrueType fonts (.ttc, .ttf) from the specified folder.

Attributes

Element	Description
Path	Specifies the absolute path to the folder.
VirtualPath	Specifies the relative path to the folder.
Recurse	When set to **True**, it can read the subfolder. When set to **False**, it cannot read the subfolder.

Child Elements

None.

Parent Elements

Element	Description
FontFactory	This is the main font factory node to which you can add fonts.

Example

```
<AddFolder VirtualPath="~/Fonts" Recurse="true"/>
```

Substitute

Maps an alternate name of fonts to their official names.

Attributes

Element	Description
Font	Specifies the abbreviated font name (e.g. "Helv").

To	Specifies the official font name (e.g. "Helvetica").

Child Elements

None.

Parent Elements

Element	Description
FontFactory	This is the main font factory node to which you can add fonts.

Example

```
<Substitute Font="Helv" To="Helvetica"/>
```

SetFallbackFont (Professional Edition only)

In the Professional Edition, sets the font to use if a) the specified font is not installed, b) the **Substitute** font is not specified or not installed, or c) the font links are not set or the needed glyphs are not found in the **AddFontLink** setting. **Substitute** adds a font substitution entry. See https://learn.microsoft.com/en-us/globalization/fonts-layout/fonts and HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\FontSubstitutes registry key.

Attributes

Element	Description
Font	Specifies the font name.

Child Elements

None.

Parent Elements

Element	Description
FontFactory	This is the main font factory node to which you can add fonts.

Example

```
<SetFallbackFont Font="Arial"/>
```

AddFontLink (Professional Edition only)

In the Professional Edition, there is the 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.

Attributes

Element	Description
Font	Specifies the font used in the reports.
List	Specifies the comma-separated list of fonts to be used in case of missing glyphs, specified in the Font attribute. See https://learn.microsoft.com/en-us/globalization/fonts-layout/fonts and HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\FontLink\SystemLink registry key. ⚠ **Caution:** The character style of the link won't be outputted in case the alternate font file does not exist in the specified folder of the **AddFolder** setting.
IsDefault	When set to **True**, indicates to use the specified list for any fonts that do not have their own font links.

Child Elements

None.

Parent Elements

Element	Description
FontFactory	This is the main font factory node to which you can add fonts.

Example

```
<AddFontLink Font="Tahoma" List="MS UI Gothic,SimSun,gulim,PMingLiU"/>
```

Other Customization Options

A few additional customizations that can be performed by DevOps are as described.

Windows Forms Viewer

The Windows Forms control allows the Window Forms-related customization. See Windows Forms Viewer topic and the Win Viewer sample for details.

WPF Viewer

You can customize the WPF Viewer by applying themes. See WPF Viewer topic and the WPF Viewer sample for details.

JS Viewer

See the JS Viewer API to learn about the customization possibilities of the JS Viewer.

For example, you can change the viewer toolbar in several ways:

- Add a custom toolbar item (or button)
- Remove an existing item from the toolbar
- Change visible toolbar items and their order

You can also change the position of the sidebar panes such as search, parameters, etc.

Windows Forms Designer

You can turn on/off almost all elements of the Report Explorer and script/preview tabs.

WebDesigner

See the WebDesigner API to learn about the customization possibilities of the Web Viewer.

You can change the visibility of many designer components, such as group editor, report explorer, toolbox, tabs, buttons, and more.

Localization

In ActiveReports, you can localize your application's UI to the supported languages: English (default), Japanese, and simplified Chinese. You can also perform custom localization into any language you need. Web components contain JSON files with English resource strings, which should be translated and then applied to the component, with the corresponding language specified using an API. .NET application localizations require resources to be translated and then satellite assemblies to be built.

MESCIUS may, from time to time and with 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.

.NET Localization

ActiveReports uses the Hub and Spoke model for localizing resources. The hub is the main executing assembly and the spokes are the satellite DLLs that contain localized resources for the application.

For example, if you want to localize the Viewer Control, the hub is MESCIUS.ActiveReports.Viewer.Win.dll and the spoke is MESCIUS.ActiveReports.Viewer.Win.resources.dll.

In your Program Files folder, the Localization folder is in a path like **....\MESCIUS\ActiveReports 18\Localization**, and contains all of the ActiveReports components that you can localize.

There are sixteen ActiveReports components in the Localization folder and most have two files.

- A **.bat** file that is used to set the Cultures to which you want to localize.
- A **.zip** file that contains the resource files (**.resx**) in which you update or change the strings.

There is one application in the Localization folder: **NameCompleter.exe**. When you run your .bat file after updating your culture, it runs this application to create a SatelliteAssembly folder with a language sub-folder containing the localized MESCIUS.ActiveReports.*AssemblyName*.resources.dll file.

Place the language folder containing the *.resources.dll file inside your main executing assembly folder to implement changes.

> **Note**: Before you can distribute or put your localization in the Global Assembly Cache (GAC), you must first send the localized MESCIUS.ActiveReports.*AssemblyName*.resources.dll file to Support Team by opening a support ticket, and get it signed with a strong name. Once you receive the signed DLL file, you can drag the language subfolder with the signed DLL file into C:\WINDOWS\ASSEMBLY, or distribute it with your solution.

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 run time sets the CurrentUICulture property or you can set it in code to force a certain UI Culture so that you can 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 ("en-US") is the default culture for ActiveReports.

> **Tip**: 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.

Js Components Localization

The NPM packages contain files for the localization of JavaScript components, namely, WebDesigner and Js Viewer. See the topics on WebDesigner Localization and Js Viewer Localization that explain this in detail.

WebDesigner Localization

WebDesigner automatically sets the UI language the same as that of the browser for the localizations: English, Japanese, and simplified Chinese. This article discusses setting a web designer UI language different from the browser for the supported localizations and adding custom localization for the languages that are not explicitly supported.

Set WebDesigner UI Language

To specify the UI language of WebDesigner, specify the language API while initializing the component. The supported values are 'en-US' (for English), 'ja-JP' (for Japanese), and 'zh-CN' (for simplified Chinese).

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <!-- get these files from the installation location for the package:
\node_modules\@mescius\activereportsnet-designer\dist -->
    <link rel="stylesheet" href="web-designer.css" />
    <script src="web-designer.js"></script>
</head>
<body>
    <div id="ar-web-designer" class="ar-web-designer">
    </div>
    <script>
        GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
            language: 'ja-JP'
        });
    </script>
```

```
</body>
</html>
```

Add Custom Localization using Resource Strings

With the WebDesigner NPM Package, you can access the resource strings from the **custom-resources.json** file available in the **activereportsnet-designer\dist\docs** folder of the package. In the 'custom-resources.json' file, you will find the resources stored in the file as shown:

Structure of custom-resources.json

```
{
   "ns": "bundle namespace",
   "lng": "bundle language",
   "resources":  Record<string,any>
}
```

To add custom localization to the web designer UI,

1. Copy the **custom-resources.json** file to the 'wwwroot' folder and rename the file to, say, 'fr-resources.json'.
2. Open 'fr-resources.json' file and translate the strings from the ‘resources’ node to a language, say French. Also, set the ‘lng’ node with an appropriate bundle language.
   For example, here we are localizing the WebDesigner component for French language, so the strings must be translated to French language and 'lng' must be set to 'fr-FR'.

   > **Note**: In web designer, resources should be assigned to their specific namespace, so do not modify the ‘ns’ node when translating resources.

   Sample fr-resources.json

```
[
   {
     "ns": "components",
     "lng": "fr-FR",
     "resources": {
       "appBar": {
         "btnPreview": "Aperçu",
         "btnFile": "Déposer",
         "btnHome": "Maison",
         "btnInsert": "Insérer",
         "textUnsavedChanges": "Modifications non enregistrées",
         "btnParameters": "Paramètres"
       },
       "menu": {
         "btnClose": "Fermer",
         "titlePin": "Broche",
         "btnLayerList": "Couches",
         "btnGroupEditor": "Éditeur de groupe",
         "btnReportExplorer": "Explorateur"
       }
```

```
    }
  }
]
```

> **Note:** As you may notice,
> • We have modified only the resource values and left the bundle namespace (e.g., components) and resource section (e.g., appBar, menu) untouched.
> • Some of the localization strings are skipped in the above example. The values for which localization strings are missing are by default displayed using EN localization data.

3. Now that we have the translated resource bundles in the 'fr-resources.json' file, load this file and use addLanguage() and language APIs to localize the web designer. .

Index.html

```
<!DOCTYPE html>
<html>
<head>
 <!--get these files from the installation location for the package:
\node_modules\@mescius\activereportsnet-designer\dist-->
    <link rel="stylesheet" href="web-designer.css" />
    <script src="web-designer.js"></script>
</head>
<body>
    <div id="ar-web-designer" class="ar-web-designer">
    </div>
    <script>
        fetch("./fr-resources.json").then(res => res.json()).then((res) => {
            var languageResource = res;
            GrapeCity.ActiveReports.Designer.addLanguage('fr-FR', languageResource);
            GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
                language: 'fr-FR'
            });
        });
    </script>
</body>
</html>
```

An alternate approach is to pass the translated resource strings from **custom-resources.json** directly as the JSON object, as shown, and use addLanguage() and language APIs to localize the web designer.

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <!--get these files from the installation location for the package:
\node_modules\@mescius\activereportsnet-designer\dist-->
```

```
    <link rel="stylesheet" href="web-designer.css" />
    <script src="web-designer.js"></script>
</head>
<body>
    <div id="ar-web-designer" class="ar-web-designer">
    </div>
    <script>
        var resourceBundles = [
            {
                "ns": "components",
                "lng": "fr-FR",
                "resources": {
                    "appBar": {
                        "btnPreview": "Aperçu",
                        "btnFile": "Déposer",
                        "btnHome": "Maison",
                        "btnInsert": "Insérer",
                        "textUnsavedChanges": "Modifications non enregistrées",
                        "btnParameters": "Paramètres"
                    },
                    "menu": {
                        "btnClose": "Fermer",
                        "titlePin": "Broche",
                        "btnLayerList": "Couches",
                        "btnGroupEditor": "Éditeur de groupe",
                        "btnReportExplorer": "Explorateur"
                    }
                }
            }
        ];
        GrapeCity.ActiveReports.Designer.addLanguage('fr-FR', resourceBundles);
        GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
            language: 'fr-FR'
            });
    </script>
</body>
</html>
```

Js Viewer Localization

Js Viewer supports English, Japanese, and Chinese localizations for which the localized files are already available. This article discusses setting a Js Viewer UI language from the supported localizations and adding custom localization for the languages that are not explicitly supported.

Set Js Viewer UI Language

To set the UI language of Js Viewer, specify the locale API while initializing the component. The supported values are 'en-US' (for English), 'ja-JP' (for Japanese), and 'zh-CN' (for simplified Chinese).

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
    <script src="jsViewer.min.js"></script>
</head>
<body onload="loadViewer()">
    <div id="viewerContainer">
    </div>
    <script>
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                locale: 'ja-JP'
            });
            viewer.openReport("Report.rdlx");
        }
    </script>
</body>
</html>
```

Add Custom Localization using Resource Strings

With the Js Viewer NPM package, you can access the resource strings from the **custom-locale.json** file available in the **activereportsnet-viewer\dist** folder of the package. By default, the custom-locale.json file contains all the strings in English that are required to localize the viewer.

To add custom localization to the Js Viewer UI,

1. Copy the 'custom-locale.json' file to the 'wwwroot' folder and rename the file to, say, 'fr-locale.json'.
2. Open 'fr-locale.json' file and translate the strings in the file to any language, say French. For example, here we are localizing the Js Viewer component for French language, so the strings must be translated to French language.

   Sample fr-locale.json

   ```
   {
      "viewer": {
        "toolbar": {
          "refresh": "Rafraîchir",
          "cancel": "Annuler"
        },
   ```

```
    "search": {
      "start-search-btn": "Recherche",
      "paneltitle": "Recherche"
    }
  }
}
```

3. Specify the URL of 'fr-locale.json' file using the localeUri API to add localization to the viewer.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
    <script src="jsViewer.min.js"></script>
</head>
<body onload="loadViewer()">
    <div id="viewerContainer">
    </div>
    <script>
     let viewer;
     function loadViewer() {
       viewer = GrapeCity.ActiveReports.JSViewer.create({
           element: '#viewerContainer',
           localeUri: './fr-locale.json'
       });
         viewer.openReport("Report.rdlx");
     }
   </script>
 </body>
 </html>
```

> **Note**: As you may notice, some of the localization strings are skipped in the above example. The values for which localization strings are missing are by default displayed using EN localization data.

An alternate approach is to use the localeData API to directly assign the localized JSON to the viewer, as shown.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
    <script src="jsViewer.min.js"></script>
</head>
<body onload="loadViewer()">
```

```
    <div id="viewerContainer">
    </div>
    <script>
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                localeData: {
                    "viewer": {
                        "toolbar": {
                            "refresh": "Rafraîchir",
                            "cancel": "Annuler"
                        },
                        "search": {
                            "start-search-btn": "Recherche",
                            "paneltitle": "Recherche"
                        }
                    }
                }
            });
            viewer.openReport("Report.rdlx");
        }
    </script>
</body>
</html>
```

WebViewer Localization

This article discusses setting the WebViewer UI language.

The localization file for WebViewer is **ar-webviewer-locale.json**. This file is automatically added to the project in the **grapecity.activereports.web\content** folder when you install the MESCIUS.ActiveReports.Web NuGet package, which is required when using WebViewer. The 'ar-webviewer-locale.json' file contains English localization so that the user can use it as a template for their own localization.

To localize WebViewer, first translate all the resources in the 'ar-webviewer-locale.json' file. Once you have translated all the resources, you must specify the location of the 'ar-webviewer-locale.json' file in the WebViewer's **LocalizationJson ('LocalizationJson Property' in the on-line documentation)** property.

Default. aspx. cs

```
WebViewer1.LocalizationJson = File.ReadAllText(Server.MapPath("~") + " ar-webviewer-locale.json ");
```

Blazor Localization

The NuGet packages contain files for the localization of Blazor components, namely, Blazor WebDesigner and Blazor Viewer. See the topics on Blazor WebDesigner Localization and Blazor Viewer Localization that explain this in detail.

Blazor WebDesigner Localization

Blazor WebDesigner automatically sets the UI language the same as that of the browser for the localizations: English, Japanese, and simplified Chinese. This article discusses setting a blazor webdesigner UI language different from the browser for the supported localizations and adding custom localization for the languages that are not explicitly supported.

Set Blazor WebDesigner UI Language

To set the UI language of Blazor WebDesigner, specify the Language API with-in the div tags while initializing the component. The supported values are 'en-US' (for English), 'ja-JP' (for Japanese), and 'zh-CN' (for simplified Chinese).

Index.razor

```
@page "/"
@inject IJSRuntime JSRuntime
<PageTitle>Index</PageTitle>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" PreviewSettings=@_preview Language="ja-JP" />
</div>
@code {
    private ReportDesigner _designer;
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
            {
                CanPreview = false
            };
    }
}
```

Check the complete HTML code provided below. Note that the code does not localize the viewer, see Blazor Viewer Localization for more information.

Add Custom Localization using Resource Strings

The localization file for Blazor WebDesigner is **custom-resources.json**. This file is automatically added to the project in the **grapecity.activereports.blazor.designer\staticwebassets\docs** folder when you install the MESCIUS.ActiveReports.Blazor.Designer NuGet package, which is required when using Blazor WebDesigner. In the 'custom-resources.json' file, you will find the resources stored in the file as shown:

Structure of custom-resources.json

```
{
   "ns": "bundle namespace",
   "lng": "bundle language",
   "resources":  Record<string,any>
```

```
}
```

To add custom localization to the blazor web designer UI,

1. Copy the **custom-resources.json** file to the 'wwwroot' folder and rename the file to, say, 'fr-resources.json'.
2. Open 'fr-resources.json' file and translate the strings from the ‘resources’ node to a language, say French. Also, set the ‘lng’ node with an appropriate bundle language.
   For example, here we are localizing the WebDesigner component for French language, so the strings must be translated to French language and 'lng' must be set to 'fr-FR'.

   > **Note**: In web designer, resources should be assigned to their specific namespace, so do not modify the ‘ns’ node when translating resources.

   Sample fr-resources.json

   ```
   [
     {
       "ns": "components",
       "lng": "fr-FR",
       "resources": {
         "appBar": {
           "btnPreview": "Aperçu",
           "btnFile": "Déposer",
           "btnHome": "Maison",
           "btnInsert": "Insérer",
           "textUnsavedChanges": "Modifications non enregistrées",
           "btnParameters": "Paramètres"
         },
         "menu": {
           "btnClose": "Fermer",
           "titlePin": "Broche",
           "btnLayerList": "Couches",
           "btnGroupEditor": "Éditeur de groupe",
           "btnReportExplorer": "Explorateur"
         }
       }
     }
   ]
   ```

   > **Note:** As you may notice,
   > • We have modified only the resource values and left the bundle namespace (e.g., components) and resource section (e.g., appBar, menu) untouched.
   > • Some of the localization strings are skipped in the above example. The values for which localization strings are missing are by default displayed using EN localization data.

3. Now that we have the translated resource bundles in the 'fr-resources.json' file, load this file and use LocalizationResources and Language APIs to localize the web designer. Here is the complete HTML code provided below. Note that the code does not localize the viewer, see Blazor Viewer Localization for more information.

Complete Index.razor

```
@page "/"
@inject IJSRuntime JSRuntime
<PageTitle>Index</PageTitle>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" PreviewSettings=@_preview
LocalizationResources=@_localizationResources Language="fr"/>
</div>
@code {
    private ReportDesigner _designer;
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
        {
                CanPreview = false
        };
    }
    private LocalizationResources[] _localizationResources;

    protected override async Task OnInitializedAsync()
    {
        _localizationResources = new LocalizationResources[]{
            new LocalizationResources(){
                Language = "fr",
                Resources =(await JSRuntime.InvokeAsync<object>("getJsonData", new
object[] { "./fr-resources.json" })).ToString()
            }
        };
    }
}
```

Blazor Viewer Localization

Blazor Viewer supports English, Japanese, and Chinese localizations for which the localized files are already available. This article discusses setting a Blazor Viewer UI language from the supported localizations and adding custom localization for the languages that are not explicitly supported.

Set Blazor Viewer UI Language

To set the UI language of Blazor Viewer, specify the Locale API while initializing the component. The supported values are 'en-US' (for English), 'ja-JP' (for Japanese), and 'zh-CN' (for simplified Chinese).

Index.razor

```
<div id="viewerContainer">
     <ReportViewer @ref="_viewer" ReportName="@_currentReport" Locale="ja-JP" />
```

```
</div>
```

Add Custom Localization using Resource Strings

The localization file for Blazor Viewer is **ar-blazor-viewer-locale.json**. The file is added to the project in the **grapecity.activereports.blazor.viewer\content** folder when you install the MESCIUS.ActiveReports.Blazor.Viewer NuGet package, which is required when using Blazor Viewer. By default, the 'ar-blazor-viewer-locale.json' file contains all the strings in English that are required to localize the viewer.

To add custom localization to the Blazor Viewer UI,

1. Copy the 'ar-blazor-viewer-locale.json' file to the 'wwwroot' folder and rename the file to, say, 'ar-blazor-viewer-fr-locale.json'.
2. Open the file and translate the strings in the file to any language, say French. For example, here we are localizing the Blazor Viewer component for French language, so the strings must be translated to French language.

Sample ar-blazor-viewer-fr-locale .json

```
{
   "viewer": {
     "toolbar": {
       "refresh": "Rafraîchir",
       "cancel": "Annuler"
     },
     "search": {
       "start-search-btn": "Recherche",
       "paneltitle": "Recherche"
     }
   }
 }
```

3. Specify the URL of 'ar-blazor-viewer-fr-locale.json' file using the LocaleUri API to add localization to the viewer.

Index.razor

```
@page "/"
@using BlazorViewerServer.Data
@inject ReportsService ReportsService
<div class="main">
    <ReportList ReportsList="@reportsList" CurrentReport="@_currentReport"
OnClickCallback="OnClick"></ReportList>
    <div id="viewerContainer">
        <ReportViewer @ref="_viewer" ReportName="@_currentReport" LocaleUri="./ar-
blazor-viewer-fr-locale.json" />
    </div>
</div>
@code{
    private ReportViewer _viewer;
    private List<string> reportsList;
    private string _currentReport = null;
    protected override void OnInitialized()
```

```
    {
        reportsList = ReportsService.GetReports().ToList();
        _currentReport = reportsList.FirstOrDefault();
    }
    private async void OnClick(string res)
    {
        _currentReport = res;
        await _viewer.OpenReport(_currentReport);
    }
}
```

> **Note**: As you may notice, some of the localization strings are skipped in the above example. The values for which localization strings are missing are by default displayed using EN localization data.

An alternate approach is to use the LocaleData API to directly assign the localized JSON to the viewer, as shown.

Index.razor

```
@page "/"
@using BlazorViewerServer.Data
@inject ReportsService ReportsService
<div class="main">
    <ReportList ReportsList="@reportsList" CurrentReport="@_currentReport"
OnClickCallback="OnClick"></ReportList>
    <div id="viewerContainer">
        <ReportViewer @ref="_viewer" ReportName="@_currentReport" LocaleData="@_localeData" />
    </div>
</div>
@code{
    private ReportViewer _viewer;
    private List<string> reportsList;
    private string _currentReport = null;

    string _localeData = "{\"viewer\":{\"toolbar\":
{\"refresh\":\"Rafraîchir\",\"cancel\":\"Annuler\"},\"search\":{\"start-search-
btn\":\"Recherche\",\"paneltitle\":\"Recherche\"} } }";

    protected override void OnInitialized()
    {
        reportsList = ReportsService.GetReports().ToList();
        _currentReport = reportsList.FirstOrDefault();
    }
    private async void OnClick(string res)
    {
        _currentReport = res;
        await _viewer.OpenReport(_currentReport);
    }
}
```

Cultures

For your convenience, here is a list of predefined System.Globalization cultures. (Source: MSDN.) For ActiveReports localization purposes, use the Culture and Language Name value in the first column.

Culture and 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	0x2C0A	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	0x1C0A	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	0x4C0A	Spanish (Nicaragua)
es-PA	0x180A	Spanish (Panama)
es-PY	0x3C0A	Spanish (Paraguay)
es-PE	0x280A	Spanish (Peru)
es-PR	0x500A	Spanish (Puerto Rico)
es-ES	0x0C0A	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)

Deployment

ActiveReports .NET can be deployed to any supported environment. Depending on the choice of environment and the specific scenario that needs to be addressed, you can choose the environment.

To deploy ActiveReports successfully, you need to have an idea about the following details related to Licensing and GAC.

- Licensing: Each environment requires its licensing scenario. Refer to the License ActiveReports topic for details.
- GAC (Global Assembly Cache): ActiveReports supports installation to the GAC. However, you should implement it manually as no special scripts are provided.

ActiveReports can be deployed in the following supported environments:

- **Windows apps**

  Learn how to deploy ActiveReports Windows projects to your system.

- **Web apps**

  Learn how to deploy ActiveReports Web projects to your Web server.

- **Non-Windows environment**

  Learn how to deploy applications across non-Windows environments Mac and Linux.

- **Cloud services**

Learn how to deploy applications across cloud services such as Azure App Service, Amazon Web Services, and Google Cloud services.

- **Using Docker on Windows environment**

  Learn how to deploy applications using Docker on Windows.

Deploy Windows Application

Before deploying a Windows application, for the report files, change the **Copy to Output Directory** property to **Copy always** from the Properties window.

It is also good to be sure that all of the packages you need for your reports are included.

XCopy Deployment

1. Open your project in Visual Studio, and set the **Solution Configuration** to 'Release'.
2. From the **Build** menu, select **Build Solution**.
3. In File Explorer, navigate to the project's bin directory, and copy all the files from the **Release** folder into a zip file.
4. Distribute the zip file.

MSI Installer Deployment

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**.
4. In the **Add a new project** dialog, select **Setup Project**, rename the file if you want and click **Create**. The **ProjectName** that you enter determines the name that is displayed for the application in folder names.
5. In the **File System** editor that appears, under File System on Target Machine, select the **Application Folder**.

   > **Note**: To show the File System editor at any time, drop down the **View** menu and select **Editor**, then **File System**.

6. Right-click **Application Folder** and from the context menu, select **Add**, then **Project Output**.
7. In the **Add Project Output Group** dialog that appears, choose your ActiveReports project name from the drop-down list.
8. In the list, select **Primary output** and click **OK**. This adds all of the existing package dependencies to your project.

   > **Note**: If you would rather use the ActiveReports .msm file, please contact our technical support team.

9. In the **Select Component** dialog that appears, select the components that you want to add and click the **OK** button.
10. From the Visual Studio **Build** menu, select **Build *YourInstallerProjectName*** to build your Installer project.

Deploy the installer application

1. Right-click the Installer project in the Solution Explorer and select **Install**.
2. 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 Application

Follow this guide to deploy ActiveReports Web projects to your Web server. See the article from MSDN for more details.

To deploy ActiveReports Web projects, you must have access to the Microsoft .NET Framework version 4.6.2 or higher and the coordinating version of ASP.NET, or ASP.NET Core with .NET Core 3.1 and above. You must also have access to Internet Information Services version 8 or higher, and you need administrative access to the server.

It is also good to be sure that all of the packages you need for your reports are included.

Install prerequisites on the server

Follow Microsoft's instructions to install each of the following on your Web server:

- The Microsoft .NET Framework version 4.6.2 or higher
- ASP.NET version 4.6.2 or higher (must be the same version as the Framework)
- ASP.NET Core with .NET Core 3.1 and above.
- Internet Information Services (IIS) version 8

Map your application to a virtual directory

1. Make sure your project is deployed to a virtual directory in IIS.
2. To map requests from the page with the WebViewer control to your virtual directory, paste the **<base>** tag with a virtual directory name inside the **<head>** tag of the page containing the WebViewer control. In this case, all requests to back end assemblies will include the specified virtual directory name. For example,

```
<head>
 <base href="/VirtDirName/">
</head>
```

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 (e.g. PDF or RDF exports) to a folder on Windows machines, the **ASPNET** user ID needs **Write** access to that folder.
- Windows 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 (e.g. SQL, Access, etc.) assign **Read** access to it.

You can also deploy **Report Parts** by placing your Page and RDLX reports with report parts to the report items library. The path to this library is set in the **ReportPartsDirectory** property of the ActiveReports.config file; see Configure ActiveReports topic.

Report Parts is a Professional Edition feature that provides you with the possibility to reuse report parts (groups of controls with data and settings) from one report in other reports. For example, you can use Chart and Tablix data regions from Report1, and a Table data region from Report2 to create a new report.

Deploy Applications in Non-Windows Environment

This page discusses on how to deploy applications across non-Windows environment such as Mac and Linux.

Mac

This page demonstrates how you can deploy your ActiveReports application on Mac locally.

Prerequisites

This tutorial uses the .Net 6 SDK to run and deploy the JSViewer_MVC_Core sample application locally. You can download the sample from our GitHub repository: https://github.com/activereports/WebSamples18/tree/main/JSViewer_MVC_Core

We will also be using some sample reports to that can downloaded from the following link:

https://github.com/activereports/WebSamples18/tree/main/JsViewerReports

Requirements

- .Net .6 SDK (x64)
- Visual Studio 2019 or later

Build and Run

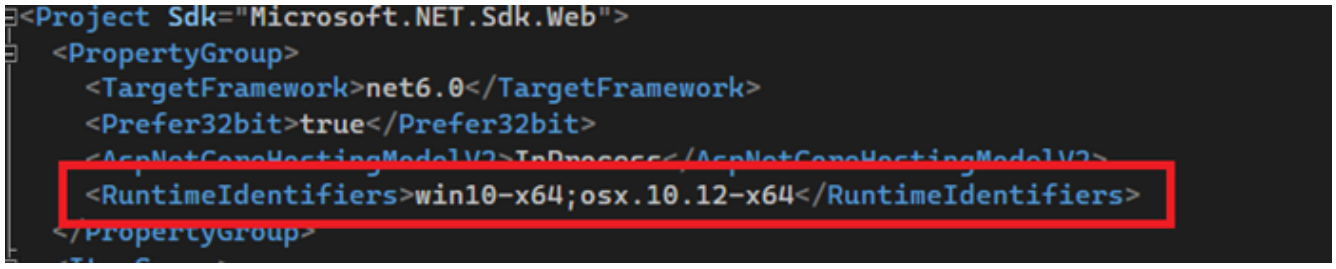
1. Open the JSViewer_MVC_Core project in Visual Studio.
2. Restore NuGet packages.
    1. For this expand the Dependencies option in the Solution Explorer.
    2. Right-click on the NuGet and select Restore in the context menu.
3. Add JsViewerReport folder that we have downloaded previously in your project.
    1. Right-click on the Project in Solution Explorer.
    2. Click on Add > Add Existing Folder
    3. Select the JSViewerReports.
4. Build and run the application.

Publish the application

Once you have confirmed that the application is running on your system, we are ready to Publish the application. Here are the steps to publish the macOS application.

1. In Visual Studio, right-click the project file of the JSViewer_MVC_Core and click Edit Project to edit the project file
2. Now add a new Runtime Identifier to the file and save it, just as highlighted here:

```
<RuntimeIdentifiers>win10-x64;osx.10.12-x64</RuntimeIdentifiers>
```



3. Now right-click the project file again and click Publish>Publish to Folder...
4. Set the publish location as “bin>release>6.0>publish” and click OK.

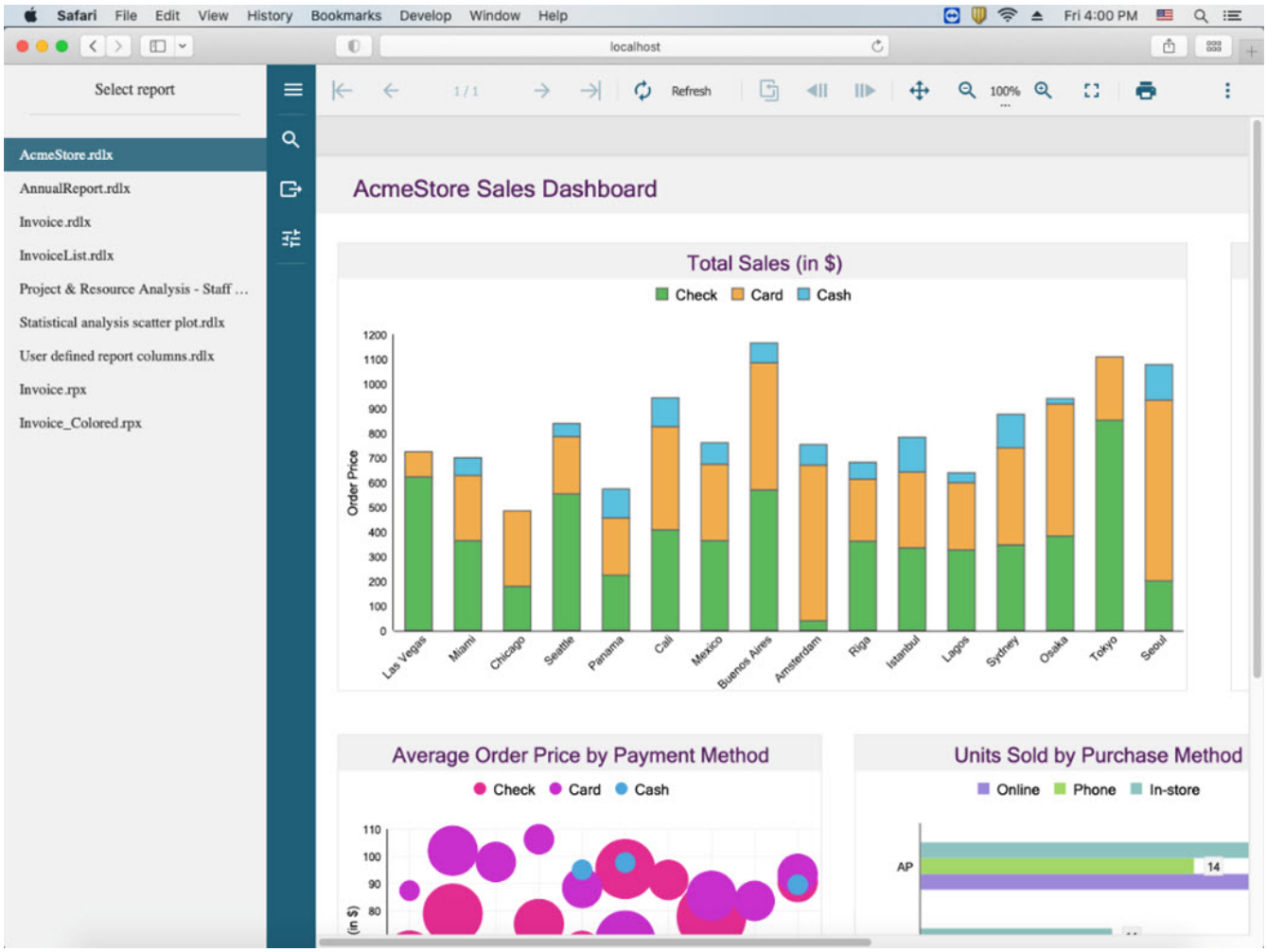
Running the applications on macOS

Now that we have published our application, we can run the application on macOS. The process is very similar as the process on Windows.
Now to run the self-contained app on macOS:

1. To run the JSViewer_MVC_Core application, we need to grant the executable permissions to run.
    1. First, Go to the bin/release/6.0 folder.
    2. Right-click on the Publish Folder and Select Services in the context menu and select New Terminal at Folder.
2. Type in sudo chmod +x JSViewer_MVC_Core and hit enter.
3. Type in the password of your admin account and hit enter to grant the executable permission to run.
4. Now type in open JSViewer_MVC_Core and hit enter
5. The app now runs in a new terminal window and the following message is displayed:

```
Hosting environment: Production
Content root path: /Users/gpctadmin
Now listening on: http://localhost:5000
Now listening on: https://localhost:5001
Application started. Press Ctrl+C to shut down.
```

6. Now you can open the application on following URL in your browser: http://localhost:5000



Linux

ActiveReports supports Linux environment with only .NET Core environment installed. WinForms and WPF controls are not supported.

> Important: You should configure the Font Resolver (with config file or through code) to obtain consistent result. Check Troubleshooting to fix the missing fonts issue.

Publish your .Net Core application

Once you have created a .Net core application on Visual Studio, open the project and follow the steps to publish the application.

1. Right-click on your project.
2. Click on Publish button.
3. Now create a new publish profile, and browse the folder where you want to publish your project dll.
4. Click on Publish so it will create your dll in the folder.

Install required .Net Module on Linux

Now we have the web application dll and we need to host it on the Linux environment. .Net applications run on Kestrel servers and we run Apache or Nginx server in Linux environments, which acts as a proxy server and handles the traffic from outside the machine and redirects it to the Kestrel server so we will have Apache or Nginx server as the middle layer.

In this article, we will use Apache as a proxy server.

First, we would need to install the .Net core module in our Linux environment. For that run the following commands,

- `sudo apt-get update`
- `sudo apt-get install apt-transport-https`
- `sudo apt-get update`
- `sudo apt-get install dotnet-sdk-3.1`
- `sudo apt-get install dotnet-runtime-3.1`
- `sudo apt-get install aspnetcore-runtime-3.1`

This will install all the required .Net packages

Install and configure Apache Server

Install the Apache server

- `sudo apt-get install apache2`
- `sudo a2enmod proxy proxy_http proxy_html proxy_wstunnel`
- `sudo a2enmod rewrite`

Next, we need to make a conf file to set up our proxy on Apache. Create the file by running the following command:

- `sudo nano /etc/apache2/conf-enabled/netcore.conf`

Now copy the following configuration in that file:

netcore.conf

```
<VirtualHost *:80>
   ServerName www.DOMAIN.COM
   ProxyPreserveHost On
   ProxyPass / http://127.0.0.1:5000/
   ProxyPassReverse / http://127.0.0.1:5000/
   RewriteEngine on
   RewriteCond %{HTTP:UPGRADE} ^WebSocket$ [NC]
   RewriteCond %{HTTP:CONNECTION} Upgrade$ [NC]
   RewriteRule /(.*) ws://127.0.0.1:5000$1 [P]
   ErrorLog /var/log/apache2/netcore-error.log
   CustomLog /var/log/apache2/netcore-access.log common
</VirtualHost>
```

The **<VirtualHost *:80>** tag defines the IP and port it will bind Apache so we will access our application from outside our Linux environment through this Ip:Port.

Now restart the Apache server using the following the commands:

- `sudo service apache2 restart`
- `sudo apachectl configtest`

Configure and Start Service

Move the dll to the defined path with the below command.

`"sudo cp -a ~/release/ /var/netcore/"`

Create a service file for our .Net application using the following command.

`"sudo nano /etc/systemd/system/ServiceFile.service"`

Copy the following configuration in that file and it will run our application,

```
[Unit]
Description=ASP .NET Web Application
[Service]
WorkingDirectory=/var/netcore
ExecStart=/usr/bin/dotnet /var/netcore/Application.dll
Restart=always
RestartSec=10
SyslogIdentifier=netcore-demo
User=www-data
Environment=ASPNETCORE_ENVIRONMENT=Production
```

```
[Install]
WantedBy=multi-user.target
```

In "ExecStart=/usr/bin/dotnet /var/netcore/**Application.dll**" line of code, replace **Application.dll** with your dll name that you want to run. For JSViewer MVC Core replace the **Application.dll** with **JSViewer_MVC_Core.dll**.

Now start the service. Instead of the service name in the below commands use the name of the file made above,

- `sudo systemctl enable {Service Name}`
- `sudo systemctl start {Service Name}`

Now your proxy server and kestrel server are running, and you can access your application through any ip with port 80.

To redeploy the code your need to replace the dll and stop and start your service again through the following commands,

- `sudo systemctl stop {Service Name}`
- `sudo systemctl start {Service Name}`

Deploy Applications to Cloud Services

Lets discuss ActiveReports deployment across cloud services such as Azure App Service, Amazon Web Services, and Google Cloud cloud services.

Azure App Service

ActiveReports can be deployed to Azure - it can be Virtual Machine, Azure Function, or Azure App Service. This page explains how to deploy (publish) a web project by using the Visual Studio. To deploy your web app, you must first create and configure a new App Service that you can publish your app to. See the MSDN documentation.

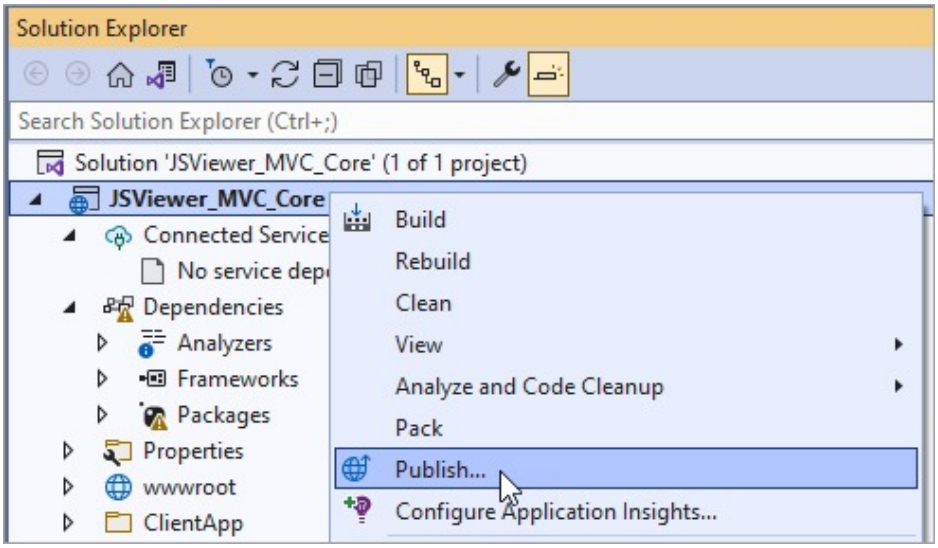
Deploy on Azure using a Windows Environment

In this tutorial, we will be deploying the JSViewer_MVC_Core sample on Azure using a Windows environment.

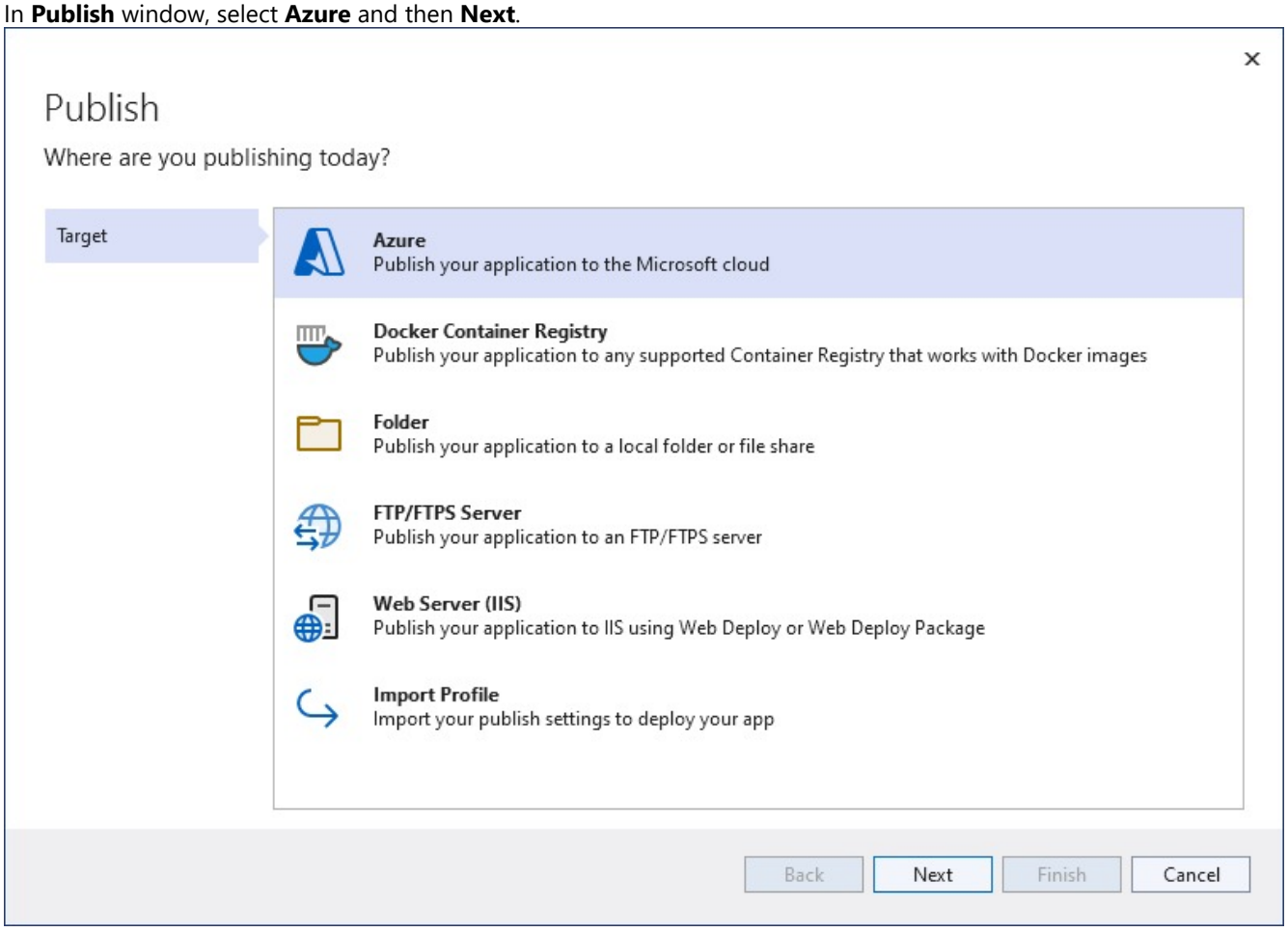
1. Download the JSViewer_MVC_Core sample from our WebSamples18 GitHub repository.
2. Open the application in Visual Studio and build and run the application.
3. Remove the **Reports** folder containing links the reports from the project since we will be embedding the reports in the application.
4. Now add a new folder and name it 'Reports'.
5. Add all the files from **JSViewerReports** in the **Reports** folder and set the **Build Action** as 'Embedded Resource'.

Follow these steps to create your App Service resources and publish your project. Let us use JSViewer_MVC_Core sample project for demonstration purpose.

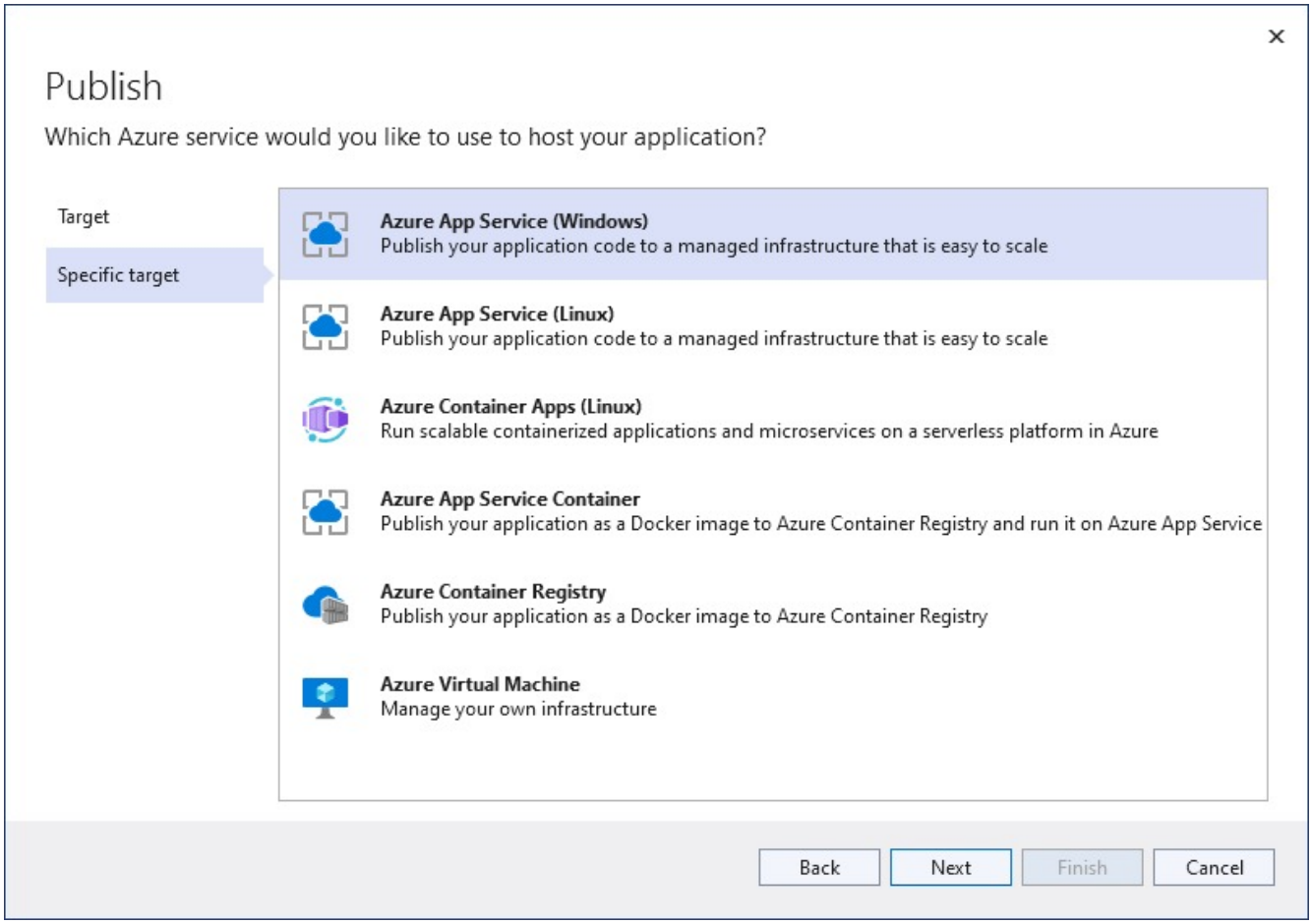
1. Open the JSViewer_MVC_Core sample project in Visual Studio 2022.
2. In the **Solution Explorer**, right-click the on your project name, and select **Publish**.



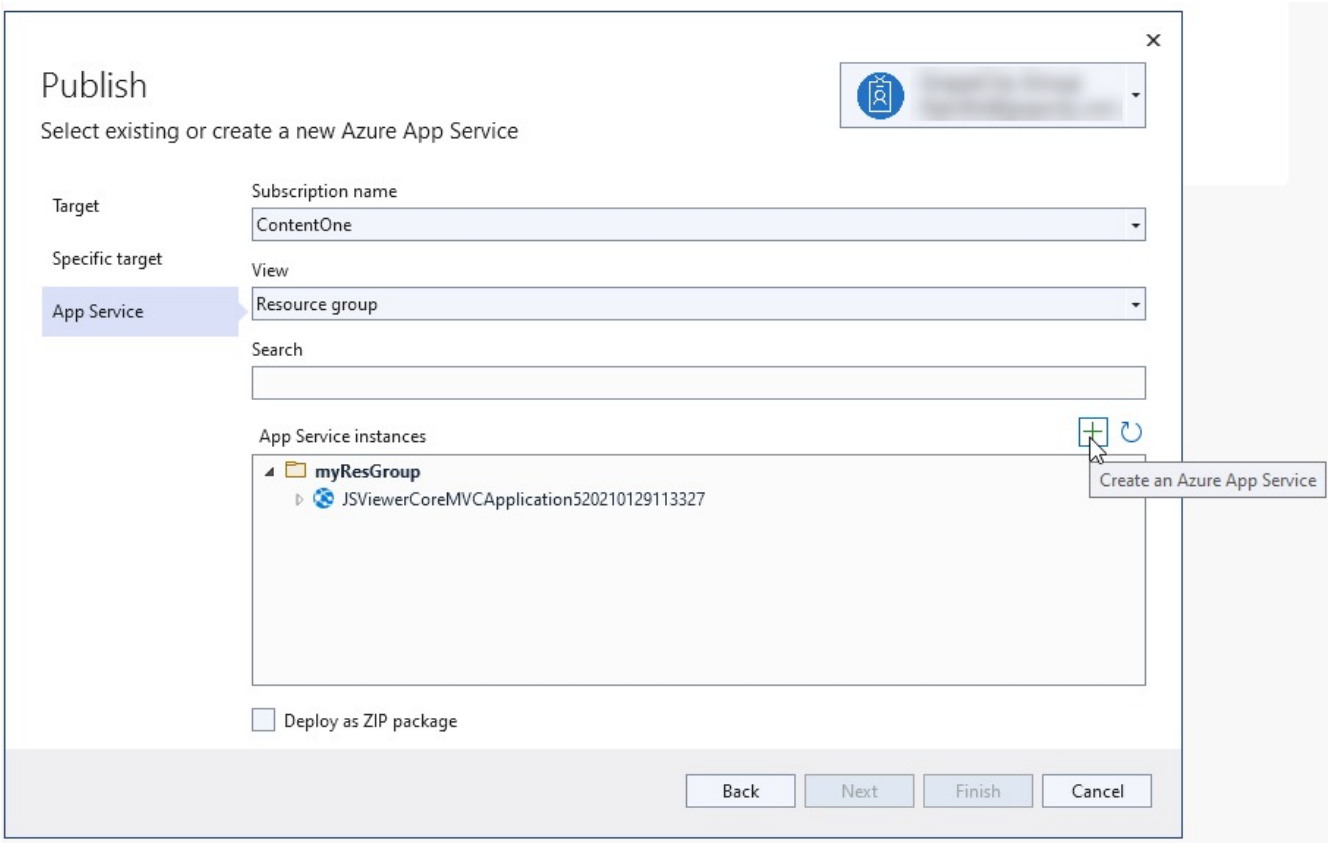
3. In **Publish** window, select **Azure** and then **Next**.



4. Choose the **Specific target** Azure App Service (Windows). Then, select **Next**.

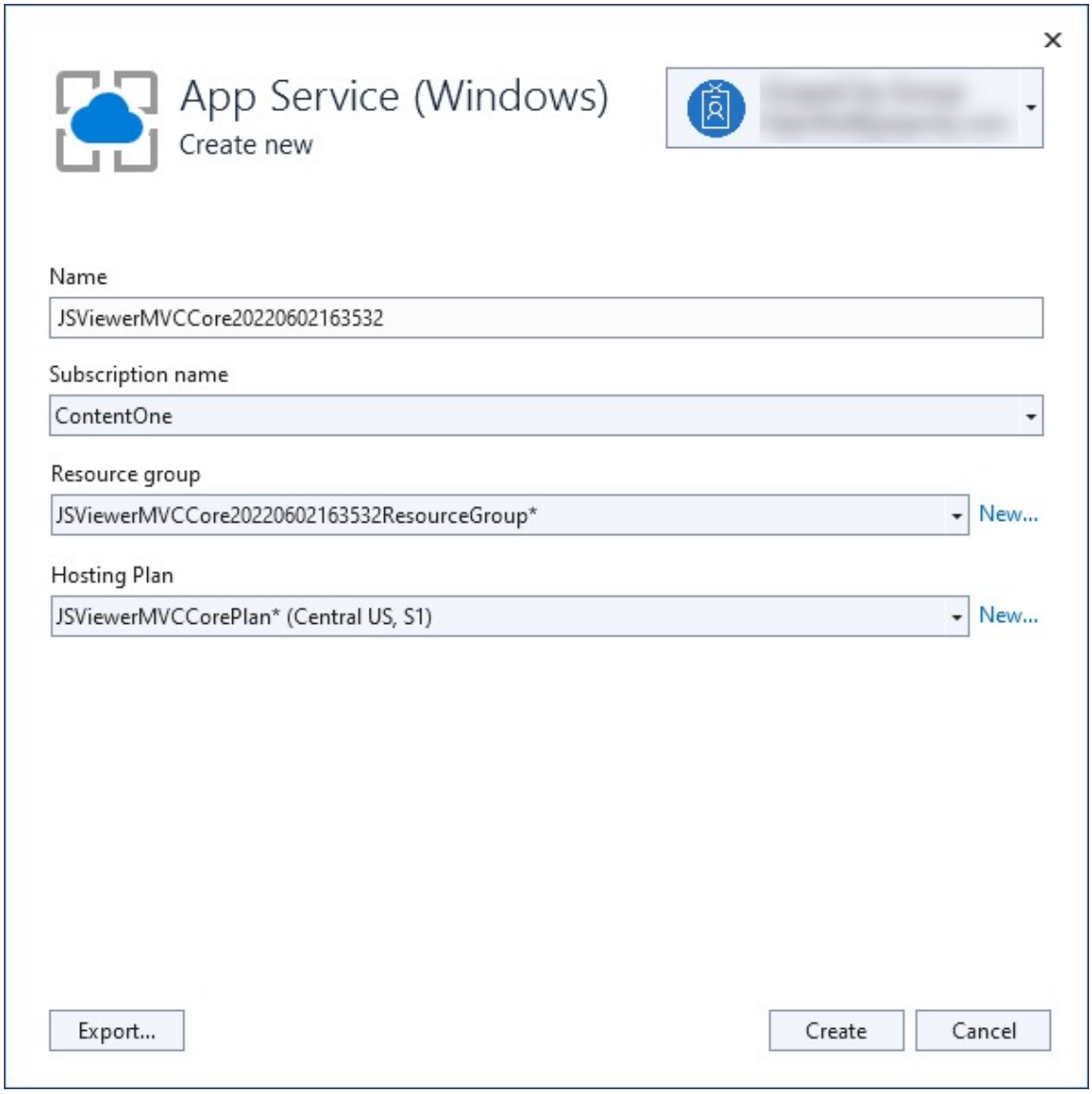


5. Your options depend on whether you're signed in to Azure already and whether you have a Visual Studio account linked to an Azure account. Select either Add an account or Sign in to sign in to your Azure subscription. If you're already signed in, select the account you want.
6. To the right of App Service instances, select +.

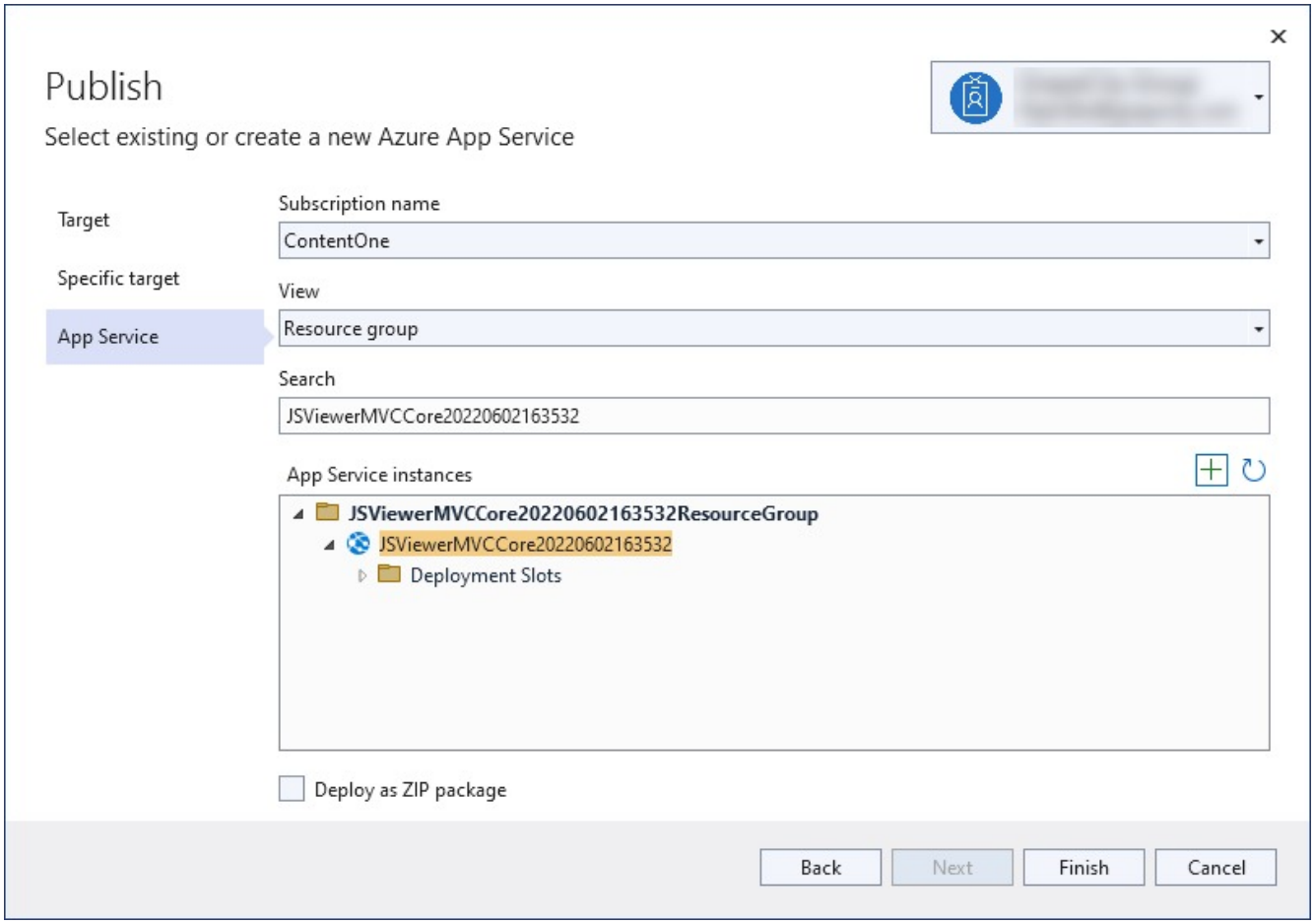


7. For **Subscription name**, accept the subscription that is listed or select a new one from the drop-down list.
8. For **Resource group**, select **New**. It will contain all of the Azure resources for the service.
9. In **New resource group name**, enter **JSViewerResourceGroup** and select OK.
10. For **Hosting Plan**, select **New**. It specifies the location, size, and features of the web server that hosts your app.
11. In the **Hosting Plan** dialog, enter the appropriate values:

    Hosting Plan: JSViewerMVCCorePlan
    Location: Central US
    Size: S1
12. In **Name** field, enter a unique app name that includes only the valid characters (a-z, A-Z, 0-9, and -). You can accept the automatically generated unique name. The URL of the web app is http://<app-name>.azurewebsites.net, where <app-name> is your app name.

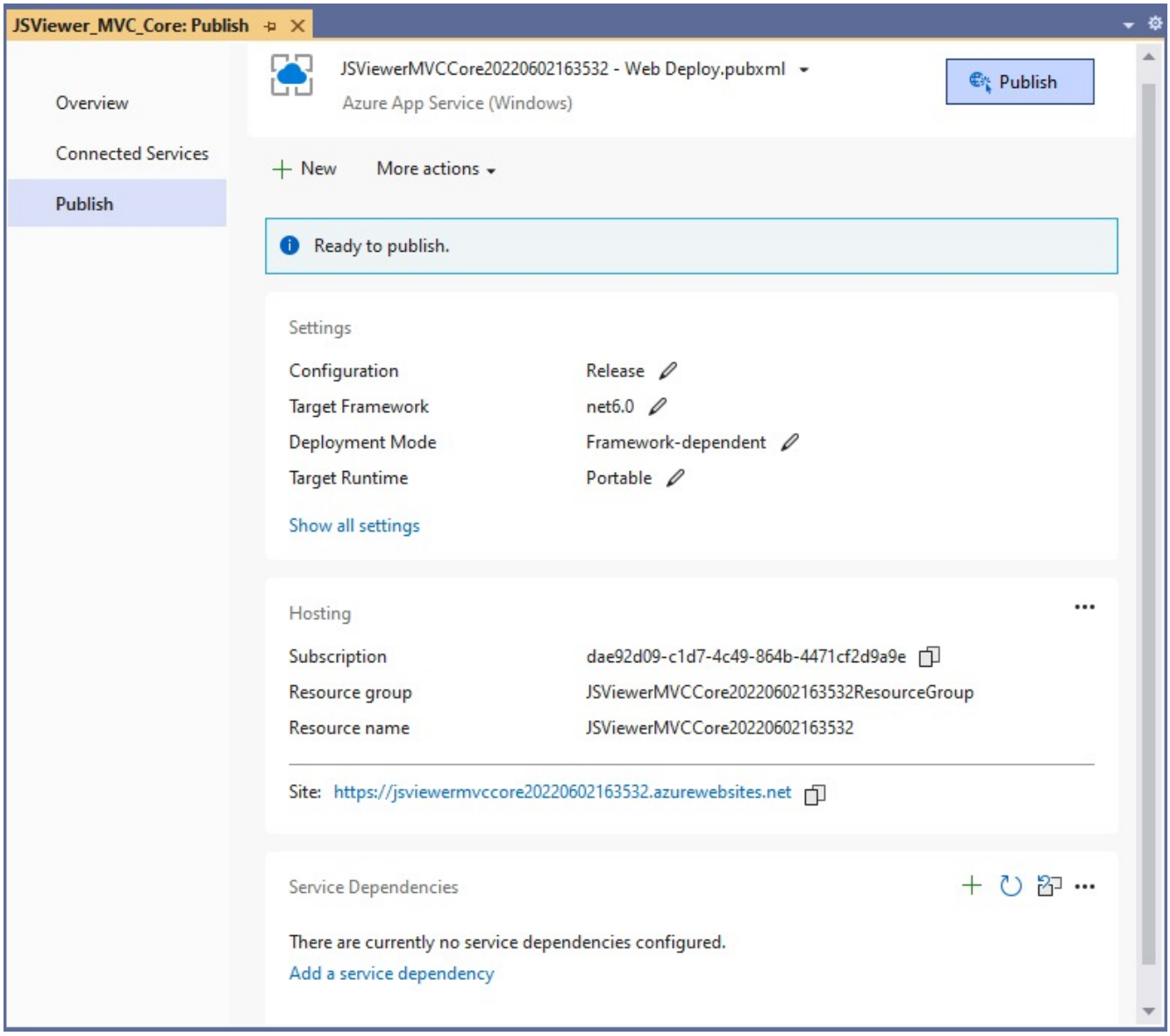


13. Select **Create**.

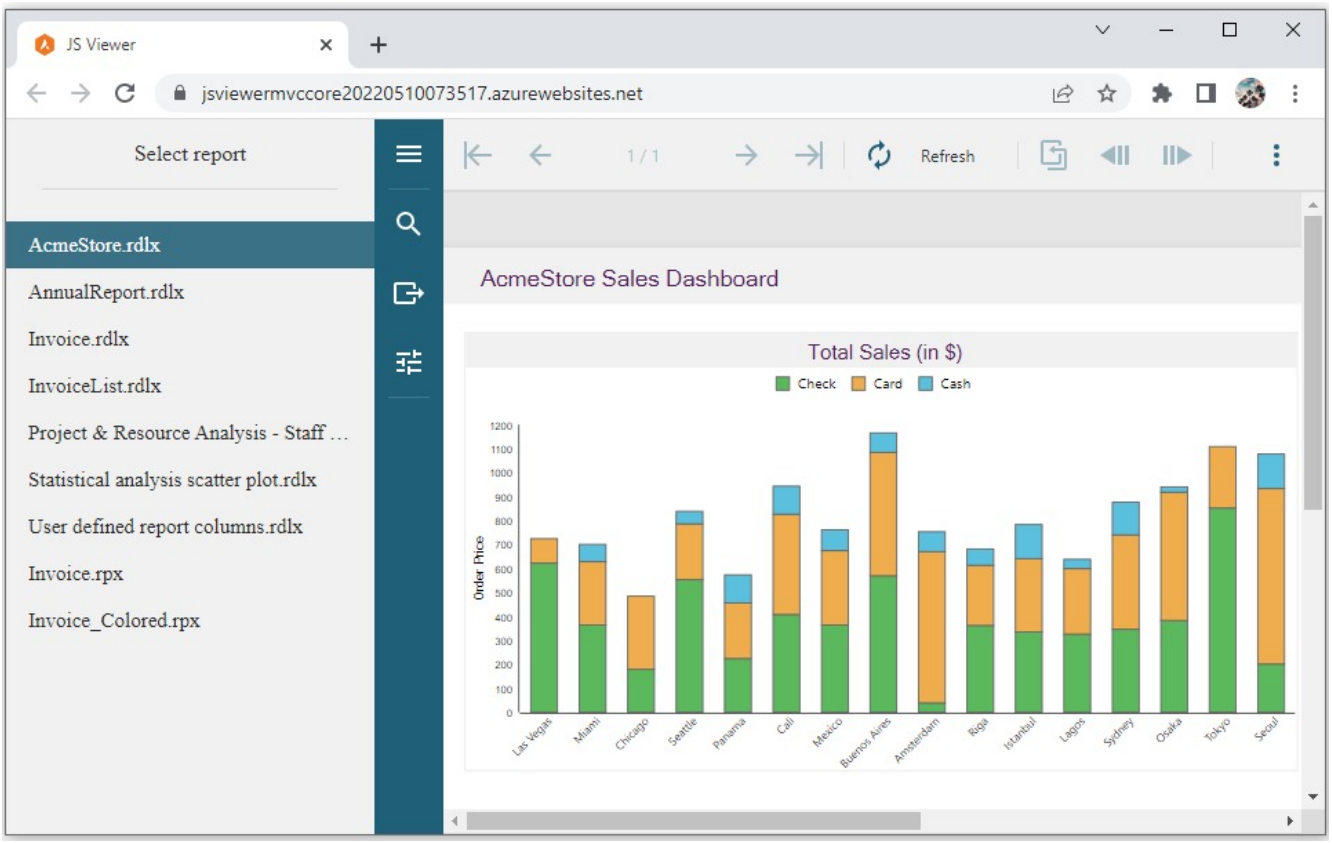


Once the wizard completes, the Azure resources are created and the project is ready to be published.

14. In the **Publish** dialog, ensure your new App Service app is selected in App Service instance, then select **Finish**. Visual Studio creates a publish profile for you for the selected App Service app.



15. In the Publish page, select **Publish**.
    Visual Studio builds, packages, and publishes the app to Azure, and then launches the app in the default browser.
    Once you have followed all the steps the website is deployed on Azure.



Deploy on Azure using a Linux Environment

In this tutorial we will be deploying the JSViewer_MVC_Core sample on Azure using a Linux environment.

1. Download the JSViewer_MVC_Core sample from our WebSamples18 GitHub repository.
2. Build and run the application.
3. Remove the **Reports** folder containing links the reports from the project since we will be embedding the reports in the application.
4. Now add a new folder and name it 'Reports'.
5. Add all the files from **JSViewerReports** in the **Reports** folder and set the **Build Action** as 'Embedded Resource'.
6. Since the published docker might not contain the fonts used by reports we will add a custom font resolver in our ActiveReports application.
    1. Add a new folder **Fonts** in the project where we will the required fonts.
    2. Add a new .cs file '**CustomFontResolver.cs**' and make sure to set the **Build Action** to 'Embedded Resource'.
    3. Add the following code in the .cs file to the add the script.

CustomFontResolver.cs

```
using GrapeCity.Documents.Text.Windows;
public sealed class CustomFontResolver : GrapeCity.ActiveReports.IFontResolver
    {
        static readonly GrapeCity.Documents.Text.FontCollection _fonts = new
GrapeCity.Documents.Text.FontCollection();
        static CustomFontResolver()
        {
            _fonts.Clear();
```

```
            var assembly = Assembly.GetExecutingAssembly();
            var fontnames = assembly.GetManifestResourceNames().Where(str =>
str.EndsWith(".ttf"));
            foreach (var fontname in fontnames)
            {
                Stream stream = assembly.GetManifestResourceStream(fontname);
                _fonts.Add(GrapeCity.Documents.Text.Font.FromStream(stream));
            }
            _fonts.DefaultFont = _fonts.FindFamilyName("Arial");
        }
        public static GrapeCity.ActiveReports.IFontResolver Instance = new
CustomFontResolver();
        private CustomFontResolver() { }
        GrapeCity.Documents.Text.FontCollection
GrapeCity.ActiveReports.IFontResolver.GetFonts(string familyName, bool isBold, bool
isItalic)
        {
            var fonts = new GrapeCity.Documents.Text.FontCollection();
            var font = _fonts.FindFamilyName(familyName, isBold, isItalic);
            if (font != null) fonts.Add(font);
            fonts.Add(_fonts.DefaultFont);
            return fonts;
        }
    }
```

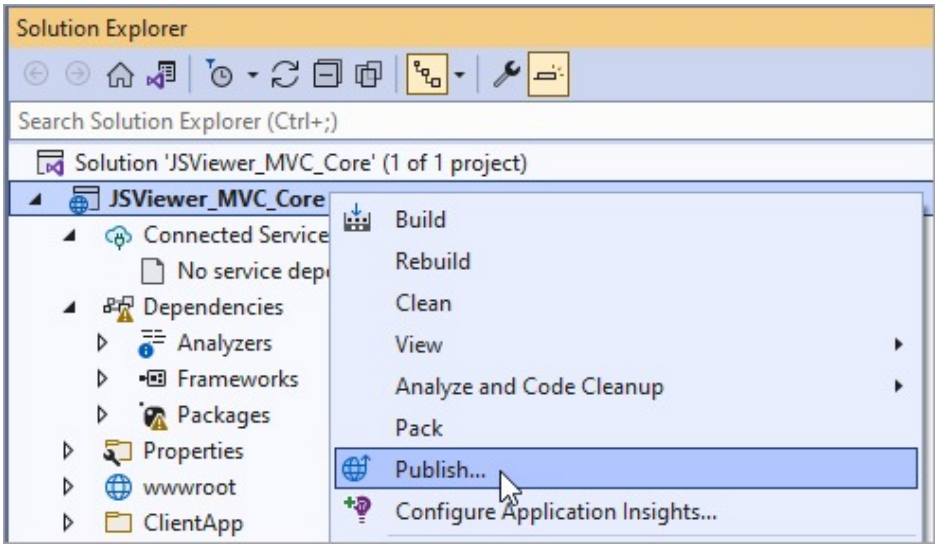
4. In Startup.cs, specify the FontResolver property in the JavaScript Viewers as in the following code example.

Startup.cs

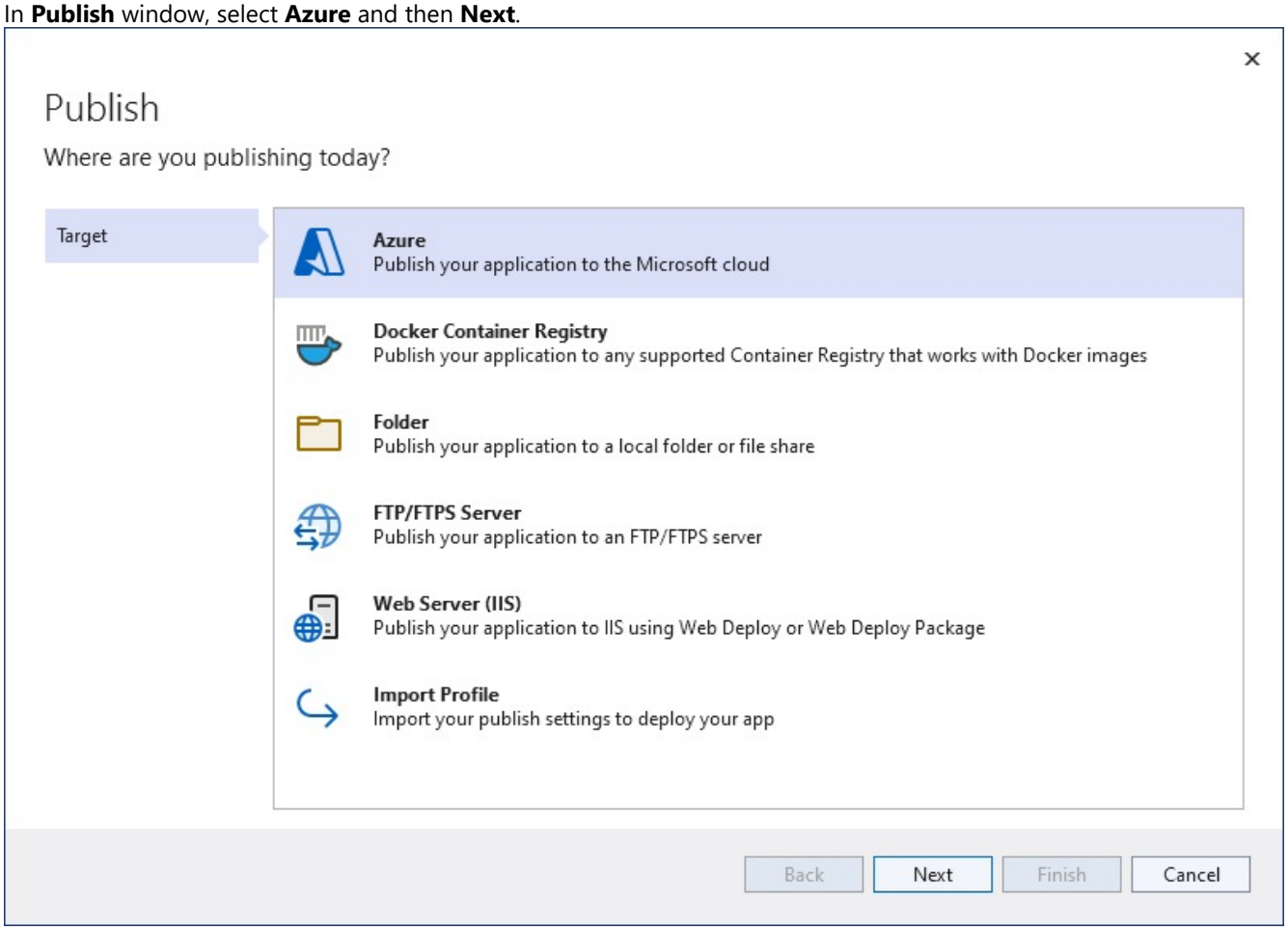
```
app.UseReportViewer(settings =>
            {
                settings.FontResolver = CustomFontResolver.Instance;
                settings.UseEmbeddedTemplates(EmbeddedReportsPrefix,
Assembly.GetAssembly(GetType()));
                settings.UseCompression = true;
            });
```

Follow these steps to create your App Service resources and publish your project. Let us use JSViewer_MVC_Core sample project for demonstration purpose.

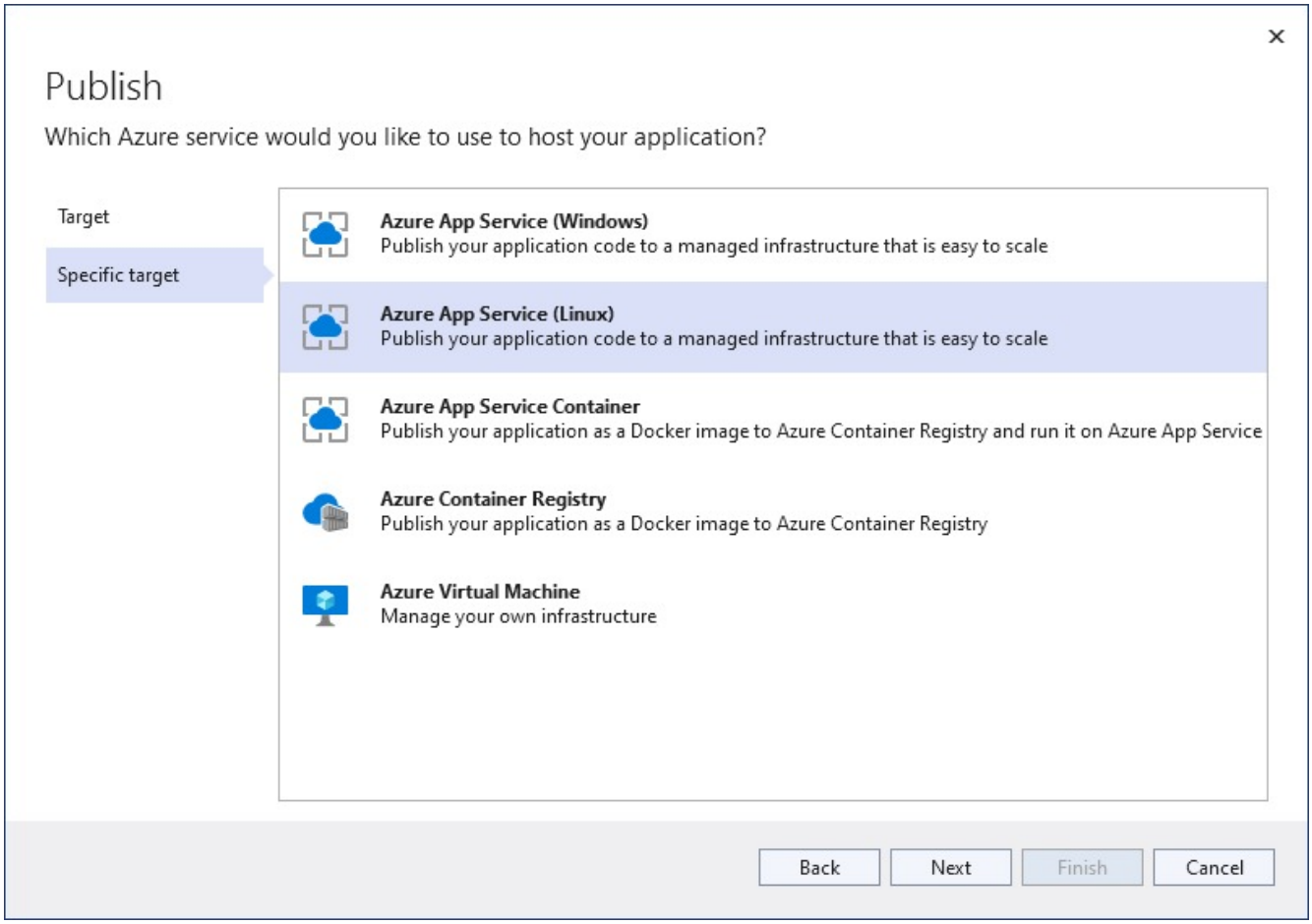
1. Open the JSViewer_MVC_Core sample project in Visual Studio 2022.
2. In the **Solution Explorer**, right-click the on your project name, and select **Publish**.



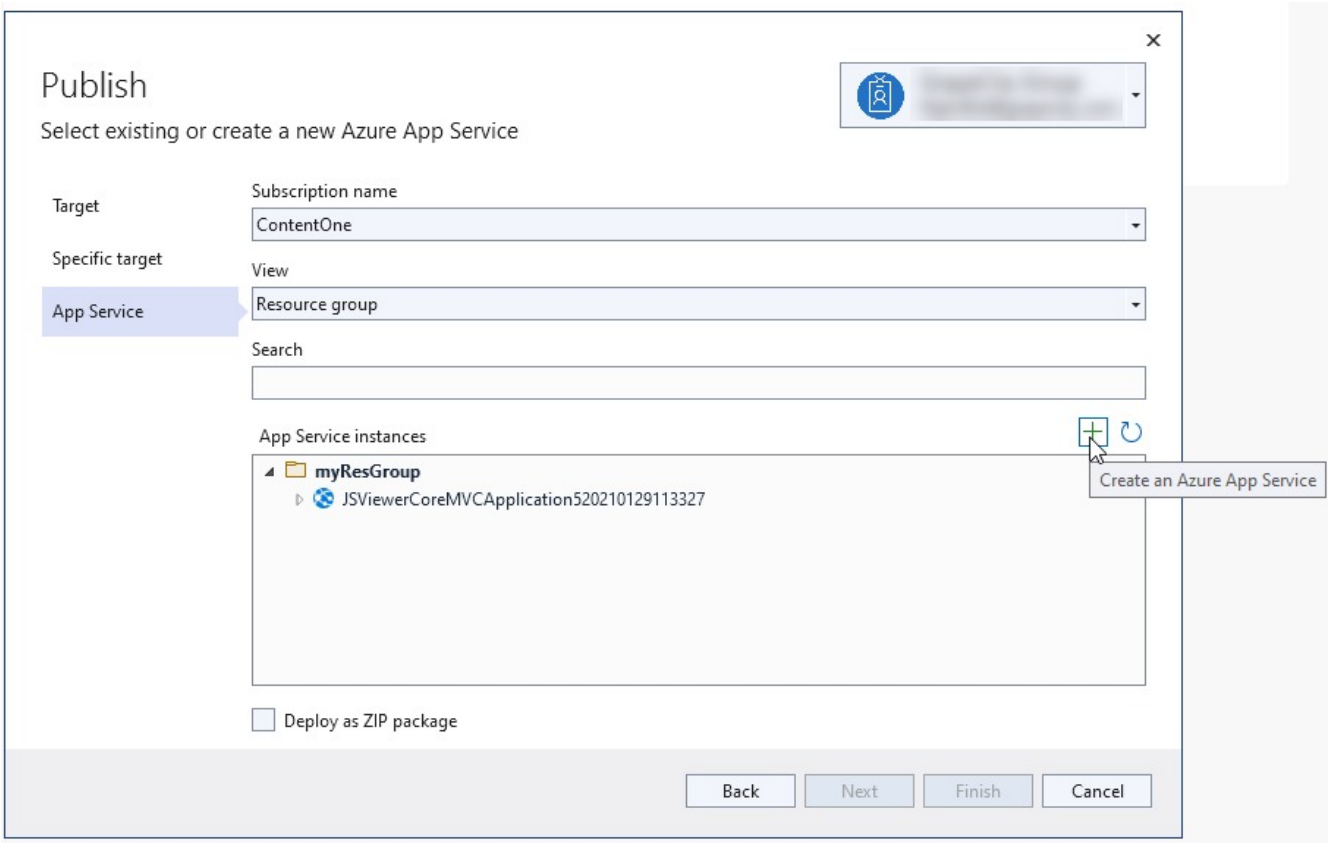
3. In **Publish** window, select **Azure** and then **Next**.



4. Choose the **Specific target** Azure App Service (Linux). Then, select **Next**.

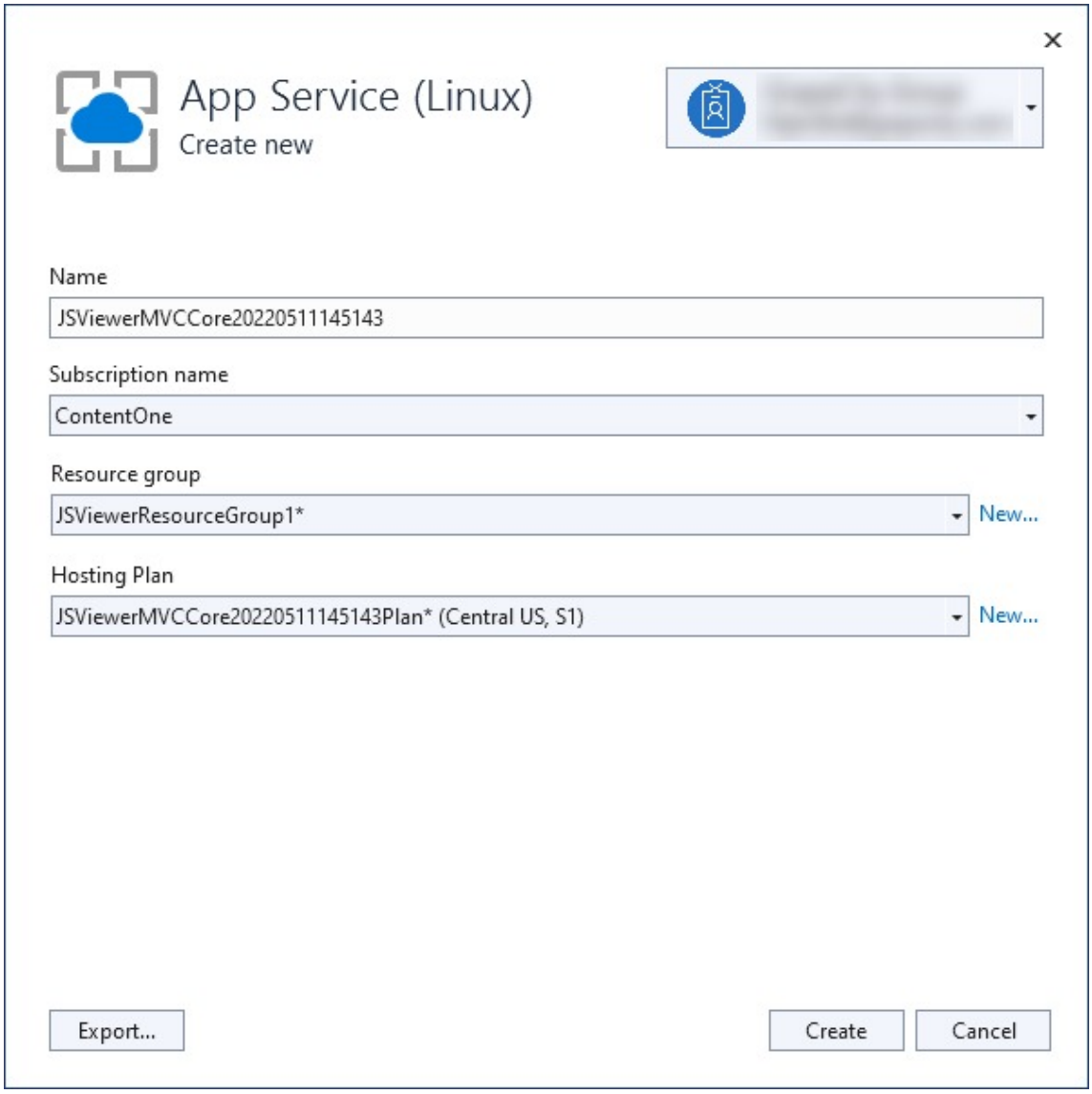


5. Your options depend on whether you're signed in to Azure already and whether you have a Visual Studio account linked to an Azure account. Select either Add an account or Sign in to sign in to your Azure subscription. If you're already signed in, select the account you want.
6. To the right of App Service instances, select +.

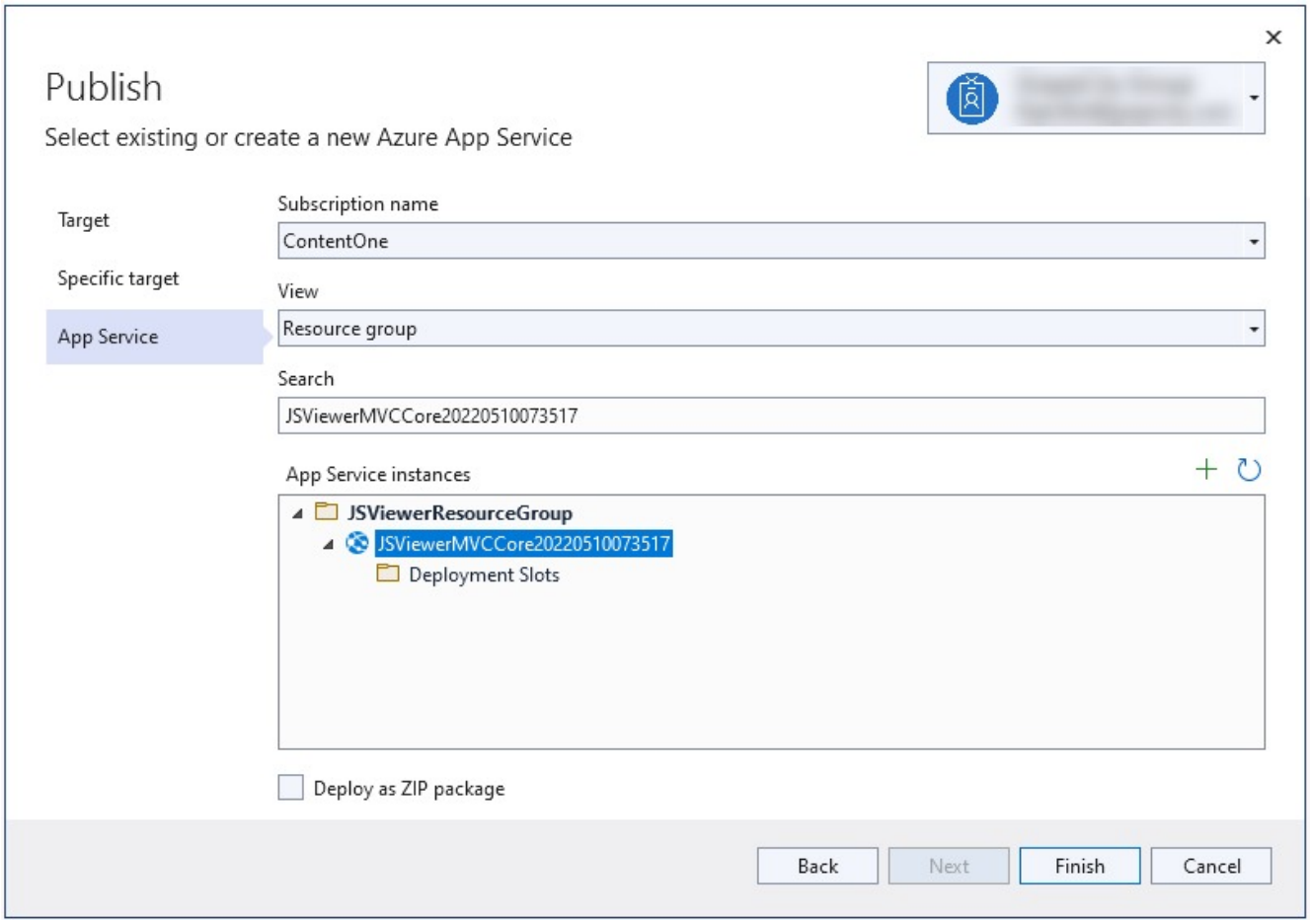


7. For **Subscription name**, accept the subscription that is listed or select a new one from the drop-down list.
8. For **Resource group**, select **New**. It will contain all of the Azure resources for the service.
9. In **New resource group name**, enter **JSViewerResourceGroup** and select OK.
10. For **Hosting Plan**, select **New**. It specifies the location, size, and features of the web server that hosts your app.
11. In the **Hosting Plan** dialog, enter the appropriate values:

    Hosting Plan: JSViewerMVCCorePlan
    Location: Central US
    Size: S1
12. In **Name** field, enter a unique app name that includes only the valid characters (a-z, A-Z, 0-9, and -). You can accept the automatically generated unique name. The URL of the web app is http://<app-name>.azurewebsites.net, where <app-name> is your app name.

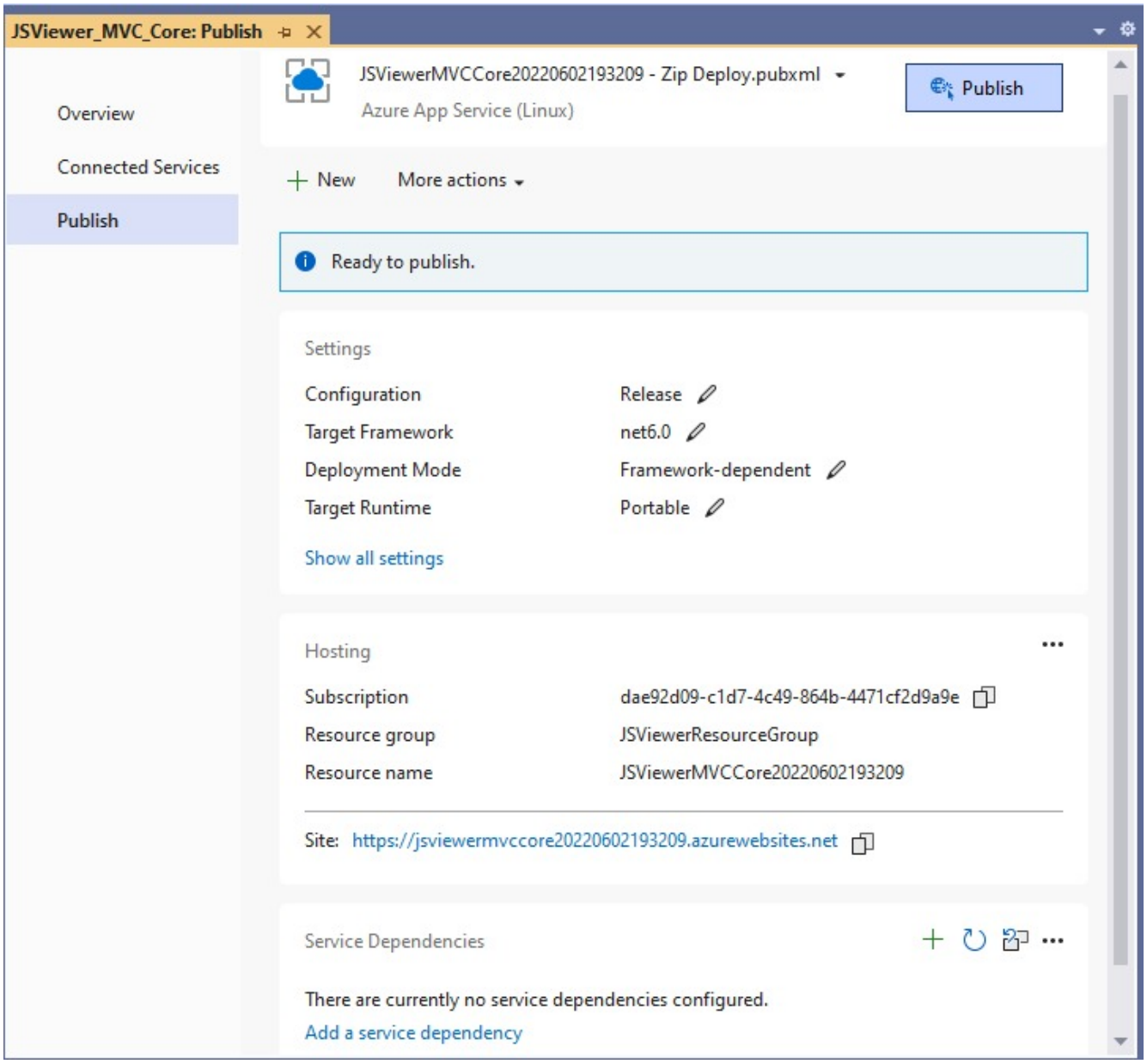


13. Select **Create**.

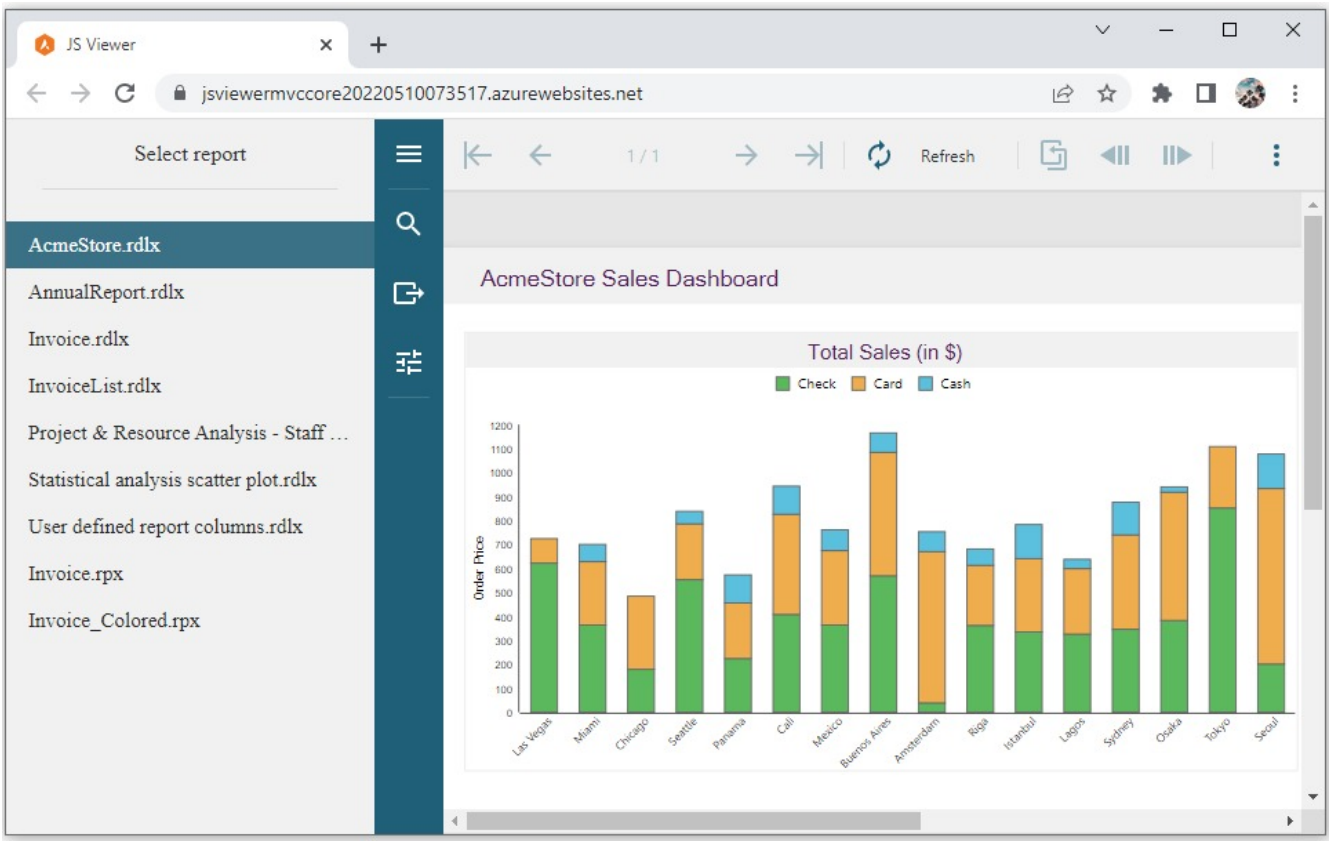


Once the wizard completes, the Azure resources are created and the project is ready to be published.

14. In the **Publish** dialog, ensure your new App Service app is selected in App Service instance, then select **Finish**. Visual Studio creates a publish profile for you for the selected App Service app.



15. In the Publish page, select **Publish**.
    Visual Studio builds, packages, and publishes the app to Azure, and then launches the app in the default browser.
    Once you have followed all the steps the website is deployed on Azure.



Amazon Web Services

We will be deploying the JSViewer_MVC_Core sample on cloud using AWS web service.

Prerequisites

This tutorial uses the .NET Core SDK to generate a basic .NET Core application, run it locally, and build a deployable package.

Requirements

- NET Core SDK x64 for Windows 1.0.1, 2.0.0, or later
  Download .NET SDK from https://dotnet.microsoft.com/en-us/download and run the installer.
- ActiveReports 18 or later
  See Install ActiveReports for more information.
- AWS Toolkit for Visual Studio 2022
  1. In Visual Studio 2022, from **Extensions** > **Manage Extensions**, search online for 'AWS Toolkit for Visual Studio 2022'.
  2. Select **Download**. The extension is scheduled for install.
  3. Close all instances of Visual Studio and restart your .Net project once you have installed the extension.

Deploy on AWS using a Windows Environment

This page demonstrates how you can deploy your ActiveReports application on AWS using a Windows Environment.

Setup your application

1. Download the JSViewer_MVC_Core sample from our WebSamples18 GitHub repository.
2. Open the application in Visual Studio and build and run the application.
3. Remove the **Reports** folder containing links the reports from the project since we will be embedding the reports in the application.
4. Now add a new folder and name it 'Reports'.
5. Add all the files from **JSViewerReports** in the **Reports** folder and set the **Build Action** as 'Embedded Resource'.

Setup a user in AWS

Once you have set up everything the next step is to create a user in your AWS Account. You can also visit AWS Docs for more information.

1. Open the Sign In Page.

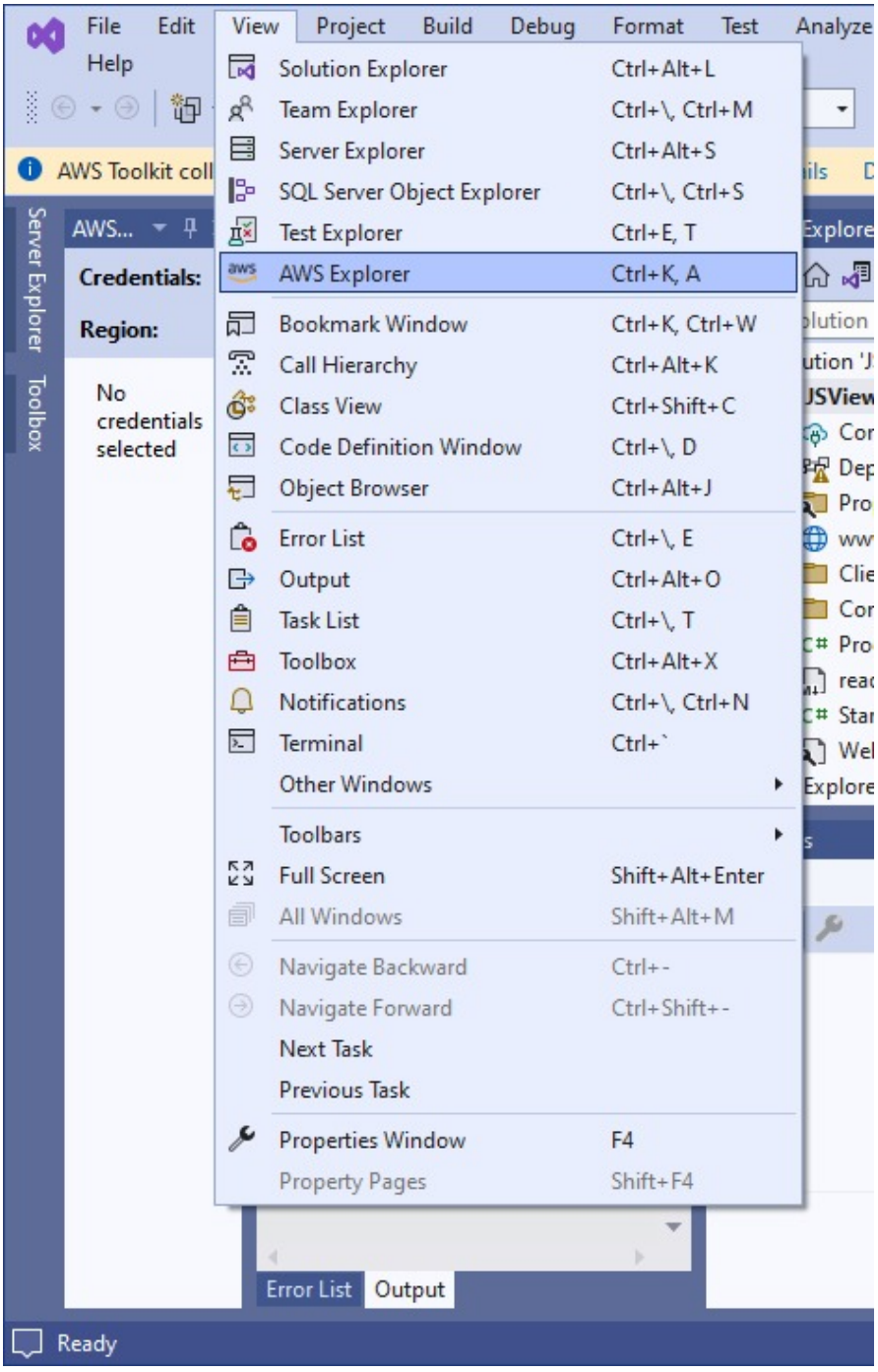
   > **Note:** If you are new to AWS then Sign Up to AWS and add MFA Authentication by logging-in as a Root User first.

2. Select **IAM user** and enter your credentials.
3. Now from the AWS Management Console under All Services, select **IAM**. This will open the IAM Dashboard.
4. Select **Users** under the **Access Management** group
5. Click **Add users** on the top right of your screen. This will open the Add user page.
6. Set the **User name**. Here, for documentation purposes we are adding 'AR_Deployment_Agent' as the username.
7. Select the **Access key-Programmatic access** that gives this user programmatic access. Then click **Next: Permissions**.
8. Now we will add user to a group. Select an existing group or to create a new group:
   a. Click on **Create a Group** option.
   b. Enter the **Group Name**. Here we are using 'AR_Deployment_Agents' as the group name.
   c. In the policies option, check **AdministratorAccess-AWSElasticBeanstalk** and **IAMFullAccess**.
   d. Click **Create** group.
9. Click **Next: Tags**. Here you can add the tags that can be used in your application for now we are going to skip this.
10. Click **Next: Review**. Here, you can review the user properties and then select **Create user**.
11. Now copy the **Access key ID** and **Secret access key** and save it.
    Note you can also download the CSV from this page.

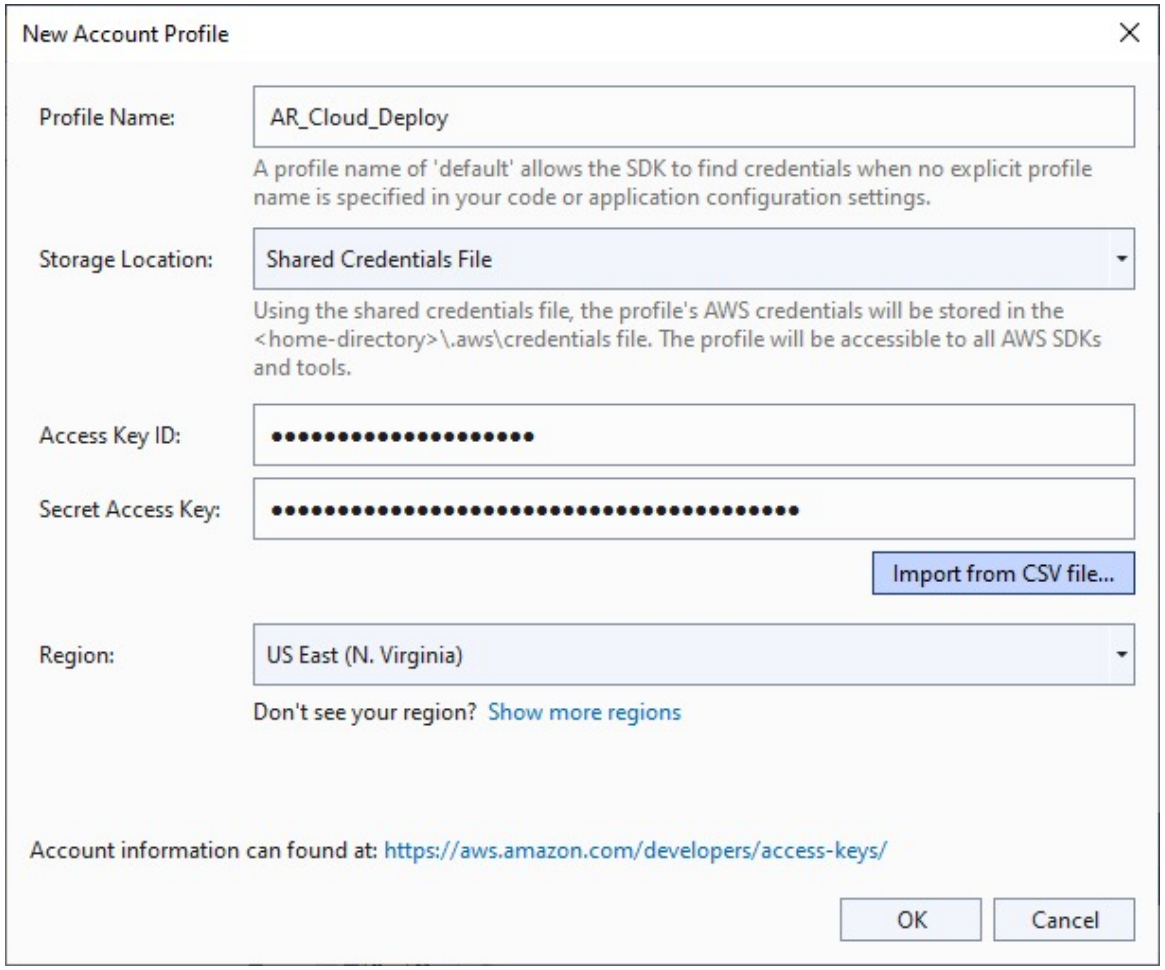
Setup AWS Profile in Visual Studio

Now that we have setup everything on the AWS account, let us setup our application with AWS and deploy it on the cloud.

1. Open the JSViewer_MVC_Core sample in Visual Studio.
2. From **View**, go to **AWS Explorer**.



3. In the AWS Explorer window, click the "+" icon to create a new profile.



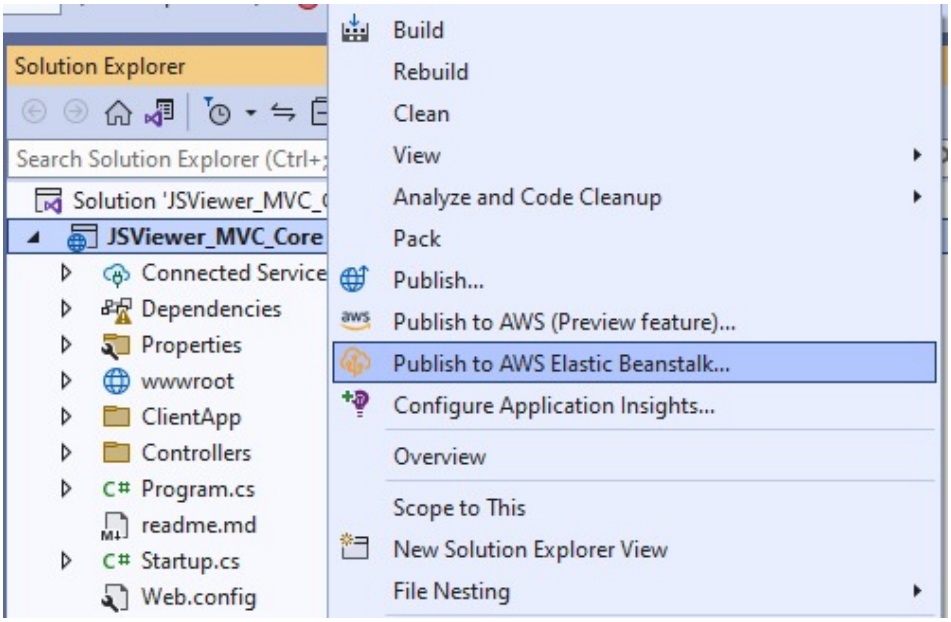
4. Enter a **Profile Name** (AR_Cloud_Deploy), the saved **Access Key ID**, and the **Secret Access Key**.

   Note that you can also import the CSV if you chose to download it while creating the user account.

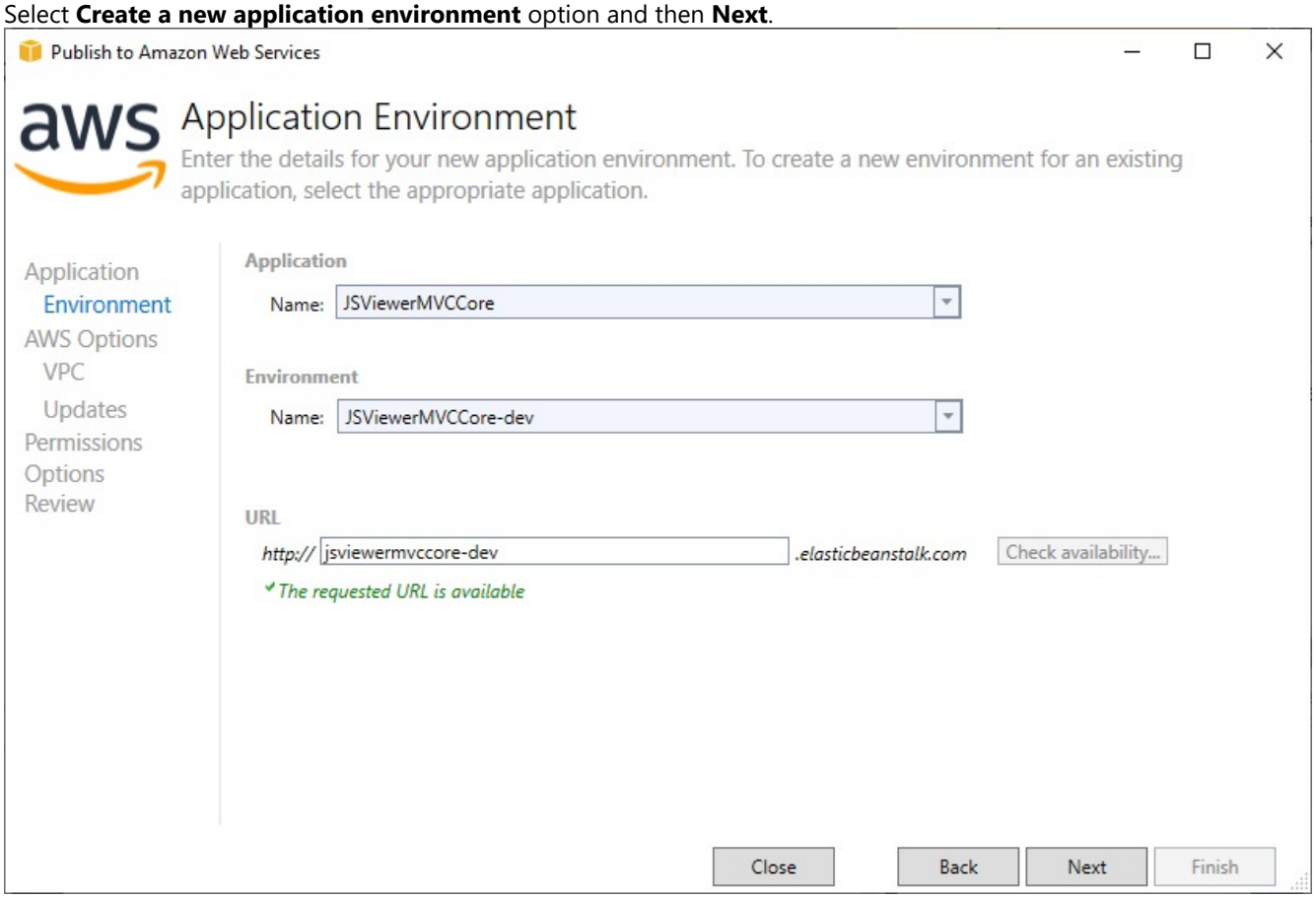
Publish and deploy your application

To publish your application for AWS Elastic Beanstalk, follow these steps.

1. Right-click on the project in the Solution Explorer and click **Publish to AWS Elastic Beanstalk**.. option in the context menu.

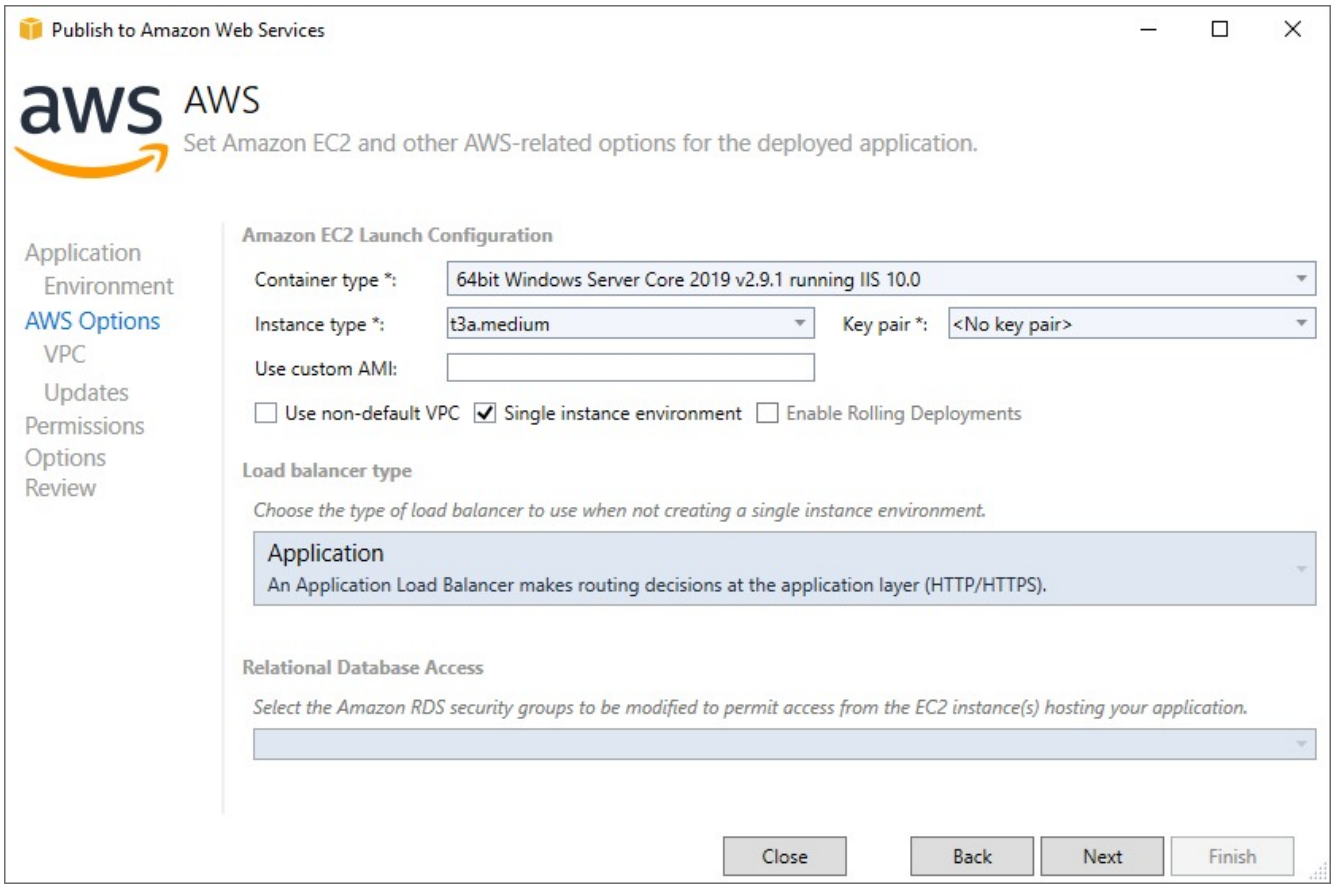


2. Now in the Publish to Amazon Web Services window, set the Region. Although AWS has many different regions for this sample, we are selecting the region as US East (N. Virginia).
3. Select **Create a new application environment** option and then **Next**.



4. Set the **Application Name** and the **Environment Name**.
   A URL should be added automatically after the previous step. You can add some other URL as well.
5. Check the URL availability by clicking the **Check availability**... button. If the URL is available, "The requested URL is available" message should appear on your screen. Now, click **Next**.

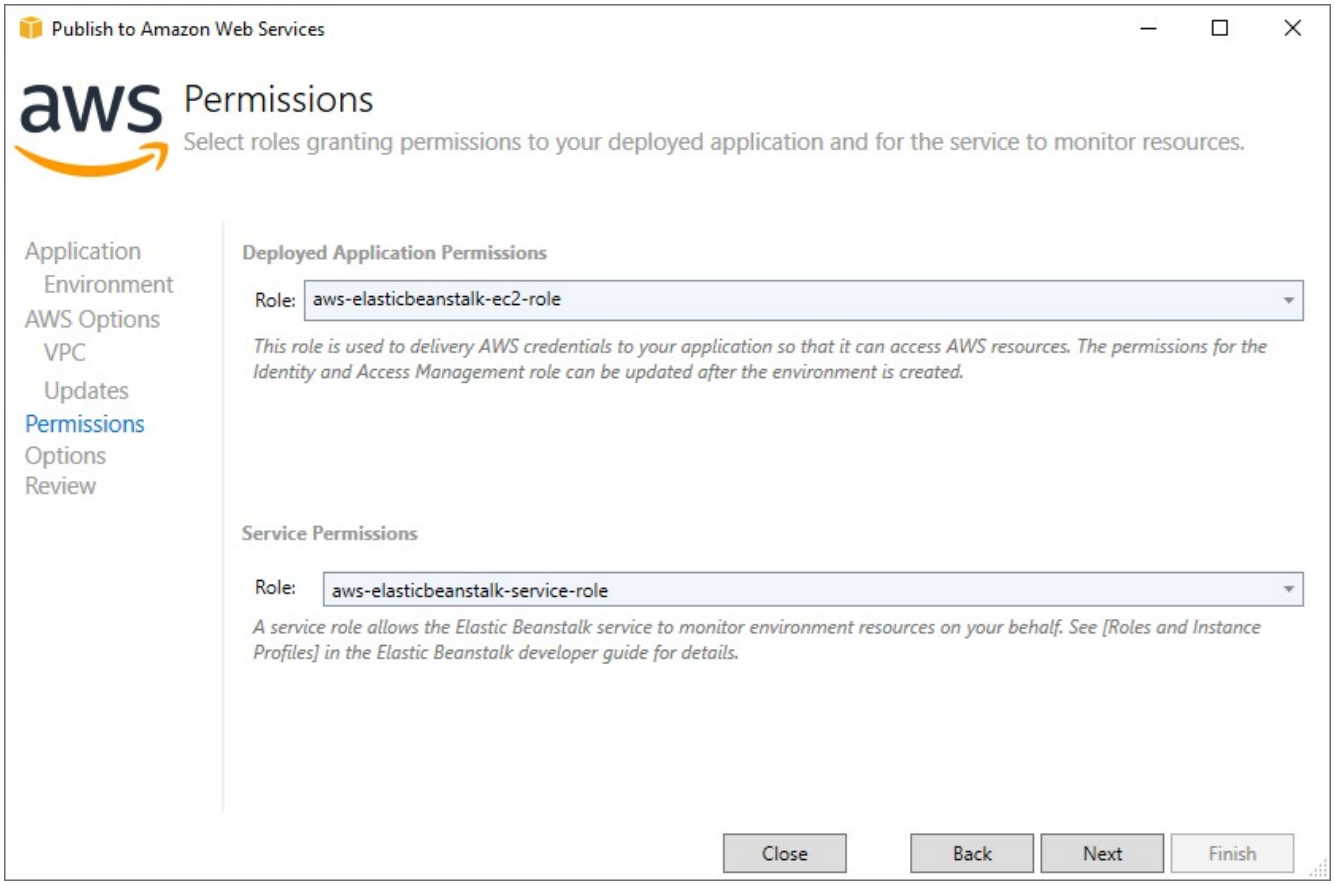
6. Set the **Container type** to **64bit Windows Server Core 2019 v2.9.1 running IIS 10.0**. Now, click Next.
   Now, we must configure all our **AWS Options**. For the sample purposes we are leaving most of the options as default.



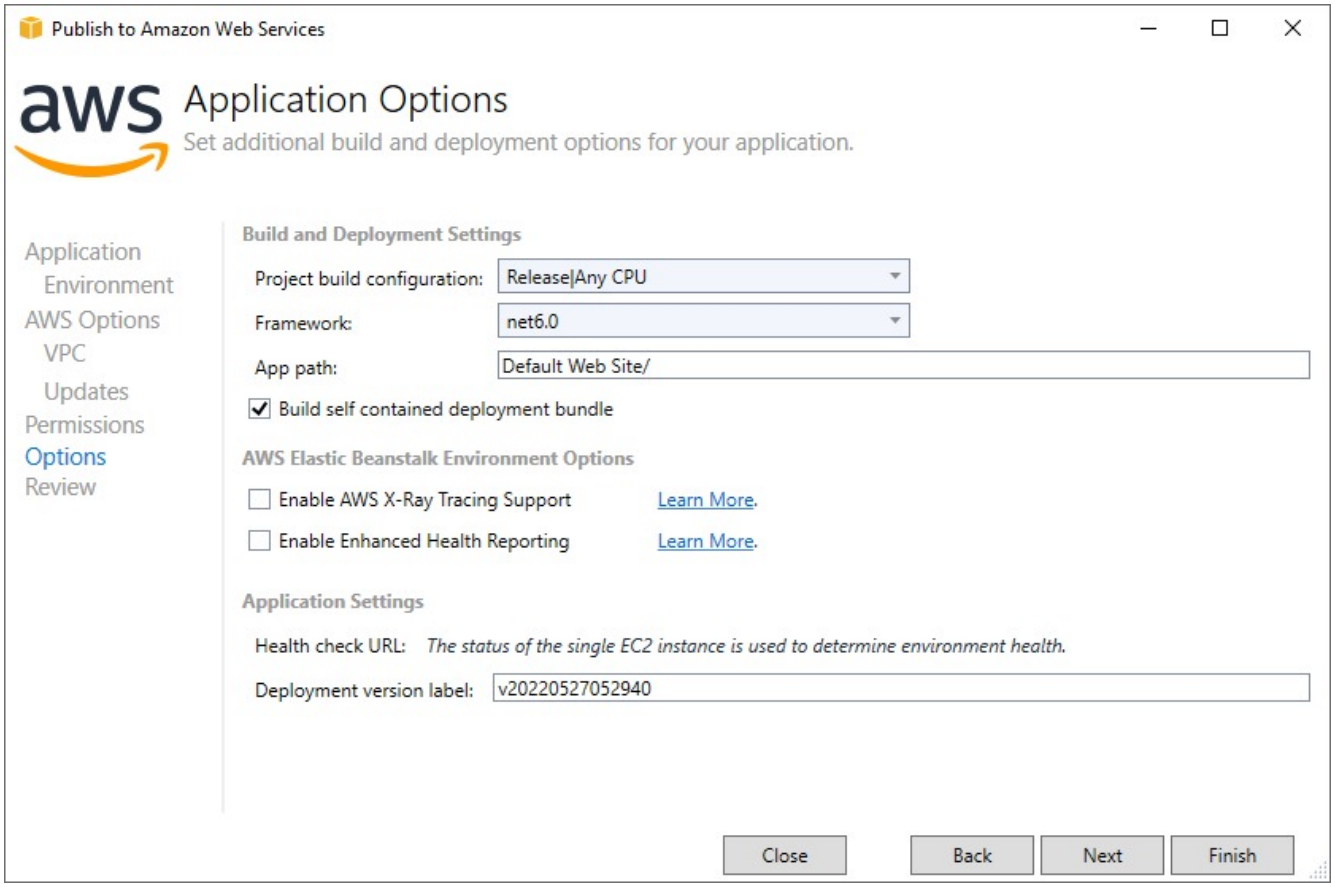
7. In the next **Permissions** window, we will select roles granting permissions to your deployed application for the service to monitor resources. Input the following values in the permissions options and then click **Next**.

   Deployment Application Permissions
   Role: aws-elasticbeanstalk-ec2-role

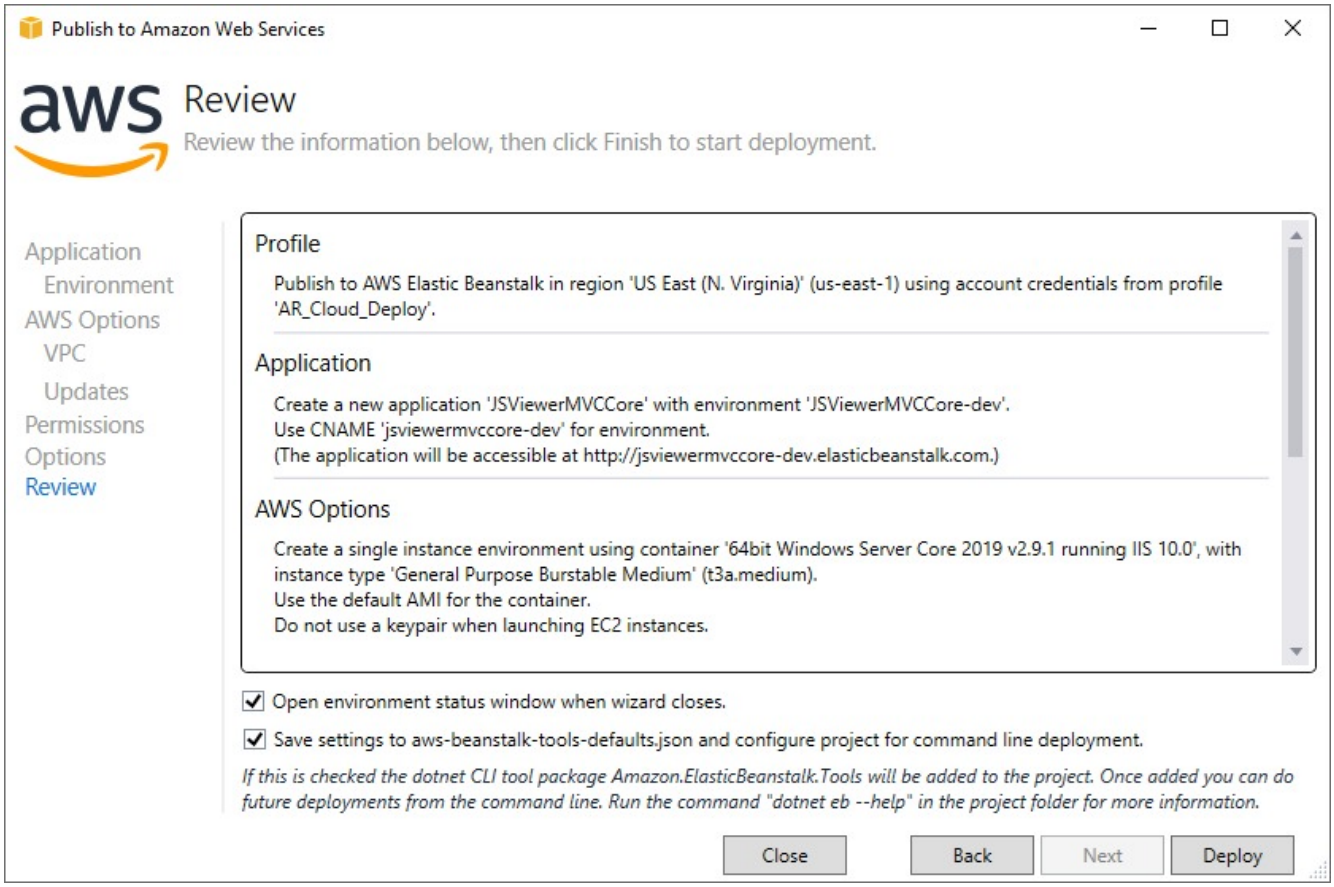
   Service Permissions
   Role: aws-elasticbeanstalk-service-role



8. In the **Application Options** window, change the **Project build configuration** from Debug|Any CPU to **Release|Any CPU**.
9. Set the **Framework** to net6.0.
10. Check the **Build self contained deployment bundle** option and then click **Next**.



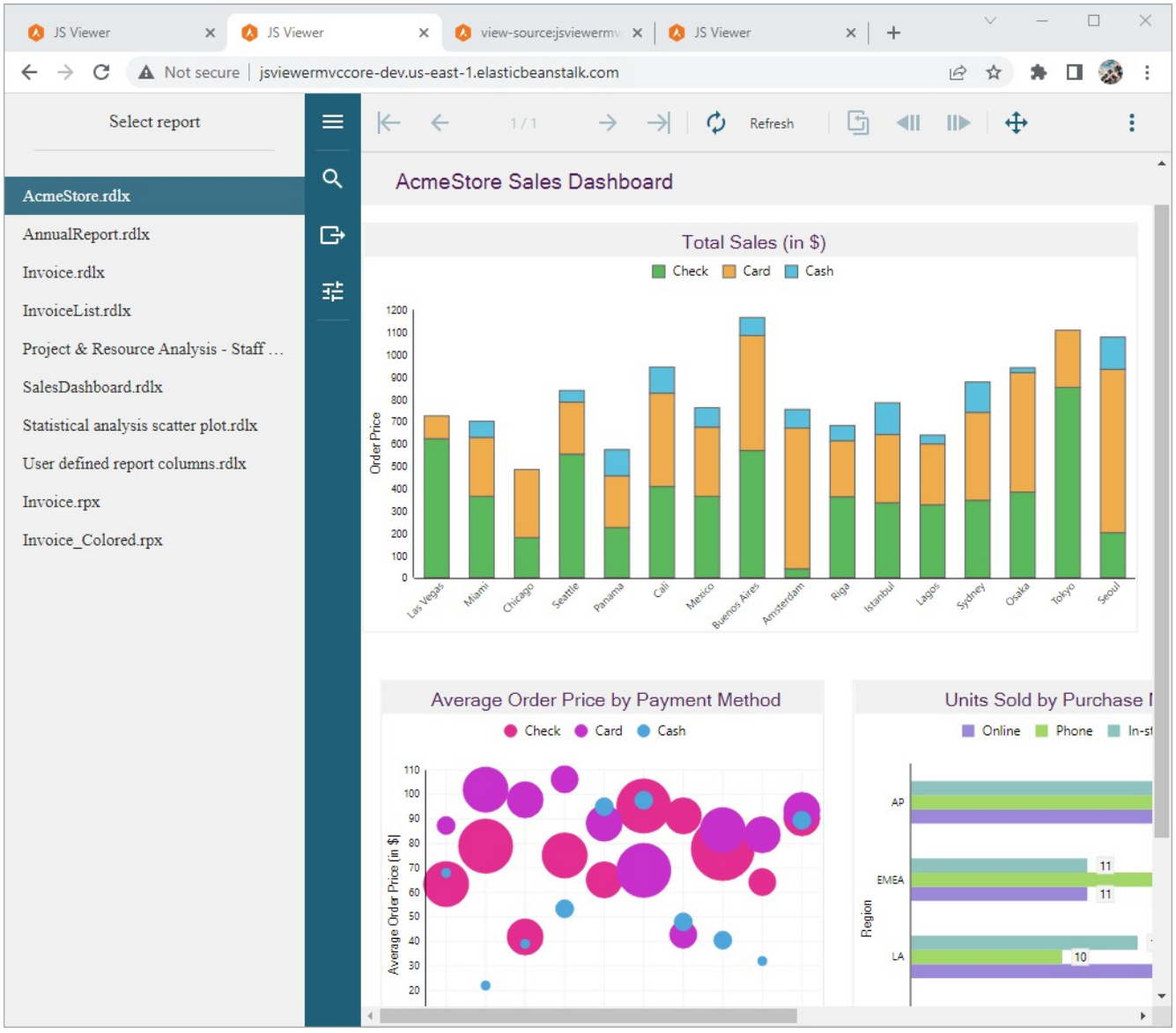
11. Go through the **Review** page to ensure you have selected all the required options as per your organization and application.
12. Click **Deploy**.
    Once you click Deploy, it takes several minutes to create the environment and launch your ActiveReports application to the cloud.



Once your application is successfully launched, you will see the following Event statement in the Event's tab in the Environment window:

> Successfully launched environment: JSViewerMVCCore-dev

Also, the Status of the environment would be Environment is healthy.



Now your application has successfully launched to the cloud. Now you can open it using the URL given at the top of the Environment window.

Clean up

To avoid any incurring charges if you are done working with Elastic Beanstalk for now, you can terminate your .NET environment. To terminate your **Elastic Beanstalk** environment:

1. Open the **Elastic Beanstalk console**, and in the Regions list, select your AWS Region
2. In the navigation pane, choose **Environments**, and then choose the name of your environment from the list.
3. Choose **Environment actions** and then choose **Terminate environment**.

Deploy on AWS using a Linux Environment

This page demonstrates how you can deploy your ActiveReports application on AWS using a Linux Environment.

Setup your application

1. Download the JSViewer_MVC_Core sample from our WebSamples18 GitHub repository.
2. Open the application in Visual Studio and build and run the application.
3. Remove the **Reports** folder containing links to the reports from the project since we will be embedding the reports in the application.
4. Now add a new folder and name it 'Reports'.
5. Add all the files from **JSViewerReports** in the **Reports** folder and set the **Build Action** as 'Embedded Resource'.
6. If you are planning to publish the application on a Linux container, it is possible that the container might not contain the fonts used by reports. We will add a custom font resolver in our ActiveReports application that we are creating. For this please refer to our documentation page Custom Font Resolver.
    1. Add a new folder **Fonts** in the project where we will add the required fonts.
    2. Add all the required fonts in the **Fonts** folder and then
        - make sure to set the **Build Action** of the font files to 'Embedded Resource'.
          OR
        - add the following lines of code to the JSViewer_MVC_Core.csproj file to copy all the fonts to the build folder:

          JSViewer_MVC_Core.csproj

          ```
          <ItemGroup>
               <Content Include="Fonts**\*.ttf**">
                 <CopyToOutputDirectory>PreserveNewest</CopyToOutputDirectory>
               </Content>
           </ItemGroup>
          ```
    3. Add the following code in the .cs file to the add the script.

       CustomFontResolver.cs

       ```
       using GrapeCity.Documents.Text.Windows;

       public sealed class CustomFontResolver : GrapeCity.ActiveReports.IFontResolver
           {
               static readonly GrapeCity.Documents.Text.FontCollection _fonts = new
       GrapeCity.Documents.Text.FontCollection();
               static CustomFontResolver()
               {
                   _fonts.Clear();
                   var assembly = Assembly.GetExecutingAssembly();
                   var fontnames = assembly.GetManifestResourceNames().Where(str =>
       str.EndsWith(".ttf"));
                   foreach (var fontname in fontnames)
                   {
                       Stream stream = assembly.GetManifestResourceStream(fontname);
                       _fonts.Add(GrapeCity.Documents.Text.Font.FromStream(stream));
                   }
                   _fonts.DefaultFont = _fonts.FindFamilyName("Arial");
               }
               public static GrapeCity.ActiveReports.IFontResolver Instance = new
       CustomFontResolver();
               private CustomFontResolver() { }
               GrapeCity.Documents.Text.FontCollection
       GrapeCity.ActiveReports.IFontResolver.GetFonts(string familyName, bool isBold, bool
       isItalic)
               {
       ```

```
            var fonts = new GrapeCity.Documents.Text.FontCollection();
            var font = _fonts.FindFamilyName(familyName, isBold, isItalic);
            if (font != null) fonts.Add(font);
            fonts.Add(_fonts.DefaultFont);
            return fonts;
        }
    }
```

4. In Startup.cs, specify the **FontResolver** property in the JavaScript Viewers as in the following code example.

Startup.cs

```
app.UseReportViewer(settings =>
            {
                settings.FontResolver = CustomFontResolver.Instance;
                settings.UseEmbeddedTemplates(EmbeddedReportsPrefix,
Assembly.GetAssembly(GetType()));
                settings.UseCompression = true;
            });
```

Setup a user in AWS

Once you have set up everything the next step is to create a user in your AWS Account. You can also visit AWS Docs for more information.

1. Open the Sign In Page.

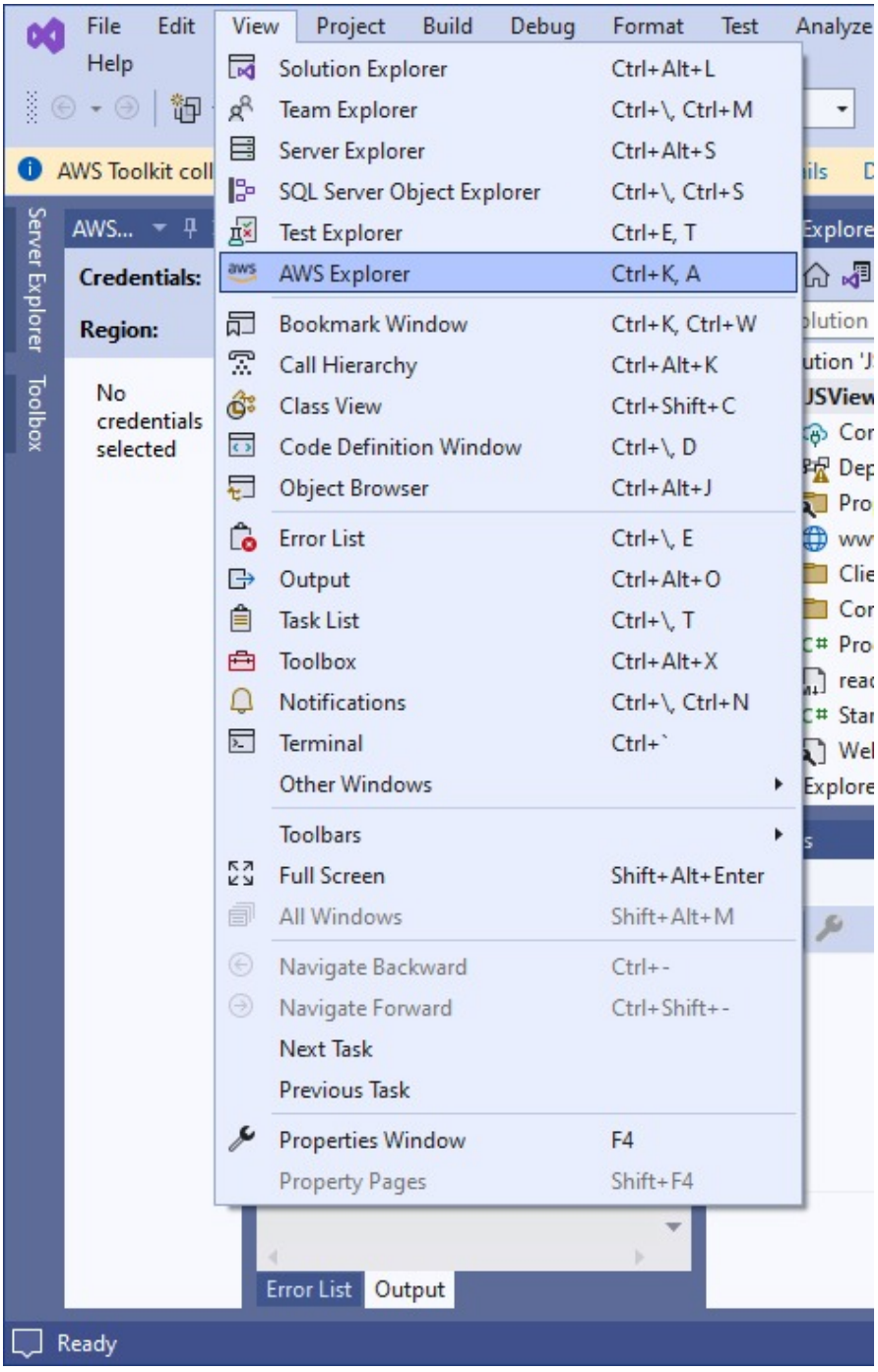
   Note: If you are new to AWS then Sign Up to AWS and add MFA Authentication by logging in as a Root User first.

2. Select **IAM user** and enter your credentials.
3. Now from the AWS Management Console under All Services, select **IAM**. This will open the IAM Dashboard.
4. Select **Users** under the **Access Management** group
5. Click **Add users** on the top right of your screen. This will open the Add user page.
6. Set the **User name**. Here, for documentation purposes, we are adding 'AR_Deployment_Agent' as the username.
7. Select the **Access key-Programmatic access** that gives this user programmatic access. Then click **Next: Permissions**.
8. Now we will add a user to a group. Select an existing group or create a new group:
   a. Click on **Create a Group** option.
   b. Enter the **Group Name**. Here we are using 'AR_Deployment_Agents' as the group name.
   c. In the policies option, check **AdministratorAccess-AWSElasticBeanstalk** and **IAMFullAccess**.
   d. Click **Create** group.
9. Click **Next: Tags**. Here you can add the tags that can be used in your application for now we are going to skip this.
10. Click **Next: Review**. Here, you can review the user properties and then select **Create user**.
11. Now copy the **Access key ID** and **Secret access key** and save it.
    Note you can also download the CSV from this page.

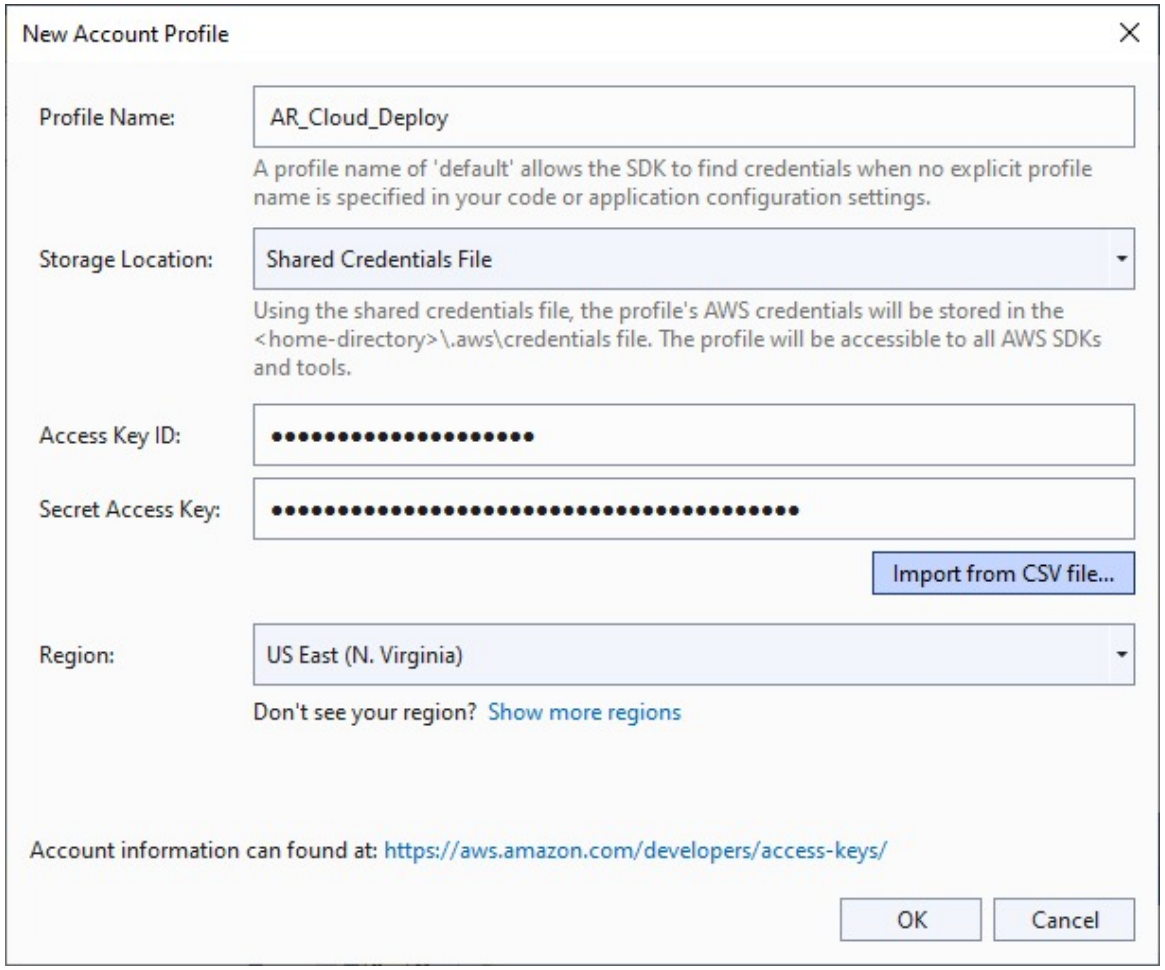
Setup AWS Profile in Visual Studio

Now that we have set up everything on the AWS account, let us set up our application with AWS and deploy it on the cloud.

1. Open the JSViewer_MVC_Core sample in Visual Studio.
2. From **View**, go to **AWS Explorer**.



3. In the AWS Explorer window, click the “+” icon to create a new profile.



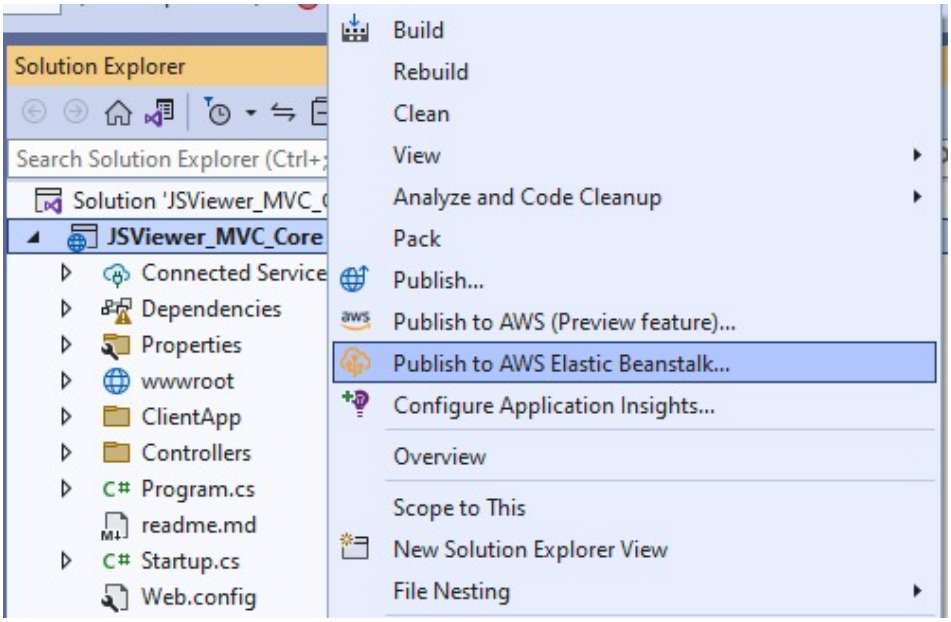
4. Enter a **Profile Name** (AR_Cloud_Deploy), the saved **Access Key ID**, and the **Secret Access Key**.

   Note that you can also import the CSV if you chose to download it while creating the user account.

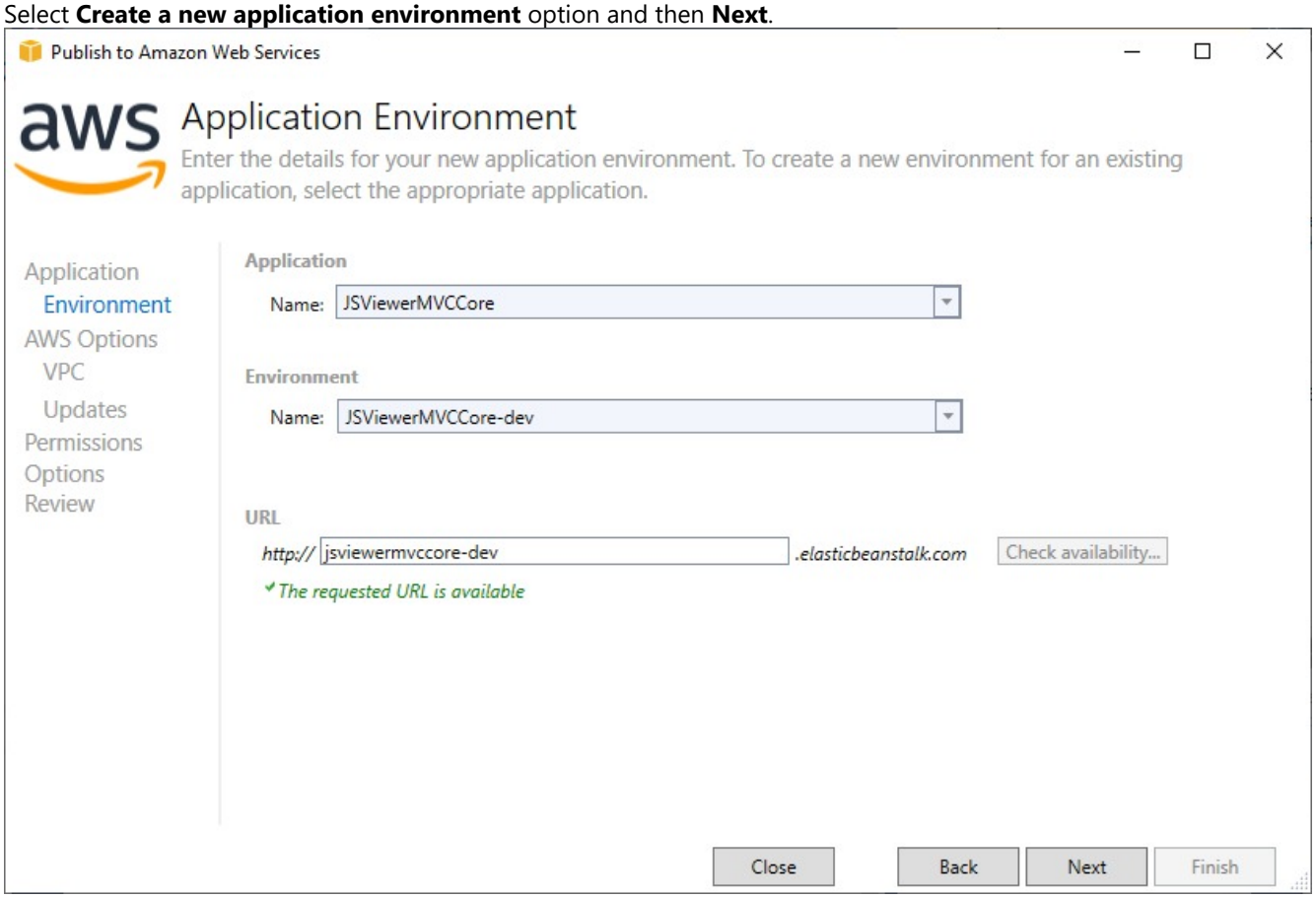
Publish and deploy your application

To publish your application for AWS Elastic Beanstalk, follow these steps.

1. Right-click on the project in the Solution Explorer and click **Publish to AWS Elastic Beanstalk**.. option in the context menu.

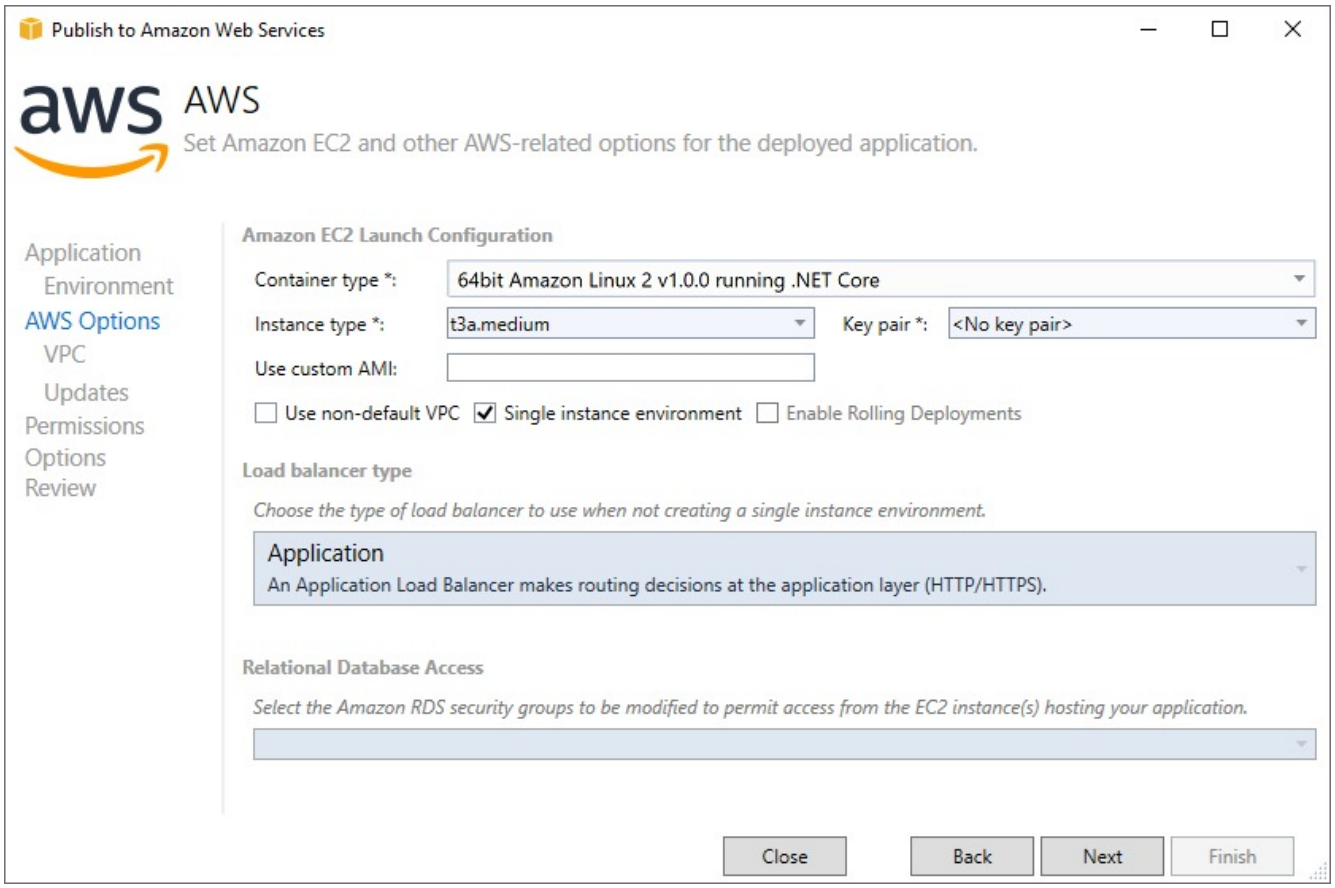


2. Now in the Publish to Amazon Web Services window, set the Region. Although AWS has many different regions for this sample, we are selecting the region as US East (N. Virginia).
3. Select **Create a new application environment** option and then **Next**.



4. Set the **Application Name** and the **Environment Name**.
   A URL should be added automatically after the previous step. You can add some other URL as well.
5. Check the URL availability by clicking the **Check availability**... button. If the URL is available, "The requested URL is available" message should appear on your screen. Now, click **Next**.

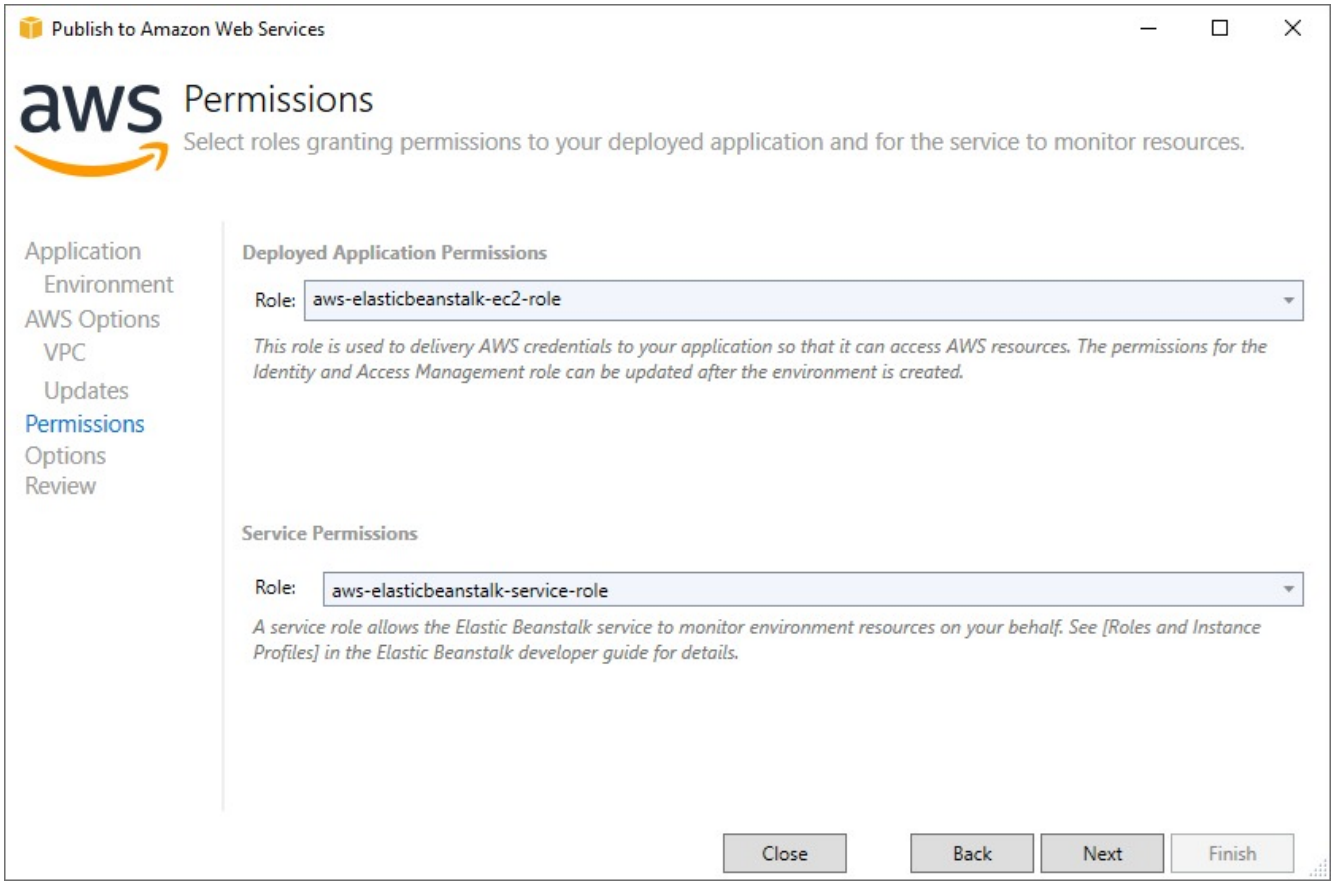
6. Set the **Container type** to **64bit Amazon Linux 2 v1.0.0 running NET Core.** Now, click Next.
   Now, we must configure all our **AWS Options**. For the sample purposes, we are leaving most of the options as default.



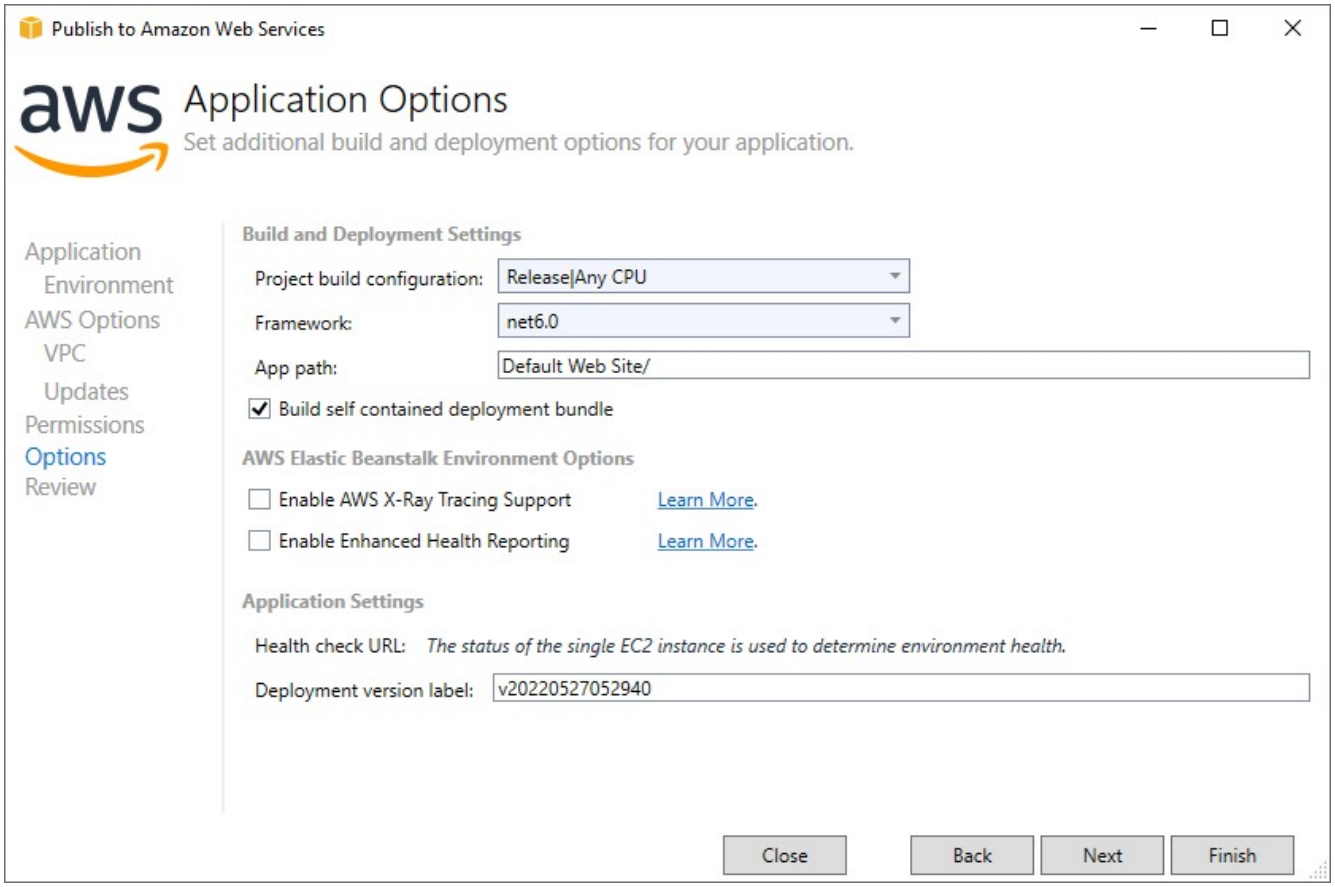
7. In the next **Permissions** window, we will select roles granting permissions to your deployed application for the service to monitor resources. Input the following values in the permissions options and then click **Next**.

   Deployment Application Permissions
   Role: aws-elasticbeanstalk-ec2-role

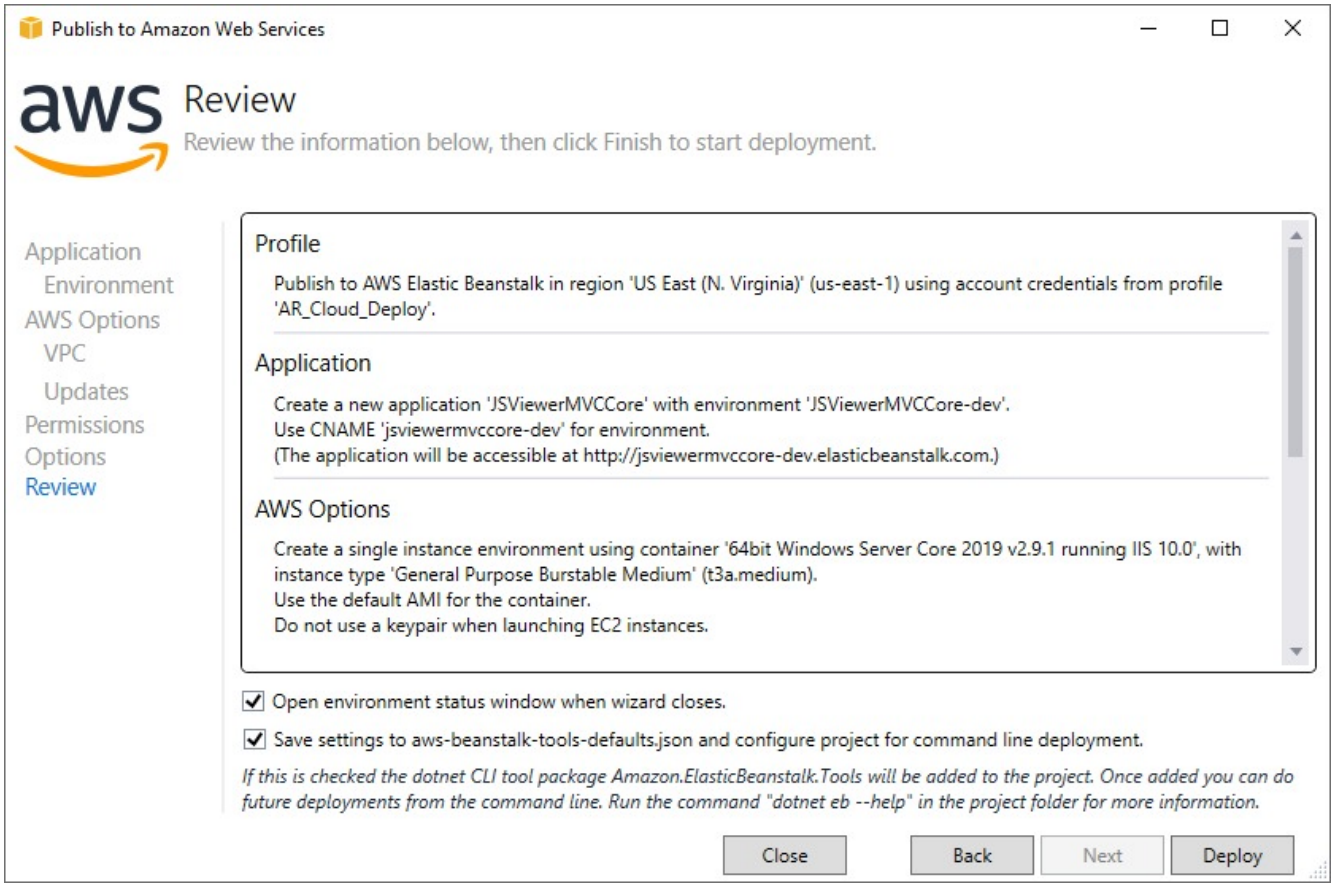
   Service Permissions
   Role: aws-elasticbeanstalk-service-role



8. In the **Application Options** window, change the **Project build configuration** from Debug|Any CPU to **Release|Any CPU**.
9. Set the **Framework** to net6.0.
10. Check the **Build self contained deployment bundle** option and then click **Next**.



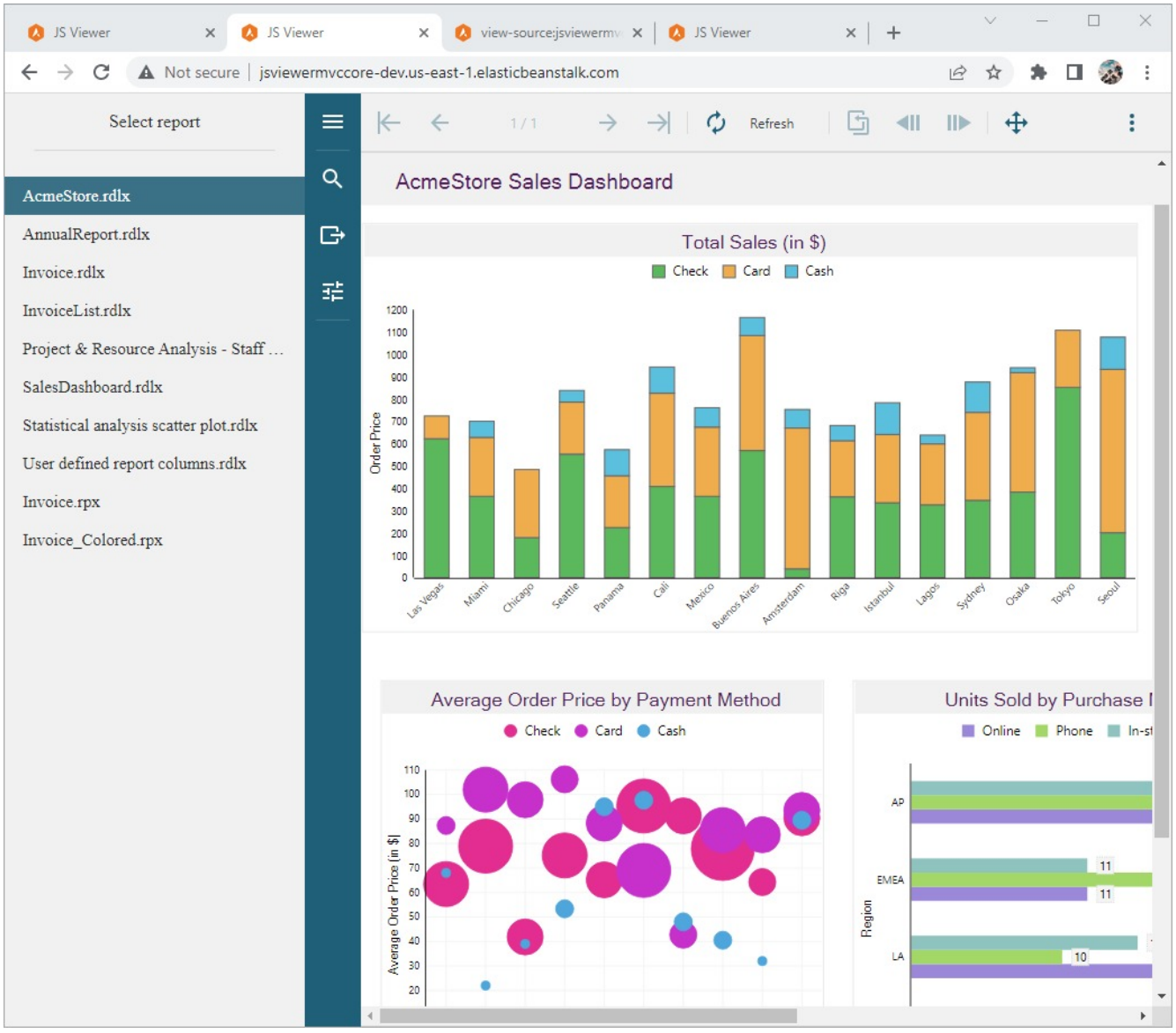
11. Go through the **Review** page to ensure you have selected all the required options as per your organization and application.
12. Click **Deploy**.
    Once you click Deploy, it takes several minutes to create the environment and launch your ActiveReports application to the cloud.



Once your application is successfully launched, you will see the following Event statement in the Event's tab in the Environment window:

> Successfully launched environment: JSViewerMVCCore-dev

Also, the Status of the environment would be Environment is healthy.



Now your application has successfully launched to the cloud. Now you can open it using the URL given at the top of the Environment window.

Clean up

To avoid any incurring charges if you are done working with Elastic Beanstalk for now, you can terminate your .NET environment. To terminate your **Elastic Beanstalk** environment:

1. Open the **Elastic Beanstalk console**, and in the Regions list, select your AWS Region
2. In the navigation pane, choose **Environments**, and then choose the name of your environment from the list.
3. Choose **Environment actions** and then choose **Terminate environment**.

Google Cloud

This section explains deploying on Google cloud.

Prerequisites

- Google Cloud SDK
- Google Cloud Tools for Visual Studio
- .Net 6.0 SDK for Windows Environment or .Net 6.0 for Linux environment
- Visual Studio 2022 or 2019, Visual Studio Code, or any other text editor
- ActiveReports 18 or later

It would also help to have basic familiarity with Google Cloud Platform (GCP), .NET Framework/.NET Core and Docker, with a GCP account to use.

Deploy on Google Cloud using a Windows Environment

In this tutorial we will be deploying our JSViewer_MVC sample on Google Cloud platform using **Google Compute Engine** environment. The approach we are using to deploy the application is to create a VM Instance and then publish the application using Web Server (IIS).

Setup Google Cloud

1. Sign in or create a new account on Google Cloud Platform.
2. Go to the Project Selector in the **Google Cloud Console** and create a new Project.
   Here we are naming the project as JSViewer MVC.
3. Make sure that billing is enabled for your Cloud project. Learn how to check if billing is enabled on a project.
4. From Console, go to the API Library and enable the 'Compute Engine API'.

Setup Visual Studio

1. Open Visual Studio 2019.
2. Go to **Extensions** > **Manage Extensions**.
3. Search online and install **Google Cloud Tools for Visual Studio**.

> **Note:** At the moment the Google Cloud Tools for Visual Studio is only supported for Visual Studio 2017 and Visual Studio 2019 which is why we will be using Visual Studio 2019.

Setup your application

1. Download the JSViewer_MVC sample from our WebSamples18 GitHub repository.
2. Open the application in Visual Studio and build and run the application.
3. Remove the **Reports** folder containing links the reports from the project since we will be embedding the reports in the application.
4. Now add a new folder and name it 'Reports'.
5. Add all the files from **JSViewerReports** in the **Reports** folder and set the **Build Action** as 'Embedded Resource'.
6. Run the application.

Create and configure a Compute Engine instance

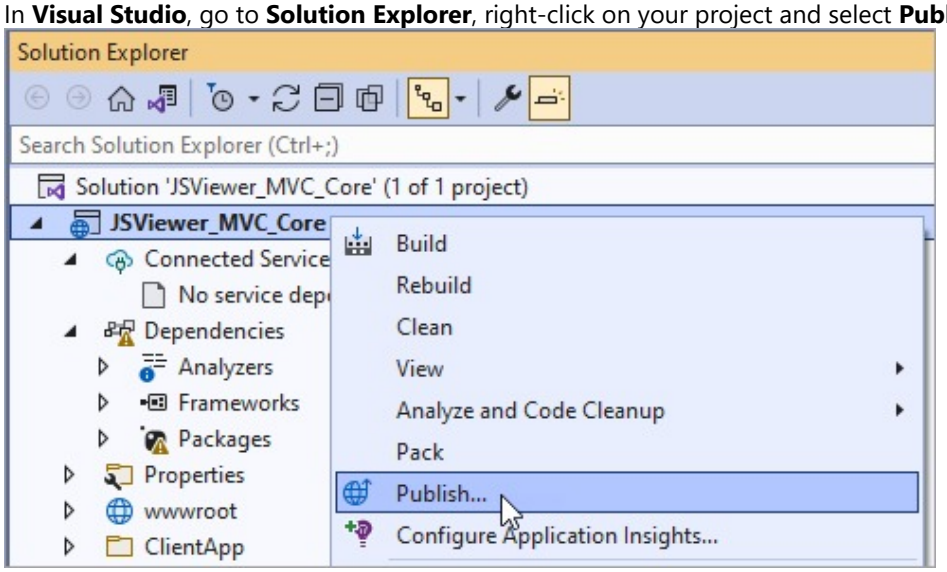
1. Navigate to Google Cloud Console from another browser tab or window, to https://console.cloud.google.com.
2. In the Google Cloud console, go to the ASP.NET Framework Cloud Marketplace page.
3. Click **Launch**.
4. Leave the settings set to their default values and click **Deploy**.
   Wait for the Compute Engine instance to deploy. It usually takes about 5 minutes to deploy.

5. Once finished, go to **Compute Engine** from the menu on the top left and select **VM Instances**. A new virtual machine should be added in the VM Instances list.
6. Open the External IP in a new browser tab. You should also see the default IIS website served from the instance.
7. Now we will create a Windows user and password to access the VM Instance in Visual Studio.
    1. Click on RDP >**Set Windows** password, then make note of the username.
    2. Copy the new Windows password and save it somewhere secure.
    3. Click **Close**.

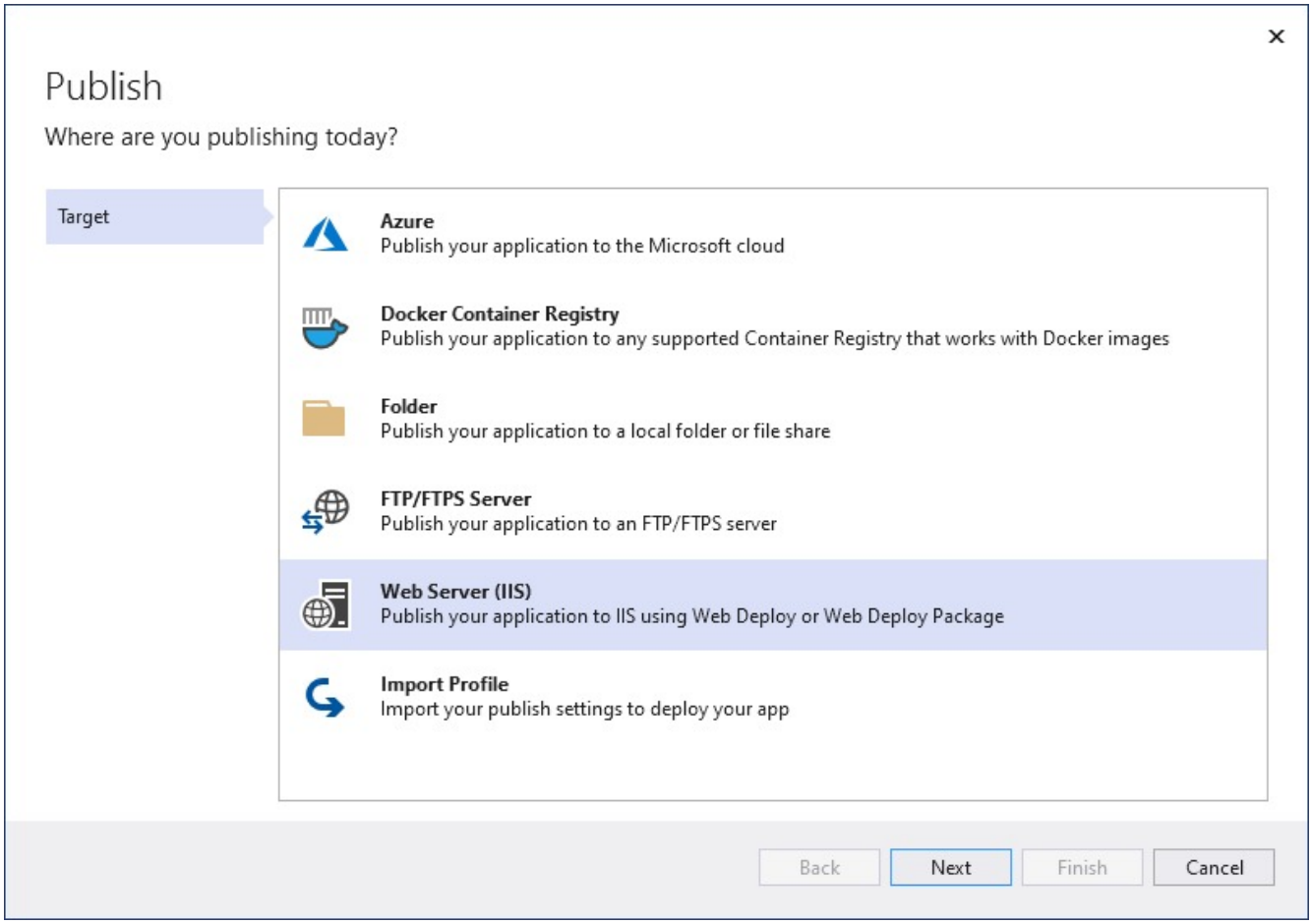
Deploy the application

Now that we have configured our Compute Engine Instance, we are ready to deploy our application.

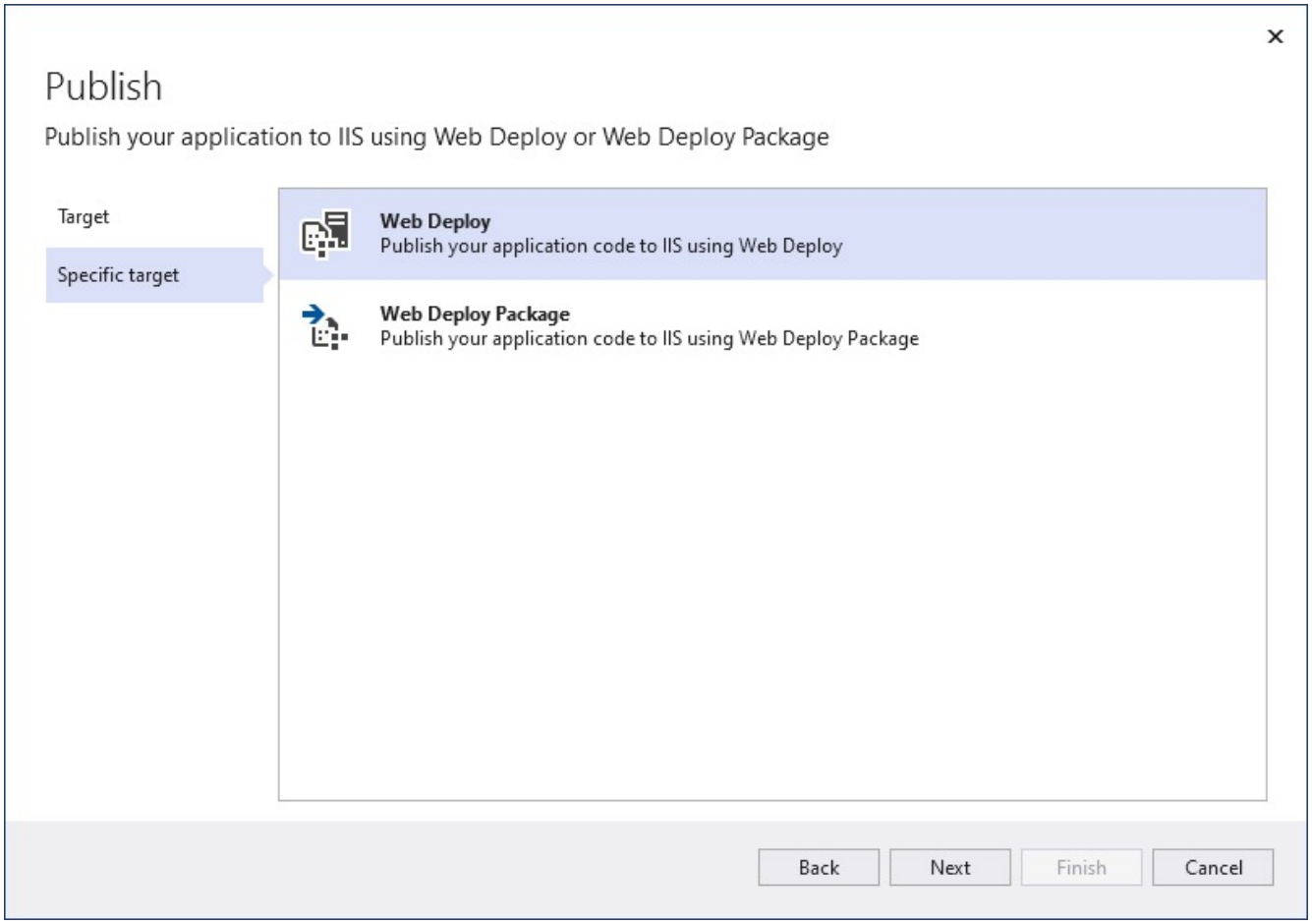
1. In **Visual Studio**, go to **Solution Explorer**, right-click on your project and select **Publish** in the context menu.



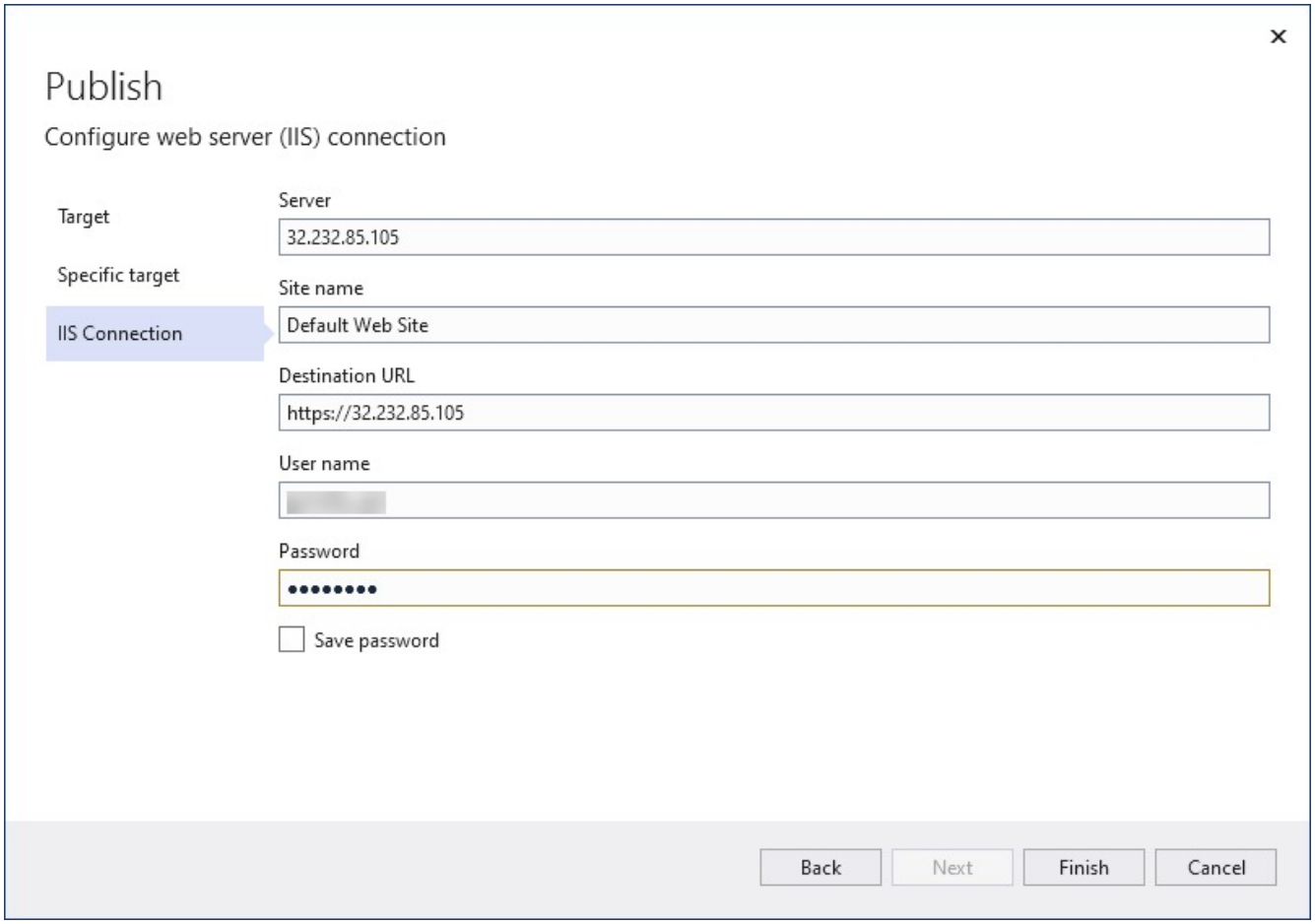
2. Click on **Web Server (IIS)**.



3. Select **Web Deploy**.

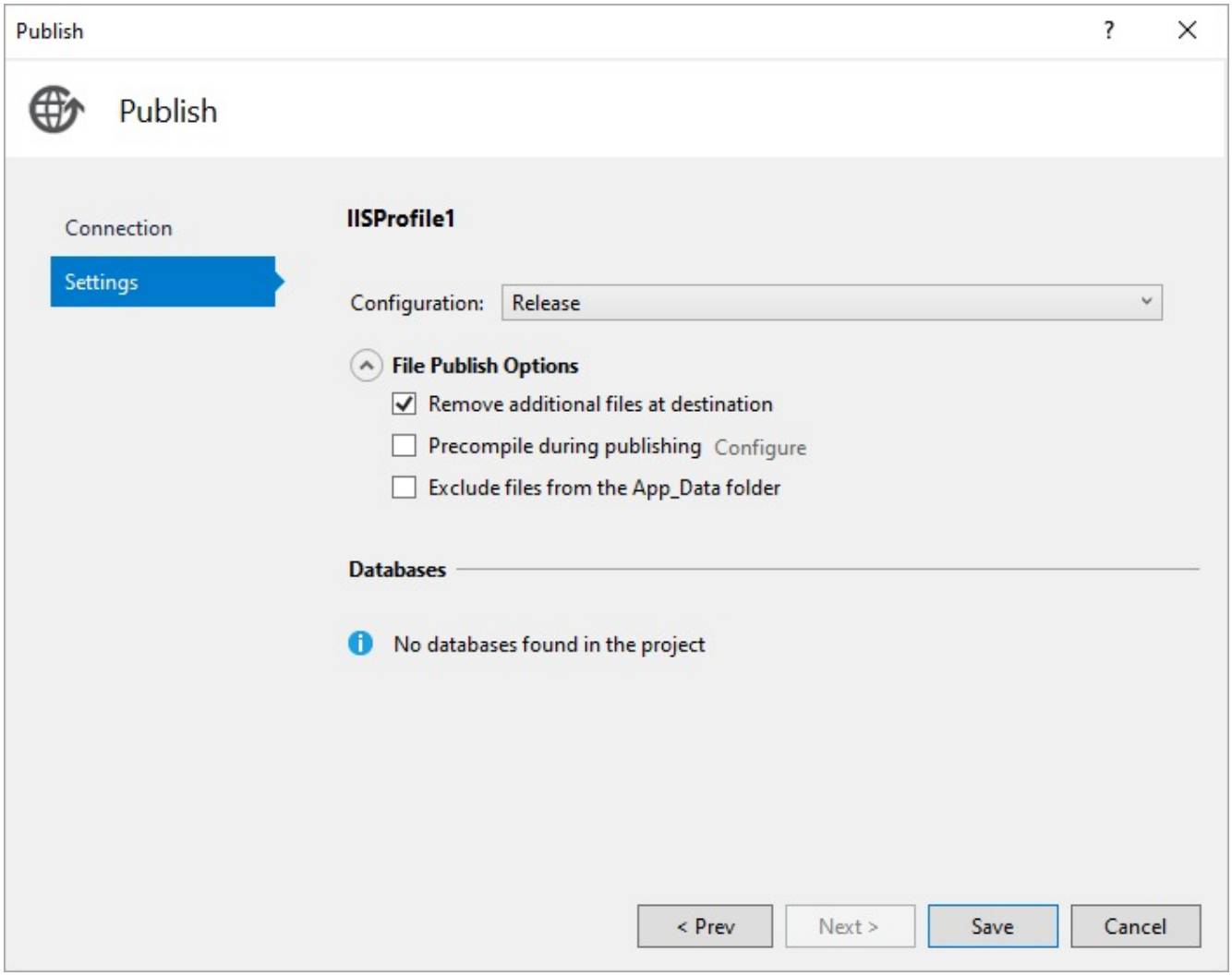


4. Configure the Web Server with the following details.

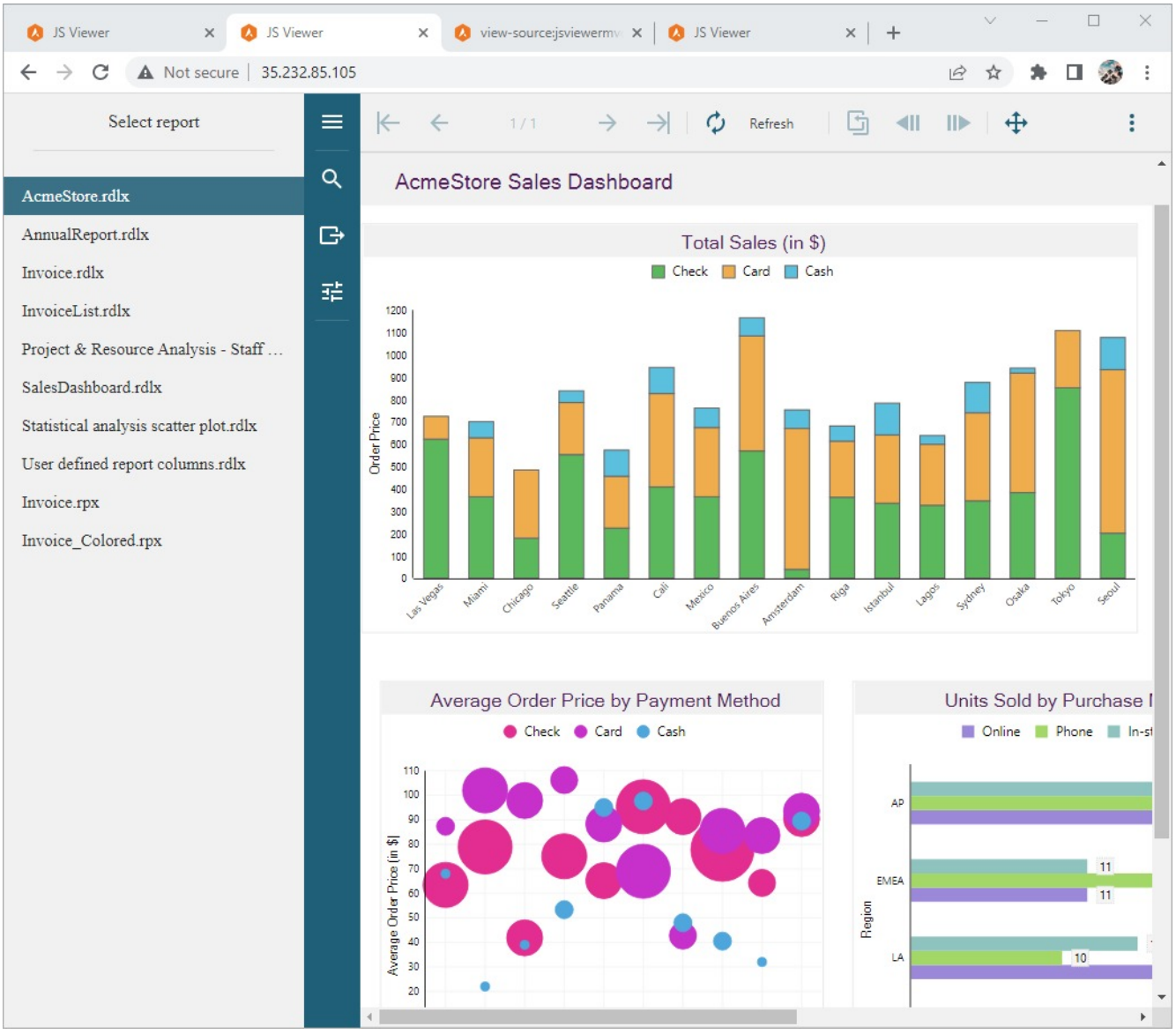


Server	The external IP address of your Compute Engine instance. This address can be found on the VM instances page in the Cloud Console.
Site Name	**Default Web Site** Note: The site name you provide here must match the name that appears in IIS Manager on your Compute Engine instance.
Username	The username of the Windows user account you created on your Compute Engine instance.
Password	The password of the Windows user account you created on your Compute Engine instance.
Destination URL	http://... . The destination URL is the address where your page will be accessible after it is deployed.

5. Click **Validate Connection**.
   Because the Microsoft IIS installation in your deployment uses a self-signed certificate by default, you will see a Certificate Error during the validation process. Check the box to Save this certificate for future sessions of Visual Studio, and click Accept to accept the certificate.
6. If your configuration is valid, click Settings. Click **File Publish Options**, and check **Remove** additional files at destination. This is important for later steps when you publish new web sites to the same Compute Engine instance.



7. Click **Publish** to deploy the application.



Clean Up

To avoid incurring charges, you can delete the instances or simply stop them, although keep in mind that stopped instances can still incur costs related to storage.
You can also delete your Cloud project to stop billing for all the resources used within that project.

1. In the Cloud console, go to the **Manage resources** page.
2. In the project list, select the project that you want to delete, and then click **Delete**.
3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

Deploy on Google Cloud using a Linux Environment

In this tutorial we will be deploying our JSViewer_MVC_Core sample on Google Cloud platform using **Google App Engine** flexible environment. App Engine flexible environment is based on Linux, it is useful for .NET Core applications.

Setup Google Cloud

1. Sign in or create a new account on Google Cloud Platform.
2. Go to the Project Selector in the **Google Cloud Console** and create a new Project.
   Here we are naming the project as JSViewer MVC.
3. Make sure that billing is enabled for your Cloud project. Learn how to check if billing is enabled on a project.
4. From Console, go to the API Library and enable the following APIs.
   - Cloud Build API
   - Google App Engine Flexible Environment
   - App Engine Admin API
5. Install and initialize the Google Cloud CLI.
6. Initialize the Application in App Engine, which we can either do through the console or using the CLI:

```
gcloud app create –JSViewer-MVC-Core
```

Setup Visual Studio

1. Open **Visual Studio 2019**.
2. Go to **Extensions** > **Manage Extensions**.
3. Search online and install **Google Cloud Tools for Visual Studio**.

Setup your application

1. Download the JSViewer_MVC_Core sample from our WebSamples18 GitHub repository.
2. Open the application in Visual Studio and build and run the application.
3. Remove the **Reports** folder containing links the reports from the project since we will be embedding the reports in the application.
4. Now add a new folder and name it 'Reports'.
5. Add all the files from **JSViewerReports** in the **Reports** folder and set the **Build Action** as 'Embedded Resource'.
6. Since the published docker might not contain the fonts used by reports we will add a custom font resolver in our ActiveReports application.
   1. Add a new folder **Fonts** in the project where we will the required fonts.
   2. Add a new .cs file '**CustomFontResolver.cs**' and make sure to set the **Build Action** to 'Embedded Resource'.
   3. Add the following code in the .cs file to the add the script.

CustomFontResolver.cs

```
using GrapeCity.Documents.Text.Windows;
public sealed class CustomFontResolver : GrapeCity.ActiveReports.IFontResolver
    {
        static readonly GrapeCity.Documents.Text.FontCollection _fonts = new
GrapeCity.Documents.Text.FontCollection();
        static CustomFontResolver()
        {
            _fonts.Clear();
            var assembly = Assembly.GetExecutingAssembly();
            var fontnames = assembly.GetManifestResourceNames().Where(str =>
str.EndsWith(".ttf"));
            foreach (var fontname in fontnames)
            {
                Stream stream = assembly.GetManifestResourceStream(fontname);
                _fonts.Add(GrapeCity.Documents.Text.Font.FromStream(stream));
            }
            _fonts.DefaultFont = _fonts.FindFamilyName("Arial");
```

```
        }
        public static GrapeCity.ActiveReports.IFontResolver Instance = new
CustomFontResolver();
        private CustomFontResolver() { }
        GrapeCity.Documents.Text.FontCollection
GrapeCity.ActiveReports.IFontResolver.GetFonts(string familyName, bool isBold, bool
isItalic)
        {
            var fonts = new GrapeCity.Documents.Text.FontCollection();
            var font = _fonts.FindFamilyName(familyName, isBold, isItalic);
            if (font != null) fonts.Add(font);
            fonts.Add(_fonts.DefaultFont);
            return fonts;
        }
    }
```

4. In Startup.cs, specify the FontResolver property in the JavaScript Viewers as in the following code example.

Startup.cs

```
app.UseReporting(settings =>
            {
                settings.FontResolver = CustomFontResolver.Instance;
                settings.UseEmbeddedTemplates(EmbeddedReportsPrefix,
Assembly.GetAssembly(GetType()));
                settings.UseCompression = true;
            });
```

7. To push your application to Google App Engine, you need to specify which environment and runtime it's going to use. To do that, you need to add an app.yaml and a dockerfile to your project file to your application.
In a perfect world scenario, we would just need to specify that we are going to use the Flex environment and the AspNetCore runtime. However, as of now there are no images tagged for .Net 3.x or higher versions we will use an image from Microsoft Container Registry so that we can deploy our application with .Net 6.0. We will be using a docker file so we must set the runtime environment as custom in the .

app.yaml

```
runtime: custom
env: flex
```

Since we have installed the Google Cloud Tool for Visual studio extension in Visual studio 2019. This can be easily done by:

a. Right Clicking on the project and Click on **Generate app.yaml and dockerfile**.
b. In the generated app.yaml change the runtime environment from aspnetcore to custom and add the following container configuration.

app.yaml

```
runtime: custom
env: flex
manual_scaling:
     instances: 1
```

```
resources:
     cpu: 1
     memory_gb: 0.5
     disk_size_gb: 10
```

Note: For testing purposes we have set the container specifications lower.

c. Clear the dockerfile and add the following text in the docker file:

Example Title

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build-env
WORKDIR /app
# Copy csproj and restore as distinct layers
COPY *.csproj ./
RUN dotnet restore
# Copy everything else and build
COPY . ./
RUN dotnet publish -c Release -o out
# Build runtime image
FROM mcr.microsoft.com/dotnet/aspnet:6.0
WORKDIR /app
COPY --from=build-env /app/out .
EXPOSE 8080
ENV ASPNETCORE_URLS=http://*:8080
ENTRYPOINT ["dotnet", "JSViewer_MVC_Core.dll"]
```

On Window's command prompt, you can create a new app.yaml in one line:

app.yaml

```
(echo runtime: aspnetcore & echo env: flex) > app.yaml
```

The app.yaml file will have to be copied over to the destination folder when you start the publishing process. You can include the file to the output through Visual Studio or editing the JsViewer_MVC_Core.csproj file:

JsViewer_MVC_Core.csproj

```
<Project Sdk="Microsoft.NET.Sdk.Web">
<PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
</PropertyGroup>
<ItemGroup>
    <None Include="app.yaml" CopyToOutputDirectory="Always" />
</ItemGroup>
</Project>
```

Publish the Application

Now we can publish the release version our application either using Visual Studio or CLI. You can use the following command to publish the application using command prompt.

```
dotnet publish -c Release
```

To publish using Visual Studio:

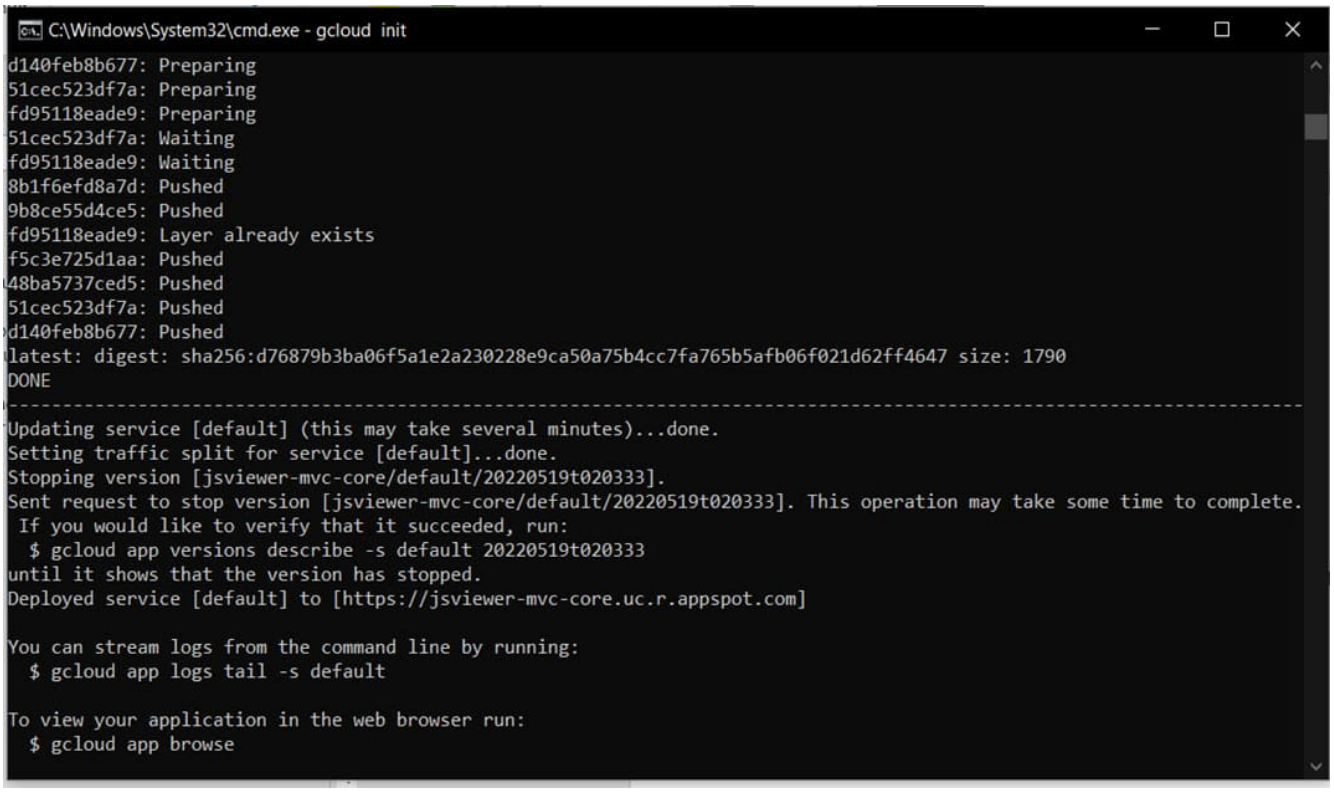
1. Right-click on the Project in Visual Studio and Click on Publish.
2. Set the **Target** as 'Folder'.
3. Set the Folder Location as: "bin\Release\net6.0\publish\"
4. Click Finish and now Click Publish.

Deploy the application

Now that we have set up everything it is time to deploy the application.

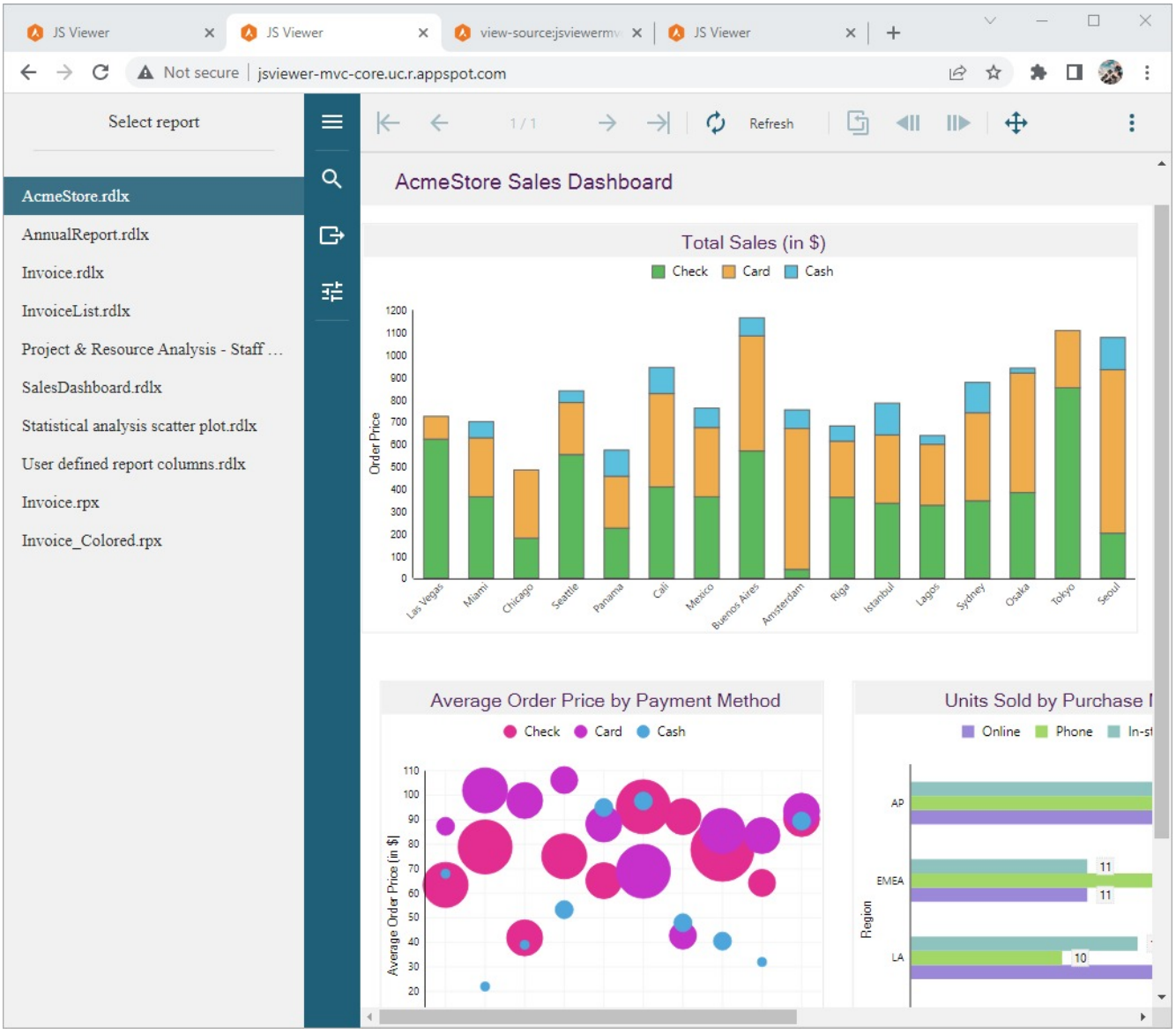
1. Open command prompt and set the directory as the project's folder.
2. Run the following command.

```
gcloud app deploy
```



It will take 10-20 minutes to deploy the application on Google Cloud.
Once deployed you can open the deployed website using the following command:

```
gcloud app browse
```



Clean Up

To avoid incurring charges, you can delete your Cloud project to stop billing for all the resources used within that project.

1. In the Cloud console, go to the Manage resources page.
2. In the project list, select the project that you want to delete, and then click Delete.
3. In the dialog, type the project ID, and then click Shut down to delete the project.

Deploy Application Using Docker

In this tutorial, we will be deploying the JSViewer_MVC_Core sample using Docker on Windows environment. Download the JSViewer_MVC_Core sample from our WebSamples18 GitHub repository.

Create Docker File

To create a docker file, follow the steps as described below:

1. Create a file named 'Dockerfile' and add it to the above sample project (make sure no extension is added like .txt is added). See creating Dockerfile on Windows for more information.
2. Use the .Net SDK image (Docker Official Image) and install dependencies like node.js in it.

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
# installing node
RUN curl -sL https://deb.nodesource.com/setup_12.x | bash - \
 && apt update \
 && apt install -y nodejs
```

3. Create a new working folder named 'src' and copy the files from our project in the local machine to 'src' folder. Also, install the required project dependencies and build the application using the `dotnet restore` and `dotnet build` commands, followed by `dotnet publish` to publish the same.

```
WORKDIR /src
COPY . .
ENV PATH="$PATH:/root/.dotnet/tools"
RUN dotnet restore "./JSViewer_MVC_Core.csproj"
RUN dotnet build "JSViewer_MVC_Core.csproj" -c Release -o /app/build
FROM build AS publish
RUN dotnet publish "JSViewer_MVC_Core.csproj" -c Release -o /app/publish
```

4. Once done, create a working directory called 'app' and copy the contents of the published application from the layer having an alias name as 'publish' and run the application as follows.

```
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "JSViewer_MVC_Core.dll"]
```

Therefore, we now have a docker file with the following commands.

```
FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
# installing node
RUN curl -sL https://deb.nodesource.com/setup_12.x | bash - \
 && apt update \
 && apt install -y nodejs
WORKDIR /src
COPY . .
ENV PATH="$PATH:/root/.dotnet/tools"
RUN dotnet restore "./JSViewer_MVC_Core.csproj"
RUN dotnet build "JSViewer_MVC_Core.csproj" -c Release -o /app/build
FROM build AS publish
RUN dotnet publish "JSViewer_MVC_Core.csproj" -c Release -o /app/publish
# final stage/image
FROM mcr.microsoft.com/dotnet/aspnet:6.0-focal
RUN apt-get update; \

ENV ASPNETCORE_URLS="http://+:5000"
```

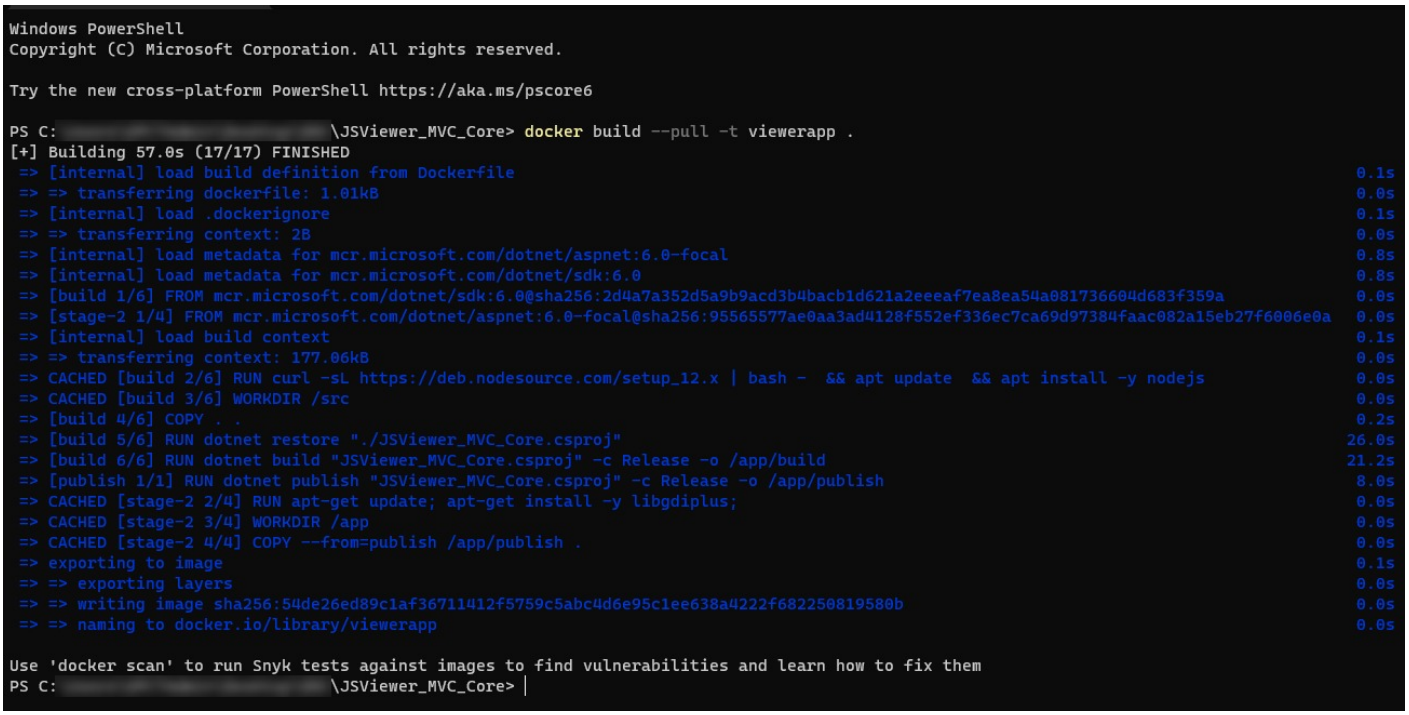
```
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "JSViewer_MVC_Core.dll"]
```

Create an image using the above-created Docker File

Open the command prompt and build an image named 'viewerapp' from the above-created docker file and use the `docker build` command as follows.

```
docker build --pull -t viewerapp
```

The console screenshot on executing the above command is shown.



For more information on the above command, please refer to the link:
https://docs.docker.com/reference/cli/docker/image/build/

Execute the above-created image inside the docker container

We will use the following `docker run` command and would bind the port 8090 of the host system to the container's 5000 port where the above specified JSViewer application is running.

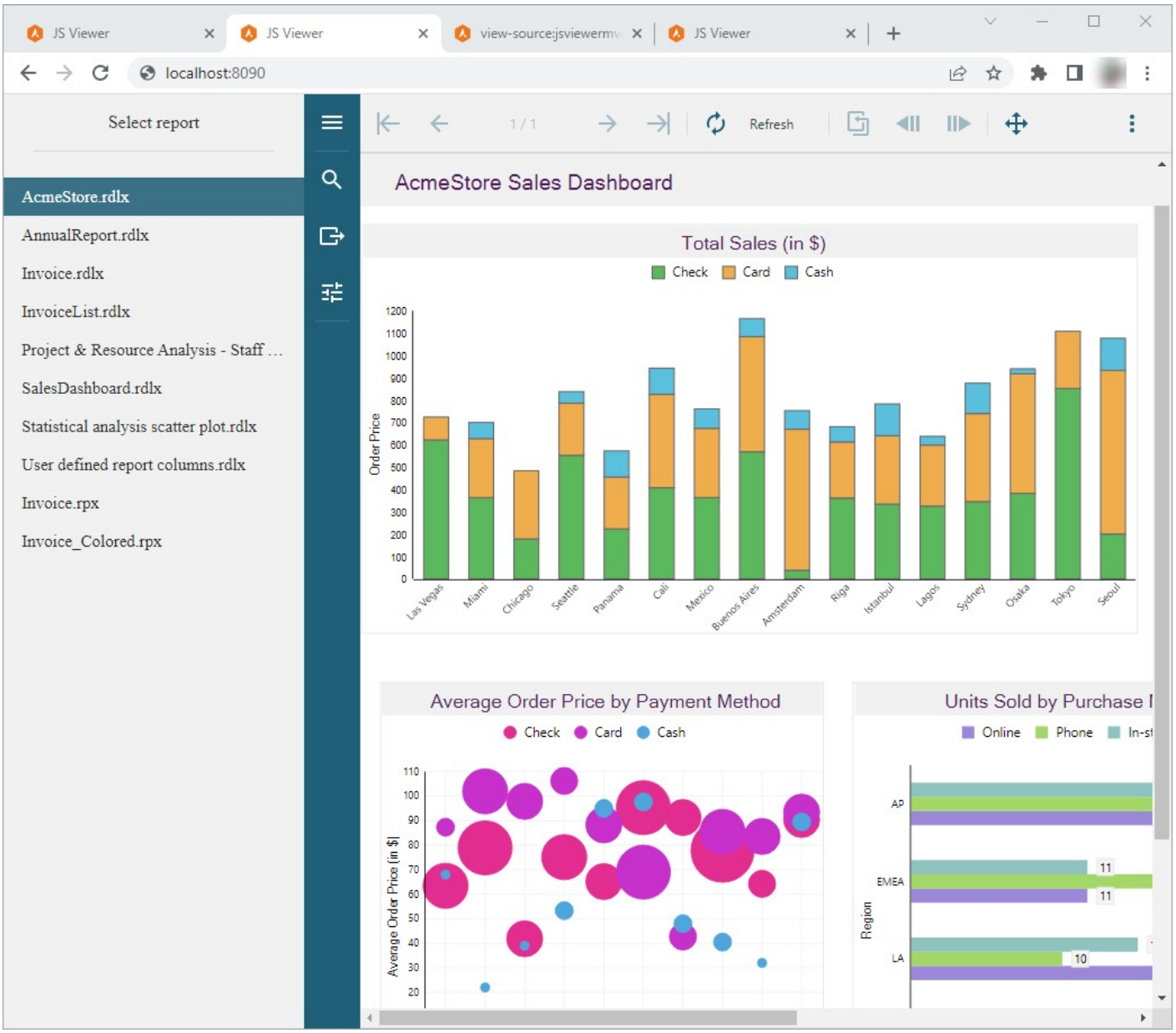
```
docker run -dp 8090:5000 viewerapp
```

The console screenshot on executing the above command is shown.

```
PS C:                                    \JSViewer_MVC_Core> docker run -dp 8090:5000 viewerapp
0a508f1d64670c937b6a33df3704903cd7498c0beadb8c85fb03c62506812566
PS C:                                    \JSViewer_MVC_Core>
```

For more information on the above command, please refer to the link:
https://docs.docker.com/reference/cli/docker/container/run/

Now, try opening the localhost:8090 in the browser of the host system, the output should be as follows.



Developers

ActiveReports.NET is mainly a product for developers. The developers are responsible for most of the use-cases related to ActiveReports, including:

- creating and storing reports, especially with Standard license and Code-Based Section reports.
- binding data sources at runtime and the custom data providers the data sources.
- creating and customizing the applications.
- licensing the applications.
- exporting reports applications.

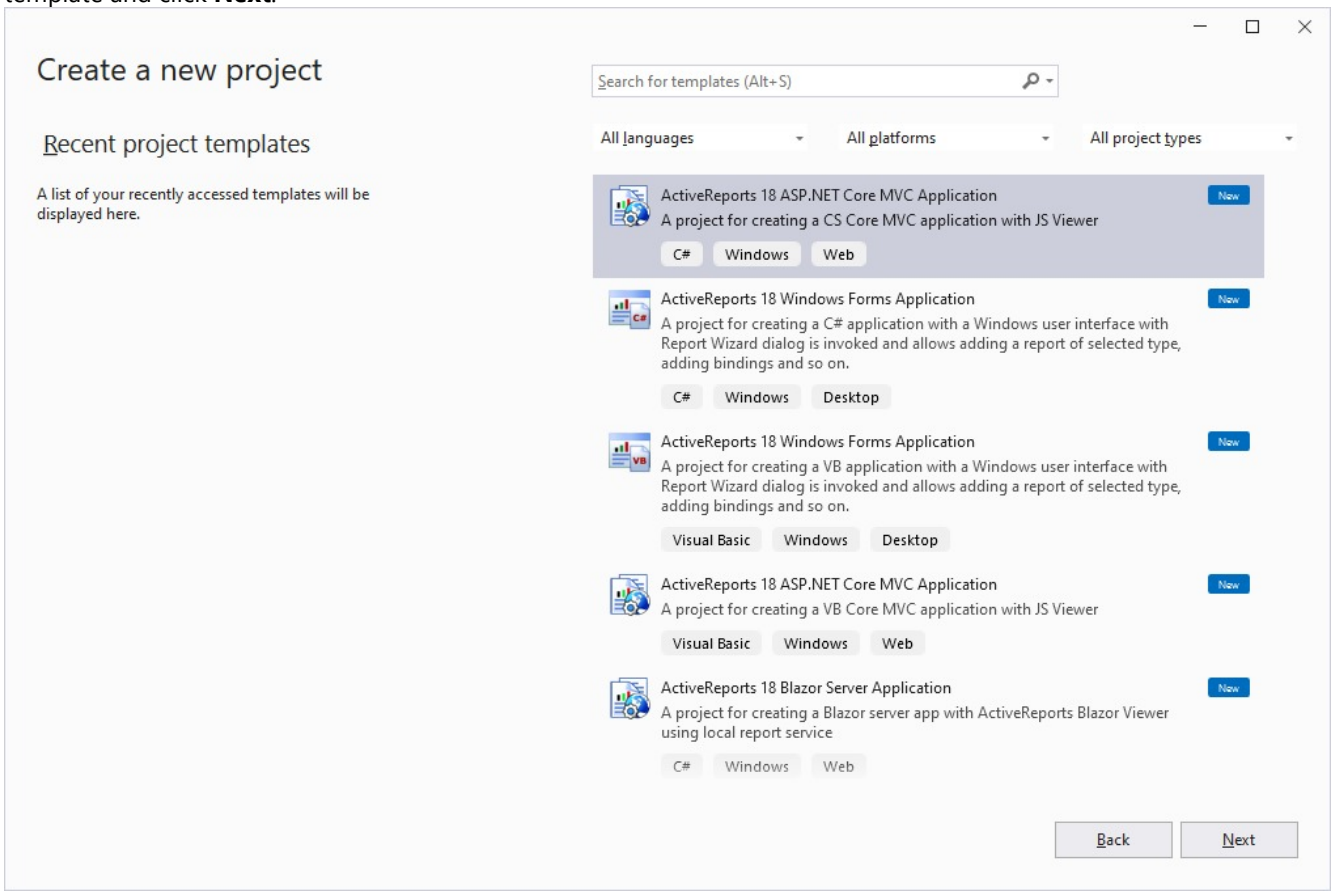
Some reports creation can be delegated to independent Report Authors, otherwise this task is primarily for developers. Some customization, configuration, and deployment tasks can be delegated to DevOps too.

In this section, let's dive into the information required by developers to perform their tasks.

Quick Start

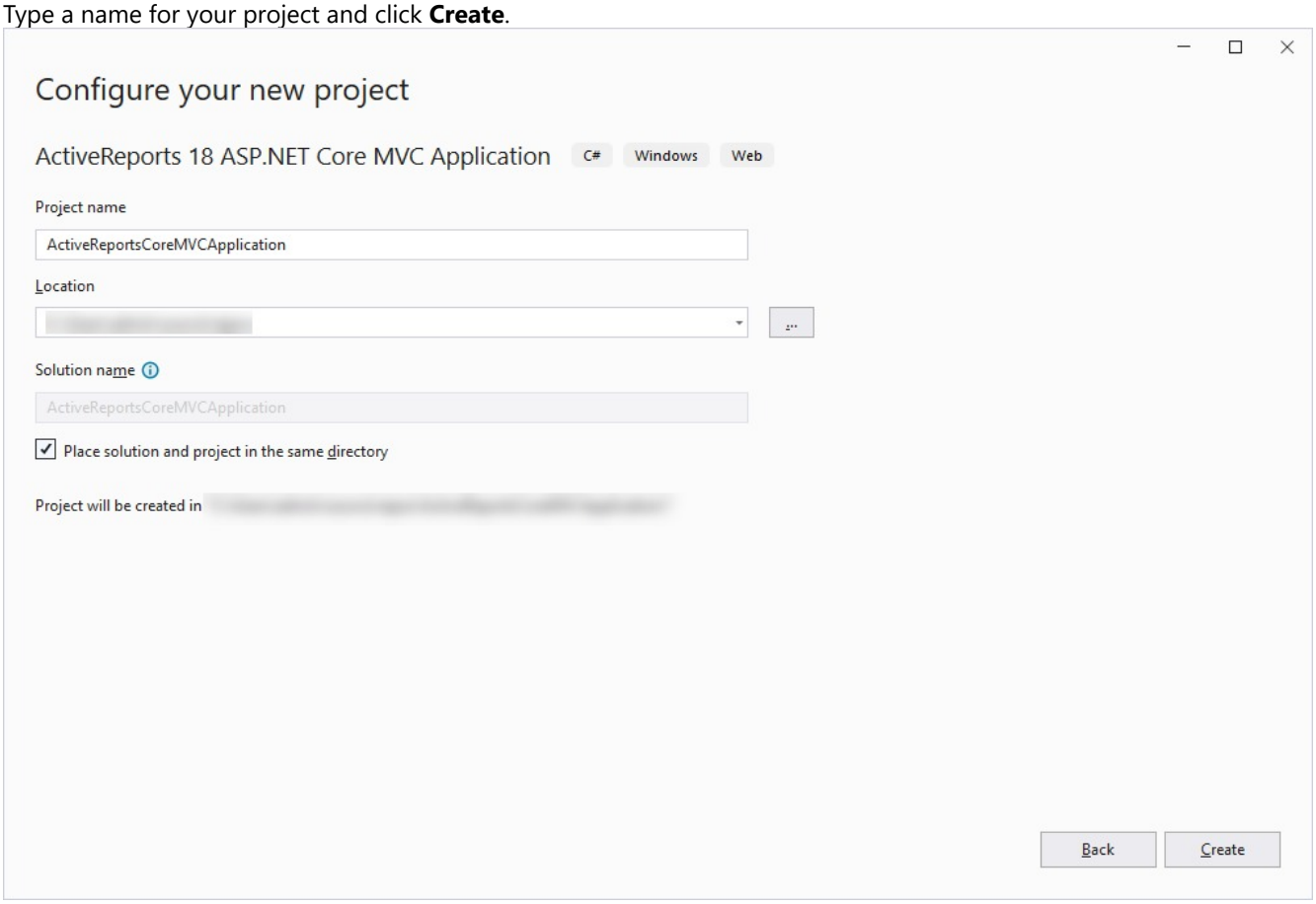
Quickly begin using ActiveReports by following the steps below.

1. Install ActiveReports.
2. In **Microsoft Visual Studio 2022** (version 17.0 or above), select **ActiveReports 18 ASP.NET Core MVC Application** template and click **Next**.



For complete list of built-in templates, see the Project Templates topic.

3. Type a name for your project and click **Create**.



4. Select RDLX from report types and click **Finish**:
   - RDLX
   - RDLX Dashboard
   - Page
   - Section

   Check Report Types topic that will help you choose a report type.
5. Open the default 'Report.rdlx' report from the 'Reports' folder and design. You can also choose to add report types from the context menu of the 'Reports' folder.
   - For information on designing a report, see Report Authors: Designer Components.
   - For information on viewing a report, see Report Readers: Viewer Components.
6. Make sure to set the **Build Action** property of the report to 'Embedded resource'.
7. Modify **index.cshtml** to provide the name of the report you want to preview in **viewer.openReport()** method: `viewer.openReport("Report.rdlx");`

   The complete index.cshtml looks as below.

index.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="~/jsViewer.min.css" rel="stylesheet">
    <link href="~/index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%; overflow-x: hidden">
        <div style="float:right;width:100%" id="viewerContainer">
        </div>
    </div>
    <script type="text/javascript" src="~/jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer'
            });
            viewer.openReport("Report.rdlx");
        }
    </script>
</body>
</html>
```

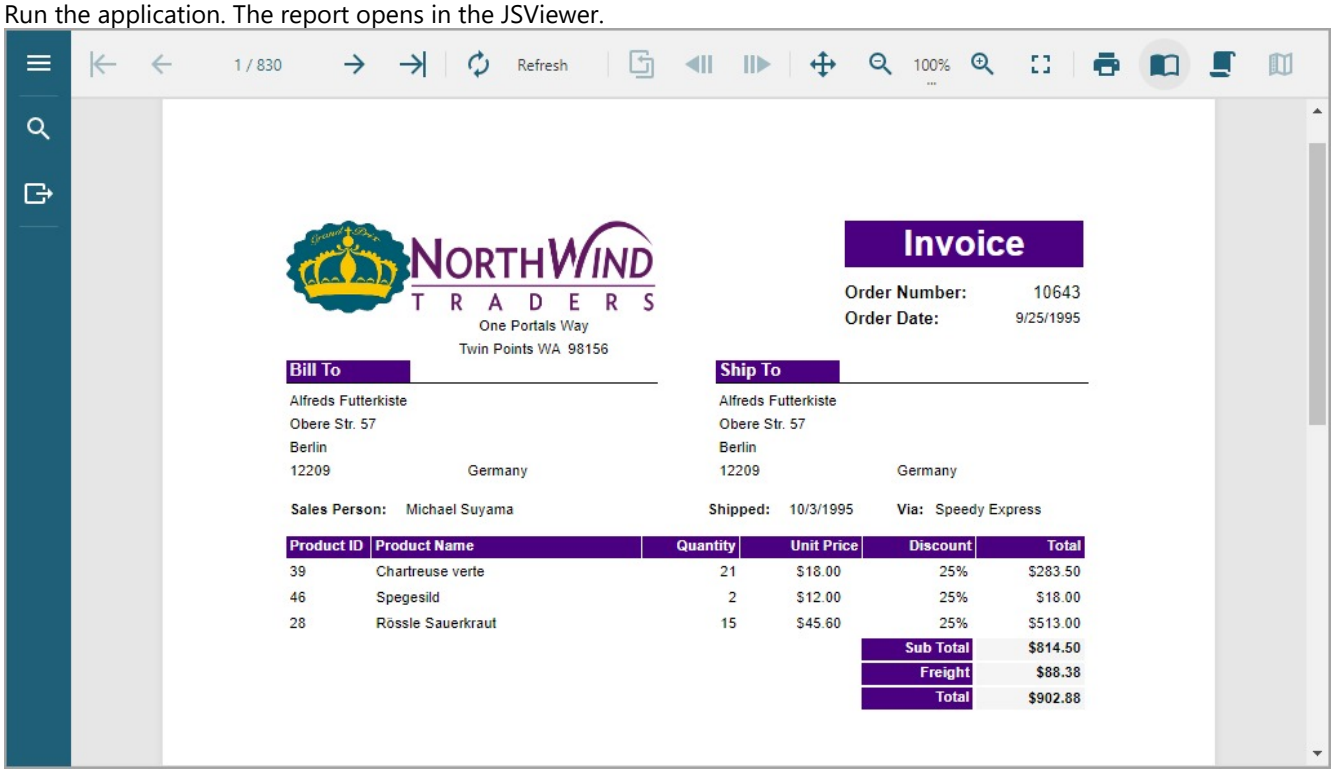
In **Startup.cs**, the root directory containing the report is resolved using config.UseFileStore(rootDir). The complete **Startup.cs** looks like below.

Startup.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using System;
using System.IO;
namespace ActiveReportsCoreMVCApplication
{
    public class Startup
    {
        // This method gets called by the runtime. Use this method to add services to the
container.
        public void ConfigureServices(IServiceCollection services)
        {
            services
                .AddLogging(config =>
                {
                    // Disable the default logging configuration
                    config.ClearProviders();
                    // Enable logging for debug mode only
```

```
                    if (Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT") ==
Environments.Development)
                    {
                        config.AddConsole();
                    }
                })
                .AddReportViewer()
                .AddMvc(option => option.EnableEndpointRouting = false);
        }
        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            app.UseReportViewer(settings =>
            {
                var reportsFolder = Path.Combine(env.ContentRootPath, "Reports");
                settings.UseFileStore(new DirectoryInfo(reportsFolder));
            });
            app.UseStaticFiles();
            app.UseMvc();
        }
    }
}
```

8. Run the application. The report opens in the JSViewer.



Breaking Changes

Breaking changes from ActiveReports 17 to ActiveReports 18

Moved Visual SQL Query Designer (VQD) to a new assembly
The GrapeCity.ActiveReports.Design.QueryDesignerImpl class was moved from the **GrapeCity.ActiveReports.Design.Win ('GrapeCity.ActiveReports.Design.Win Namespace' in the on-line documentation)** assembly to the **GrapeCity.ActiveReports.QueryDesigner.Implementation ('GrapeCity.ActiveReports.QueryDesigner.Implementation Namespace' in the on-line documentation)** namespace in the newly created **MESCIUS.ActiveReports.QueryDesigner ('MESCIUS.ActiveReports.QueryDesigner Assembly' in the on-line documentation)** assembly.
Removed data source dialog automatic launch

In ActiveReports 17, the **LaunchDataSourceWizard** flag was added. A customer used this flag to enable or disable the automatic launch of the data source dialog when creating a report.
In ActiveReports 18, the new Report Wizard has been implemented to combine the report creation dialog with the report data binding steps. The **LaunchDataSourceWizard** setting has become unnecessary and is now simply ignored.
`<appSettings>`
`  <add key="LaunchDataSourceWizard" value="true"/>`
`</appSettings>`

Removed RDL report type from the WinForms and Web Designers
The legacy RDL was removed, and RDL Multi-Section was renamed to RDLX. The RDL report type is no longer available for new reports in ActiveReports 18. RDL reports, created in the previous versions of ActiveReports, are automatically converted to the RDLX format when opened in the Designer. The **GrapeCity.ActiveReports.Design.DesignerReportType.Rdl** enum is now Obsolete. We recommend that you postpone upgrade and contact our Support Team in case backward compatibility with old AR versions is required.
Calendar Sample Changes
The Calendar sample is undergoing a number of changes related to the public API of the sample controls. If you are unable to assemble the sample, contact our Support Team.
Section Report design-time related support
TypeConverters, related to section report, have been hidden. If you encounter problems with using these TypeConverters in your project, contact our Support Team.
Transition from GrapeCity to MESCIUS
GrapeCity and **Gcef** in all ActiveReports packages names have been changed to **MESCIUS**. To manage this transition and update the packages, you can use the ActiveReports File Converter or perform the update manually by following the reference migration steps.
.NET 8 in all Windows Forms and Web Samples
All Windows Forms and Web Samples have been retargeted to .NET 8.
DataProvidersFactory class is moved to GrapeCity.ActiveReports.Rendering.Data
The DataProvidersFactory class is moved from the **DataProviders ('GrapeCity.ActiveReports.ReportsCore.Data.DataProviders Namespace' in the on-line documentation)** assembly to the **GrapeCity.ActiveReports.Rendering.Data ('GrapeCity.ActiveReports.Rendering.Data Namespace' in the on-line documentation)** namespace in the **MESCIUS.ActiveReports.Core.Rendering ('MESCIUS.ActiveReports.Core.Rendering Assembly' in the on-line documentation)** assembly

CommonDbConnectionAdapter class is moved to GrapeCity.ActiveReports.ReportsCore.Data

The CommonDbConnectionAdapter class is moved from the **DataProviders ('GrapeCity.ActiveReports.ReportsCore.Data.DataProviders Namespace' in the on-line documentation)** assembly to the ActiveReports assembly in the **GrapeCity.ActiveReports.ReportsCore.Data ('GrapeCity.ActiveReports.ReportsCore.Data Namespace' in the on-line documentation)** namespace.

System.Data.SqlClient is replaced with Microsoft.Data.SqlClient for .NET Core/.NET Standard assemblies versions

The dependency for .NET Standard assemblies versions of the AR.Core has been changed. Now Microsoft.Data.SqlClient is used for the MSSQL data provider by default. This change affects users, developing .NET Core applications with SQL connections, and does not require any special migration steps. For details, see Troubleshooting.

Breaking changes from ActiveReports 16 to ActiveReports 17

Watermark Group of Settings Changes

The watermark group of settings has been removed from **PdfExportOptions.PdfReExclusiveOptions ('PdfExportOptions.PdfReExclusiveOptions Class' in the on-line documentation)** class. To set a watermark, you should use the **Watermark ('Watermark Property' in the on-line documentation)** property in the **PdfExportOptions ('PdfExportOptions Class' in the on-line documentation)** class instead.

Removed **GrapeCity.ActiveReports.Document** assembly and NuGet package

> **GrapeCity.ActiveReports.Document ('GrapeCity.ActiveReports.Document Namespace' in the on-line documentation)** assembly and NuGet package are merged with GrapeCity.ActiveReports assembly and NuGet package.

WebDesigner API Changes

The WebDesigner Settings API is now updated for simplified mutable settings definition as shown:

Old API

```
designerCore.init({
  /* < ... > */
  mutableAppSettings: ['units', 'fonts'],
});
```

New API

```
designerCore.init({
  /* < ... > */
  mutableAppSettings: {
    fonts: true;
    userPreferences: {
      units: true;
    }
  },
});
```

Removed dependency on System.Drawing (GDI+)

We have removed dependency on System.Drawing from our code for functionalities such as image handling, text measuring, printing, etc.

For example, API dependencies such as:
System.Drawing.Font and **System.Drawing.Image** in Section report have been removed from **System.Drawing.Common**.
GDI references from **GrapeCity.ActiveReports.Document.SectionReport**, **GrapeCity.ActiveReports.Document.SectionDocument**, and **GrapeCity.ActiveReports.Document.PageDocument** classes have been removed.
See System.Drawing.Common only supported on Windows. This makes the Section reports engine run under Linux independent of libgdiplus library. You may need to make some changes in the code and scripts for the applications to work. All API changes are listed next.

API Changes

GrapeCity.ActiveReports.SpreadBuilder.dll **changes**

- Class **GrapeCity.SpreadBuilder.Printing.PageSetup ('PageSetup Class' in the on-line documentation)**
  old property: public System.Drawing.Printing.PaperKind PaperSize
  new property: public **GrapeCity.SpreadBuilder.Printing.PaperKind PaperSize ('PaperSize Property' in the on-line documentation)**
- Class **GrapeCity.SpreadBuilder.DDSheet ('DDSheet Class' in the on-line documentation)**
  Removed obsolete methods:
  - public void AddImage(System.Drawing.Image img, ImageInfo imageOptions,
    System.Drawing.Color lineColor, System.Drawing.Color backColor,
    short colL, short dxL, short rwT, short dyT,
    short colR, short dxR, short rwB, short dyB,
    string hyperlink)
  - public void AddImage(System.Drawing.Image img, ImageInfo imageOptions,
    System.Drawing.Color lineColor, System.Drawing.Color backColor,
    int columnLeft, short dxL, int rowTop, short dyT,
    int columnRight, short dxR, int rowBottom, short dyB,
    string hyperlink)
  - public void AddImage(byte[] imageData, bool isMetafile, SizeF sizeInInches, ImageInfo imageOptions,
    System.Drawing.Color lineColor, System.Drawing.Color backColor,
    int columnLeft, short dxL, int rowTop, short dyT,
    int columnRight, short dxR, int rowBottom, short dyB,
    string hyperlink)

  Added new method:

  - public void **AddImage ('AddImage Method' in the on-line documentation)**(byte[] imageData, ImageInfo imageOptions,
    System.Drawing.Color lineColor, System.Drawing.Color backColor,
    int columnLeft, short dxL, int rowTop, short dyT,
    int columnRight, short dxR, int rowBottom, short dyB,
    string hyperlink)

- Class **GrapeCity.SpreadBuilder.Workbook ('Workbook Class' in the on-line documentation)**
  - Removed old default constructor
  - Added new **constructor ('Workbook Constructor' in the on-line documentation)**:
    /// <param name="measureString">Function to measure string in case of auto-height row behavior.
    /// Parameters: string, font, fontsize, maxwidth.</param>
    public Workbook(Func<string, StringMeasurementParams, SizeF> measureString = null)

GrapeCity.ActiveReports.Chart.dll ('MESCIUS.ActiveReports.Chart Assembly' in the on-line documentation) changes

- Class **GrapeCity.ActiveReports.Chart.BackdropItem ('BackdropItem Class' in the on-line documentation)**
  - old constructor: public BackdropItem(System.Drawing.Image picture, PicturePutStyle pictureAlignment)
    new constructor: public BackdropItem(GrapeCity.ActiveReports.Chart.Drawing.Image picture, PicturePutStyle pictureAlignment)
  - old constructor: public BackdropItem(System.Drawing.Image picture, PicturePutStyle pictureAlignment, AntiAliasMode antiAliasMode)
    new constructor: public BackdropItem(GrapeCity.ActiveReports.Chart.Drawing.Image picture, PicturePutStyle pictureAlignment, AntiAliasMode antiAliasMode)
- Class **GrapeCity.ActiveReports.Chart.FontInfo ('FontInfo Class' in the on-line documentation)**
  - old property: public System.Drawing.Font
    new property: public GrapeCity.ActiveReports.Chart.Drawing.Font Font
  - old constructor: public FontInfo(Color color, System.Drawing.Font font, float angle)
    new constructor: public FontInfo(Color color, GrapeCity.ActiveReports.Chart.Drawing.Font font, float angle)

    - old constructor: public FontInfo(Color color, System.Drawing.Font font)
      new constructor: public FontInfo(Color color, GrapeCity.ActiveReports.Chart.Drawing.Font font)

- Class **GrapeCity.ActiveReports.Chart.GraphXmlSerializationContext ('GraphXmlSerializationContext Class' in the on-line documentation)**
    - old property: public IDictionary<string, System.Drawing.Font> Fonts
      new property: public IDictionary<string, GrapeCity.ActiveReports.Chart.Drawing.Font> Fonts
- Class **GrapeCity.ActiveReports.Chart.SharpGraphCore ('SharpGraphCore Class' in the on-line documentation)**
  Removed methods:
    - public DrawContent(System.Drawing.Graphics graphics, Rectangle rectangle)
    - public HitTest HitTest(int x, int y)
- Class **GrapeCity.ActiveReports.Chart.Graphics.Backdrop ('Backdrop Class' in the on-line documentation)**
    - old property: public System.Drawing.Image Picture
      new property: public GrapeCity.ActiveReports.Chart.Drawing.Image Picture
    - old property: public System.Drawing.HatchStyle Pattern
      new property: public GrapeCity.ActiveReports.Chart.Drawing.HatchStyle Pattern
    - old constructor: public Backdrop(System.Drawing.Image picture, PicturePutStyle pictureAlignment)
      new constructor: public Backdrop(GrapeCity.ActiveReports.Chart.Drawing.Image picture, PicturePutStyle pictureAlignment, AntiAliasMode antiAliasMode)
- Class **GrapeCity.ActiveReports.Chart.Graphics.TextStyle ('TextStyle Class' in the on-line documentation)**
    - old property: public System.Drawing.Font Font
      new property: public GrapeCity.ActiveReports.Chart.Drawing.FontFont Font

Printing API changes

- GrapeCity.ActiveReports.Extensibility.Printing namespace has been removed, instead use **GrapeCity.ActiveReports.Printing ('GrapeCity.ActiveReports.Printing Namespace' in the on-line documentation)**.
- GrapeCity.ActiveReports.Extensibility.Printing.Printer has been removed, instead use **GrapeCity.ActiveReports.Win.Printing.Printer ('Printer Class' in the on-line documentation)** and **GrapeCity.ActiveReports.Printing.Printer ('Printer Class' in the on-line documentation)** classes.
- The type for the following classes is changed to **GrapeCity.ActiveReports.Printing.Printer ('Printer Class' in the on-line documentation)**:
    - SectionDocument.Printer
    - PageDocument.Printer
- The following classes have been moved to **GrapeCity.ActiveReports.Win.Printing ('GrapeCity.ActiveReports.Win.Printing Namespace' in the on-line documentation)** namespace:
    - ActiveReportPrintController
    - DDPaperSize
- Removed some System.Drawing.Printing enumerations as follows:
    - Removed PageSettings.PaperSourceKind enumeration, new type is **GrapeCity.ActiveReports.Printing.PaperSourceKind ('PaperSourceKind Enumeration' in the on-line documentation)**.
    - Removed PageSettings.PaperKind type enumeration, new type is **GrapeCity.ActiveReports.Printing.PaperKind ('PaperKind Enumeration' in the on-line documentation)**.
    - Removed PageSettings.Duplex type enumeration, new type is **GrapeCity.ActiveReports.Printing.Duplex ('Duplex Enumeration' in the on-line documentation)**.

GrapeCity.ActiveReports.SectionReportModel ('GrapeCity.ActiveReports.SectionReportModel Namespace' in the on-line documentation) Changes

- Removed some System.Drawing references from public API:

- System.Drawing.Font, new type is **GrapeCity.ActiveReports.Document.Drawing.Font ('Font Class' in the on-line documentation)**.
- System.Drawing.Image, new type is **GrapeCity.ActiveReports.Document.Drawing.Image ('Image Class' in the on-line documentation)**.
- System.Drawing.FontStyle, new type is **GrapeCity.ActiveReports.Document.Drawing.FontStyle ('FontStyle Enumeration' in the on-line documentation)**.
- System.Drawing.GraphicsUnit, new type is **GrapeCity.ActiveReports.Document.Drawing.GraphicsUnit ('GraphicsUnit Enumeration' in the on-line documentation)**.
- System.Drawing.StringAlignment, new type is **GrapeCity.ActiveReports.SectionReportModel.StringAlignment ('StringAlignment Enumeration' in the on-line documentation)**.
- System.Drawing.ContentAlignment, new type is **GrapeCity.ActiveReports.SectionReportModel.ContentAlignment ('ContentAlignment Enumeration' in the on-line documentation)**.
- System.Drawing.HatchStyle type, new type is **GrapeCity.ActiveReports.SectionReportModel.HatchStyle ('HatchStyle Enumeration' in the on-line documentation)**.

- Other Affected GrapeCity.ActiveReports.SectionReportModel APIs are as under:
  - **GrapeCity.ActiveReports.Document.Section.Page.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.InputFieldText.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.Barcode.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.Barcode.Alignment ('Alignment Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.CheckBox.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.CheckBox.CheckAlignment ('CheckAlignment Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.Shape.BackgroundPattern ('BackgroundPattern Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.Label.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.TextBox.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReportModel.ReportInfo.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.Printing.WatermarkOptions.Font ('Font Property' in the on-line documentation)**
  - **GrapeCity.ActiveReports.SectionReport.Watermark ('Watermark Property' in the on-line documentation)**

  - GrapeCity.ActiveReports.SectionReportModel.RichTextBox.Font
  - GrapeCity.ActiveReports.SectionReportModel.RichTextBox.SelectionFont

GrapeCity.ActiveReports.Export.Pdf Changes

- GrapeCity.ActiveReports.Export.Pdf.Section.PdfWatermarkSettings.FontStyle
- GrapeCity.ActiveReports.Export.Pdf.Section.Signing.PdfStamp.Font
- GrapeCity.ActiveReports.Export.Pdf.Section.Signing.PdfStamp.Image

Breaking changes from ActiveReports 15 to ActiveReports 16

Dropped Microsoft Visual Studio 2015 Support
ActiveReports 16 is not supported in Microsoft Visual Studio 2015.
Dropped Internet Explorer 11 Support
ActiveReports 16 no longer supports Internet Explorer 11 owing to the Microsoft announcement.
The Web package no longer includes a dependency from the Web.Design package
For using the **WebViewer** control in the **ASP.NET Designer**, you need to install an additional **GrapeCity.ActiveReports.Web.Design** (or **GrapeCity.ActiveReports.Web.Design.VS2022**) package that corresponds to the Visual Studio version you are using. Installing the **GrapeCity.ActiveReports.Web** package does not install the Web Design package, thus you may see some design time or compilation errors.
Extended IRenderingExtension with a cancellation token

If you implemented a custom rendering extension, it needs to be updated with the implementation of the new Render Signature. Use **IRenderingExtension ('Render(IRenderingExtension,StreamProvider,NameValueCollection,Boolean,Boolean,CancellationToken,IProgress<ProgressInfo>) Method' in the on-line documentation)** as follows: `void Render(IReport report, StreamProvider streams, NameValueCollection settings, CancellationToken token);`
Now the `-out` path in the console for importing reports supports only a folder path
Previously, while using the Import tool via command line, you could use `-out` to specify a path to a file or a path to a folder. Now, `-out` specifies the path to a folder only.
Scripts with WinForms dependencies will no longer work

You may need to load WinForms dependencies manually by using code similar to the following.
`var rpt = new SectionReport();`
`rpt.LoadLayout("MySectionReport.rpx");`
`rpt.AddAssembly(System.Reflection.Assembly.Load("System.Windows.Forms, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"));`
 	
The sample of the script that worked earlier without above code is as shown:
`public void detail_Format()`
`{`
`    System.Windows.Forms.MessageBox.Show("Detail_Format");`
`}`

PDF export settings in ASP.NET WebForms viewer
If you specified export settings on the ASPX page, it may require some updates due to this breaking change.

For breaking changes in previous versions, see the documentation page of ActiveReports 16.

ActiveReports Version Compatibility and Migration

ActiveReports 18 supports the side-by-side installation with previous versions and allows to migrate from version to version.

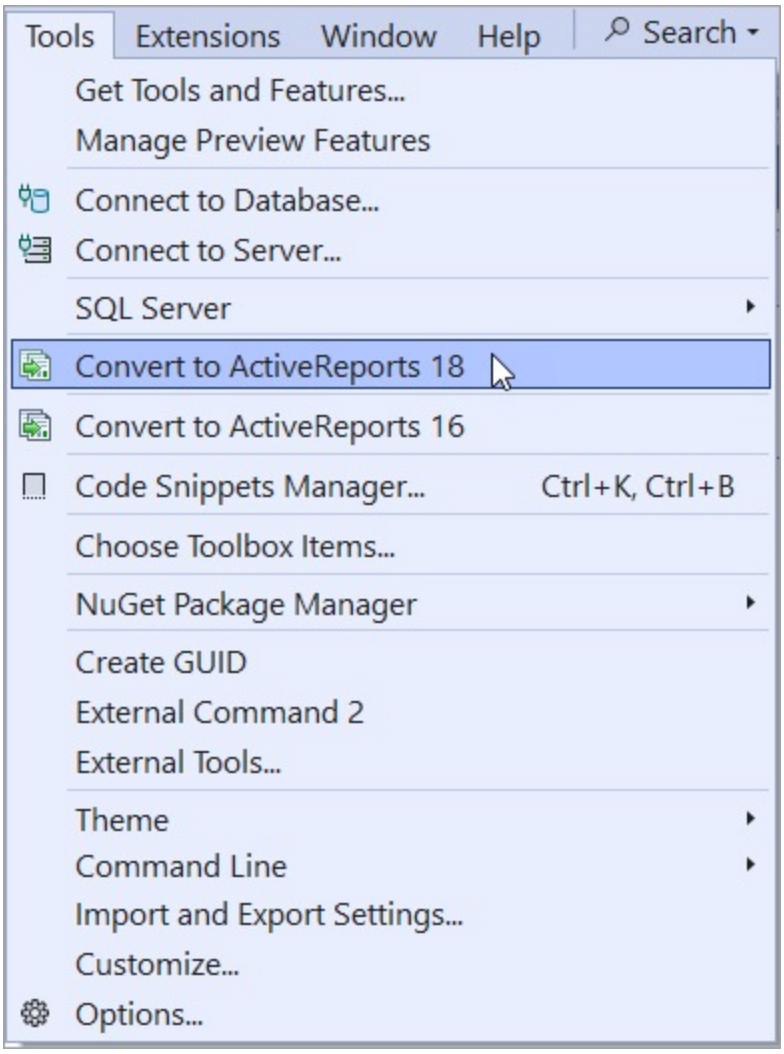
We recommend that you follow certain rules for migrating from one version to another. In case of any question, please contact our **Support Team** by using any of these channels:

Web site	https://developer.mescius.com/support/contact
E-mail	activereports.sales@mescius.com

Note: As not all old versions of ActiveReports are supported, you should check migration possibilities without delay. This will help us fix problems that may appear.

Migration Rules

- Perform migration in testing branches. Do not migrate in production without a possibility to roll back.
- We recommend that you first use the **ActiveReports Upgrade** tool.



This tool upgrades reports and links but it cannot process:
    - JS code/dependencies
    - Some breaking changes
    - Some problems in Web environments
- Verify affected functions and breaking changes.
- Contact our **Support Team** in case of any problem.

Compatibility of Visual Studio Integrated Designers

ActiveReports 18 can be installed and used on the same machine in which other older versions of ActiveReports for .NET (except ActiveReports 13) have been installed.

Compatibility of ActiveReports Visual Studio Integrated Designers with Visual Studio

ActiveReports Visual Studio integrated designer of different versions can be used by integrating the designer with Visual Studio IDE (Integrated development environment). The following table shows the Visual Studio versions corresponding to the integrated designer versions that are supported.

	VS .NET 2002	VS .NET 2003	VS 2005	VS 2008	VS 2010	VS 2012	VS 2013	VS 2015	VS 2017	VS 2019	VS 2022
ActiveReports 1	○	○	×	×	×	×	×	×	×	×	×
ActiveReports 2	×	○	○	×	×	×	×	×	×	×	×
ActiveReports 3	×	○	○	×	×	×	×	×	×	×	×
ActiveReports 6	×	×	○	○	○	×	×	×	×	×	×
ActiveReports 7	×	×	×	○	○	○	○	×	×	×	×
ActiveReports 8	×	×	×	○	○	○	○	×	×	×	×
ActiveReports 9	×	×	×	×	○	○	○	○	×	×	×
ActiveReports 10	×	×	×	×	○	○	○	○	×	×	×
ActiveReports 11	×	×	×	×	○	○	○	○	○	×	×
ActiveReports 12	×	×	×	×	×	○	○	○	○ (VS SP2+)	×	×
ActiveReports 13	×	×	×	×	×	○	○	○	○ (VS SP2+)	○	×
ActiveReports 14	×	×	×	×	×	×	○	○	○ (VS SP2+)	○	×
ActiveReports 15	×	×	×	×	×	×	×	○	○ (VS SP2+)	○	×
ActiveReports 16	×	×	×	×	×	×	×	×	○ (VS SP2+)	○	○
ActiveReports 17	×	×	×	×	×	×	×	×	○ (VS SP2+)	○	○
ActiveReports 18	×	×	×	×	×	×	×	×	○ (VS SP2+)	○	○

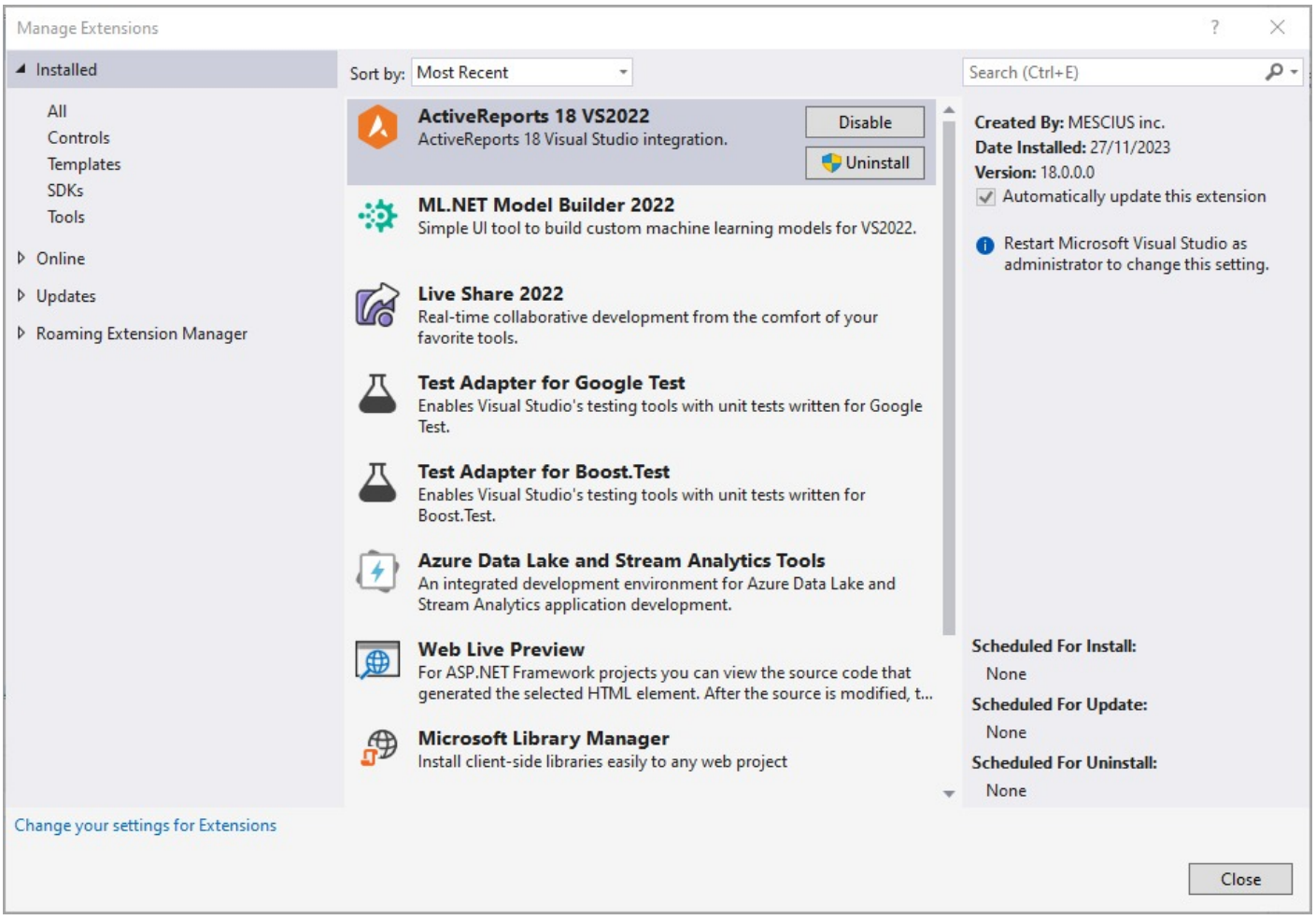
To switch the designers of versions prior to ActiveReports 13, use the switcher tool. See ActiveReports 13 help file for more information.

Starting from **ActiveReports 14**, you should use the Visual Studio menu to enable/disable the ActiveReports integration.

Manage Visual Studio Integration

In case you want to disable or uninstall ActiveReports 18 Visual Studio integration, do the following.

1. In Visual Studio, go to **Tools > Extensions and Updates...** or **Extensions** > **Manage Extensions**.
2. Navigate to **ActiveReports 18 VS2022** and click Disable or Uninstall.



In case you want to install the integration again, follow these steps.

1. Go to the installation folder - C:\Program Files\MESCIUS\ActiveReports 18\VisualStudio (C:\Program Files (x86)\MESCIUS\ActiveReports 18\VisualStudio on a 64-bit Windows operating system).
2. Run the VSIX as per your Visual Studio Version.

Note:

- It is not possible to install ActiveReports 18 and ActiveReports 13 side-by-side.
- It is not possible to use different versions of ActiveReports for .NET within a single Visual Studio project.
- If you are creating an ActiveReports 17 application in Visual Studio 2017 or Visual Studio 2019 or Visual Studio 2022, and you also have ActiveReports 13 installed in the supported Visual Studio version, then you may get a few errors in the property grid such as 'Object does not match target type' or 'Object reference not set to an instance of Object'. This issue is related to Visual Studio; you just need to restart Visual Studio

to resolve the issue. Before restart, save the project, if necessary.
- When using our Integrated Designer in Visual Studio 2022, you may observe invisible controls in our smart panels or tool windows (such as Report Explorer or Layers List). To solve this issue, navigate to Tools > Options > Environment > General and clear the 'Optimize rendering for screens with different pixel densities' checkbox.

Upgrade Reports and References

Generally, there are just a few breaking changes between ActiveReports versions. However, advanced use cases often require manual upgrade. We recommend that you start by using the ActiveReports Upgrade tool that converts links and reports (see ActiveReports File Converter). After that, you should do the following:

- Manually upgrade links if necessary.
- Manually upgrade namespace/code changes.
- Manually upgrade JS dependencies and JS code.
- Contact the **Support Team** in case of any problem.

ActiveReports File Converter

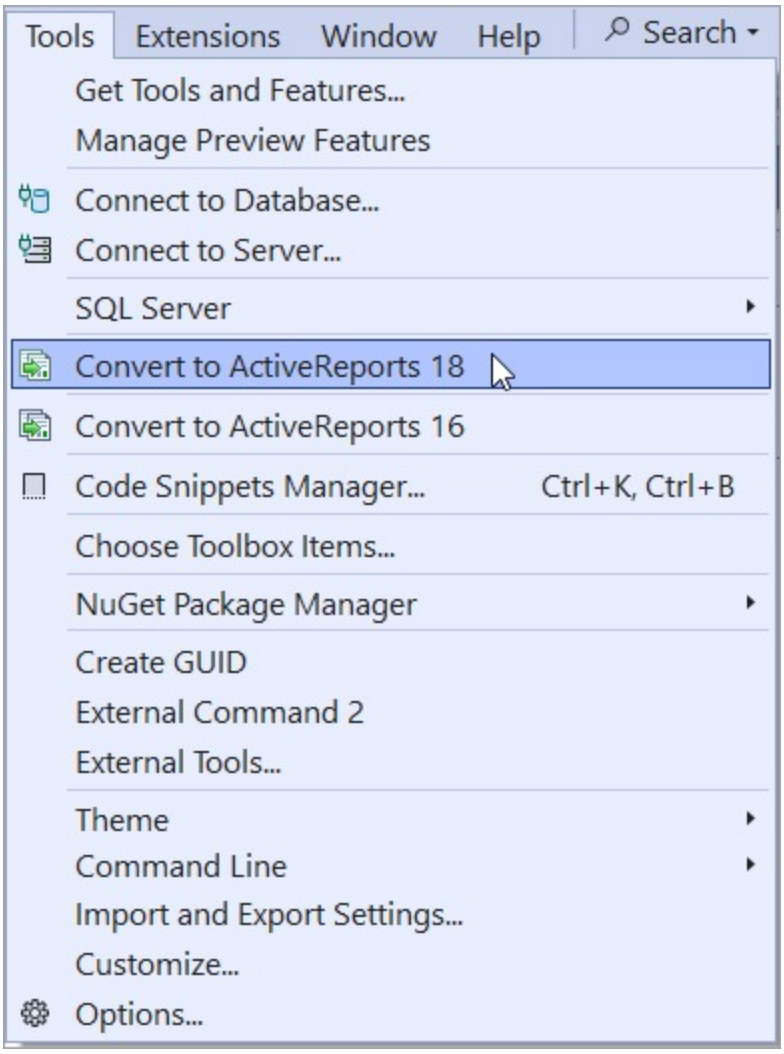
The ActiveReports file converter allows you to upgrade your existing reports from previous versions of ActiveReports (ActiveReports 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 3, 2, and 1) to the latest version.

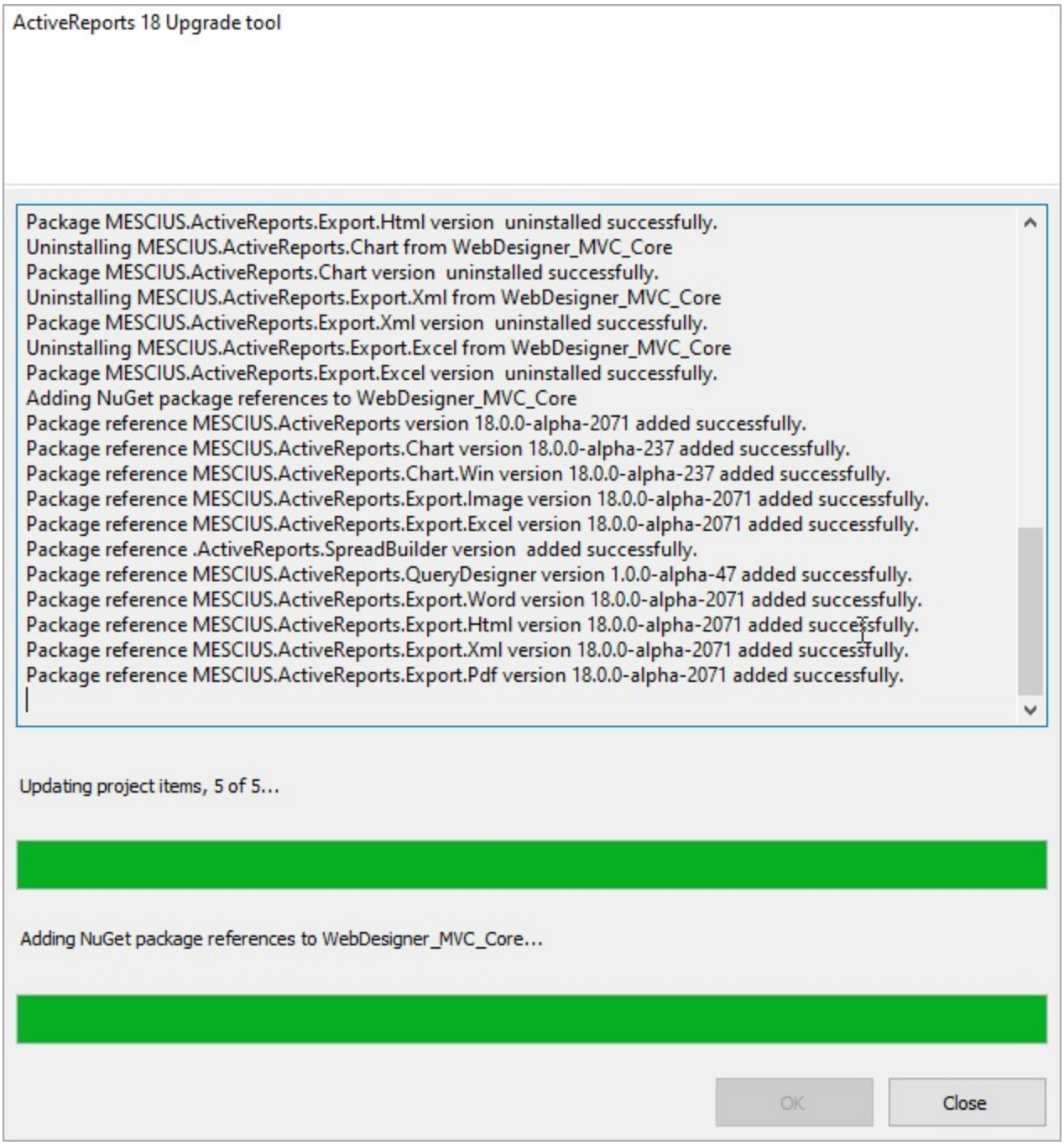
To use the file converter

1. In Visual Studio, open an existing ActiveReports project.

   ⚠ **Caution:** If you are migrating a project from a different version of Visual Studio, a Visual Studio conversion wizard will run automatically to convert the project.

2. From the Visual Studio **Tools** menu, select **Convert to ActiveReports 18**. In the ActiveReports 18 Upgrade tool window that appears, you can see a list of report files to be converted.





File Type	Extension
Report file	*.rdlx, *.rpx, *.vb, *.cs
Form with a viewer	*.vb, *.cs, *.xaml
ASP.NET Web Form with a WebViewer	*.aspx, *.aspx.vb, *.aspx.cs
Project file	*.vbproj, *.csproj, Web.confg
License file	licenses.licx

3. Click **OK** to convert the files. After the files are successfully converted, all of the reference files are updated and

the reports are converted to C# or Visual Basic class files. The old files are copied to **ARConverterBackup** folder generated under the root folder of the project. You can delete these files from the project if you do not need them.

> ⚠ **Caution:** When converting from version 6 or below, you may observe the following warnings on migrating a form with the viewer control. These warnings are due to API modifications in ActiveReports 7 or above. You can ignore these warnings or delete them manually.
>
> - 'Public Property Text() As String' is an old format: 'Not used anymore'
> - 'Public Property TabTitleLength() As Integer' is an old format: 'Not used anymore'

> **Note:** In addition to the above conversions, you may also observe some formatting changes, such as code breaks, while using this converter tool.

4. After running the converter, you need to manually update all the project references. For more details, see Upgrade Reports and References.

Migrate to ActiveReports 18

ActiveReports 18 has several major breaking changes, so you should consider this information while migrating to ActiveReports 18.

If you are going to migrate from very old versions of ActiveReports, please see corresponding User Guides or contact our Support Team.

Reference Migration

ActiveReports uses NuGet/NPM to publish components since ActiveReports 14. We cannot manage all references without problems in our converter, so please remove extra NuGet packages and fix NPM references/dependencies.

If you are going to update references manually, please follow this plan:

- Remove all ActiveReports NuGet/NPM references.
- Add only important references (other references will be added automatically):
  - https://www.NuGet.org/packages/MESCIUS.ActiveReports to use only .NET API and packages MESCIUS.ActiveReports.Export.[Excel/Word/Pdf/Xml/Html/Image] to use corresponding exports.
  - https://www.NuGet.org/packages/MESCIUS.ActiveReports.Serializer or https://www.NuGet.org/packages/MESCIUS.ActiveReports.Serializer.VS2022 to design code-based Section reports under .NET Framework 4.6.2-4.8.1. Please note that it is better to remove such dependencies from the final application.
  - https://www.NuGet.org/packages/MESCIUS.ActiveReports.Viewer.Win or https://www.NuGet.org/packages/MESCIUS.ActiveReports.Viewer.Wpf to use only the Desktop Viewer (without Designer).
  - https://www.NuGet.org/packages/MESCIUS.ActiveReports.Design.Win to use the Desktop Designer.
  - https://www.NuGet.org/packages/MESCIUS.ActiveReports.Aspnetcore.Viewer or https://www.NuGet.org/packages/MESCIUS.ActiveReports.Aspnet.Viewer to use Blazor/JS viewers, https://www.NuGet.org/packages/MESCIUS.ActiveReports.Blazor.Viewer or https://www.npmjs.com/package/@mescius/activereportsnet-viewer as client.
  - https://www.NuGet.org/packages/MESCIUS.ActiveReports.Aspnetcore.Designer or https://www.NuGet.org/packages/MESCIUS.ActiveReports.Aspnet.Designer to use Blazor/JS designer + https://www.NuGet.org/packages/MESCIUS.ActiveReports.Blazor.Designer or https://www.npmjs.com/package/@mescius/activereportsnet-designer as client.
  - https://www.NuGet.org/packages/MESCIUS.ActiveReports.Web to use ASP.NET WebForms viewer +

https://www.NuGet.org/packages/MESCIUS.ActiveReports.Web.Design or https://www.NuGet.org/packages/MESCIUS.ActiveReports.Web.Design.VS2022 to enable ASP.NET WebForms designer. Please note that it is better to remove design-time dependencies from the final application.

- Check the JS code manually.
- Update the license ("licenses.licx", ".gclicx" and other places).
- Try to fix other issues. In case you are unable to fix the issues, please contact our Support Team.

License Migration

You may encounter two basic scenarios with the ActiveReports license migration:

1. Development environment
2. Run-time environment

Development environment

To use ActiveReports development environments, you should buy a corresponding license.

Run-time environment

A common run-time license is embedded into the application with a general .NET mechanism - the **licenses.licx** file. You should update references in the file to use new ActiveReports versions.

> **Note**: ActiveReports allows generating the embeddable license resource **.gclicx** that you should regenerate manually for any new ActiveReports version.

Web Viewer Migration

You may run into problems while migrating in Web due to the following reasons:

- No more Flash application framework support
- No more Silverlight application framework support
- New Server API since ActiveReports 13
- New JS API since ActiveReports 15

Hence, to perform the Web Viewer migration, ensure following:

- Check for any web related breaking changes.
- If you are using old technologies (like Silverlight), we suggest migrating to Blazor.
- If JS API does not work, check the new JS Viewer API and Web Samples and restore functions.
- Contact our Support Team for help.

Migrate to .NET 6/.NET 7/.NET 8

An existing ActiveReports application can be migrated to .NET 6/.NET 7/.NET 8 by following these simple steps provided on MSDN page. To migrate ActiveReports application to .NET 6/.NET 7/.NET 8, you need Visual Studio 2022 with .NET 6/.NET 7/.NET 8 installed.

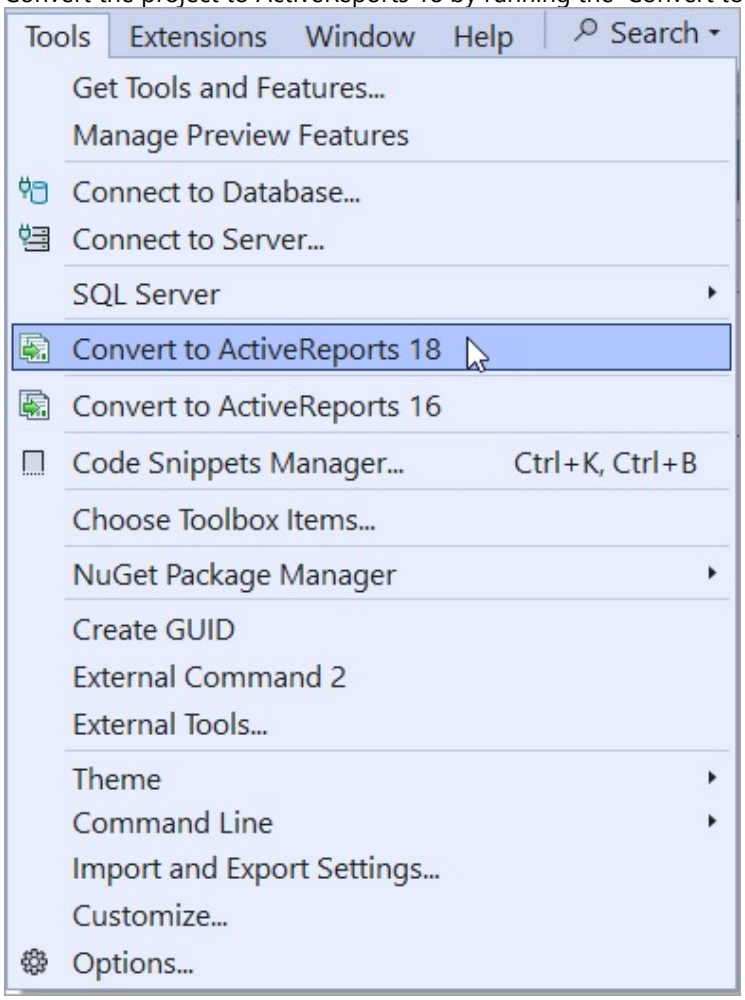
The following steps consider an ActiveReports 17 WinViewer desktop application as a sample .NET Framework project that needs to be migrated to .NET 6/.NET 7/.NET 8 platform. The sample can be found on GitHub.

For more information, please refer to the following Microsoft documentation:

- Overview of porting from .NET Framework to .NET Core

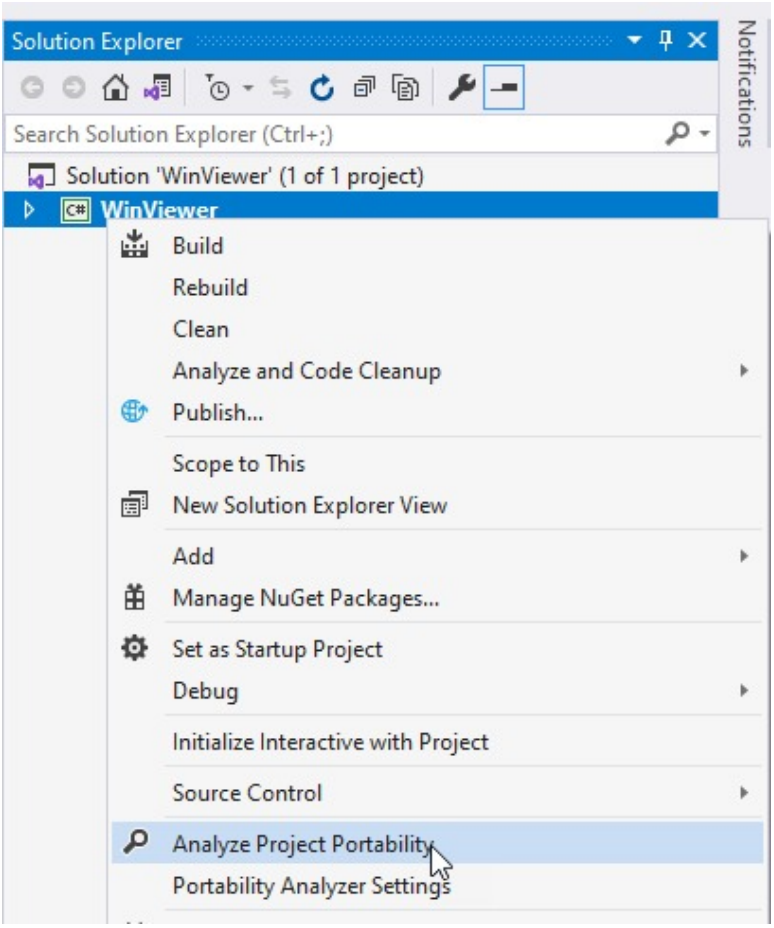
Upgrade the project to ActiveReports 18
(skip this if you are already in ActiveReports 18)

1. Open the \Desktop\WinViewer\C#**WinViewer.sln** file in **Visual Studio 2022 v17.8+**.
2. Convert the project to ActiveReports 18 by running the 'Convert to ActiveReports 18' tool.

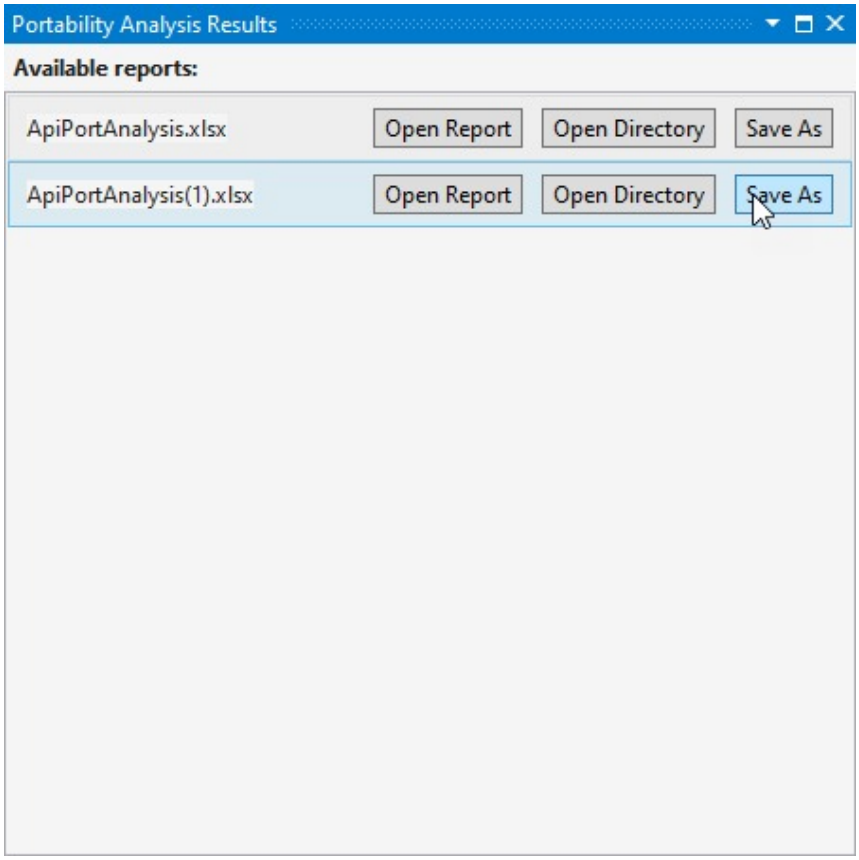


Analyze Portability
(this step is for the analysis purpose to see if the assemblies are portable to .NET Core; it can be skipped)

3. Click on **Extensions** > **Manage Extensions** and in the Manage Extensions dialog, search for '.NET Portability Analyzer' and download it. You may need to reopen the project for VSIX installer to run.
4. Right-click on the WinViewer solution and click 'Analyze Assembly Portability'.

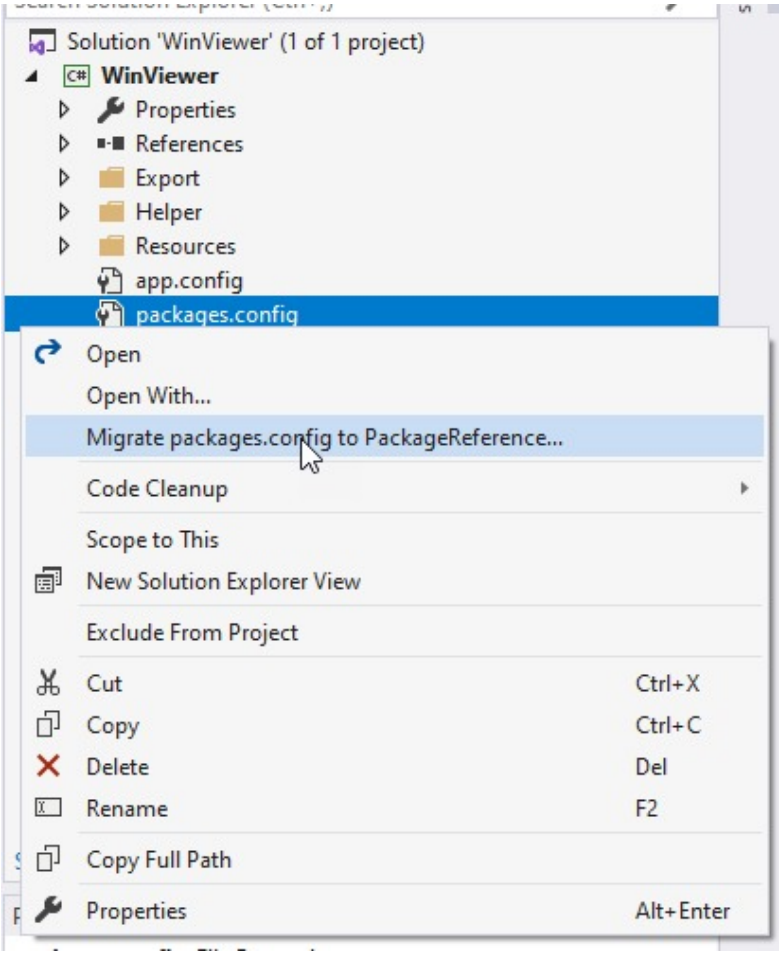


5. Save the Analysis report on your system and open it to analyze the portability details of your project.



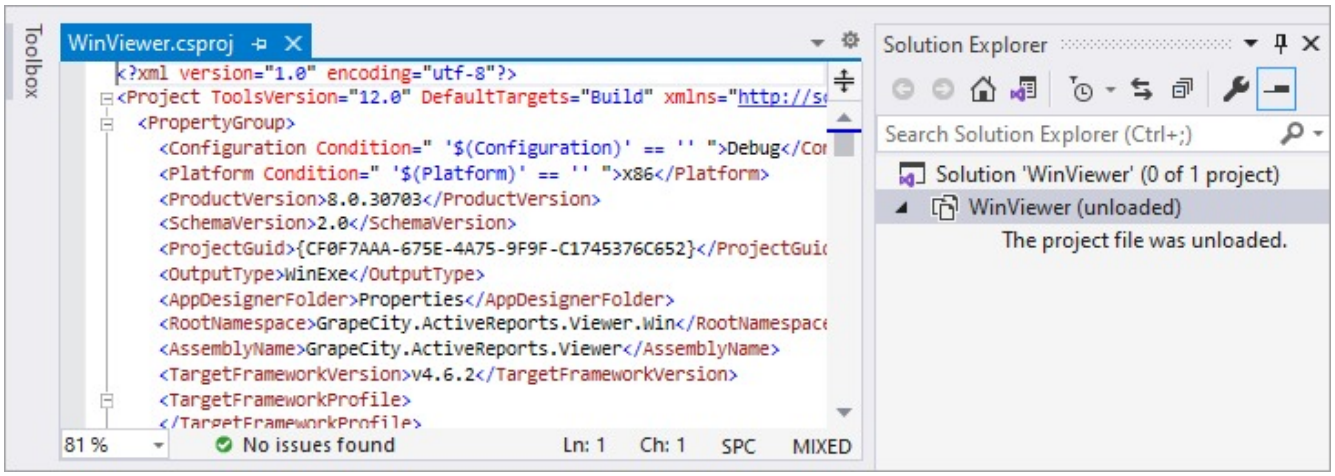
Migrate to .NET 6/.NET 7/.NET 8 platform

6. Right-click 'packages.config' in the project, click 'Migrate packages.config to PackageReference..' and press OK.



You will see a NuGet migration log page with details.

7. Right-click the project and select 'Unload Project'. Double-click 'WinViewer' in the Solution Explorer to view WinViewer.csproj file.



8. Remove the content of the WinViewer.csproj file and copy it in a text file so that the file appears blank and you have its backup in a text file.
9. Add the following code to blank WinViewer.csproj file to change it to project SDK type.

> **Note**: The OutputType is "library" because the project is a class library project. Also, 'auto-generate assembly info' should be turned off.

Show code

.csproj

```
<Project Sdk="Microsoft.NET.Sdk">
<PropertyGroup>
<OutputType>library</OutputType>
<TargetFramework>net6.0-windows</TargetFramework>
<GenerateAssemblyInfo>false</GenerateAssemblyInfo>
</PropertyGroup>
</Project>
```

10. Find the text "PackageReference" in the backup text file and copy the whole <ItemGroup> inside <Project> to WinViewer.csproj file. The final WinViewer.csproj will look like below:

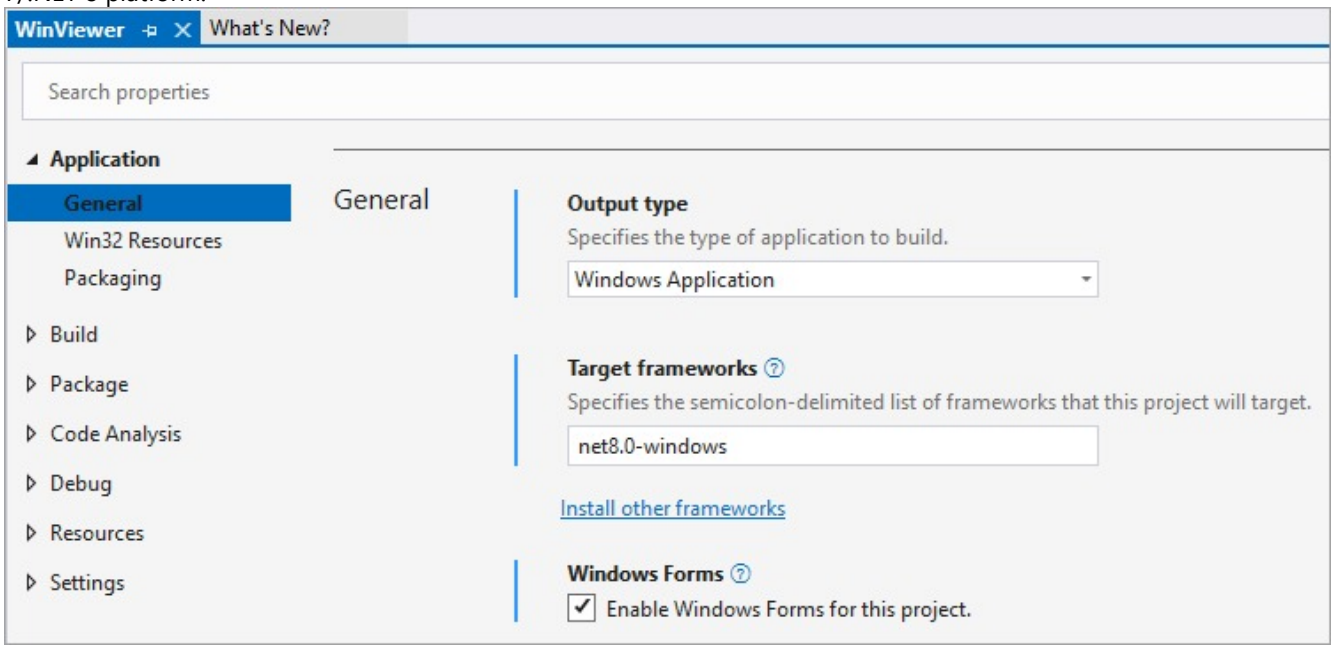
Show code

.csproj

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>WinExe</OutputType>
    <TargetFramework>net6.0-windows</TargetFramework>
    <GenerateAssemblyInfo>false</GenerateAssemblyInfo>
    <ApplicationIcon />
    <StartupObject />
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="MESCIUS.ActiveReports.Chart.Win">
      <Version>16.0.0-alpha-106</Version>
    </PackageReference>
    <PackageReference Include="MESCIUS.ActiveReports.Export.Excel">
      <Version>16.0.0-beta-1201</Version>
    </PackageReference>
    <PackageReference Include="MESCIUS.ActiveReports.Export.Html">
      <Version>16.0.0-beta-1201</Version>
    </PackageReference>
    <PackageReference Include="GrapeCity.ActiveReports.Export.Pdf">
      <Version>16.0.0-beta-1201</Version>
    </PackageReference>
    <PackageReference Include="GrapeCity.ActiveReports.Export.Word">
      <Version>16.0.0-beta-1201</Version>
    </PackageReference>
    <PackageReference Include="GrapeCity.ActiveReports.Export.Xml">
      <Version>16.0.0-beta-1201</Version>
    </PackageReference>
    <PackageReference Include="GrapeCity.ActiveReports.Viewer.Win">
      <Version>16.0.0-beta-1201</Version>
    </PackageReference>
    <PackageReference Include="GrapeCity.DataVisualization.Chart">
      <Version>0.4.53</Version>
    </PackageReference>
```

```
    <PackageReference Include="Microsoft.NETCore.Platforms">
      <Version>1.1.0</Version>
    </PackageReference>
    <PackageReference Include="Microsoft.Win32.Primitives">
      <Version>4.3.0</Version>
    </PackageReference>
  </ItemGroup>
</Project>
```

11. Right-click the project in Solution Explorer and click on Reload project. You may see version errors for 'Compile Time Assemblies'. Resolve them by manually correcting the version as suggested in the window.
12. Build the project to observe that the build is successful and the project has been successfully migrated to .NET 6/.NET 7/.NET 8 platform.



License ActiveReports

Users can access the MESCIUS License Manager utility to license ActiveReports at the time of installation or if there is already a trial version installed. The following sub-topics give an overview of all aspects of the licensing procedure in ActiveReports .NET.

License Types

Learn what license types are available.

Upgrading License from Trial to Purchase

Learn how to upgrade license from trial to purchase.

Licensing a Developer Machine

Learn about licensing a developer machine.

Licensing a Project

Learn about licensing a project.

Licensing Compiled Code

Learn how to license compiled code.

Licensing Build Agents/Pipelines

Learn how to license build agents/pipelines.

Fixing Licensing Errors

Learn how to fix license errors.

Contacting Support

Learn what channels you can use to contact our Support Team.

License Types

See ActiveReports Editions to learn which features are exclusive to the Professional Edition.

License Type	Description
Evaluation	**Trial key is required.** Evaluation banners display on all reports and controls, and the product stops functioning after 30 days from the date of installation. The first key is already activated when you download the trial. If needed, you can request a new key from the Sales department for an additional 30 day trial.
Standard	**Standard Edition product key is required.** Evaluation banners appear only on features that are exclusive to the Professional Edition. You receive this key by email when you purchase ActiveReports Standard Edition or upgrade from a previous version of ActiveReports Standard Edition.
Professional	**Professional Edition product key is required.** All reporting functionality and controls appear without any evaluation banners. You receive this key by email when you purchase ActiveReports Professional Edition or upgrade from a previous version of ActiveReports Professional Edition.

If you cannot find your email with the product key, please contact activereports.sales@mescius.com to have it looked up.

Upgrading License from Trial to Purchase

ActiveReports .NET provides a trial version to experiment and evaluate all features of Professional Edition for a total of 30 days without a product key. To completely use the benefits of ActiveReports for production, you need to upgrade the license from trial to purchase, which can be done by purchasing a product key (Standard or Professional Edition). After purchasing the license, the user can activate the key using MESCIUS License Manager. For detailed steps, refer to the Licensing a Developer Machine topic.

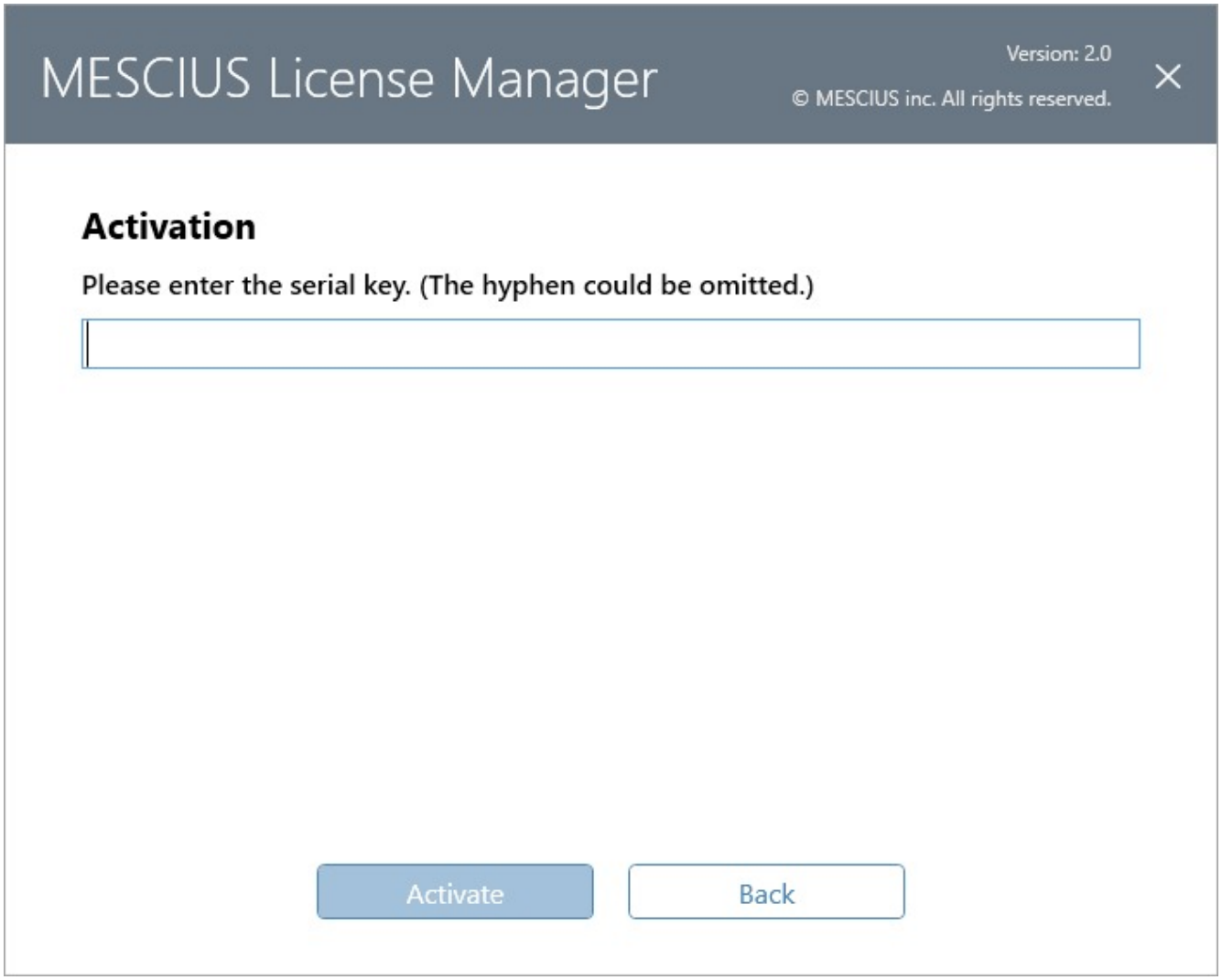
Licensing a Developer Machine

In order to open and effectively run the ActiveReports .NET or compile ActiveReports projects in Visual Studio, the user needs to license the developer machine. In this topic, we'll cover licensing a machine (or multiple machines) with ActiveReports during installation, and deactivating ActiveReports license.

> **Note**: Always run the License Manager as an **Administrator**.

License a machine with ActiveReports during installation or to license a trial without reinstalling

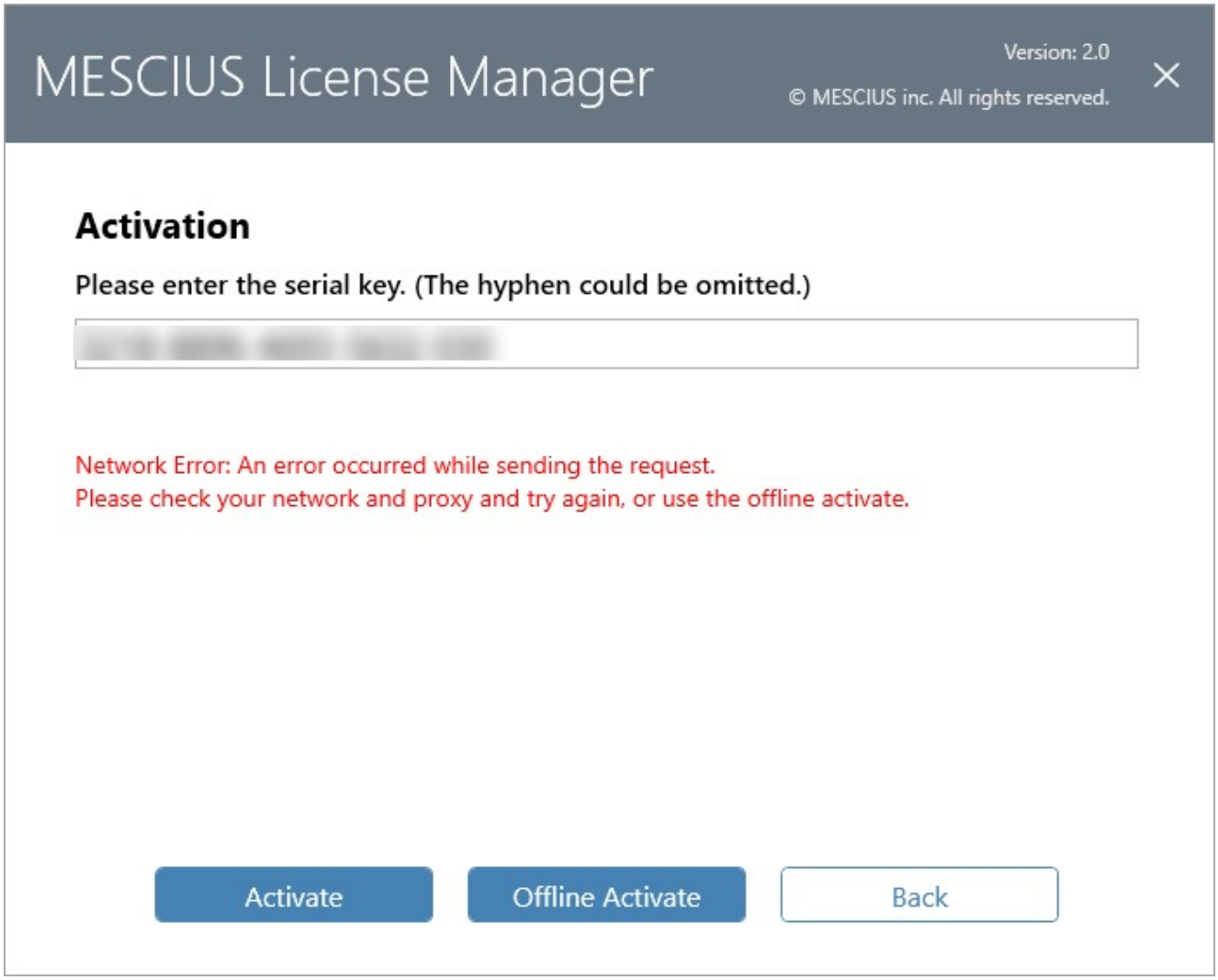
1. Go to the Start menu and select **MESCIUS License Manager** to open the licensing manager window. This window appears automatically during the product installation.
2. In the **MESCIUS License Manager** window, click **Activate**.
3. Enter the serial key, you have received from MESCIUS (including all the capital letters and special characters), and click **Activate**.



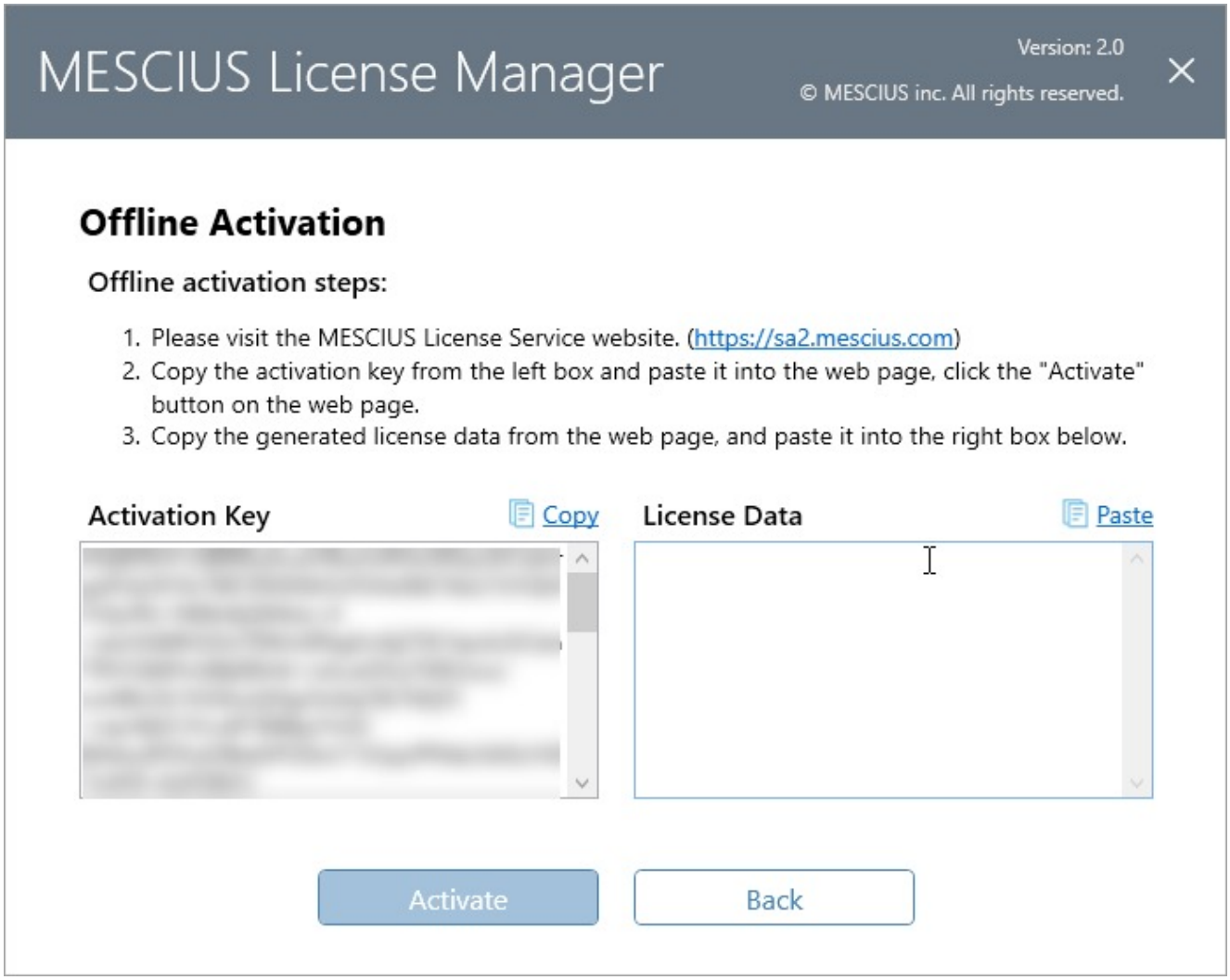
4. An activation message appears on the window. Click **OK** to complete the licensing process.
   To see the license details such as the license type (Trial or Product), edition (Professional or Standard), and the date of activation and expiry, click **Details** in the licensing window.

License ActiveReports on a machine without an internet connection

1. Go to the Start menu and select **MESCIUS License Manager** to open the licensing manager window. This window appears automatically during the product installation.
2. In the **MESCIUS License Manager** window, if you try to enter the serial key and click **Activate**, a network error is shown on the window. So, in such a case, click **Offline Activate** to proceed further with the licensing process.

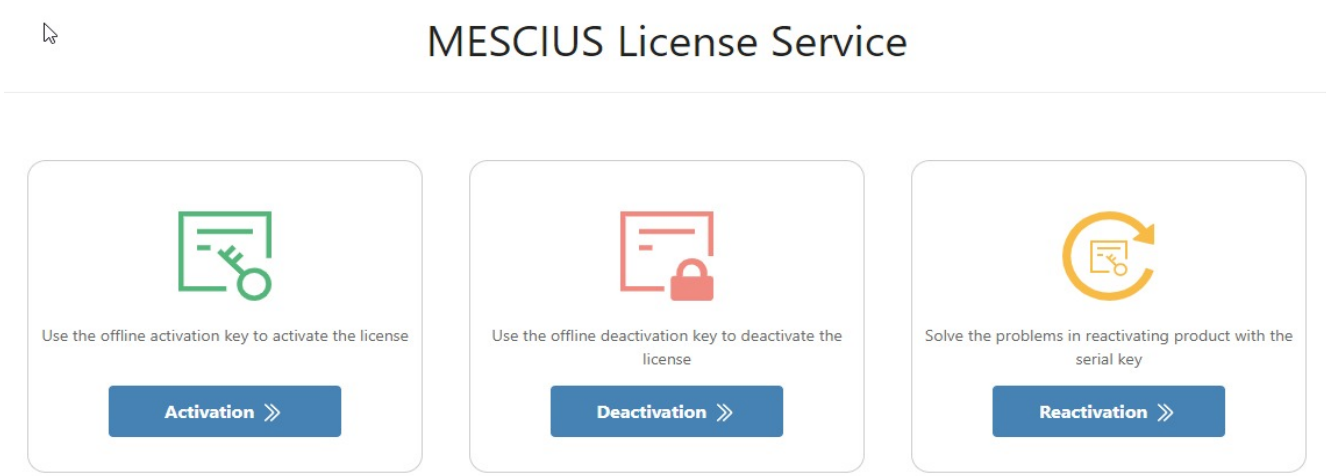


3. Copy the Activation Key from the **MESCIUS License Manager** window.

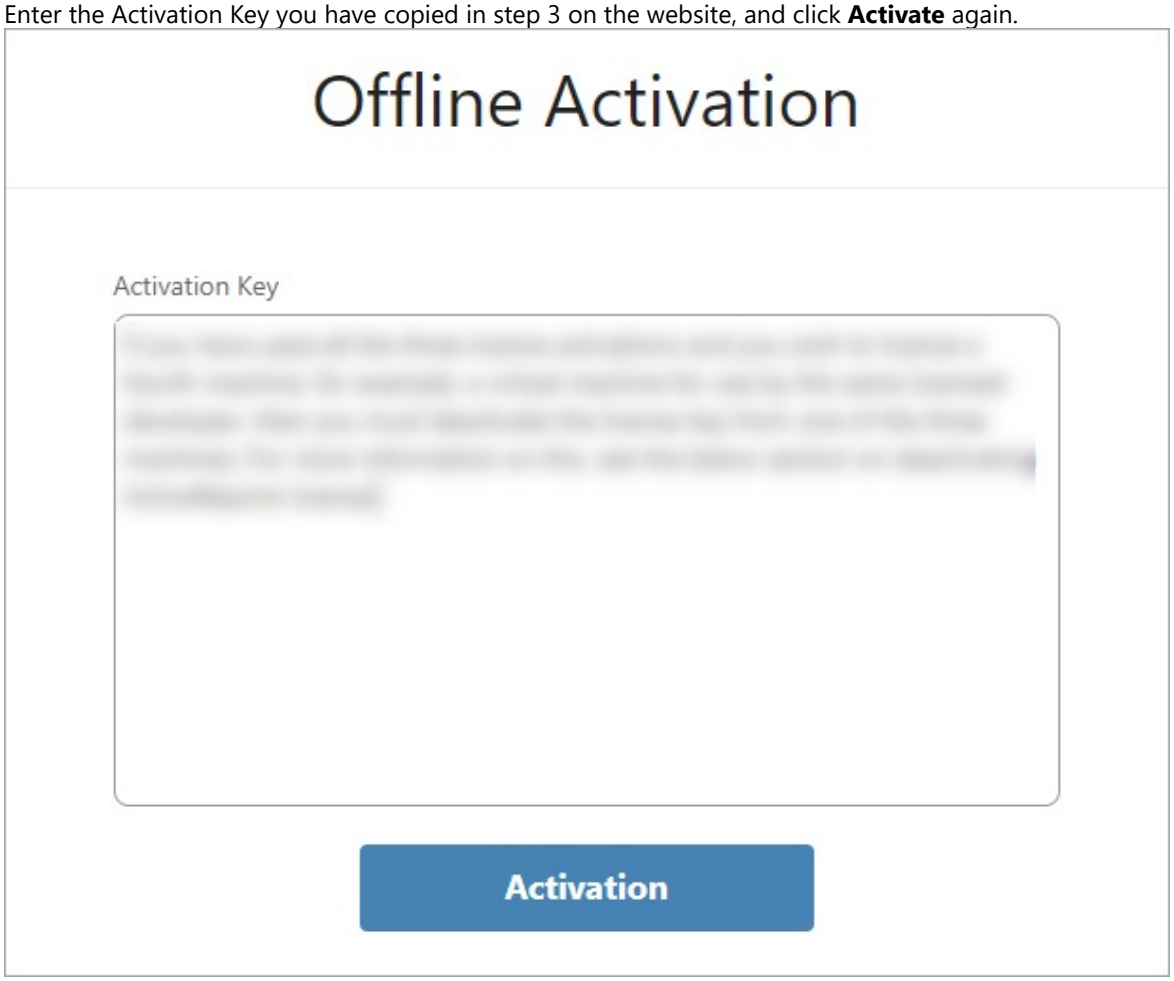


4. Visit the following website https://sa2.mescius.com on the other machine (having an internet connection), and click **Activation** to use offline Activation key to activate the license.

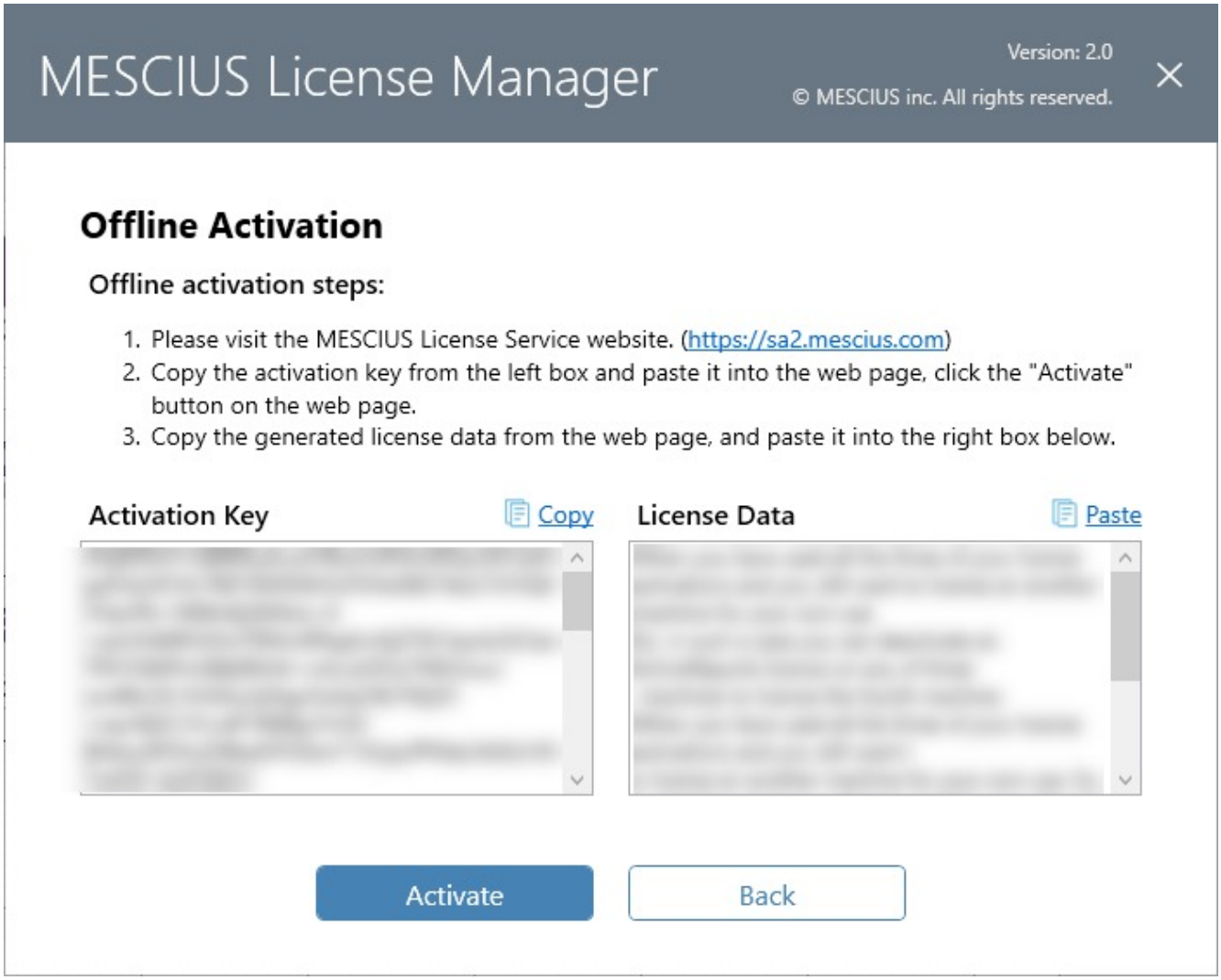
> **Note**: Do not close the activation dialog on your original machine until the activation process is completed.



5. Enter the Activation Key you have copied in step 3 on the website, and click **Activate** again.



6. After a successful activation on the website, copy the License Data from the website and paste it in the licensing window.



7. Click **Activate** and then **OK** to complete the licensing process.

Activate a license for ActiveReports on multiple machines

You can activate a single developer license key for ActiveReports on **three machines** for use by **one developer**.

If you have used all the three license activations and you wish to license a fourth machine, for example, a virtual machine for use by the same licensed developer, then you must deactivate the license key from one of the three machines. For more information on this, see the below section on deactivating ActiveReports license.

After you have deactivated licensing on one of the machines, you can then activate ActiveReports on the other machine.

Deactivate an ActiveReports license

When you have used all the three of your license activations and you still want to license an another machine for your own use. So, in such a case you can deactivate an ActiveReports license on any of three machines to license the fourth machine.

Follows these steps to deactivate an ActiveReports license on a machine:

1. Go to the Start menu and select **MESCIUS License Manager** to open the licensing manager window.
2. In the **MESCIUS License Manager** window, click **Deactivate**.

3. In the **Deactivation** page, click **Deactivate** again to confirm the deactivation process.
4. Click **OK** to complete the deactivation process.

Determine whether a machine is licensed

1. Open any Sample in Visual Studio from the included samples and click the **Preview** tab.
2. Scroll to the bottom of the report and check for any red evaluation text. If there is any, it means your machine is not licensed.

> **Note**: To check for Professional Edition licensing, open a sample from the **Designer Pro** folder, run it, and look for the evaluation messages.

Licensing a Project

> **Tips:**
>
> - To deploy using XCOPY, you must include the DLLs for all of your ActiveReports references in your bin/debug folder. To do this, in the Visual Studio Solution Explorer, select each reference, and in the Properties window, set **Copy Local** to **True** and rebuild your solution.
> - To avoid having to change the version number each time you install an ActiveReports service pack. You need to select each reference in the Visual Studio Solution Explorer, and in the Properties window, set **Specific Version** to **False**.

License Windows Forms projects made on the trial version

These steps assume that you have an ActiveReports licensed edition installed on your system.

1. Open the project in Microsoft Visual Studio.

   > **Note:** If another application calls the one containing ActiveReports features, you must license the calling application to avoid evaluation banners after deployment.

2. In Visual Studio, go to the **Build** menu and select **Rebuild Solution**.
3. To verify that the application is licensed, open the licenses.licx file and compare it to the Required references section below.

The executable application is now licensed, and no nag screens or evaluation banners will appear when you run it. You can also distribute the application to the unlicensed machines without facing any nag screens or evaluation banners.

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

The licenses.licx file must contain the following references to ActiveReports if you are using both the Section and Page reports, and both the Windows and WPF viewers. See the table for the references to the ActiveReports version and the reference to the Viewer control.

> **Note:** The Version, Culture, and PublicKeyToken information is added automatically, but they can be removed, and are preferred to be removed, if the version is wrong.

Paste INSIDE the licenses.licx file.
MESCIUS.ActiveReports.SectionReport, MESCIUS.ActiveReports MESCIUS.ActiveReports.PageReport, MESCIUS.ActiveReports MESCIUS.ActiveReports.Viewer.Win.Viewer, MESCIUS.ActiveReports.Viewer.Win MESCIUS.ActiveReports.Viewer.Wpf.Viewer, MESCIUS.ActiveReports.Viewer.Wpf

The below table lists all the license strings that you may need:

Component	License String
Section Report engine	MESCIUS.ActiveReports.SectionReport, MESCIUS.ActiveReports
Page and RDLX report engine	MESCIUS.ActiveReports.PageReport, MESCIUS.ActiveReports
WinForms viewer control	MESCIUS.ActiveReports.Viewer.Win.Viewer, MESCIUS.ActiveReports.Viewer.Win
WPF viewer control	MESCIUS.ActiveReports.Viewer.Wpf.Viewer, MESCIUS.ActiveReports.Viewer.Wpf
PRO (some features) PDF export	MESCIUS.ActiveReports.Export.Pdf.Section.PdfExport, MESCIUS.ActiveReports.Export.Pdf
PRO ONLY: WebViewer, HTTP handlers	MESCIUS.ActiveReports.Web.WebViewer, MESCIUS.ActiveReports.Web
PRO ONLY: End-user designer	MESCIUS.ActiveReports.Design.Designer, MESCIUS.ActiveReports.Design.Win
PRO ONLY: JSViewer	MESCIUS.ActiveReports.Aspnet.WebViewer, MESCIUS.ActiveReports.Aspnet.Viewer
PRO ONLY: WebDesigner	MESCIUS.ActiveReports.Aspnet.WebDesigner, MESCIUS.ActiveReports.Aspnet.Designer

Note: When using the PDF export filter in your project, make sure you check the licenses.licx file for reference to the PDF Export Assembly.

License Web Forms projects made on the trial version

Follow these steps after you have licensed ActiveReports on your machine:

1. Open the Web Forms project in Microsoft Visual Studio.
2. In Visual Studio, go to the **Build** menu and select **Rebuild Solution**.
3. The web site application is now licensed. You can distribute the web site application to unlicensed machines and no evaluation banners will appear.

License Web Site applications

Follow these steps after you license ActiveReports on your machine.

1. Open the project in Visual Studio.
2. In the Solution Explorer, right-click the licenses.licx file and select **Build Runtime Licenses** to create the App_Licenses.dll file.

3. The web site application is now licensed. You can distribute the web site application to unlicensed machines without facing any evaluation banners.

License a class library project

Follow these steps after you have licensed ActiveReports on your machine.

1. Open the project for the root-level calling application, that is, the executable that calls your class library, in Visual Studio.
2. From the **Project** menu, select **Add New Item**, and select an ActiveReports report. (You can delete it later. This is only to add the references to the project.)
3. In Visual Studio, go to the **Build** menu and select **Rebuild Solution**.
4. Check the licenses.licx file and verify that ActiveReports licensing is added for all of the features used in your project.

> **Note:**
>
> - If you use some other features of ActiveReports in your class library that are still showing an evaluation banner, for example, features exclusive to the Professional Edition, you can add those references manually and rebuild the solution.
> - The **Web Key Generator** and **Application License Generator** utilities have been removed in ActiveReports 15. You should use above steps to license the project.

License a project that use WebDesigner and JSViewer nuget packages

The WebDesigner and JSViewer nuget packages do not include a licenses.licx file, so the file needs to be added and configured manually in your project. Follow these steps to add license file in your Web Application in Visual Studio:

1. Go to **Project** menu and select **Add New Item**.
2. Select a new text file and rename it to **licenses.licx**.
3. Open the blank licenses.licx file and populate the file with the following entries:

Paste INSIDE the licenses.licx file.

`MESCIUS.ActiveReports.SectionReport, MESCIUS.ActiveReports`
`MESCIUS.ActiveReports.PageReport, MESCIUS.ActiveReports`
`MESCIUS.ActiveReports.Export.Pdf.Section.PdfExport, MESCIUS.ActiveReports.Export.Pdf`

4. Run the project.
   You should not face any licensing errors, assuming that you have a licensed ActiveReports Professional Edition.

Licensing Compiled Code

This section describes how to generate the license for the compiled code or specifically for the applications deployed on Azure Functions application. By default, licensing is applied to your main application. In case ActiveReports is embedded in a custom library, that is called by another application, a "license not found" error is shown. Therefore, you need to generate a license file for the target application.

For example, there is a 'UserControlLibrary' project with ActiveReports libraries embedded and a 'MainApp' project that references the 'UserControlLibrary'. To generate license for the specified target application, follow these steps:

1. Open the command line on Windows and change the working directory to:

```
C:\ProgramData\MESCIUS\gclm
```

2. Run the command as:
   ```
   gclm.exe "463c4179-288c-48fe-a4df-3b609586667d"  -lc [output dir].gclicx "[entry assembly name].[calling assembly name].dll"
   ```

   For example,
   ```
   C:\ProgramData\MESCIUS\gclm>gclm.exe "463c4179-288c-48fe-a4df-3b609586667d" -lc ./.gclicx "MainApp.UserControlLibrary.dll"
   ```

   To license an application on an **Azure Functions** application, run the following command:
   ```
   gclm.exe "463c4179-288c-48fe-a4df-3b609586667d" -lc .\.gclicx Microsoft.Azure.WebJobs.Script.WebHost.[assembly name].dll
   ```

   It generates a **.gclicx** file in the output directory.

   > **Note:** You must specify the application name while generating a license. Also, the generated license can not be used in other applications with different names.

3. Copy the **.gclicx** file and paste it into your application (in this case, UserControlLibrary) where ActiveReports assemblies are used, in any folder.
4. Change the build action of **.gclicx** to **Embedded Resource**.
5. Add the following line in the **.csproj** file:

   ```
   <PropertyGroup>
     <DisableGclm>true</DisableGclm>
   </PropertyGroup>
   ```
6. Rebuild the solution and run the project.
   Our control will now be able to lookup license for the 'MainApp' from the 'UserControlLibrary' assembly.
   In case of licensing the application on **Azure Functions** application, deploy the application on Azure Function.

   > - This licensing process requires an internet connection.
   > - For the license to apply at runtime, you must call **gclm.exe** with the target name, for example, 'AppName.PluginLibraryName.dll'.
   > - The build action of **.gclicx** file in the project must set to **Embedded Resource**.

Licensing Build Agents/Pipelines

MESCIUS licensing is based upon the development machine. However, in a pipeline environment, the license needs to be activated on a virtual machine or docker on the cloud. A pipeline license is designed for the build agent of DevOps pipelines like Azure, AWS, and Bitbucket. If your pipeline has a dynamic agent (for example, Azure pipeline with Microsoft-hosted agents) you must activate the license before each build and deactivate the license after each build and, depending on the type of license, you may need to deactivate the license after each build. See the instructions below for your type of license.

Activation with Pipeline License

This method of activation is the recommended one.

With a pipeline license you may build and deploy applications from a dynamic build agent with fewer limitations than a standard developer license. A pipeline license allows unlimited activations per day with no need to deactivate the key. The serial key must still be activated before each build, but it does not require network access and it cannot be used for design-time development.

You should contact our sales team at activereports.sales@mescius.com or call 1.800.858.2739 for issuing the pipeline license. A pipeline license includes three things:

1. Serial Key
2. Local Activation Package File
3. Password

To activate your pipeline serial key on the build server follow these steps:

1. Install the MESCIUS License Manager tool in the pipeline.
    1. If ActiveReports or any MESCIUS inc. product is installed, then the gclm tool is already installed at *C:\ProgramData\MESCIUS\gclm*
    2. If ActiveReports or any MESCIUS inc. product is not installed, you may download the gclm deployment tool:
        - Windows: https://cdn.mescius.com/license/gclm_deploy.exe
        - Linux/MacOS: https://www.nuget.org/packages/GrapeCity.LicenseManagerTool/ or you may install the gclm tool from command line:
          `dotnet tool install -g GrapeCity.LicenseManagerTool`

2. Copy the Local Activation Package file to the gclm install location.
    - Windows: *C:\ProgramData\MESCIUS\gclm*
    - Linux/MacOS: *~/.local/share/MESCIUS/gclm*

3. Before the build, call the gclm tool to activate the license using your serial key and provided password.
    - Windows:
      `C:\ProgramData\MESCIUS\gclm\gclm.exe "463c4179-288c-48fe-a4df-3b609586667d" -a "key" “password”`
    - Linux/MacOS:
      `gclm "463c4179-288c-48fe-a4df-3b609586667d" -a "key" “password”`

> **Note:** You may also perform the activation (step 3) using the MESCIUS License Manager GUI if command line interface is not required. See Licensing a Developer Machine and follow the steps for activating your pipeline key. It will prompt you to enter the password separately.

Activation with Standard Developer License

With a standard developer license you may build and deploy applications from a dynamic build agent with some limitations.

1. Your serial key must be activated and deactivated for each build.
2. You may only activate and deactivate a single key up to 9 times per day.

To activate your serial key on the build server follow these steps:

1. Install the MESCIUS License Manager tool (gclm) in the pipeline.
    a. If ActiveReports or any MESCIUS inc. product is installed, then the gclm tool is already installed at *C:\ProgramData\GrapeCity\gclm*

   b. If ActiveReports or any MESCIUS inc. product is not installed, you may download the gclm deployment tool:
      - Windows: https://cdn.mescius.com/license/gclm_deploy.exe
      - Linux/MacOS: https://www.nuget.org/packages/GrapeCity.LicenseManagerTool/ or you may install the gclm tool from command line:
        `dotnet tool install -g GrapeCity.LicenseManagerTool`

2. Before the build, call the gclm tool to activate the license
   - Windows:
     `C:\ProgramData\GrapeCity\gclm\gclm.exe "463c4179-288c-48fe-a4df-3b609586667d" -a "key"`
   - Linux/MacOS:
     `gclm "463c4179-288c-48fe-a4df-3b609586667d" -a "key"`

3. After the build, call the gclm tool to deactivate the license
   - Windows:
     `C:\ProgramData\GrapeCity\gclm\gclm.exe "463c4179-288c-48fe-a4df-3b609586667d" -d`
   - Linux/MacOS:
     `gclm "463c4179-288c-48fe-a4df-3b609586667d" -d`

Notes:

- The key should be deactivated even if the build is failed, otherwise the license cannot be activated for next build agent. You may contact our support team to resolve the issue if a key cannot be deactivated. If deactivation may be a problem, we recommend upgrading to a pipeline license.
- The environment should have the network to access https://sa2.grapecity.com/ website.
- By default, there is a limit of nine activations/deactivations per 24 hour period. This is known as reactivation.

Check the License Status

To check whether the license is activated or not, follow these steps:

1. Create a file, say 'info.txt' with the product name and GUID specified

   ```
   C:\ProgramData\GrapeCity\gclm>gclm "463c4179-288c-48fe-a4df-3b609586667d" > "info.txt"
   ```

2. Check the license details: license status, license type, serial key, activation date, expiration date, and edition

   ```
   C:\ProgramData\GrapeCity\gclm>more info.txt
   ```

Fixing Licensing Errors

Here are some of the common licensing errors and their causes.

Error	Cause
Application cannot run because it was built with no license.	Licensing is not present in the application or the calling application. See below for information on how to license the calling application.
License for XXXX (control name) could not be found.	Extra lines for components that you do not use are in the licenses.licx file. Delete unnecessary information and Rebuild the project.

Licensing has not been correctly applied to the application.	Check the three key points below.
Exception (LicenseException)	Check the three key points below.
'No License' message on running **.NET Core** applications in Visual Studio 2017	This is so because Visual Studio 2017 does not support core license compilation.

1. **Ensure that the license file is added to the appropriate project.**

   The licenses.licx file is automatically generated in the project where ActiveReports is used. But if your application is composed of multiple projects and another project calls the reports defined in your class library, you need to register it in the calling project rather than just in your report project.

   When you add an ActiveReports web service to a Page report, RDLX report, or XML-based Section report project, the licenses.licx file is not created automatically, and the license strings are not added. You also need to manually add licensing to your application if you want to create a control at run time or use the HTTP handlers.

   To manually add licensing to the calling application:

   > **Note:** In a C# Windows Forms project, the license file is in the **Properties** folder. In a Visual Basic project, it is in the **My Project** folder.

   1. In your application that contains ActiveReports components, check that the proper licensing strings are in the licenses.licx file. (See the table of license strings below.)
   2. Copy the ActiveReports strings from the file into the licenses.licx file in the calling application.
      a. If there is no license file, from the **Project** menu, select **Add New Item**.
      b. From the listed templates, select **Text** file and change the name to "**licenses.licx.**"
      c. In the Solution Explorer, double-click the newly created licenses.licx file to open it, and paste in the licensing strings for all components you use.
   3. From the **Build** menu, select Rebuild Project to embed the licensing.

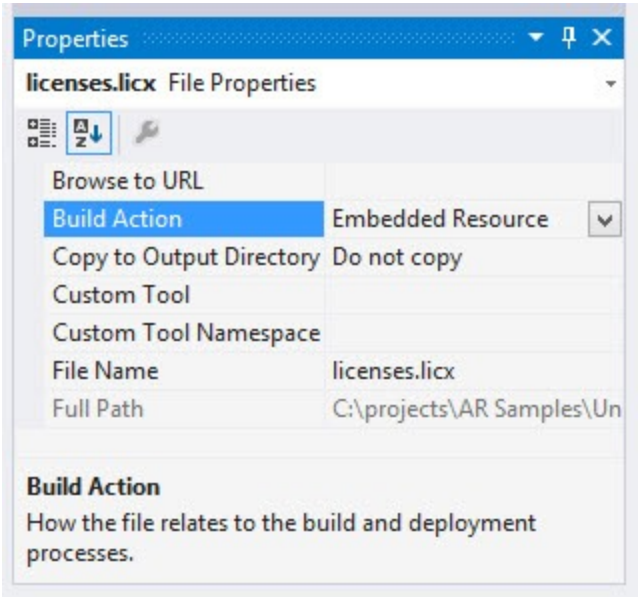
   > **Note**: If your project is a web site, the bin folder has the licenses embedded in the App_Licenses.dll file.

2. **Ensure that the contents of the license file are correct.**

   Depending on which features of ActiveReports you use in your application, the license file may need to contain multiple license strings.

   You will find a full list of license strings that you may need in the **Required references in the licenses.licx file (for Standard and Professional Editions)** section above.

3. **Ensure that the Build Action property is configured correctly for the license file.**



1. In the Solution Explorer, select licenses.licx (you may need to click the **Show all files** button to see it).
2. In the Properties window, ensure that the **Build Action** property is set to **Embedded Resource**.

4. **Ensure that .NET Core applications are always run on Visual Studio 2019 or above**

   .NET Core applications should be run only in **Visual Studio 2019** or above since Visual Studio 2017 does not support core license compilation.

To license a WebSite application

1. In the Solution Explorer, right-click the licenses.licx file and select **Build Runtime Licenses** to create the App_Licenses.dll file.
2. To generate the .gclicx file, using the GCLM, use one of these commands:
   C:\ProgramData\MESCIUS\gclm\gclm.exe "[AR GUID]" -lc "[output dir]/.gclicx" "[entry assembly name].dll"
   C:\ProgramData\MESCIUS\gclm\gclm.exe "[AR GUID]" -lc "[output dir]/.gclicx" "[executing assembly name].dll"
   C:\ProgramData\MESCIUS\gclm\gclm.exe "[AR GUID]" -lc "[output dir]/.gclicx" "[entry assembly name].[executing assembly name].dll"
3. Copy the .gclicx file and paste it into your project, in any folder.
4. Change the **Build Action** property for the .gclicx file to **Embedded Resource**.
5. Rebuild the solution and run the project.

Contacting Support

If you still face problems licensing ActiveReports, please contact our **Support Team** using any of these channels:

Web site	https://developer.mescius.com/support/contact
E-mail	activereports.sales@mescius.com
Phone[1]	(412) 681-4738

[1]Phone support is available for customers that have **Platinum Support** included with their purchased products or **issues** related to licensing or installation of a product. Monday through Friday during regular business hours (EST).

For information on Platinum Support, visit our website at https://developer.mescius.com/support/plans/.

Working with Reports

ActiveReports .NET is a developer reporting tool to embed reports within web and desktop applications. This section provides information on the tasks performed by developers while creating comprehensive reports. The report creation utilizes the .NET's Visual Studio-Integrated Designer for data visualization within the Visual Studio IDE while Page/RDLX Reports and Section Reports

Before going through this section, we recommend that you read the Report Authors section to have a minimum basic knowledge, required for a report author.

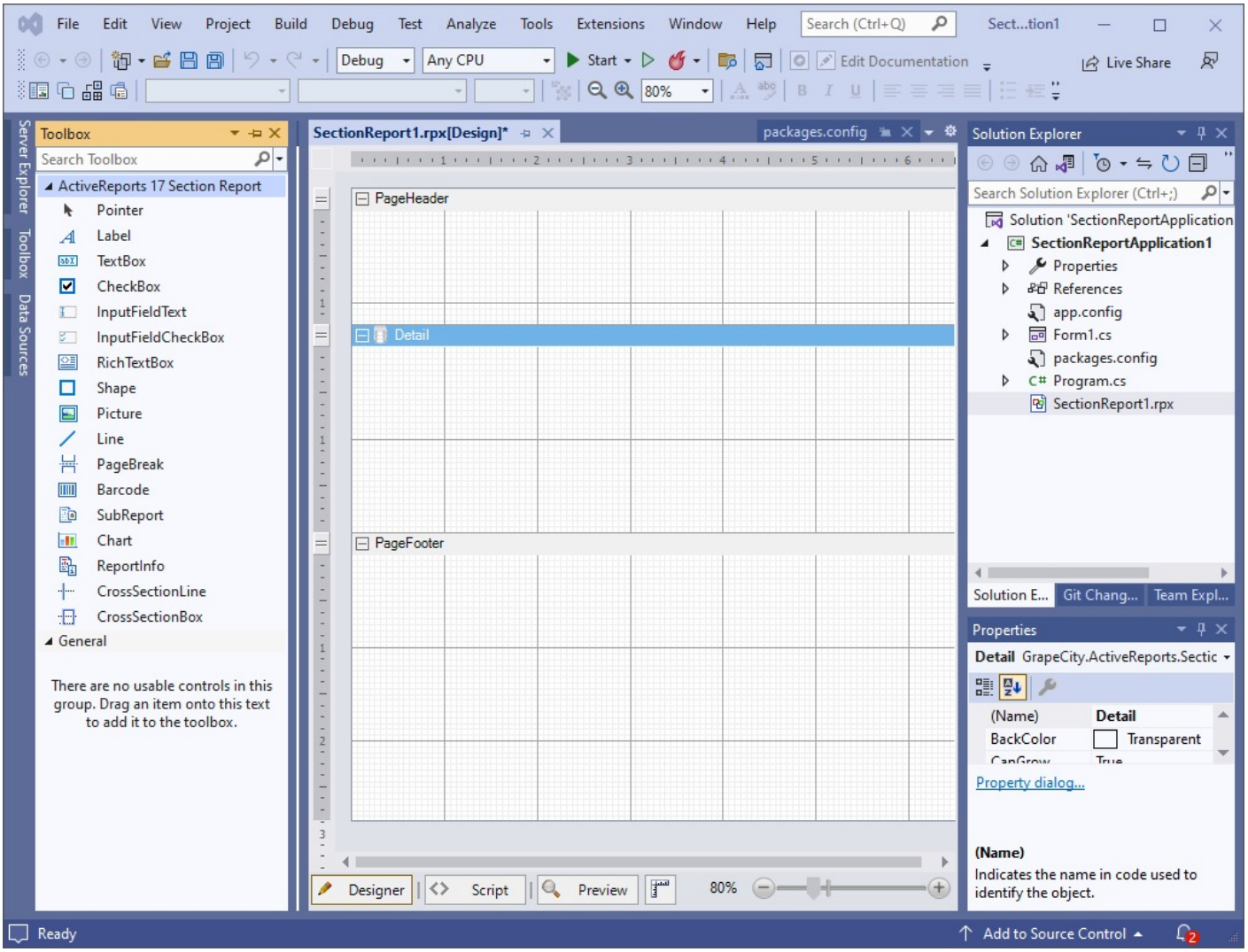
Visual Studio Integrated Designer

ActiveReports offers an integrated designer that lets you create report layouts in Visual Studio and edit them at design time, visually, and through code, script, or regular expressions. Like any form in Visual Studio, it includes a Properties panel with extensive properties for each element of the report, and also adds its own Toolbox filled with report controls, and a Report Explorer with a tree view of report controls.

The designer supports three types of report layouts: section layout, page layout, and rdl layout.

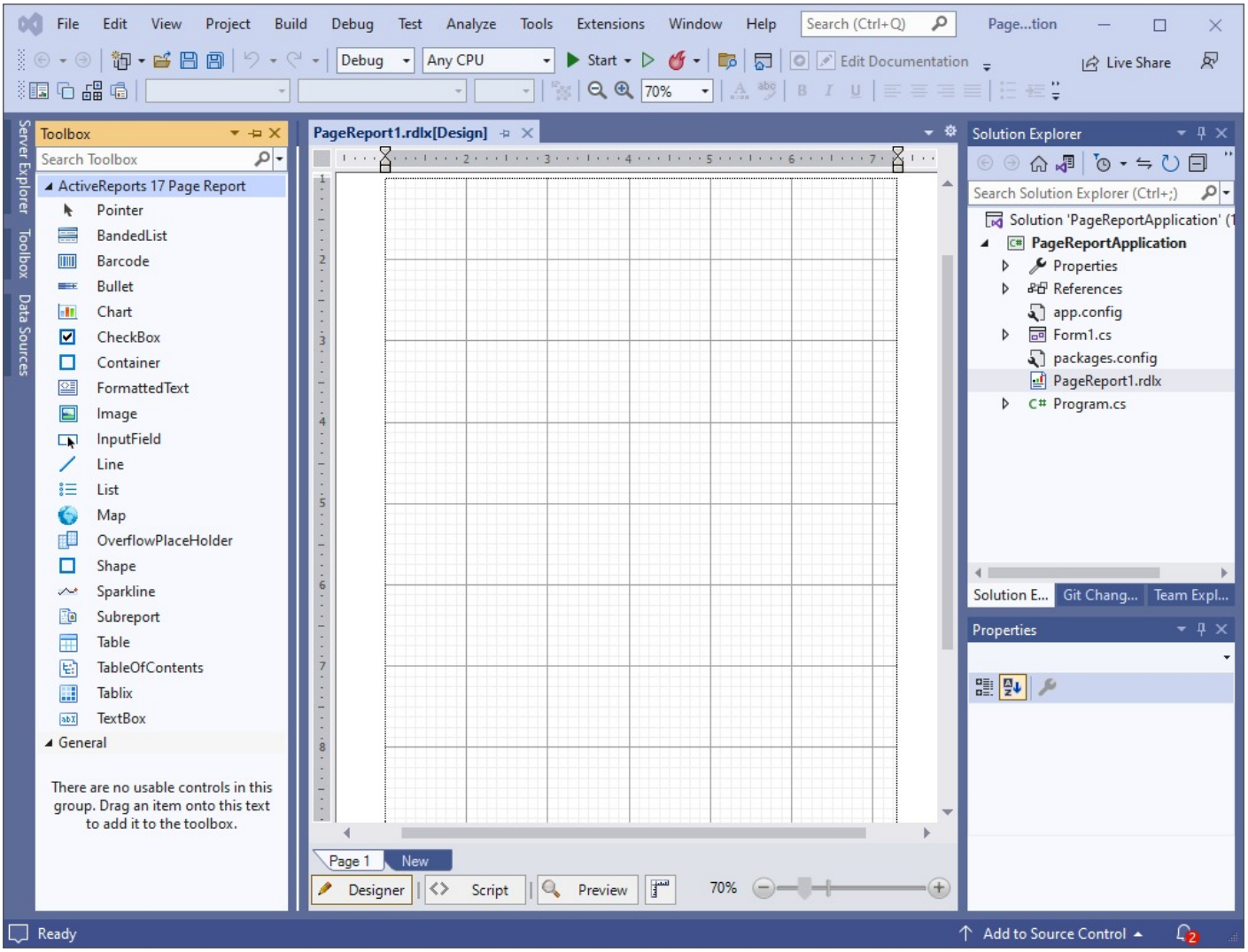
Section Report

This layout presents reports in three banded sections by default: page header, detail and page footer. You can remove the page header and footer, add a report header and footer, and add up to 32 group headers and footers. Drag controls onto these sections to display your report data. Reports designed in this layout are saved in RPX format.

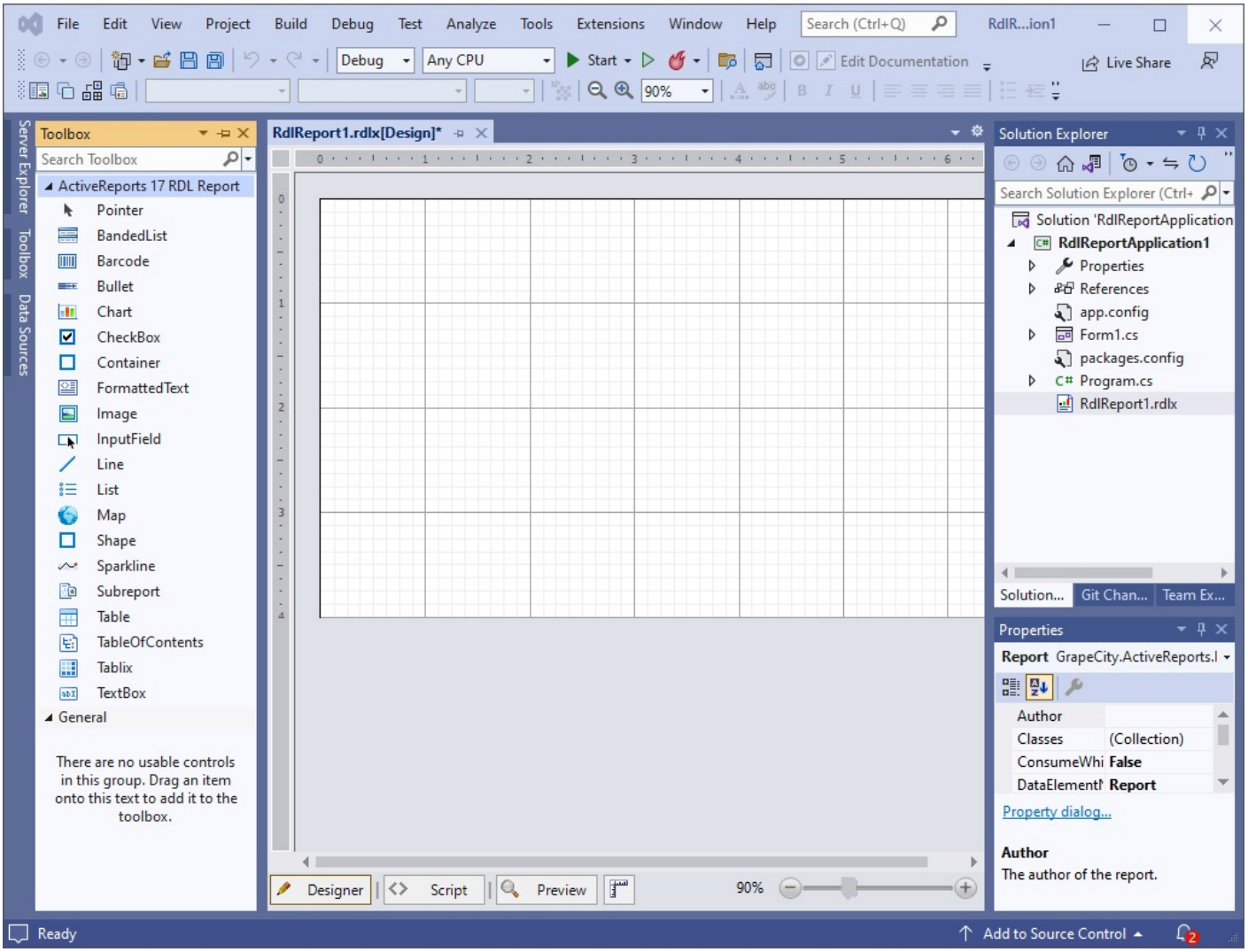


Page Report

This layout defines reports in pages where the same page layout can be used throughout the report or separate layout pages are designed for complex reports. Reports designed in this layout are saved in Rdlx format.



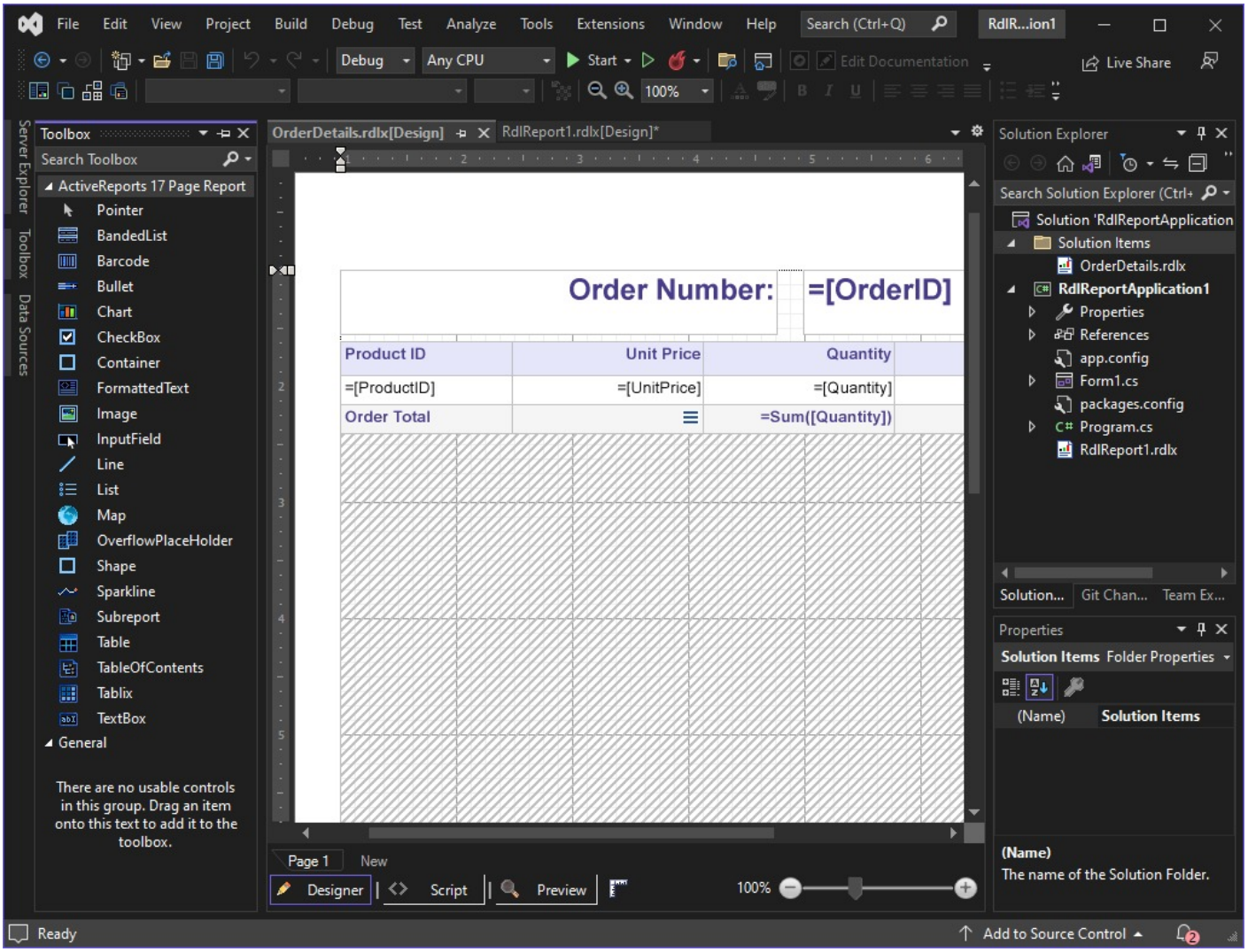
RDLX Report



This layout defines reports where controls grow vertically to accommodate data. Reports designed in this layout are saved in Rdlx format.

Note that a theme you select for your Visual Studio, is automatically applied to the ActiveReports Designer. For example, if you select the **Dark** theme for your Visual Studio, this theme is applied automatically to the Designer. The Visual Studio theme is extended to such ActiveReports Designer elements as Reports Library, Layer List, Report Explorer, and Group Editor.

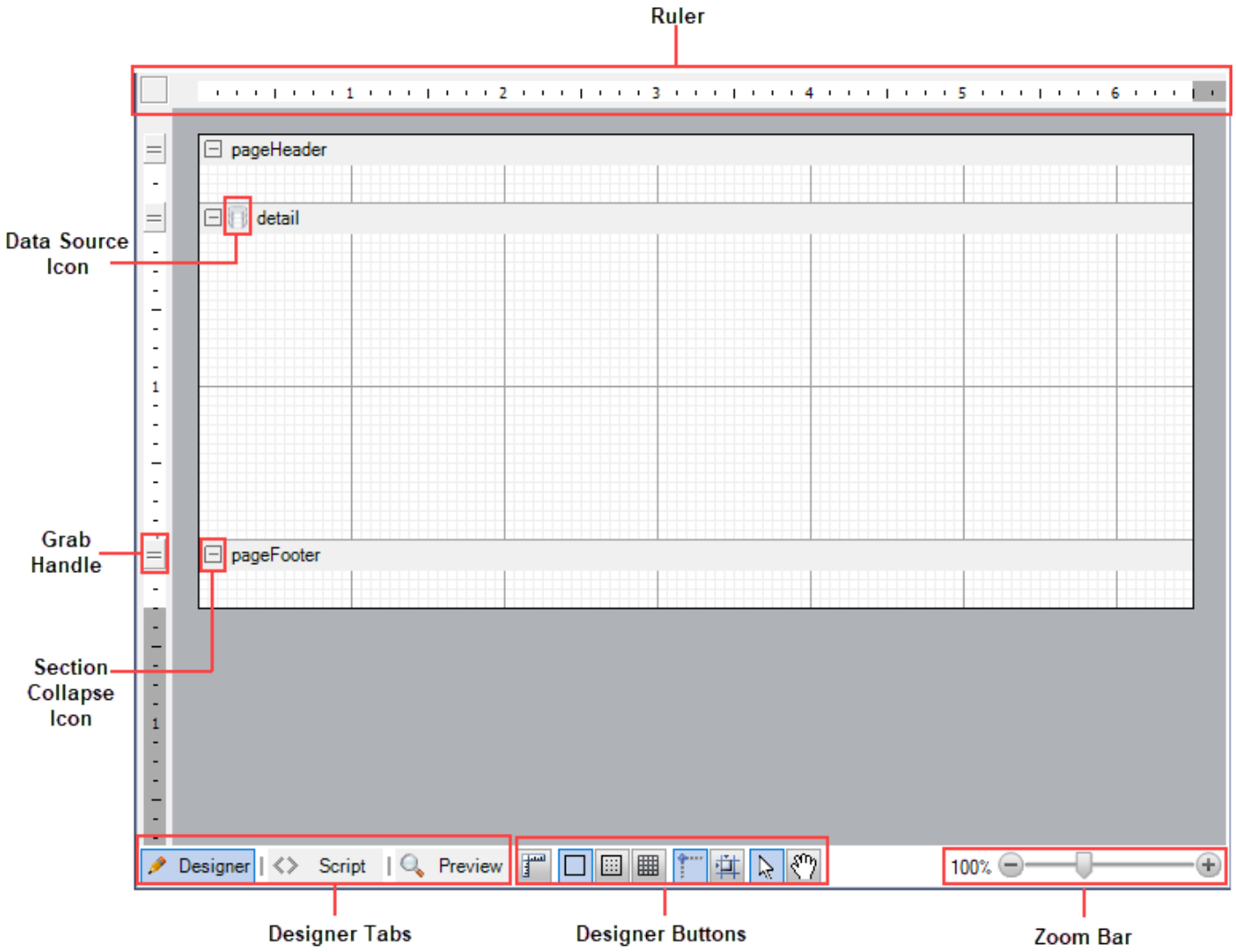
The theme integration works for all reports (Page, RDLX, and Section) opened in supported Visual Studio versions.



Design View

The report designer is fully integrated with the Microsoft Visual Studio IDE. In this topic, we introduce the main parts of the designer in Section Report, Page Report, and RDLX report to help you select the one to best suit your specific needs.

Report Designer in Section Reports



Design Surface

The design surface offers a default report structure that contains a page header, a detail section, and a page footer along with some grey area below these sections. Drag report controls and fields onto these sections to display your data. Use section grab handles to drag a section's height up or down. Right click the report and select **Insert** to add other types of header and footer section pairs.

Data Source Icon

The Data Source icon is located in the band along the top of the detail section. Click this icon to open the **Report Data Source** dialog, where you can bind your report to any OLE DB, SQL, or XML data source.

Section Collapse Icon

A Section Collapse icon (-) appears on each band adjacent to the section header. When you click the collapse icon the section collapses and an expand icon (+) appears. Please note that section collapse is only available in the **Designer** tab. All sections of the report are visible in the **Preview** tab or when the report is rendered.

Tip: In order to make a section invisible, set the **Height** property of the section to 0 or the **Visible** property to False.

Rulers

Rulers are located at the top and left of the design view. They help a user visualize the placement of controls in the report layout and how they appear in print. Please note that you have to add the right and left margin widths to determine whether your report fits on the selected paper size. The left ruler includes a grab handle for each section to resize the section height. See Rulers for more information.

Grab Handles

Grab handles on the vertical ruler indicate the height of individual sections. You can drag them up or down to change section heights, or double-click to automatically resize the section to fit the controls in it.

Designer Tabs

The designer provides three tabs: **Designer**, **Script** and **Preview**. You can create your report layout visually in the Designer tab, add script to report events in the Script tab to implement .NET functionality, and see the result in the Preview tab. See Designer Tabs for more information.

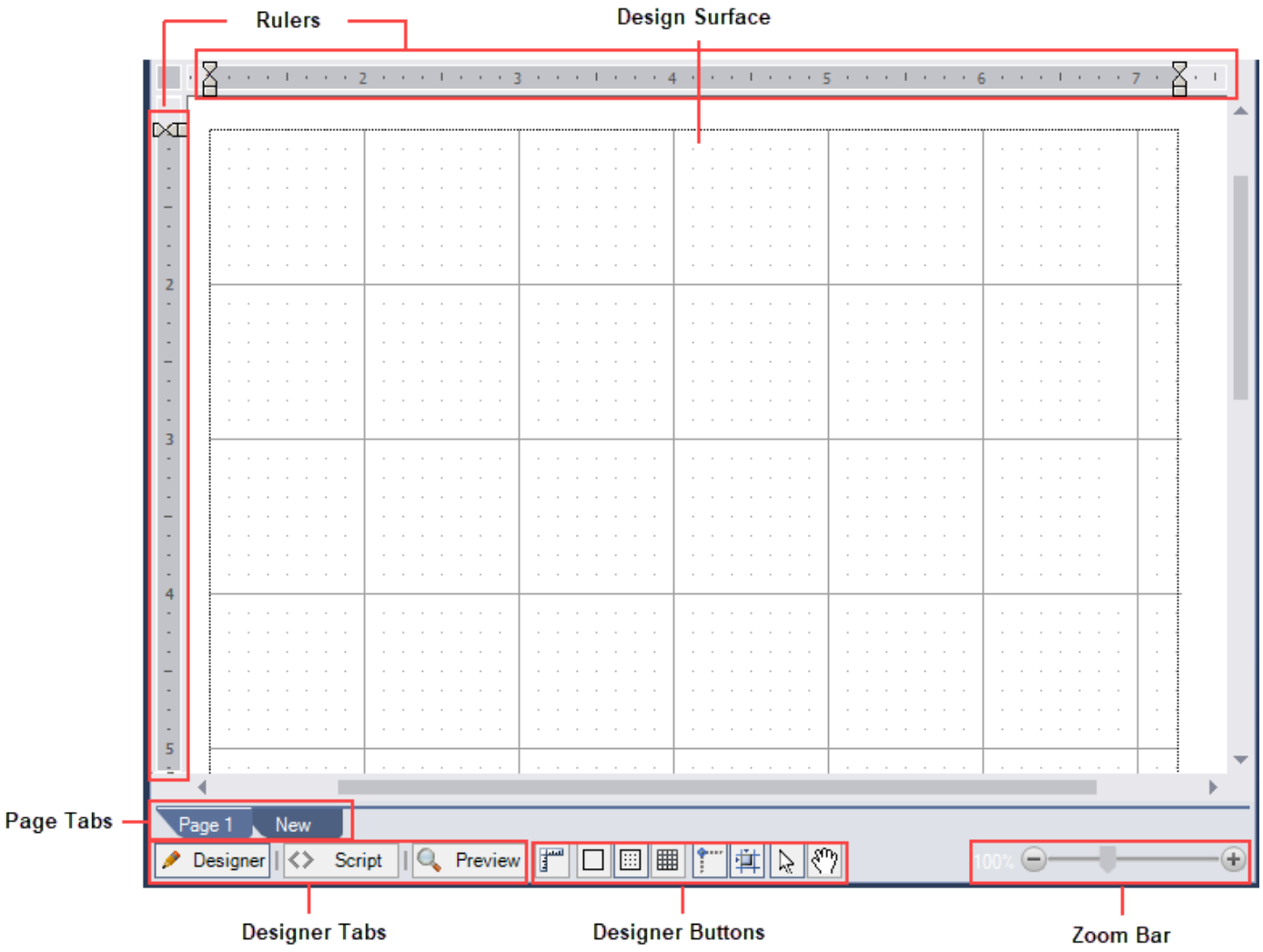
Designer Buttons

Designer buttons are located below the design surface next to the designer tabs. **Dimension Lines**, **Hide Grid**, **Dots**, **Lines**, **Snap to Lines**, and **Snap to Grid** buttons help you to align report controls and data regions. The **Select Mode** and **Pan Mode** buttons determine whether you select controls on the design surface, or move the visible area of a zoomed-in report. See Designer Buttons for more information.

Zoom Bar

The zoom bar provides a slider that you drag to zoom in and out of the design surface, or you can use the **Zoom in** and **Zoom out** buttons at either end of the slider. See Zoom Support for more information.

Report Designer in Page reports/RDLX reports



In a Page Report or an RDLX Report, the designer offers the following features that you can use to create, design and edit a report.

Design Surface

The design surface of a report appears initially as a blank page and grid lines. You can create your own layout and drag report controls and fields onto the design surface to display your data.

Rulers

Use the ruler to determine how your report will look on paper. Please note that you have to add the right and left margin widths to determine whether your report will fit on the selected paper size. See Rulers for more information.

Designer Tabs

The designer provides three tabs: **Designer**, **Script** and **Preview**. You can create your report layout visually in the Designer tab, add script to report events in the Script tab to implement .NET functionality, and see the result in the Preview tab. See Designer Tabs for more information.

Page Tabs(Page Report)

By default, the designer provides two page tabs, **Page 1** and **New**, below the design surface. Each page tab represents a layout page of the report. **Page 1** represents the first page of your report, and you can click **New** to add another page to your report. See Page Tabs for more information.

Designer Buttons

Designer buttons are located below the design surface next to the designer tabs. **Dimension Lines**, **Hide Grid**, **Dots**, **Lines**, **Snap to Lines**, and **Snap to Grid** buttons help you to align report controls and data regions. The **Select Mode** and **Pan Mode** buttons determine whether you select controls on the design surface, or move the visible area of a zoomed-in report. See Designer Buttons for more information.

Zoom Bar

The Zoom Bar provides a slider that you drag to zoom in and out of the design surface, or you can use the **Zoom in** and **Zoom out** buttons at either end of the slider. See Zoom Support for more information

> **Tip:** ActiveReports provides some useful keyboard shortcuts for the controls placed on the design surface.
>
> - **Arrow Keys:** To move control by one grid line.
> - **[Ctrl] + Arrow Keys:** To move control by 1/100 inch (around 0.025 cms)
> - **[Shift] + Arrow Keys:** To increase or decrease the size of the control by one grid line.

Report Menu

The Report menu provides access to common reporting operations. To show the Report Menu in the Visual Studio menu bar, select the Design View of the report in the ActiveReports Designer. This menu does not appear in the menu bar when the report is not selected.

The following drop-down sections describe the Report menu items. Menu items differ based on the type of report layout in use.

> **Note**: The Report menu in Visual Studio 2019 and Visual Studio 2022 is available as submenu under **Extensions**.

Report Menu for Section Reports

Menu Item	Description
Save Layout	Opens the **Save As** dialog to save the newly created report in RPX file format.
Load Layout	Opens the **Open** dialog where you can navigate to any RPX file and open it in the designer. Note that any changes to the current report are lost, as the layout file replaces the current report in the designer.
Data Source	Opens the **Report Data Source** dialog to bind a data source to the report.
Settings	Opens the **Report Settings** dialog.
View • Designer • Script • Preview	Opens the Designer, Script or Preview tab. See Designer Tabs for more details.

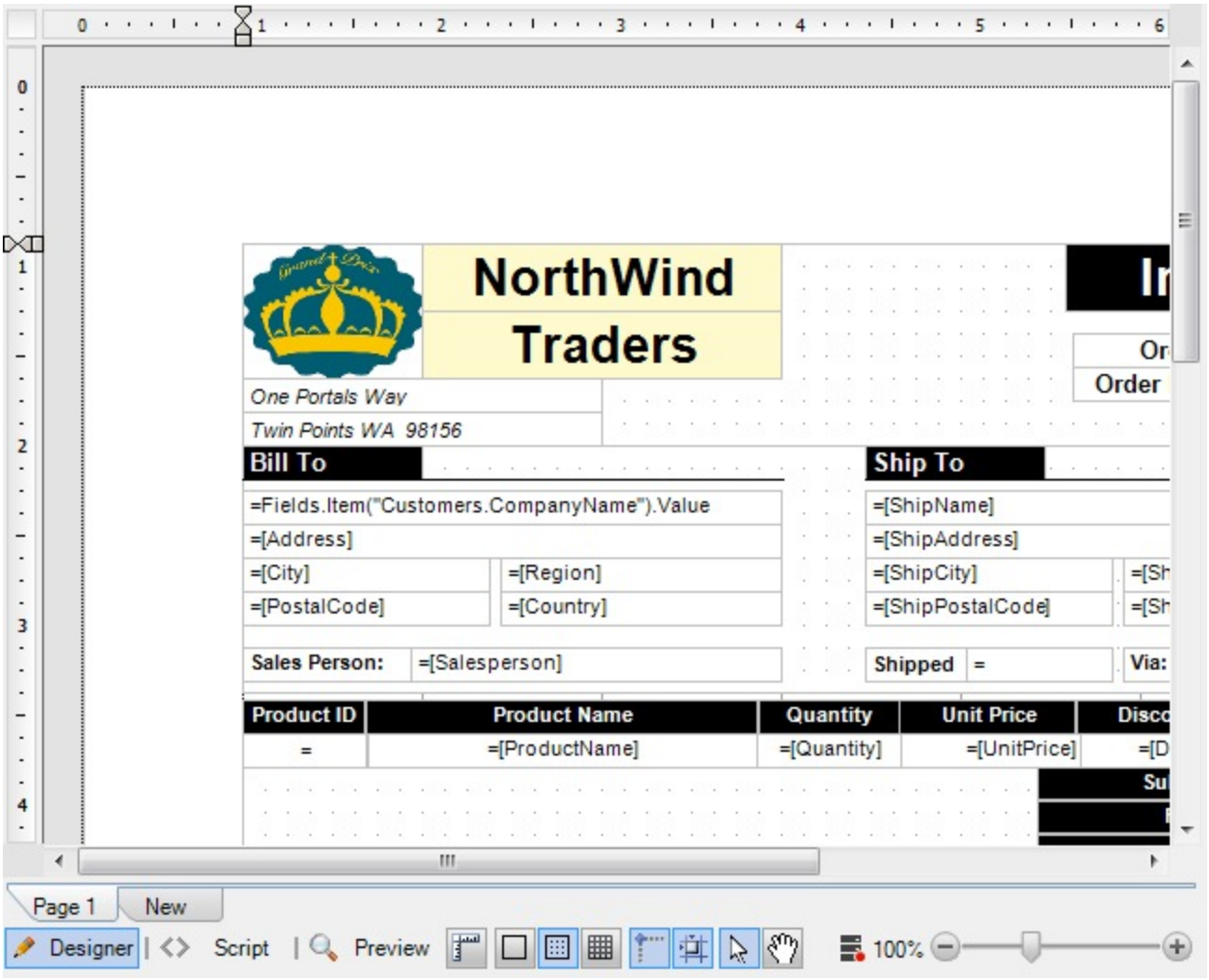
Report Menu for Page and RDLX reports

Menu Item	Description
Save Layout	Opens the **Save As** dialog to save the newly created report in RDLX file format.
Load Layout	Opens the **Open** dialog where you can navigate to any RDLX, RDLX, RDLX-master file and open it in the designer. Note that any changes to the current report are lost, as the layout file replaces the current report in the designer.
Convert to Master Report (RDLX Reports only)	Converts an RDLX Report to a Master Report. It disappears from the Report menu when a master report is applied to the report through **Set Master Report**. See Master Report (RDLX Report) for more details.
Report Parameters	Opens the **Report** dialog to the **Parameters** page where you can manage, add and delete parameters.
Embedded Images	Opens the **Report** dialog to the **Images** page, where you can select images to embed in a report. Once you add images to the collection, they appear in the Report Explorer under the **Embedded Images** node.
Report Properties	Opens the **Report** dialog to the **General** page where you can set report properties such as the author, description, page header and footer properties, and grid spacing.
Stylesheet Editor	Opens the **Stylesheet Editor** dialog, where you can create, edit or remove styles. You can also embed these styles in a style sheet or save them externally in ***.rdlx-styles** format. Embedded style sheets appear under the **Embedded Stylesheets** node in the Report Explorer.
Report Parts	Opens the **Exported Report Parts** dialog, where you can add or remove report parts. You can also modify the added report part properties, exposed in the dialog. See Report Parts for details.
Set Master Report (RDLX Reports only) • Open Local File	Select Open Local File option to open the **Open** dialog, and then select a master report.
View • Designer • Script • Preview	Opens the Designer, Script or Preview tab. See Designer Tabs for more details.
Page Header (RDLX Reports only)	Toggles the report Page Header on or off.
Page Footer (RDLX Reports only)	Toggles the report Page Footer on or off.

Designer Tabs

The Designer has three tabs located at the bottom of the report design surface. Create a report layout in the Designer tab, write a script in the Script tab to implement .NET functionality and see the result in the Preview tab.

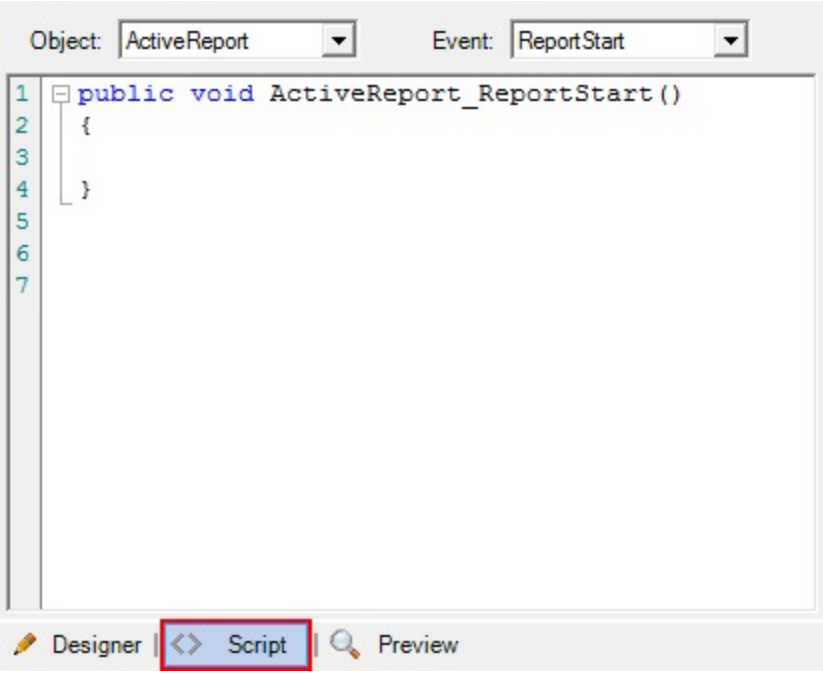
Designer Tab



The Designer tab appears by default on your designer. This tab is used to design your report layout visually. You can implement most of the design-time features here, drag controls from the toolbox to create a layout, bind data regions to data, and set properties for the report and controls through the context menu.

> **Tip:** Layout-related features like designer buttons and zoom slider can be used in this tab to help you manage your report display efficiently.

Script Tab



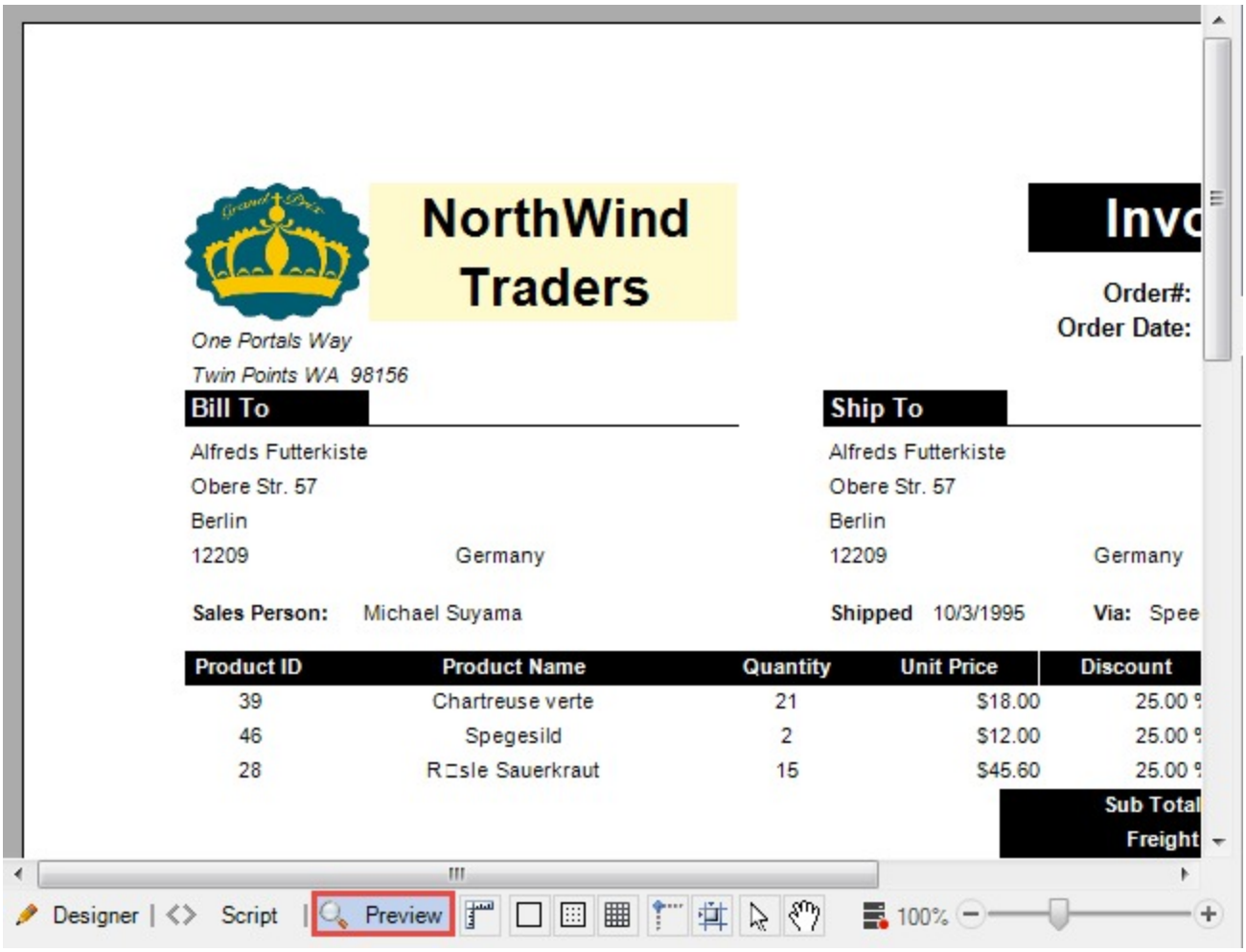
The Script tab opens the script editor, where you can provide VB.NET or C# functionality to the reports without compiling the .vb or .cs files. You may use either Visual Basic or C# script in this tab with Section Reports, or Visual Basic with Page reports and RDLX reports.

The generated reports serve as standalone reports which you can load, run, and display in the viewer control without the designer. This feature is useful when distributing reports without recompiling.

In Page reports/RDLX reports, you can embed code blocks that you can reference in the expressions you use on report controls. See Add Code to Layouts Using Script for more information about using script.

In Section Reports, you can add code to object events. The two drop-down boxes in the script editor 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. See Add Code to Layouts Using Script for more information about scripting in Section Reports.

Preview Tab



The **Preview tab** allows you to view your report 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.

This tab does not display data in the following conditions:

- Code or script in the report class is incorrect.
- Report class constructor has been changed.
- Report data source has not been set correctly.
- Settings have been implemented outside the report class
- .mdb file is being copied in the project

When the report is inherited from a class other than ActiveReports, preview is possible only when the base class is in the same project. If the base class is not in the same project and is referencing an external class library, you will not get a preview in the **Preview** tab.

When adding a report directly to ASP.NET Web site, the **Preview** tab is not visible and thus you cannot preview.

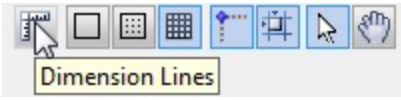
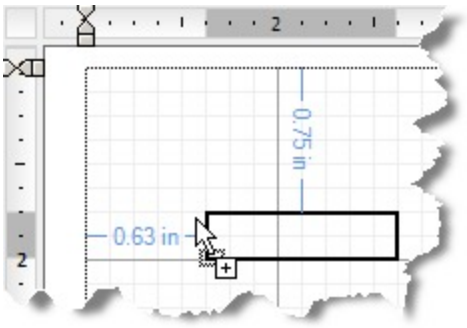
Designer Buttons

Designer buttons are located to the right of the designer tabs along the bottom of the designer, and are enabled when you are on the Designer tab. They allow you to control settings for the design surface.

Grid Settings

Dimension Lines

Dimension lines appear during a drag operation, and run from the borders of the report control or data region being moved or resized to the edges of the report designer surface. Dimension lines let you track the location of the control as you move it by displaying the distance between the control and the edge of the writable area of the report.

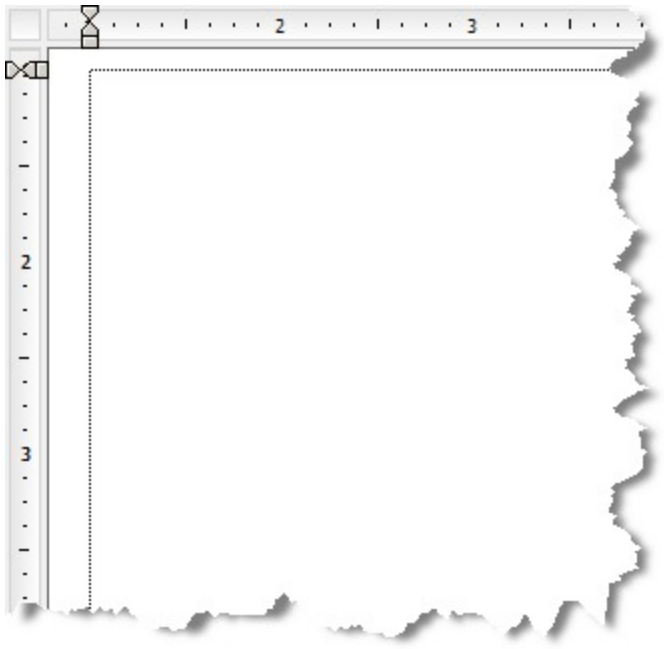
Button Type	Behavior
	

Note: With Section Reports, you can change the number of grid columns and rows in the Report Settings dialog on the Global Settings tab. With Page reports and RDLX reports, you can change the grid spacing in the Report Properties dialog on the General tab.

Hide Grid

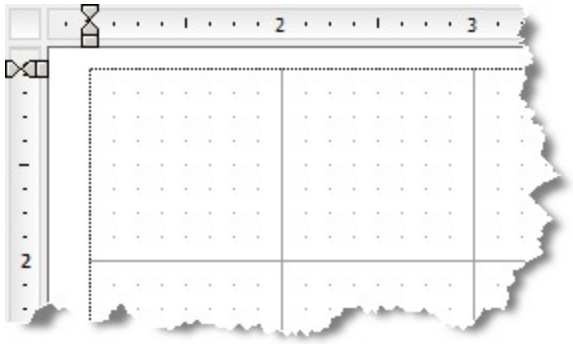
By default, grid lines and dots appear on the report design surface. You can click this button to hide the grid and design your report on a blank page. Lines or dots are also removed from the design surface when you hide the grid, but Snap to Lines or Snap to Grid settings remain unaffected.

Button Type	Behavior

Show Dots

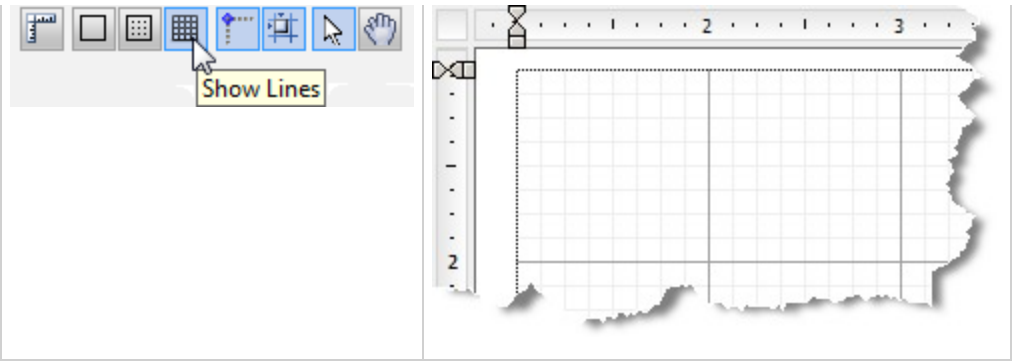
You can click this button to have dots appear on the design surface in between the grid lines to guide you in the placement of controls.

Button Type	Behavior
	

Show Lines

You can click this button to have faint grey lines appear on the design surface in between the grid lines to guide you in the placement of controls.

Button Type	Behavior



Note: Only one option out of Hide Grid, Show Dots and Show Lines can be selected at one time.

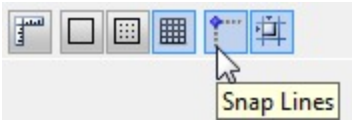
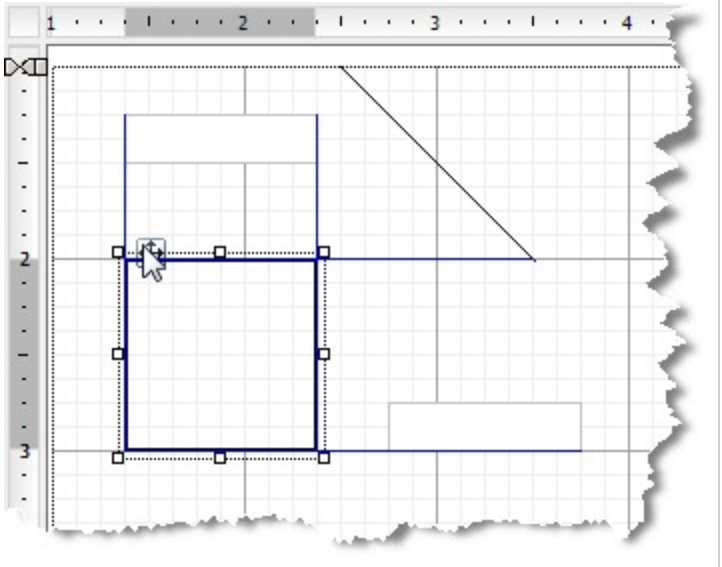
Control Drag and Drop Settings

These settings allow you to specify how you want controls to behave when you drag and drop them on the design surface.

Tip: If you plan to export a report to Excel format, use Snap Lines or Snap to Grid to ensure that your controls are aligned in columns and rows as it prevents overlapping. This makes the export to excel closer to how a report looks at run or design time.

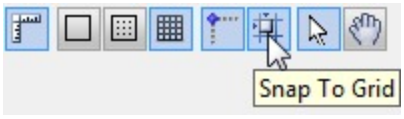
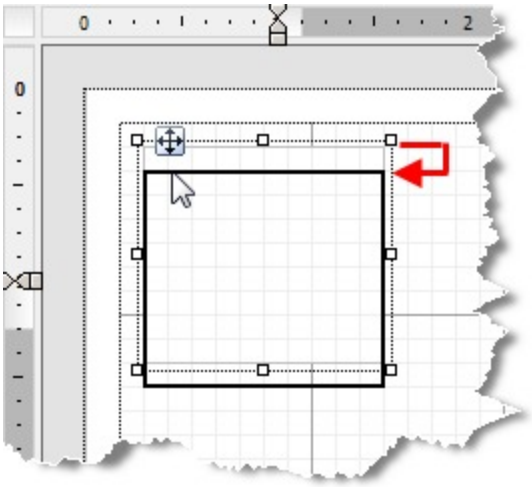
Snap Lines

This setting aligns the control you are dragging with other controls on the report design surface. When you drag the control around, snap lines appear when it is aligned with other controls or with the edges of the report, and when you drop it, it snaps into place in perfect alignment. See Snap Lines for more information.

Button Type	Behavior
	

Snap to Grid

This setting aligns the control you are dragging with grid lines on the report design surface. When you drop the control, it snaps into place in alignment with the nearest grid mark. To place your controls freely on the report design surface, turn this setting off.

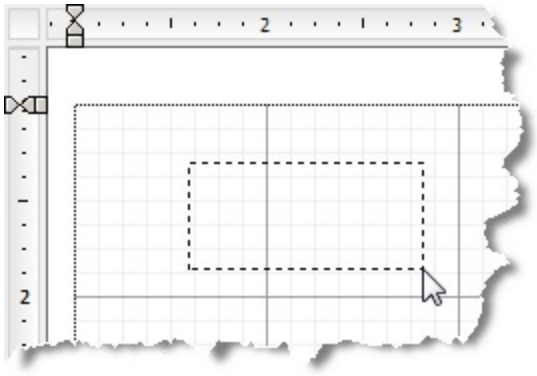
Button Type	Behavior
	

Mouse Modes

These settings allow you to specify how you want the mouse to behave in the designer.

Select Mode

In Select mode, when you click items on the report designer surface, you select them. Use this mode for editing, data binding and styling in the Designer tab. An arrow cursor appears in the Select mode.

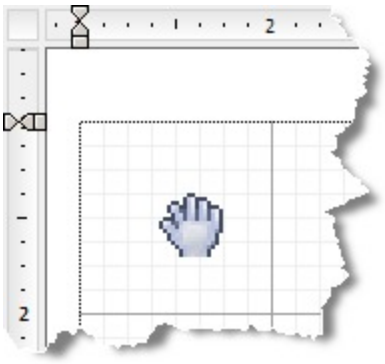
Button Type	Behavior
	

Pan Mode

Use the Pan mode to make navigation easier. In this mode, you cannot select, edit, add or delete a control from the design surface.In this mode, you cannot select, edit, or delete a control from the design surface. A hand cursor appears in Pan mode and you can navigate through your report by pressing the left mouse button and dragging the report to

the desired position.

> **Tip:** To enable Pan mode while you are in Select Mode, hold down the middle mouse button and move to the desired location with the hand cursor.

Button Type	Behavior
	

Toolbar

ActiveReports provides a toolbar integrated with the Visual Studio IDE for quick access to report designing commands. This toolbar is composed of buttons and dropdown lists which offer functions for a number of commonly used commands.

Show or Hide the Toolbar in Visual Studio

1. Create a new project or open an existing project in Visual Studio.
2. Right click on the Visual Studio toolbar and from the context menu that appears, select **ActiveReports**.

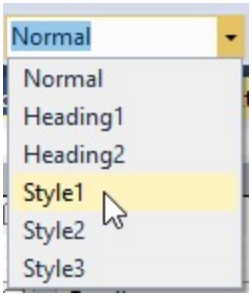
The **ActiveReports** toolbar appears under the Visual Studio menu bar. The toolbar options may differ based on whether you have a Section Report, Page Report or an RDLX Report open.

See the description of each toolbar option in the tables below.

> **Note:** Toolbar descriptions are grouped in a logical order for understanding. The buttons and dropdowns may appear in a different order in the **ActiveReports** toolbar.

Text Decoration

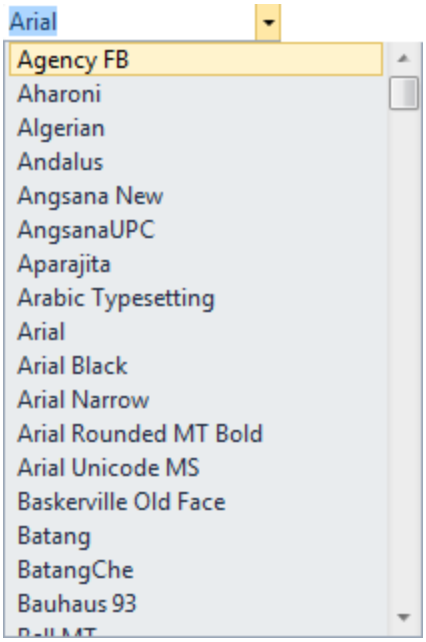
Command	Description
Style	Sets the style for the selected report control.



For details on setting styles in Page and RDLX reports, see Working with Styles.

For details on setting styles in Section Report, see Working with External Style Sheets.

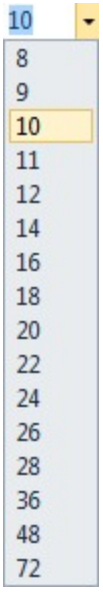
Font



Sets the typeface of all the text in a control.

In a Section Report, for the RichTextBox control, typeface of only the selected text changes. In a Page Report or an RDLX Report, in a data region like Tablix or Table, you can change the typeface of the entire data region or only the selected TextBox within the region.

Font Size



Sets the font size of all the text in a control.

In a Section Report, for the RichTextBox control, font size of only the selected text changes. In a Page Report or an RDLX Report, for a data region like Tablix or Table, you can change the font size of the entire data region or only the selected TextBox within the region.

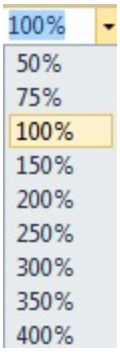
Fore Color 	Opens a **Choose Color** dialog to set the text color of controls.
Back Color 	Opens a **Choose Color** dialog to set the background color of controls.
Bold 	Sets or removes text emphasis from the entire text of the control. In Section Report, for the RichTextBox control, bold applies to the selected text only. In a Page Report or an RDLX Report, for a data region like Tablix or Table, you can change the emphasis of the entire text or only the text of the selected TextBox within the region.
Italic 	Sets or removes text slant for the entire text of the control. In a Section Report, for the RichTextBox control, italic applies to the selected text only. In a Page Report or an RDLX Report, for a data region like Tablix or Table, you can italicize the entire text or only the text of the selected TextBox within the region.
Underline 	Sets or removes the text underline for the entire text of the control. In a Section Report, for the RichTextBox control, underline applies to the selected text only. In a Page Report or an RDLX Report, for a data region like Tablix or Table, you can also underline the entire text or only the text of the selected TextBox within the region.

Text Alignment

Command	Description
Align Left 	Aligns the text to the left in the control area.
Center 	Aligns the text to the center in the control area.
Align Right 	Aligns the text to the right in the control area.
Align Justify 	Justifies the text in the control area.

Layout Editing

Command	Description
Zoom Out	Reduces the magnification level of the design surface and any elements within it.

	
Zoom In 	Increases the magnification level of the design surface and any elements within it.
Zoom 	Opens a dropdown list to set the magnification level of the design surface between 50% and 400%. Zoom percentage is set to 100% by default.

Control Alignment

Command	Description
Align to Grid 	Snaps the top left of the selected control to the closest gridline.
Align Lefts 	Aligns the selected controls with their left border coinciding with the left border of the primary control. The vertical space separating the controls remains the same.
Align Rights 	Aligns the selected controls with their right border coinciding with the right border of the primary control. The vertical space separating the controls remains the same.
Align Tops 	Aligns the selected controls with their top border coinciding with the top border of the primary control. The horizontal space separating the controls remains the same.
Align Middles 	Aligns the selected controls vertically to the middle with respect to the primary control. The horizontal space separating the controls remains the same.
Align Bottoms 	Aligns the selected controls with their bottom border coinciding with bottom border of the primary control. The horizontal space separating the controls remains the same.

Control Resizing

Command	Description
Make Same Width	Resizes the width of the selected controls to the width of the primary control.

	
Make Same Height 	Resizes the height of the selected controls to the height of the primary control.
Make Same Size 	Resizes the size (width and height) of the selected controls to the size of the primary control.
Size to Grid 	Snaps the selected control to the closest gridline by resizing the control on all four sides.

Control Spacing

Command	Description
Make Horizontal Spacing Equal 	Creates equal space between the selected controls with respect to the primary control, using the outermost edges of the controls as end points.
Increase Horizontal Spacing 	Increases the horizontal spacing by one grid unit with respect to the primary control.
Decrease Horizontal Spacing 	Decreases the horizontal spacing by one grid unit with respect to the primary control.
Remove Horizontal Spacing 	Removes the horizontal space so that the selected controls move to the nearest edge of the top-left control.
Make Vertical Spacing Equal 	Creates equal space between the selected controls with respect to the primary control, using the top and bottom edges of the control as the end points.
Increase Vertical Spacing 	Increases the vertical spacing by one grid unit with respect to the primary control.
Decrease Vertical Spacing 	Decreases the vertical spacing by one grid unit with respect to the primary control.
Remove Vertical Spacing	Removes the vertical spacing so that the selected controls move to the nearest edge of the top-left control.

Z-order Alignment

Command	Description
Bring to Front 	Moves the selected controls to the front of all other controls on the report.
Send to Back 	Moves the selected controls behind all other controls on the report.

RichTextBox commands

Command	Description
Bullets 	Adds or removes bullets from the selected text inside a RichTextBox control in a Section Report.
Increase Indent 	Increases the indent of selected text in the RichTextBox control area in a Section Report.
Decrease Indent 	Decreases the indent of selected text in the RichTextBox control area in a Section Report.

Others

Command	Description
View ReportExplorer 	Shows or hides the Report Explorer window. See Report Explorer for further details.
Reorder Groups 	Opens the Group Order dialog, where you can drag and drop groups to rearrange them. This button is enabled when you have multiple groups in a Section Report.
Layer List 	Opens the Layer List window to displays a list of Layers in the report along with their visibility and lock options. Layer List Explorer also allows you to add or remove Layers from your reports. See Layers for further details.

Note: *Primary control* is the control in a selected group of controls, to which you align all other controls. It is generally the first control selected in the group and has sizing handles (white boxes) which are different from the rest of the selected controls.

Toolbox

In ActiveReports, the Visual Studio integrated toolbox tabs display all of the controls specific to the type of report that has focus, or the ActiveReports controls that you can use on Web Forms or Windows Forms.

When a Section report has focus, the **ActiveReports 18 Section Report** toolbox becomes available. For information about the report controls available in this toolbox, please see the Section Report page.

When a Page report has focus, the **ActiveReports 18 Page report** toolbox becomes available.

When an RDLX report has focus, the **ActiveReports 18 RDLX report** toolbox becomes available. For information about the report controls available in this toolbox, please see the Page/RDLX Report topic.

When a Windows Form has focus, the ActiveReports 18 toolbox group offers the following Windows Forms controls:

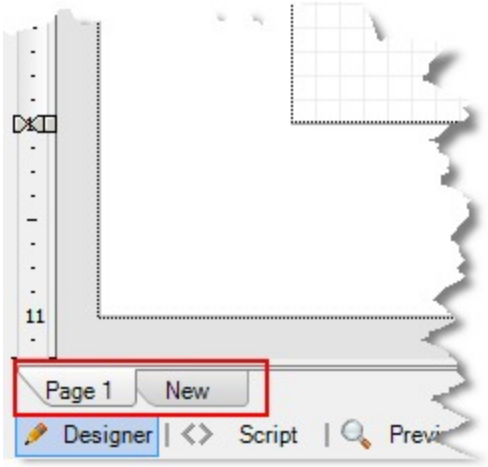
- ReportExplorer (requires Professional Edition license)
- **Toolbox ('Toolbox Class' in the on-line documentation)** (requires Professional Edition license)
- **Designer ('Designer Class' in the on-line documentation)** (requires Professional Edition license)
- **Viewer ('Viewer Class' in the on-line documentation)**

When a Web Form has focus, the ActiveReports 18 toolbox group offers one Web control: the WebViewer (requires Professional Edition license).

Page Tabs

Page tabs appear in an Excel-like bar below the report design surface. This feature is only available in Page Report, where report layouts are designed on separate pages and you can control the way each page appears. Using page tabs, you can select which page to view or edit, add new pages, remove existing pages, reorder pages, and create duplicate pages.

By default, a new report has a **Page 1** tab and a **New** tab.



- **Page 1:** This is the layout for the first page of the report. If no other page layouts exist, the layout on this page is applied to the entire report.
- **New:** Click to add a new page where you can create a layout for pages displayed after the first page.

Right-click any page tab (except the **New** tab) to get a context menu that allows you to **Insert** a new page, **Duplicate** the page, or **Delete** the page.

Adding a new page

To add a new page, click the **New** tab.

A new page tab with an incremented page number appears to the right of any existing page tabs. This page has the same page size and margins as the previous page. The **New** tab moves to the right of the newly added page.

Inserting a page

To insert a page, right-click the page tab and select **Insert**. A page is inserted to the left of the selected page. It has the same page size and margins as the selected page.

Deleting a page

To delete a page, right-click the page tab that you want to remove and select **Delete**. This option is disabled if there is only one page in the report.

Creating a copy of a page

To create a copy of a page, right-click on the page tab that you want to copy and select **Duplicate**. A copy of the selected page appears to the right of the selected page.

> **Note:** When the duplicate page contains a data region, ActiveReports replaces the data region with an OverflowPlaceHolder on the new page. Reset the **GrapeCity.ActiveReports.PageReportModel.DataRegion.OverflowName** property for the duplicated page to maintain the overflow data chain between page tabs.

Reordering pages

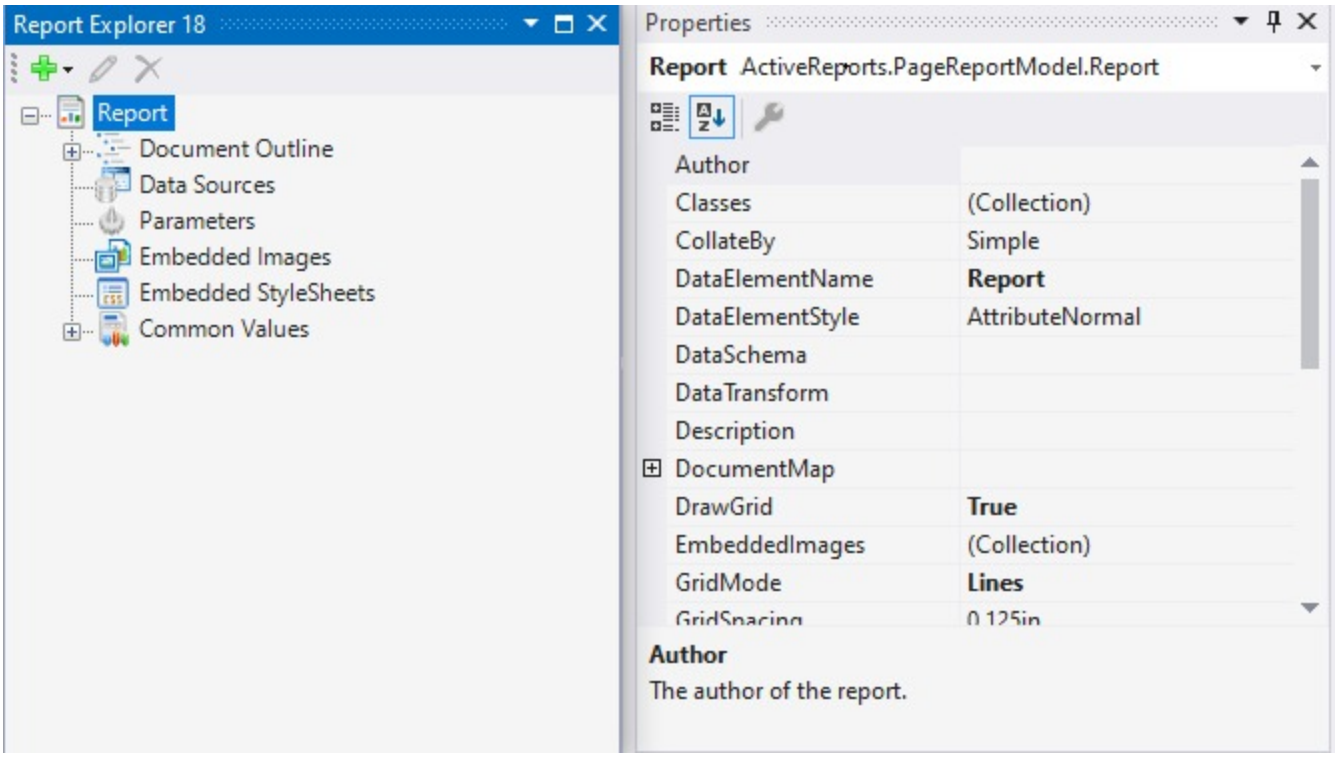
To change the order of page tabs, drag a tab and drop it at the desired location. The tab is inserted in the chosen location and the page number is updated according to its position. The page numbers of other tabs also change automatically.

You can cancel the move operation by pressing the **[Esc]** key while dragging the tab.

Report Explorer

The **Report Explorer** gives you a visual overview of the report elements in the form of a tree view where each node represents a report element.

Using the Report Explorer with any type of report, you can remove controls, add, edit or remove parameters, add a data source, add a dataset, and drag fields onto the report. You can also select the report or any element in the report to display in the Properties panel, where you can modify its properties.



ActiveReports supports four types of reports:

- Section Report (in your choice of XML-based RPX or code-based CS or VB files)
- Page Report (in XML-based RDLX files)
- RDLX Report
- RDLX Dashboard Report

These reports are composed of different types of report elements, so the Report Explorer shows different elements in the report tree depending on the type of report you have open.

Show or hide the Report Explorer in Visual Studio

Once you add the Report Explorer in Visual Studio, it appears every time you create a new Windows application. Use the steps below to hide it when you do not need it.

1. Right-click on the Visual Studio toolbar and select **ActiveReports** to display the report designer toolbar. See Toolbar for further details.
2. On the Designer toolbar, click the **View ReportExplorer** button. The Report Explorer window appears.
3. To hide the Report Explorer, follow the steps above and toggle **View ReportExplorer** back off.

> **Tip**: Another way to show the Report Explorer window in Visual Studio, is from the **View** menu, select **Other Windows**, then **Report Explorer**.

Change control properties

1. In the Report Explorer, select the control for which properties are to be changed. In the Properties Panel, all of the properties for the item appear.
2. Change property values by entering text or selecting values from drop-down lists. With some properties, for example, **OutputFormat** or **Filters**, when you click the property, an ellipsis button appears to the right. Click the

ellipsis button to open a dialog where you can make changes.

Delete a control

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

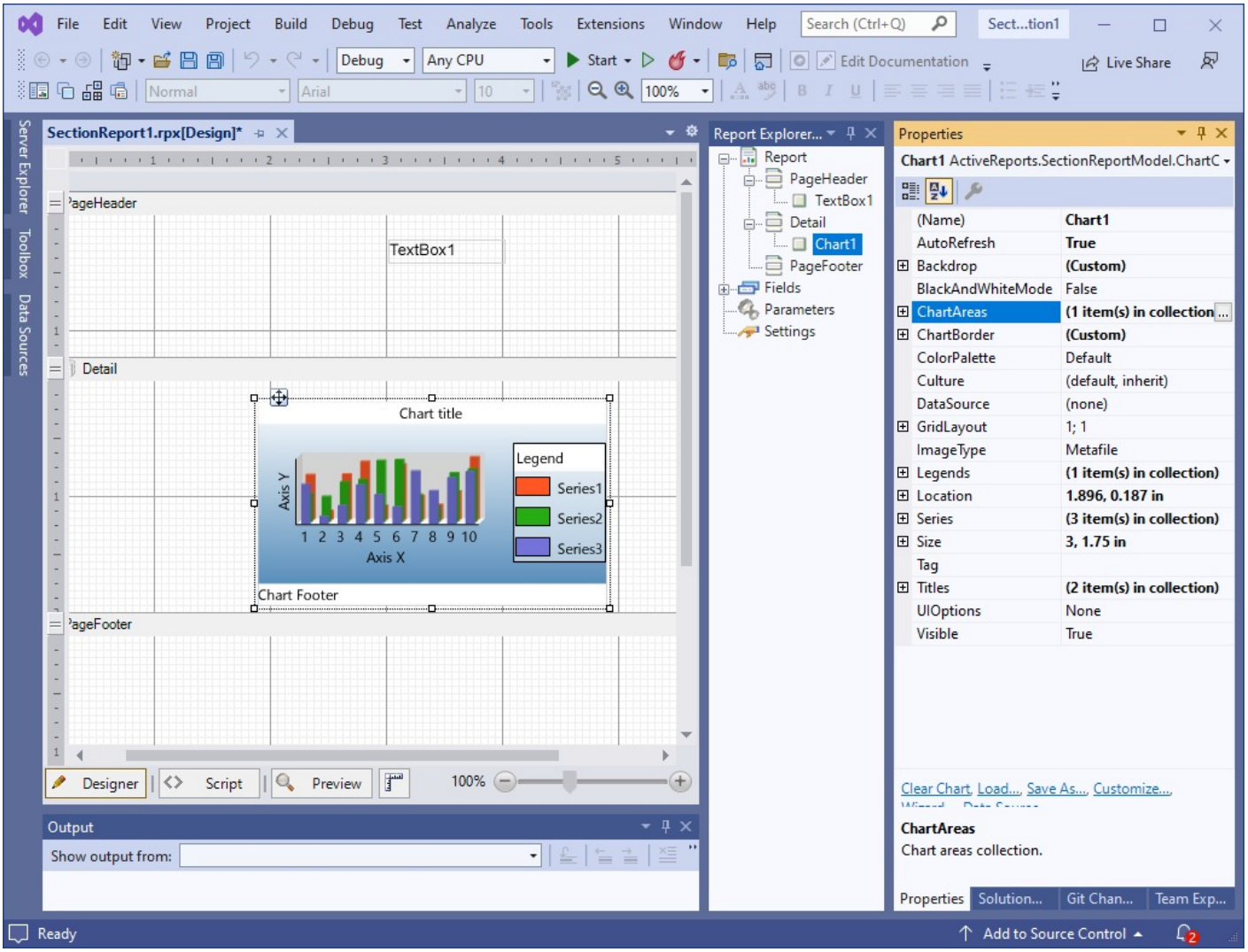
Properties Panel

The Visual Studio Properties window is an important tool when you design a report. Select any page, section, data region, control or the report itself to gain access to its properties in the Properties window. By default, this window is placed to the right of the report design area, or wherever you may have placed it in Visual Studio. You can show the list of properties by category or in alphabetical order by clicking the buttons at the top of the Properties window.

Select a property to reveal a description at the bottom of the window. Just above the description is a commands section that contains commands, links to dialogs that give you access to further properties for the item. You can resize the commands or description sections by dragging the top or bottom edges up or down.

> **Tip**: If the commands or description section is missing in Visual Studio, you can toggle it back on by right-clicking anywhere in the Properties window and clicking **Commands** or **Description**.

In the image below, you can see a chart control selected on the designer surface, revealing its properties in the Properties window, along with any associated commands, and a description of the selected property.



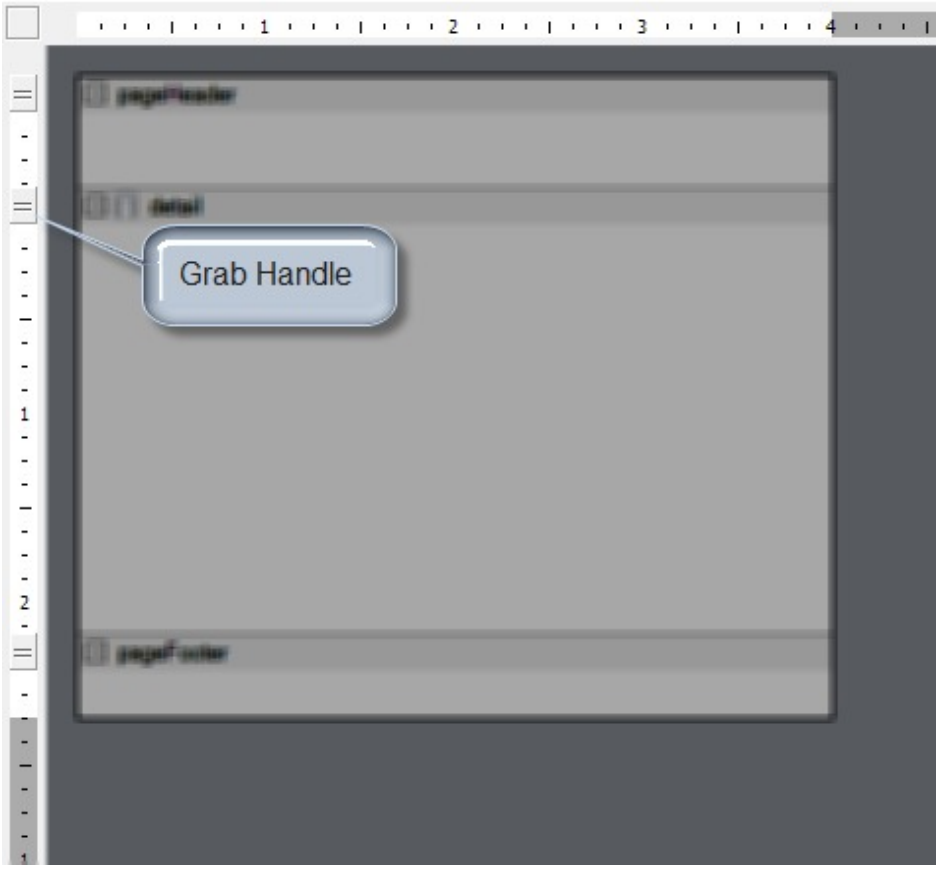
Rulers

In ActiveReports, rulers appear to the top and left of the Design View to guide you in vertically and horizontally aligning items in the report. They have large tick marks to indicate half-inch points and smaller tick marks to indicate eighths of an inch.

> **Note**: The numbers indicate the distance in inches from the left margin, not from the edge of the page.

Section Report

In Section reports, the white area on the ruler indicates the designable area of the report. The grey area at the bottom of the vertical ruler and at the right of the horizontal ruler indicates the report margins. Grab handles on the vertical ruler indicate the height of individual sections. You can drag them up or down to change section heights, or double-click to automatically resize the section to fit the controls in it.

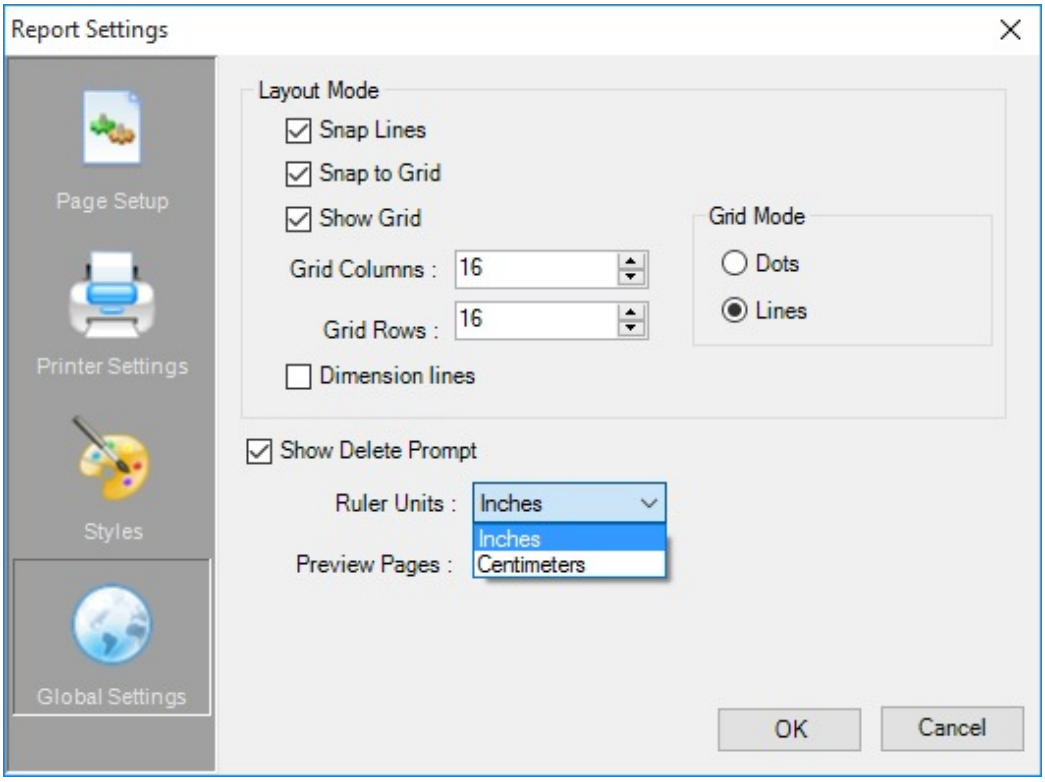


In a Section Report, you can change ruler measurements from inches to centimeters and centimeters to inches. Use the following instructions to modify ruler measurements at design- time.

To change ruler measurements at design-time in Section Report

At design time, you can change the ruler measurements from the Report Settings Dialog.

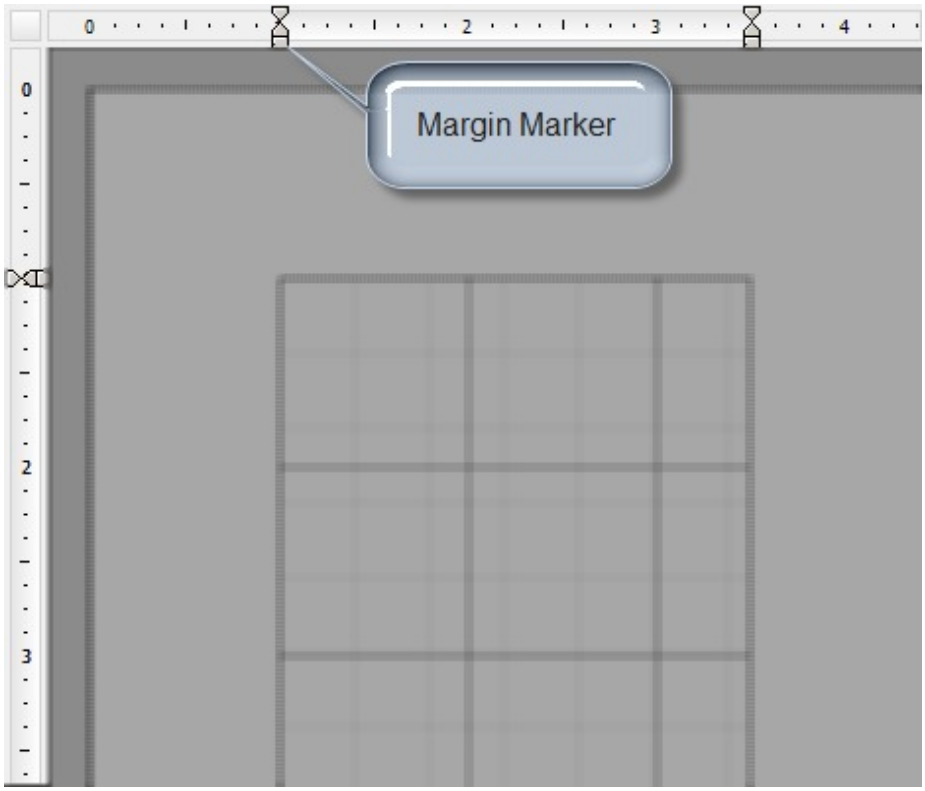
1. In the Report Explorer, double-click the **Settings** node.
2. In the Report Settings dialog that appears, click **Global Settings**.
3. From the **Ruler Units** dropdown select Centimeters or Inches.



In Section Report, you can change the units of measure for the rulers.

Page Report/RDLX Report

In Page or RDLX reports, margin markers indicate the designable area of the report. The area inside the margin markers is designable, and the areas outside the markers are the margins. To change the margins, you can drag the margin markers to the desired locations.

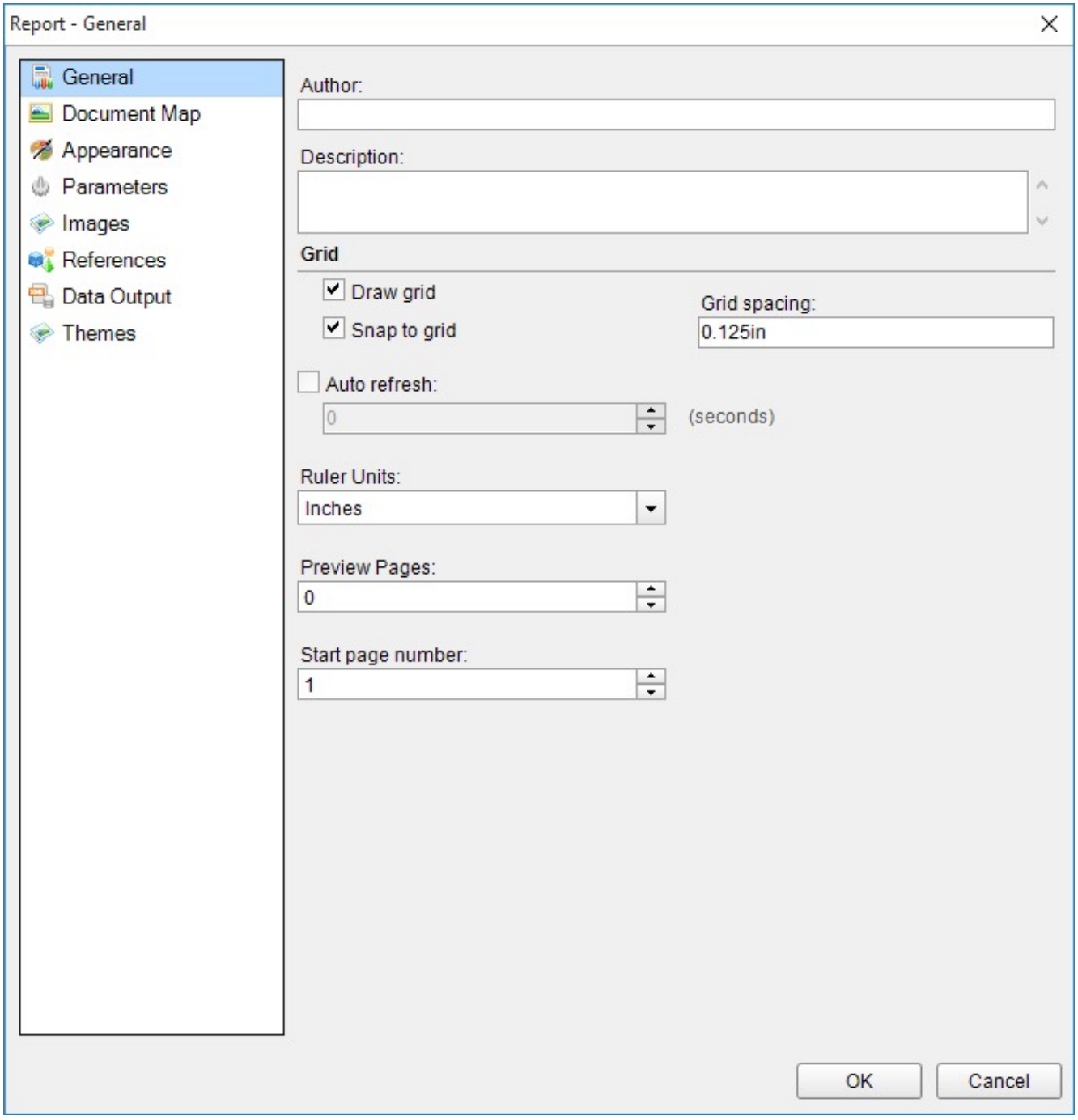


In a page layout, you can change ruler measurements from inches to centimeters and centimeters to inches. Use the following instructions to modify ruler measurements in Page reports.

To change ruler measurements at design-time in Page Report

At design time, you can change the ruler measurements from the Report Dialog.

1. In the Report Explorer, select the **Report** node.
2. In the command section of the Properties Panel, click the Property dialog.
3. In the dialog that appears, go to the **General** tab.
4. From the **Ruler Units** dropdown select Centimeters or Inches.



Scroll Bars

Scroll Bars appear automatically when controls or data regions do not fit the visible area of the report design surface. A scroll bar consists of a shaded column with a scroll arrow at each end and a scroll box (also called a thumb) between the arrows. You can scroll up, down, right or left using the scroll arrow buttons, scroll box or mouse wheel.

Auto Scrolling

When a user drags a control beyond the edge of the design surface and the mouse pointer reaches near the surface edge, scrolling starts automatically in the direction of the mouse movement. Auto scrolling works in all four directions. This feature is useful while designing reports with a magnified design view.

Note: In Section Layout and Report Definition Language (RDLX), when the mouse button is released during auto scrolling at a location outside the design surface, the surface extends to accommodate the control.

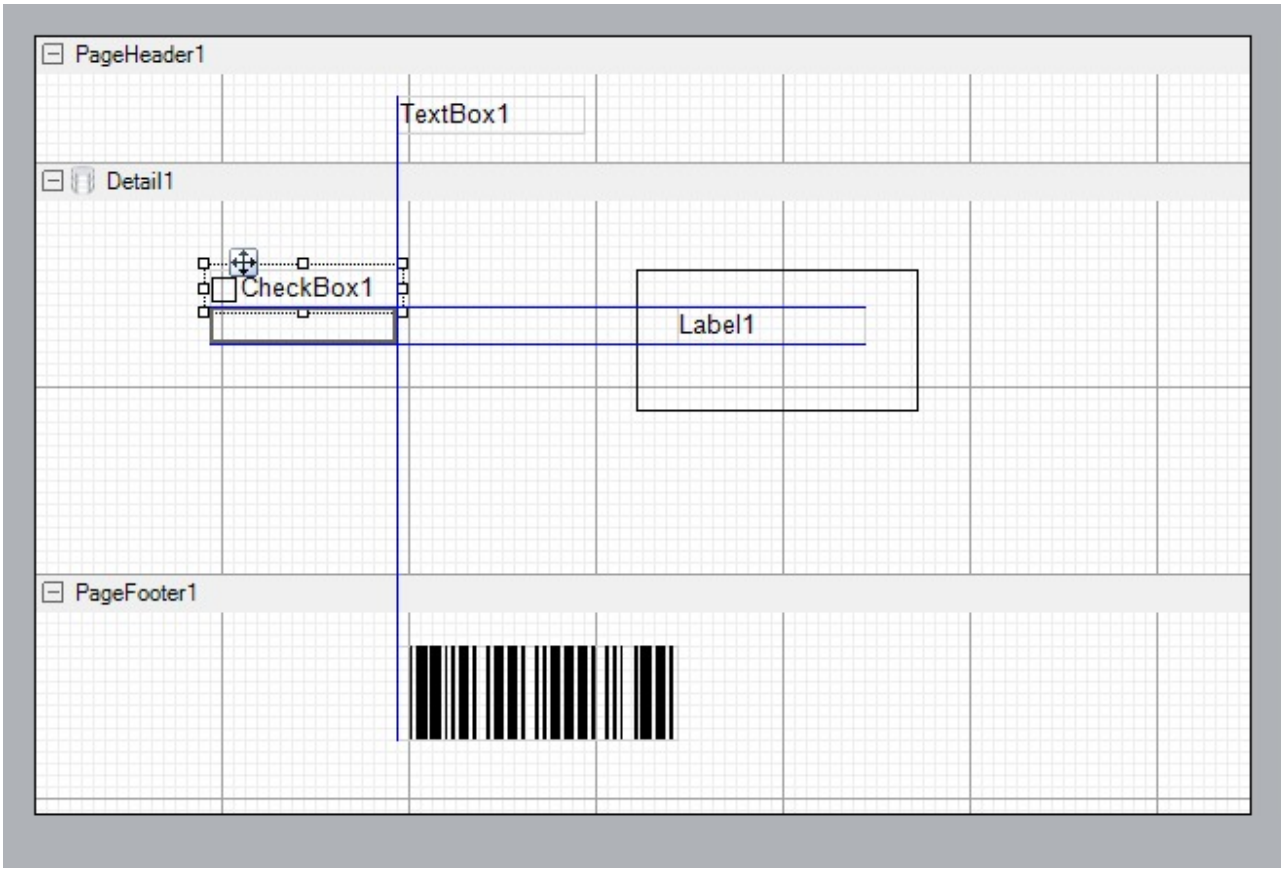
Scrolling stops in the following scenarios:

- The user stops dragging the mouse (Mouse Up).
- The user moves the mouse in the opposite direction.
- The **[Esc]** key is pressed while dragging the mouse.

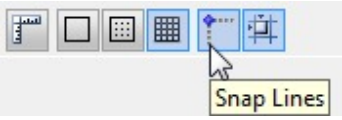
Tip: To enable auto scrolling for multiple controls, hold down the **[Ctrl]** or **[Shift]** key to select the controls. Drag them together to the edge of the design surface and enable auto scrolling.

Snap Lines

Snap lines assist in accurate positioning of elements on a report design surface while you drag report controls on it. These dynamic horizontal and vertical layout guidelines are similar to the ones found in Visual Studio. You can see snap lines on the Visual Studio Integrated Designer as well as the Standalone Designer application.



Snap lines appear on the design surface by default. In order to disable them, click the **Snap Lines** button below the design surface, or in Section reports, hold down the **[Alt]** key while dragging a control to temporarily hide the snap lines.



When you drag a control on the design surface, blue snap lines appear and the control slows down as it aligns with another control or a section edge. Unless you are also using the **Snap to Grid** setting, with **Snap Lines**, the control can move freely around the report and can be placed anywhere on the design surface.

> **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 or overlapping of controls in the spreadsheet.

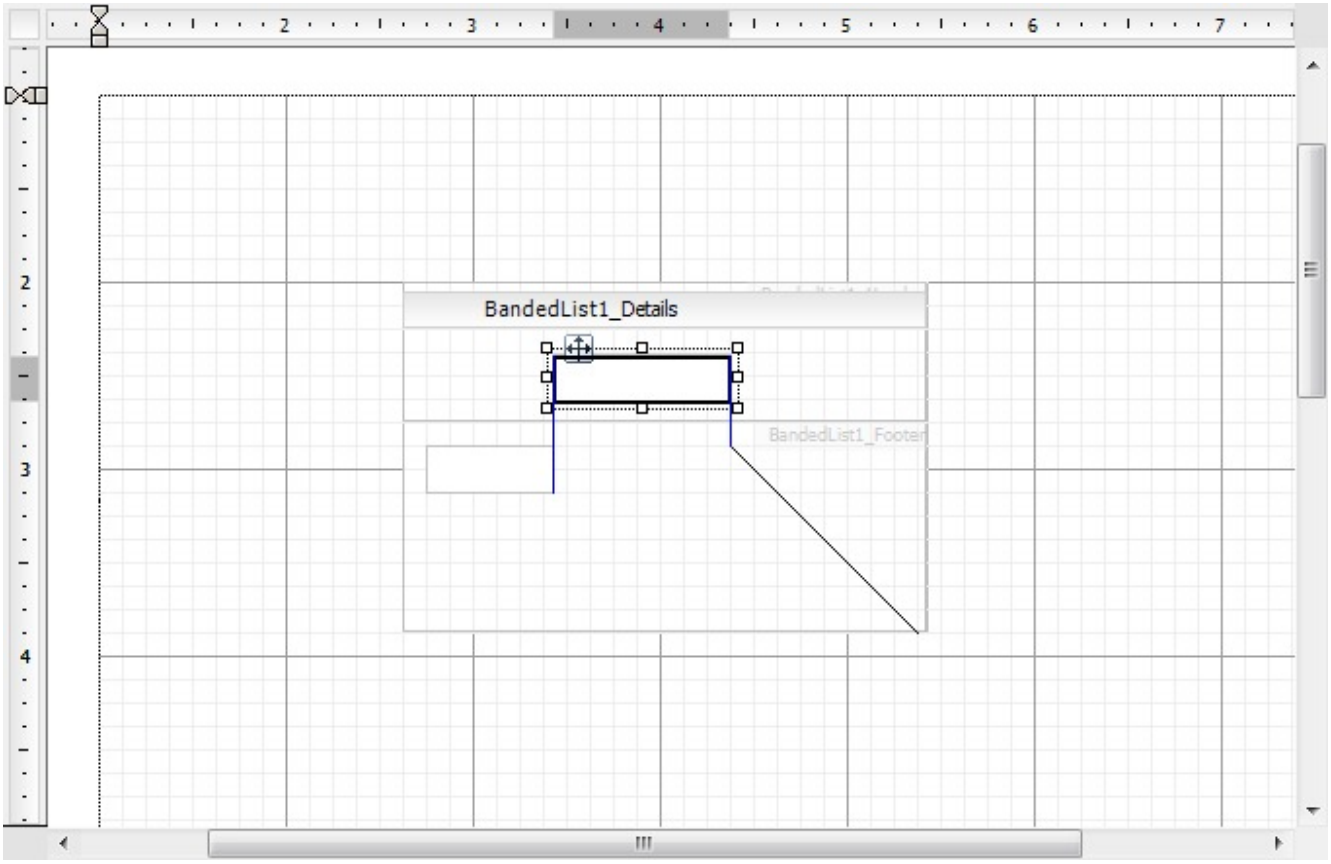
Snap Line Behavior

On dragging with a mouse

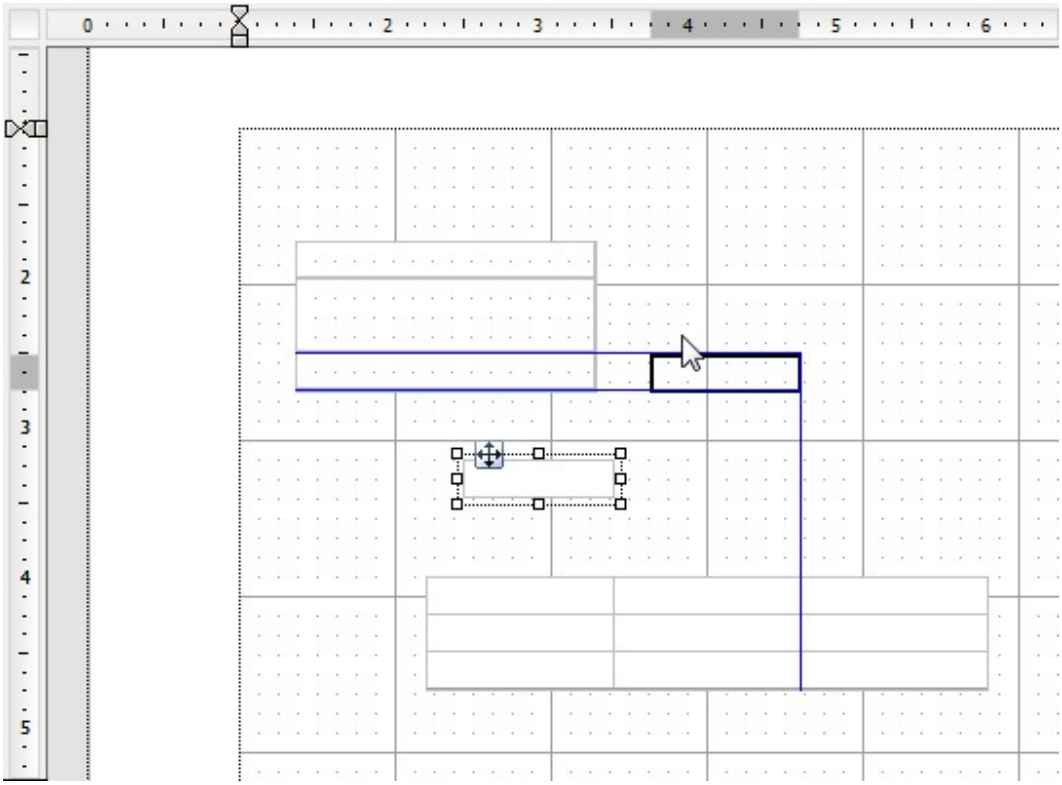
When you drag report controls across the design surface, they snap to other controls, report and section edges. Snap lines appear when the control you are dragging aligns with any edge of any of the following:

- Any control inside any section of the report.

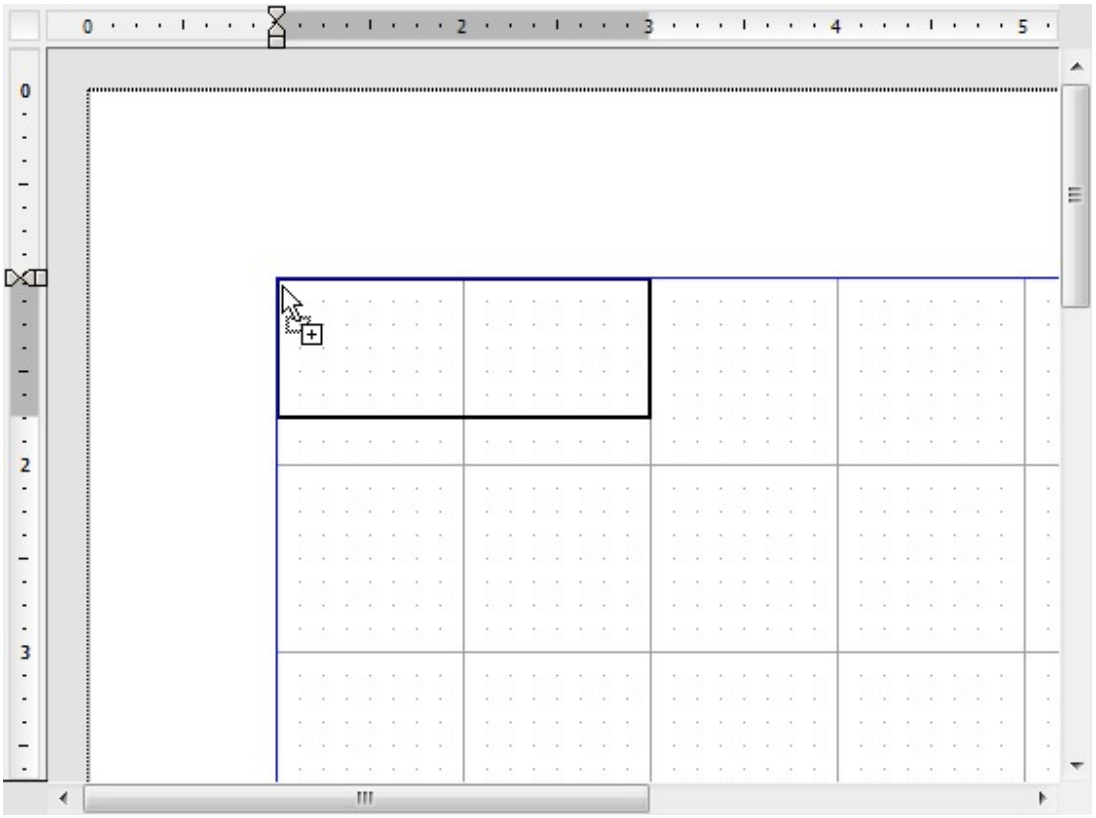
- Another control inside the same data region.



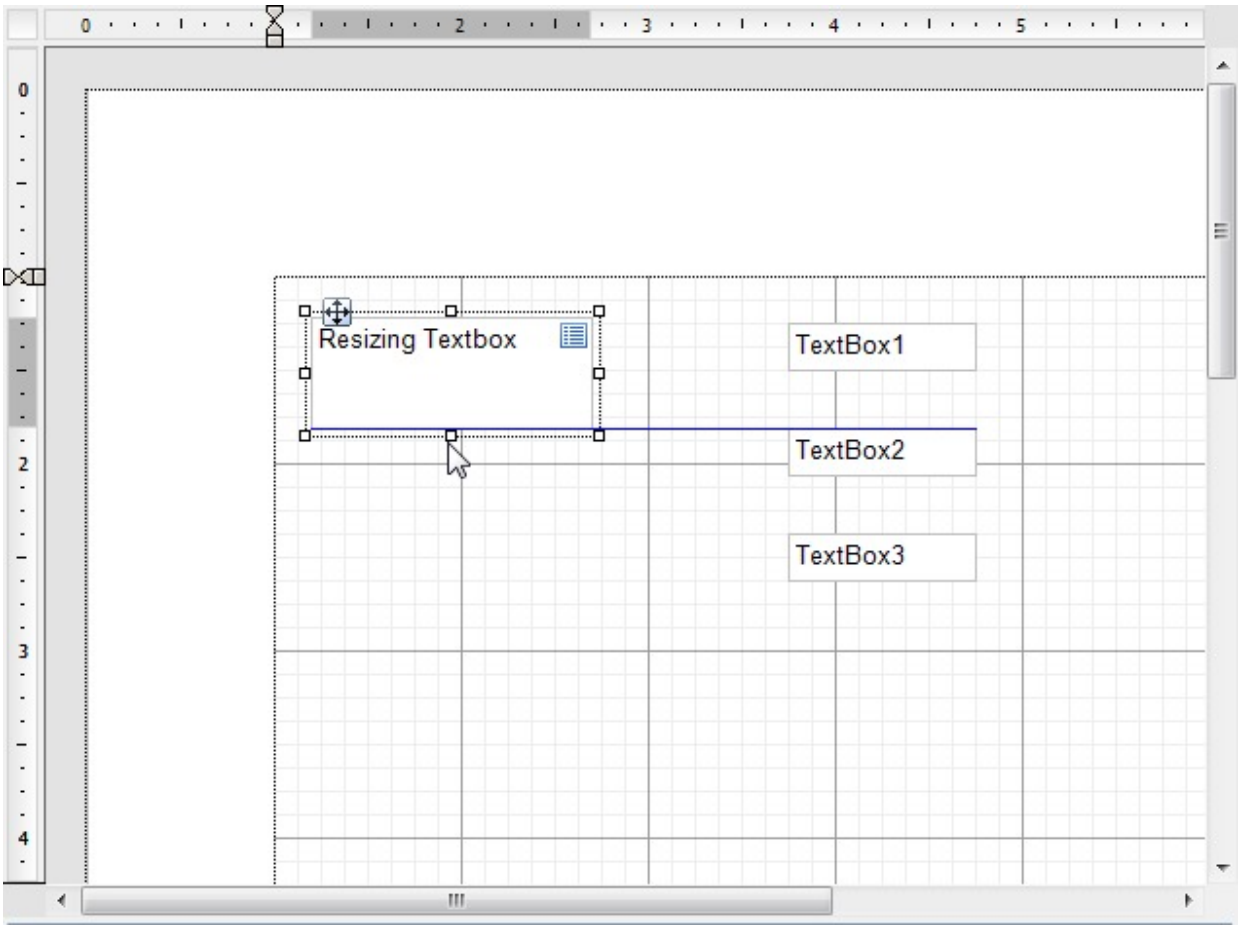
- Parts of a data region (bands in a Banded List, or columns and rows in a Table).



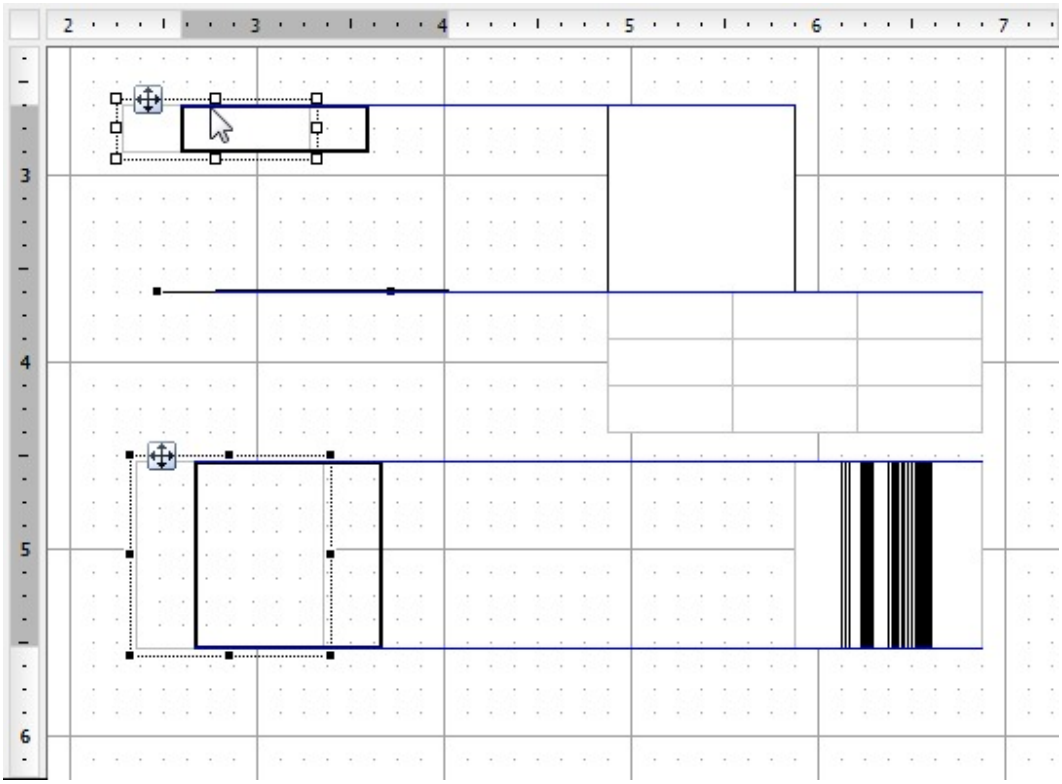
- Report edges and section edges.



- Other control edges while resizing with a mouse.



- On selecting multiple items where all the items move as a single unit, snap lines appear for all items in the selection.



With keyboard actions

- Use [Ctrl] + [Shift] + Arrow keys to resize the selected control from one snap line to the next.
- Use [Ctrl] + Arrow keys to move the selected control to the next snap line.
- Use [Ctrl] + Left mouse button to copy the control and see snap lines appear between the edges of the copied control being dragged and the original control.

Note: Snap lines do not appear when you move a control with arrow keys.

Zoom Support

ActiveReports allows you to zoom in or out on the report design surface for better control over your report layout. As you zoom in or out, the size of every item on the design surface changes.

In the designer, you can access the zoom feature from the Zoom bar below the report design surface where the slider thumb is set to 100% by default. The slider allows you to zoom in and out of the report designer surface. Using this slider you can magnify the layout from 50% to 400%. You can also use the zoom in (+) and zoom out (-) buttons at either end of the slider to change the zoom level.

Zoom settings are also available on the ActiveReports toolbar where you can change the zoom percentage or use the zoom in/zoom out buttons. See Toolbar for further information.

Keyboard Shortcuts

You can hold down the **Ctrl** key and use the mouse wheel to zoom in and zoom out of the design surface.

You can also use keyboard shortcuts for the following functions:

- **[Ctrl] + [+] :** Zoom in
- **[Ctrl] + [-] :** Zoom out
- **[Ctrl] + 0 :** Return to 100%

Page/RDLX Report

Page/RDLX reports is the ActiveReports extensions of the MS RDLX specification. Developers should clearly understand the behavior of these reports because it can conflict with other requirements.

- Imagine an RDLX report as an infinite canvas (as well as a multi-section report as many canvases tied together) that can grow in two directions, depending on data. The pagination in this type of report works according to predefined RDLX rules - a page break before/after, orphaned headers/footers and other.
- A Page report is a set of predefined pages with static and dynamic data content. If data doesn't fit to a page, the entire page (static content) is cloned and filled with new dynamic data content. A Page report has a special control - **OverflowPlaceHolder**, to keep place to dynamic data.

From the developer's perspective, such difference makes sense only in case of requirements to exports or master reports because:

- Word DOCX export works well only with RDLX reports.
- Excel export in the single page mode works well only with RDLX reports.
- Excel Data export is similar to the RDLX report template.
- Master reports support only RDLX reports.
- Galley (not paginated) mode is available in RDLX reports only.
- Only RDLX reports support header/footer. Per page static content (not only on header/footer) can be implemented in Page reports.

Create a Report or Load an Existing Report

This topic discusses about creating and loading a Page or an RDLX report in designer using code. We recommend you to create a new report directly from the standalone designer or you using the built-in project templates from Visual Studio.

Create a Report

Use **NewReport() ('NewReport Method' in the on-line documentation)** method in the **Designer ('Designer Class' in the on-line documentation)** class to initialize the designer with a new report layout, with specified type. Make sure to add **MESCIUS.ActiveReports.Design.Win** package to your project.

C#. Add using statements on the top of Form.cs

```
using System.Windows.Forms;
using GrapeCity.ActiveReports.Design;
```

C# code. Paste INSIDE the Form Load event.

```
var _designer = new Designer() { Dock = DockStyle.Fill };
_designer.NewReport(DesignerReportType.Section);
Controls.Add(_designer);
```

For details on using additional Designer class methods, properties, and events, refer to this page.

Load an Existing Report

Use the **DefaultResourceLocator ('DefaultResourceLocator Class' in the on-line documentation)** class especially in case when you are dealing with master reports (RDLX reports). You can create a new report with the master report and load it in the designer as shown. Make sure to add **MESCIUS.ActiveReports.Core.Rdl** package to your project.

C#. Add using statements on the top of Form.cs

```
using System.Windows.Forms;
using GrapeCity.ActiveReports.Design;
using GrapeCity.ActiveReports;
```

C# code. Paste INSIDE the Form Load event.

```
var _designer = new Designer() { Dock = DockStyle.Fill };
_designer.ResourceLocator = new DefaultResourceLocator(new Uri(path to master));
_designer.LoadReport(new FileInfo(path to report));
```

Bind a Page/RDLX Report to Data

The following topics provide information on data binding scenarios with Page/RDLX reports:

Bind Page/RDLX Report to Entity Framework and Other Data Providers

Bind Page/RDLX Report to SQL, OLEDB, and ODBC Data Providers

Bind Page/RDLX Report to Custom Data Providers (Oracle, PostgreSQL, SQLite)

Bind Page/RDLX Report to CSV, JSON and XML Data

Bind Page/RDLX Report to Data at Runtime

Use Dynamic Connection String in Data Source

Bind Page/RDLX Report to Entity Framework and Other Data Providers

You can bind your Page/RDLX reports to a data provider like Entity Framework, NHibernate, etc.

To connect to the Entity Framework data source in ActiveRepors, follow these basic steps:

1. Initialize a selected data provider.
2. Create a data layer that uses the selected data provider.
3. Bind the data that you want to link to the report, using the data binding mechanism with objects. For more information, see the **Object Provider** section below.

Below is the example that illustrates binding to the Entity Framework in details. The example assumes that you have a database with the **Product** entity, which has **ProductName** and **ProductID** fields. A tuple of data retrieved from the database needs to be mapped to a table in a report.

Note: The sample report has already pre-configured fields in the table to match the fields from the entities.

1. Initialize Entity Framework as a data provider.

```
// Your binding model
private class Product
{
    // Key
    public int ProductID { get; set; }
    // Fields
    public string ProductName { get; set; }
}

// Your database context
private class ProductsDatabaseContext : DbContext
{
    public ProductsDatabaseContext(string connectionstring) : base(connectionstring) { }
    public DbSet<Product> Products { get; set; }
}
```

2. Create a data layer.

```
// Your data layer
private class ProductDataLayer
{
    private ProductsDatabaseContext _context;
    internal ProductDataLayer()
    {
        _context = CreateDataProvider();
    }
    private ProductsDatabaseContext CreateDataProvider()
    {
        // Your connection string to database
        string connectionString = "";
        return new ProductsDatabaseContext(connectionString);
    }
    internal IEnumerable<Product> GetProducts() => _context?.Products;
}
```

3. Bind data by the data binding mechanism that uses objects.

```
private PageReport LoadReport()
{
    // Your path to the report
    string path = "";

    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateDataSource += OnLocateDataSource;

    return report;
```

```
}
private void OnLocateDataSource(object sender, LocateDataSourceEventArgs args)
{
    ProductDataLayer dataLayer = new ProductDataLayer();
    args.Data = dataLayer.GetProducts();
}
```

This approach allows you to work with any data provider and gives you the flexibility to customize your applications when working with reports.

DataSet Provider

To bind your Page/RDLX report to a DataSet, you must subscribe to the event that occurs when a report needs to locate a data source from the calling application:

```
report.Document.LocateDataSource += new LocateDataSourceEventHandler(OnLocateDataSource);
```

To connect to an unbound data source at run time, you can use the DataSet provider with the **LocateDataSource** event. The reporting engine raises the **LocateDataSource** event when it needs input on the data to use.

```
private PageReport LoadReport()
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateDataSource += new LocateDataSourceEventHandler(OnLocateDataSource);

    return report;
}
private void OnLocateDataSource(object sender, LocateDataSourceEventArgs args)
{
    DataTable dataTable = new DataTable("Example");

    // Add the desired data to the dataset
    //
    // dataTable.Columns.Add(...);
    // dataTable.Rows.Add(...);
    //

    args.Data = dataTable;
}
```

Note: We assume that the report has defined data fields and datasets, which can be done by various ways, e.g., in the Web or Windows Forms Designer.

The DataSet provider has the following limitations:

- Relationship names with periods are not supported.

- Fields in nested relationships only traverse parent relationships (e.g. FK_Order_Details_Orders.FK_Orders_Customers.CompanyName).

To request a field from a parent table, prefix the field name with the name of the relation(s) that must be traversed to navigate to the appropriate parent table. Separate field names and relations with periods.

For example, consider a main table named **OrderDetails** with a parent table named **Orders**. A relation named **Orders_OrderDetails** defines the relationship between the two tables. Use a field with the syntax below to access the OrderDate from the parent table: **Orders_OrderDetails.OrderDate**.

Use this same technique to traverse multiple levels of table relations. For example, consider that the **Orders** table, used in the example above, has a parent table named **Customers** and a relation, binding the two, called **Customers_Orders**. If the CommandText specifies the main table as OrderDetails, use the following syntax to get the CustomerName field from the parent table: **Customers_Orders.Orders_OrderDetails.CustomerName**.

> **Note**: An ambiguity can occur if a field and a relation have the same name. This is not supported.

Object Provider

Use the API to bind a Page/RDLX report to the Object data source.

In the example below, the report calls the **LocateDataSource** event, used to set data into the report. To do this, we create the **OnLocateDataSource** event handler and pass in it the data that will be transmitted to the report. The data is added as a new data source in the **LocateDataSource** event handler.

```
private PageReport LoadReport()
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateDataSource += new LocateDataSourceEventHandler(OnLocateDataSource);
    return report;
}
private void OnLocateDataSource(object sender, LocateDataSourceEventArgs args)
{
    ArrayList data = new ArrayList();

    // Add the desired data to the collection
    //
    // data.Add(...);
    //

    args.Data = data;
}
```

> **Note**: We assume that the report has defined data fields and datasets, which can be done by various ways, e.g., in the Web or Windows Forms Designer.

Bind Page/RDLX Report to SQL, OLEDB, and ODBC Data Providers

Using a database as a data source is one of the most common scenarios when you work with reports. This topic explains how to bind your ActiveReports report to a SQL, OLEDB or ODBC data source.

The example below uses a SQL database as a data source.

```
private PageReport LoadReport()
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateCredentials += OnLocateCredentials;

    string dataProviderName = "SQL"; // Or use "OLEDB" or "ODBC" if you need it

    DataSource dataSource = report.Report.DataSources[0]; // Your data source
    if(dataSource.ConnectionProperties.DataProvider == dataProviderName)
    {
        dataSource.ConnectionProperties.ConnectString =
ConfigurationManager.ConnectionStrings["YourConnectionString"].ConnectionString;
    }

    return report;
}
private void OnLocateCredentials(object sender, LocateCredentialsEventArgs args)
{
    args.Password = ""; // Your password if you need it
    args.UserName = ""; // Your username if you need it
}
```

Use the following strings to connect to different data providers:

- **SQL** - Use this if you connect using the MS SQL Server. An example of a connection string:

  ```
  Data Source=[sourceName];Initial Catalog=[dbName];Integrated Security=True;Connect
  Timeout=30;Encrypt=False;
  ```

- **OLEDB** - Use this if you connect using the OLEDB driver. An example of a connection string:

  ```
  Provider=Microsoft.Jet.OLEDB.4.0;Data Source=[dbname];
  ```

- **ODBC** - se this if you connect using the ODBC driver. An example of a connection string:

  ```
  DRIVER=SQLite3 ODBC Driver;Database=[dbname]; LongNames=0; Timeout=1000; NoTXN=0;
  SyncPragma=NORMAL; StepAPI=0;
  ```

> **Note:** When you are using an ODBC/OLEDB data provider, make sure that the required driver is installed on your computer. Otherwise, you cannot use this data provider.

When working with a data provider, you may need to process user credentials to work with the target source. The **LocateCredentials** event is used for this purpose.

```
report.Document.LocateCredentials += OnLocateCredentials;
```

Assign a handler that specifies the password and username when they are requested.

```
private void OnLocateCredentials(object sender, LocateCredentialsEventArgs args)
{
    args.Password = ""; // Your password if you need it
    args.UserName = ""; // Your username if you need it
}
```

The event handler receives an argument of the **LocateCredentialsEventArgs** type that contains data, related to this event. The following **LocateCredentialsEventArgs** properties provide information, specific to this event.

- **Password** - Gets ot sets the password of the credentials to be located.
- **PromptText** - Gets ot sets the password of the credentials to be located.
- **ReportPath** - Gets the text of prompt of the locate credentials request.
- **UserName** - Gets ot sets the user name of the credentials to be located.

> **Note**: We assume that the report has defined data fields and datasets, which can be done in various ways, e.g., in the Web or Windows Forms Designer.

Bind Page/RDLX Report to Custom Data Providers (Oracle, PostgreSQL, SQLite)

> **Note:** When you bind a report to any custom data provider, you need to modify the ActiveReports configuration file. See Configure ActiveReports for more information.

This topic explains how to connect to the following data providers:

- **Oracle** – Oracle Database RDBMS. This is not a built-in data source and requires an external provider. This data source has a built-in support to locate credentials.
- **PostgreSQL** – PostgreSQL RDBMS. This is not a built-in data source and requires an external provider. This data source has a built-in support to locate credentials.
- **SQLite** - compact embedded RDBMS. This is not a built-in data source and requires an external provider.

When working with a custom data provider, you may need to process user credentials to work with the target source. The **LocateCredentials** event is used for this purpose.

The event handler receives an argument of the **LocateCredentialsEventArgs** type, containing related data. The following **LocateCredentialsEventArgs** properties provide information, specific to this event.

- **Password** - Gets ot sets the password of the credentials to be located.
- **PromptText** - Gets ot sets the password of the credentials to be located.
- **ReportPath** - Gets the text of prompt of the locate credentials request.
- **UserName** - Gets ot sets the user name of the credentials to be located.

Oracle Database

The Oracle database is not a built-in data source and requires an external provider. However, this data source has a built-in support to locate credentials.

The example below shows what code you should add to the configuration file.

```
<ReportingConfiguration>
    <Extensions>
        <Data>
             <Extension Name="ORACLE"
Type="Oracle.ManagedDataAccess.Client.OracleClientFactory, Oracle.ManagedDataAccess"
DisplayName="Oracle Provider" />
        </Data>
    </Extensions>
</ReportingConfiguration>
```

Note: To use the Oracle data, you must install the Oracle.ManagedDataAccess or Oracle.ManagedDataAccess.Core NuGet package.

Add the following code to connect to the Oracle Database as a new data source.

```
private PageReport LoadReport()
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateCredentials += OnLocateCredentials;
    DataSource dataSource = report.Report.DataSources[0]; // Your data source
    if(dataSource.ConnectionProperties.DataProvider == "ORACLE")
    {
        dataSource.ConnectionProperties.ConnectString = ""; // Your connection string
    }

    return report;
}
private void OnLocateCredentials(object sender, LocateCredentialsEventArgs args)
{
    args.Password = ""; // Your password if you need it
    args.UserName = ""; // Your username if you need it
}
```

Note: We assume that the report has defined data fields and datasets, which can be done in various ways, e.g., in the Web or Windows Forms Designer.

See an example of a connection string below.

```
Data Source=[database];User Id=[username];Password=[password];Integrated Security=no;
```

To handle user credentials, you should subscribe to the **LocateCredentials** event as follows.

```
report.Document.LocateCredentials += OnLocateCredentials;
```

You should also add a handler to manage the data.

```
private void OnLocateCredentials(object sender, LocateCredentialsEventArgs args)
{
    args.Password = ""; // Your password if you need it
    args.UserName = ""; // Your username if you need it
}
```

PostgreSQL Database

The PostgreSQL database is not a built-in data source and requires an external provider. However, this data source has a built-in support to locate credentials.

The example below shows what code you should add to the configuration file.

```
<ReportingConfiguration>
    <Extensions>
        <Data>
            <Extension Name="POSTGRESQL" Type="Npgsql.NpgsqlFactory, Npgsql"
DisplayName="PostgreSQL Provider" />
        </Data>
    </Extensions>
</ReportingConfiguration>
```

Note: You must install the Npgsql NuGet package to use the PostgreSQL data.

Add the following code to connect PostgreSQL as a new data source.

```
private PageReport LoadReport()
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateCredentials += OnLocateCredentials;

    DataSource dataSource = report.Report.DataSources[0]; // Your data source
    if(dataSource.ConnectionProperties.DataProvider == "POSTGRESQL")
    {
        dataSource.ConnectionProperties.ConnectString = ""; // Your connection string
    }
    return report;
}
```

```
private void OnLocateCredentials(object sender, LocateCredentialsEventArgs args)
{
    args.Password = ""; // Your password if you need it
    args.UserName = ""; // Your username if you need it
}
```

Note: We assume that the report has defined data fields and datasets, which can be done in various ways, e.g., in the Web or Windows Forms Designer.

See an example of a connection string below.

```
Uid=[username];Pwd=[password];Host=[host];Port=[port];Database=[dbName];
```

The code example illustrates that you are adding a PostgtreSQL data provider as a custom data provider. Subscribing to the **LocateCredentials** event allows you to handle user credentials.

SQLite or MS SQLite Database

The example below shows what code you should add to the configuration file.

```
<ReportingConfiguration>
    <Extensions>
        <Data>
            <Extension Name="SQLITE" Type="System.Data.SQLite.SQLiteFactory,
System.Data.SQLite" DisplayName="SQLite Provider" />
        </Data>
    </Extensions>
</ReportingConfiguration>
```

Note: System.Data.SQLite or System.Data.SQLite.Core NuGet package must be installed for using Sqlite data.

You can also use MS SQLite. Add the following to the configuration file.

```
<ReportingConfiguration>
    <Extensions>
        <Data>
            <Extension Name="MSSQLITE" Type="Microsoft.Data.Sqlite.SqliteFactory,
Microsoft.Data.Sqlite” DisplayName="MS SQLite Provider" />
        </Data>
    </Extensions>
</ReportingConfiguration>
```

Note: You must install the Microsoft.Data.Sqlite or Microsoft.Data.Sqlite.Core NuGet packages for using the MS SQLite data.

Add the following code to connect to SQLite database as a new data source.

```
private PageReport LoadReport()
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));

    DataSource dataSource = report.Report.DataSources[0]; // Your data source
    if(dataSource.ConnectionProperties.DataProvider == "SQLITE") // Or "MSSQLITE" if you use
it
    {
        dataSource.ConnectionProperties.ConnectString = ""; // Your connection string
    }
    return report;
}
```

See an example of a connection string below.

```
Data Source=[dbName];Version=3;
```

Note: We assume that the report has defined data fields and datasets, which can be done in various ways, e.g., in the Web or Windows Forms Designer.

Bind Page/RDLX Report to CSV, JSON and XML Data

You can use common data formats such as CSV, JSON, XML to bind your Page/RDLX report to data. You can use a provider with the **LocateDataSource** event to connect to unbound data sources at runtime. The reporting engine calls the **LocateDataSource** event when it needs data. To do this, you need to add a handler for this event and set the data for the report in it.

When using the **LocateDataSource** event as a data binding method, the general approach is as follows:

```
string path = ""; // Your path to the report
PageReport report = new PageReport(new FileInfo(path));
report.Document.LocateDataSource += (sender, args) => { // Data binding  };
```

The event handler receives an argument of the **LocateDataSourceEventArgs** type with data, related to this event. The following **LocateDataSourceEventArgs** properties provide information, specific to this event.

- **Data** - The data returned by the event handler.
- **DataSet** - Gets the report's GrapeCity.ActiveReports.PageReportModel.IDataSet object to locate data for.
- **Parameters** - Gets the Parameters collection specified for a given report instance.
- **Report** - Gets the Report that is trying to locate the data set.

CSV Data Source

To connect to the CSV data source, use the following code:

```
private PageReport LoadReport()
```

```
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateDataSource += new LocateDataSourceEventHandler(OnLocateDataSource);

    return report;
}
private void OnLocateDataSource(object sender, LocateDataSourceEventArgs args)
{
    string data = ""; // Your csv string
    args.Data = data;
}
```

Note: We assume that the report has defined data fields and datasets, which can be done in various ways, e.g., in the Web or Windows Forms Designer.

To interact with the CSV data source, you can specify a connection string that includes the following parameters:

- Specify the path to the CSV file
- Specify the encoding
- Specify the rows separator
- Specify the columns separator
- Specify the text qualifier
- Specify the columns

You can also use the **LocateDataSource** event to add new data. Data is added as a new data source in the **OnLocateDataSource** event handler. To control this, you previously subscribed to this event. The reporting engine raises the **LocateDataSource** event when it needs data.

The **LocateDataSource** event handler sets the data in the required format as you can see in the sample below.

```
private void OnLocateDataSource(object sender, LocateDataSourceEventArgs args)
{
    string data = ""; // Your csv string
    args.Data = data;
}
```

JSON Data Source

To demonstrate how to bind to JSON data, let's pass a string in the target format as a data source. To do this, you need to subscribe to the **LocateDataSource** event.

```
private void OnLocateDataSource(object sender, LocateDataSourceEventArgs args)
{
    string data = ""; // Your json  string
    args.Data = data;
}
private PageReport LoadReport()
```

```
{
    string path = ""; // Your path to the report
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateDataSource += OnLocateDataSource;

    return report;
}
```

Note: We assume that the report has defined data fields and datasets, which can be done in various ways, e.g., in the Web or Windows Forms Designer.

Data in the JSON format is assigned when the application calls the **LocateDataSource** event. Data is added as a new data source in the **OnLocateDataSource** event handler. To control this, you previously subscribed to this event. The reporting engine raises the **LocateDataSource** event when it needs data. To do this, we create the **OnLocateDataSource** event handler.

XML Data Source

The example below shows how to bind a report to the XML data source by subscribing to the **LocateDataSource** event and assigning a handler to it.

```
private void OnLocateDataSource(object sender, LocateDataSourceEventArgs args)
{
    XmlTextReader xmlReader = new XmlTextReader(""); // Your path to the xml file
    args.Data = xmlReader;
}
private PageReport LoadReport()
{
    string path = ""; // Your path to the report file
    PageReport report = new PageReport(new FileInfo(path));
    report.Document.LocateDataSource += OnLocateDataSource;

    DataSource dataSource = report.Report.DataSources[0]; // Your data source
    if(dataSource.ConnectionProperties.DataProvider == "XML")
    {
        dataSource.ConnectionProperties.ConnectString = "xmldoc="; // Your connection string
    }

    return report;
}
```

In the event handler, an XML file is assigned as a data source by using one of the allowed binding methods. For an XML file, these methods include XmlReader, XmlDocument, IXPathNavigable.

The reporting mechanism calls the **LocateDataSource** event, which is used to set the data into the report. To do this, you create the **OnLocateDataSource** event handler and pass in it the data that will be transmitted to the report.

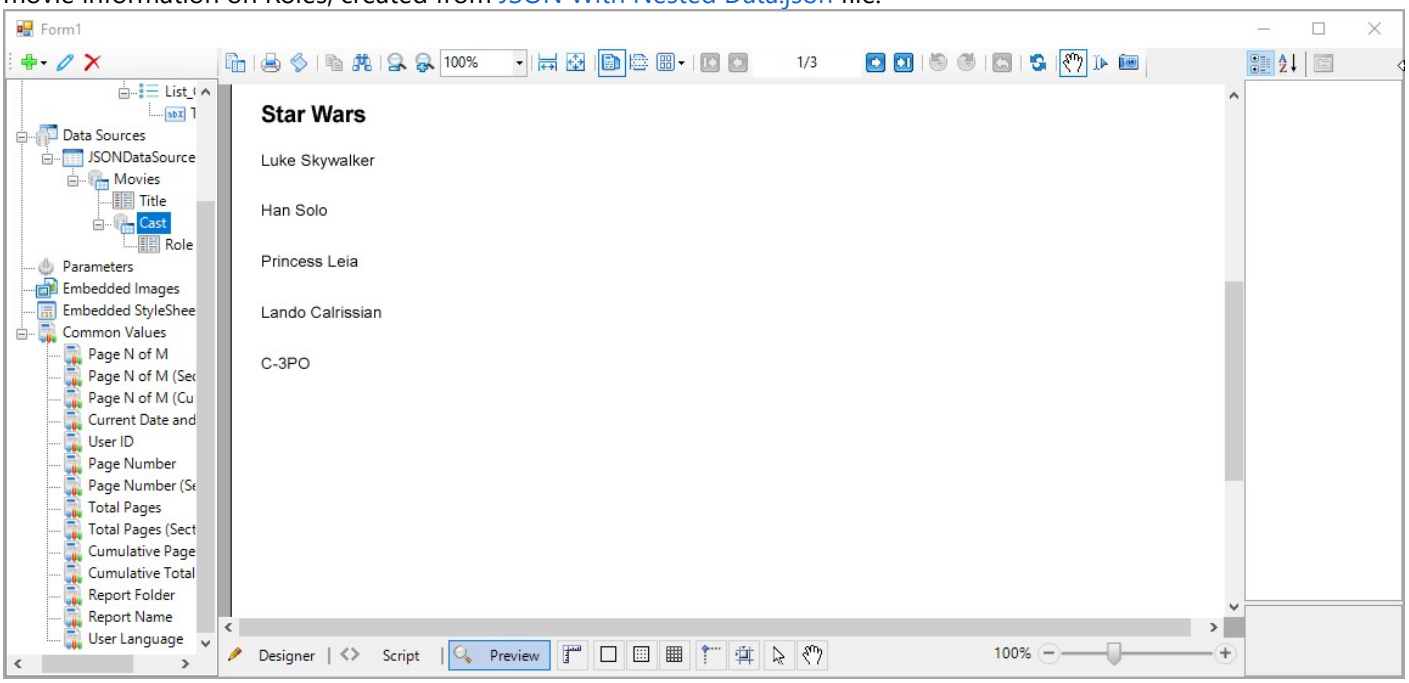
Note: We assume that the report has defined data fields and datasets, which can be done in various ways, e.g., in

> the Web or Windows Forms Designer.

Nested Datasets

A nested dataset represents JSON or XML data as an hierarchical structure. A nested JSON or XML dataset is most commonly used in a bound data region, where you can partially use the dataset nodes for different data regions of your report.

Follow these steps to create a report with a JSON nested dataset that contains data on Movies and the movie information on Roles, created from JSON With Nested Data.json file.



1. In Visual Studio 2022, create a new Windows Forms project.
2. Right-click the project in the Solution Explorer and select **Manage NuGet Packages for Solution**.
3. On the **Browse** tab of the opened **NuGet - Solution** window, search and install the **MESCIUS.ActiveReports** and **MESCIUS.ActiveReports.Design.Win** NuGet packages.
4. On the Form.cs, double-click the title bar to create the Form1_Load event.
5. At the top of the Form1 code view, add these using directives.

   Paste at the top of the Form1 code view

   ```
   using GrapeCity.ActiveReports.Design.ReportExplorer;
   using GrapeCity.ActiveReports.Design;
   using GrapeCity.ActiveReports.PageReportModel;
   using GrapeCity.ActiveReports;
   using System.Xml;
   using TextBox = GrapeCity.ActiveReports.PageReportModel.TextBox;
   using DataSet = GrapeCity.ActiveReports.PageReportModel.DataSet;
   using DataSource = GrapeCity.ActiveReports.PageReportModel.DataSource;
   using Field = GrapeCity.ActiveReports.PageReportModel.Field;
   ```

6. In the existing code, add the following code.

   Example Title

```
namespace NestedDataSetSample
{
    public partial class Form1 : Form
    {
        Designer designer;
        PropertyGrid propertyGrid;
        ReportExplorer reportExplorer;
        PageReport report;
        ReportSection section;
        public Form1()
        {
            InitializeComponent();
        }
```

7. Add the following code to the Form1_Load event.

C# code. Paste INSIDE the Form1_Load event.

```
{
    CreateLayout();
    CreateReport();
    CreateDataRegions();
    CreateDataSource();
    designer.LoadReport(XmlReader.Create(new StringReader(report.ToRdlString())),
DesignerReportType.Page);
}
```

8. In the existing code, add the following code to create the Designer, PropertyGrid and ReportExplorer objects.

C# code. Paste AFTER the Form1_Load event.

```
private void CreateLayout()
{
    designer = new Designer() { Dock = DockStyle.Fill };
    propertyGrid = new PropertyGrid() { Dock = DockStyle.Right };
    designer.PropertyGrid = propertyGrid;
    reportExplorer = new ReportExplorer()
    {
        Dock = DockStyle.Left,
        ReportDesigner = designer
    };
    Controls.Add(designer);
    Controls.Add(propertyGrid);
    Controls.Add(reportExplorer);
}
```

9. Add the following code to create the PageReport and ReportSection objects.

C# code. Paste AFTER CreateLayout()

```
private void CreateReport()
{
     report = new PageReport();
     section = new ReportSection();
```

```
        section.Name = "Section1";
        section.Width = "15cm";
        section.Body.Height = "20cm";
        report.Report.ReportSections.Add(section);
    }
```

10. Add the following code to create data regions that will contain report data.

C# code. Paste AFTER CreateReport()

```
private void CreateDataRegions()
        {
            List listMovies = new List()
            {
                Name = "List_Movies",
                Top = "0.8cm",
                Left = "0.6cm",
                Width = "11cm",
                Height = "5cm",
                Style = new Style() { FontFamily = "Arial" },
                DataSetName = "Movies",
                ZIndex = 1
            };
            listMovies.ReportItems.Add(new TextBox()
            {
                Name = "TextBox_Title",
                Top = "0.6cm",
                Left = "1cm",
                Width = "2.5cm",
                Height = "0.75cm",
                Style = new Style()
                {
                    FontFamily = "Arial",
                    PaddingBottom = "2pt",
                    PaddingLeft = "2pt",
                    PaddingRight = "2pt",
                    PaddingTop = "2pt"
                },
                Value = "=Fields!Title.Value",
                DataElementName = "Title",
                ZIndex = 1
            });
            List listCast = new List()
            {
                Name = "List_Cast",
                Top = "1.8cm",
                Left = "1.4cm",
                Width = "5cm",
                Height = "2.2cm",
                Style = new Style() { FontFamily = "Arial" },
```

```
            ZIndex = 2,
            DataSetName = "Cast"
        };
        listCast.ReportItems.Add(new TextBox()
        {
            Name = "TextBox_Name",
            Top = "0.6cm",
            Left = "0.6cm",
            Width = "2.5cm",
            Height = "0.75cm",
            Style = new Style()
            {
                FontFamily = "Arial",
                PaddingBottom = "2pt",
                PaddingLeft = "2pt",
                PaddingRight = "2pt",
                PaddingTop = "2pt"
            },
            DataElementName = "Name",
            Value = "=Fields!Name.Value",
        });
        listMovies.ReportItems.Add(listCast);
        section.Body.ReportItems.Add(listMovies);
    }
```

11. Add the following code to create a JSON data source, add a dataset and a nested dataset, and add fields to both datasets.

C# code. Paste AFTER CreateDataRegions()

```
private void CreateDataSource()
    {
        DataSource dataSource = new DataSource();
        dataSource.Name = "JSONDataSource";
        dataSource.ConnectionProperties = new ConnectionProperties()
        {
            DataProvider = "JSON",
            ConnectString = "jsondoc={path to file}\\JSON With Nested Data.json"
        };
        report.Report.DataSources.Add(dataSource);
        report.Report.DataSets.Add(CreateDataSet(dataSource));
        report.Report.DataSets.Add(CreateSecondDataSet());
    }
    private DataSet CreateDataSet(DataSource _DataSource)
    {
        Query query = new Query
        {
            Timeout = 30,
            CommandText = "$.Movie[*]",
            DataSourceName = _DataSource.Name,
```

```
            };
            DataSet dataSet = new DataSet
            {
                Query = query,
                Name = "Movies",
            };
            CreateFieldsToDataSet(dataSet);
            return dataSet;
        }
        private DataSet CreateSecondDataSet()
        {
            Query query = new Query
            {
                Timeout = 30,
                CommandText = "$",
                DataSourceName = "$dataset:Movies/Cast",
            };
            DataSet dataSet = new DataSet
            {
                Query = query,
                Name = "Cast",
            };
            CreateFieldsToSecondDataSet(dataSet);
            return dataSet;
        }
        private void CreateFieldsToDataSet(DataSet dataSet)
        {
            Field field = new Field
            {
                DataField = "Title",
                Name = "Title"
            };
            dataSet.Fields.Add(field);
        }
        private void CreateFieldsToSecondDataSet(DataSet dataSet)
        {
            Field field = new Field
            {
                DataField = "Name",
                Name = "Name"
            };
            dataSet.Fields.Add(field);
        }
```

12. Add the following code to load the report to the Designer in Form1_Load event.

C# code. Paste INSIDE the Form1_Load event.

```
designer.LoadReport(XmlReader.Create(new StringReader(report.ToRdlString())),
DesignerReportType.Page);
```

13. Run the project and preview the report.

Bind Page/RDLX Report to Data at Run Time

Create a data source

Before creating a data source, you must first decide what type of data source you need. ActiveReports supports all most common types of data sources, providing a flexible interface for interacting with them.

An example below demonstrates how to create your data source and add it to the report.

```
private PageReport LoadReport()
{
    // Your path to the report
    string path = "";

    PageReport report = new PageReport(new FileInfo(path));

    DataSource dataSource = CreateDataSource();
    report.Report.DataSources.Add(dataSource);
    return report;
}
private DataSource CreateDataSource()
{
    ConnectionProperties connectionProperties = new ConnectionProperties
    {
        ConnectString = "", // Your connection string to data source
        DataProvider = "", // Your data provider name
        IntegratedSecurity = true, // or set it to False if you don't need it
    };
    DataSource dataSource = new DataSource
    {
        Name = "", // Your data source name
        ConnectionProperties = connectionProperties,
    };
    return dataSource;
}
```

The **CreateDataSource** method returns an instance of the DataSource class. This method encapsulates the logic of creating a new data source.

There are two main steps in creating a data source.

1. Creating connection parameters.

```
ConnectionProperties connectionProperties = new ConnectionProperties
{
    ConnectString = "", // Your connection string to data source
```

```
    DataProvider = "", // Your data provider name
    IntegratedSecurity = true, // or set it to False if you don't need it
};
```

When creating a data source, you can use the following most common data provider names and your own data providers from the configuration file:

- **SQL** - Connection with MS SQL Server.
- **OLEDB** - Connection with OleDB drivers.
- **ODBC** - Connection with ODBC drivers.
- **CSV** - Connection with CSV data.
- **JSON** - Connection with JSON data.
- **XML** - Connection with XML data.

2. Creating a data source.

```
DataSource dataSource = new DataSource
{
    Name = "", // Your data source name
    ConnectionProperties = connectionProperties,
};
```

Note that creating your own data source may require additional settings for the report. If you need to modify an existing data source in the report, then use this series of articles.

- Bind Page/RDLX Report to Entity Framework and Other Data Providers (Entity Framework, DataSet, Object data providers)
- Bind Page/RDLX Report to Data Providers (SQL, OLEDB. ODBC)
- Bind to Custom Data Providers (CSV, JSON, XML)

Create a data set

Data sets are an integral part of almost any report you create in ActiveReports. An example below demonstrates how to create your data set and add it to the report.

```
private PageReport LoadReport()
{
    // Your path to the report
    string path = "";

    PageReport report = new PageReport(new FileInfo(path));
    // Create your own data source or use an existing one
    DataSource dataSource = new DataSource();
    report.Report.DataSources.Add(dataSource);
    DataSet dataSet = CreateDataSet(dataSource);
    report.Report.DataSets.Add(dataSet);
    return report;
}
```

```
private DataSet CreateDataSet(DataSource dataSource)
{
    // Your request for data
    Query query = new Query
    {
        Timeout = 30, // Your timeout
        CommandText = "", // Your SQL command
        DataSourceName = dataSource.Name,
    };
    // Your data set
    DataSet dataSet = new DataSet
    {
        Query = query,
        Name = "", // Your dataset name
    };
    // Adding fields to the created dataset
    CreateFieldsToDataSet(dataSet);
    return dataSet;
}
private void CreateFieldsToDataSet(DataSet dataSet)
{
    // Your field to add the data set
    Field field = new Field
    {
        DataField = "", // Your name of the field in the query
        Name = "", // Your name to use for the field within the report
    };
    dataSet.Fields.Add(field);
}
```

In this example, the creation of a data set is divided into the following steps:

1. Create a data source that will be used to build a data set. See the **Create a data source** section above for details.
2. Create a query instance that contains a description of the query that must be executed to retrieve the data.

   ```
   // Your request for data
   Query query = new Query
   {
       Timeout = 30, // Your timeout
       CommandText = "", // Your SQL command
       DataSourceName = dataSource.Name,
   };
   ```
3. Create a data set instance and pass a previously created query instance into it.

   ```
   // Your data set
   DataSet dataSet = new DataSet
   {
   ```

```
    Query = query,
    Name = "", // Your data set name
};
```

4. Add fields to your previously created data set if you need them.

```
private void CreateFieldsToDataSet(DataSet dataSet)
{
    // Your field to add the data set
    Field field = new Field
    {
        DataField = "", // Your name of the field in the query
        Name = "", // Your name to use for the field within the report
    };
    dataSet.Fields.Add(field);
}
```

Note: The sample report must contain controls to work with the specified fields in the data set.

Use Dynamic Connection String in Data Source

Adding a dynamic connection string to a data source is a convenient tool that allows you to change the connection string during the report execution. This is achieved by using parameterized parameters in the connection string, or by using completely different connection strings for completely different databases.

This topic describes two basic ways of working with dynamic connection strings:

1. The full connection string as a parameter.
2. Individual elements of the connection string as a parameter.

Using the dynamic connection string as a parameter makes it much easier to manage the connection string at runtime. You can set the connection string as a parameter in several ways:

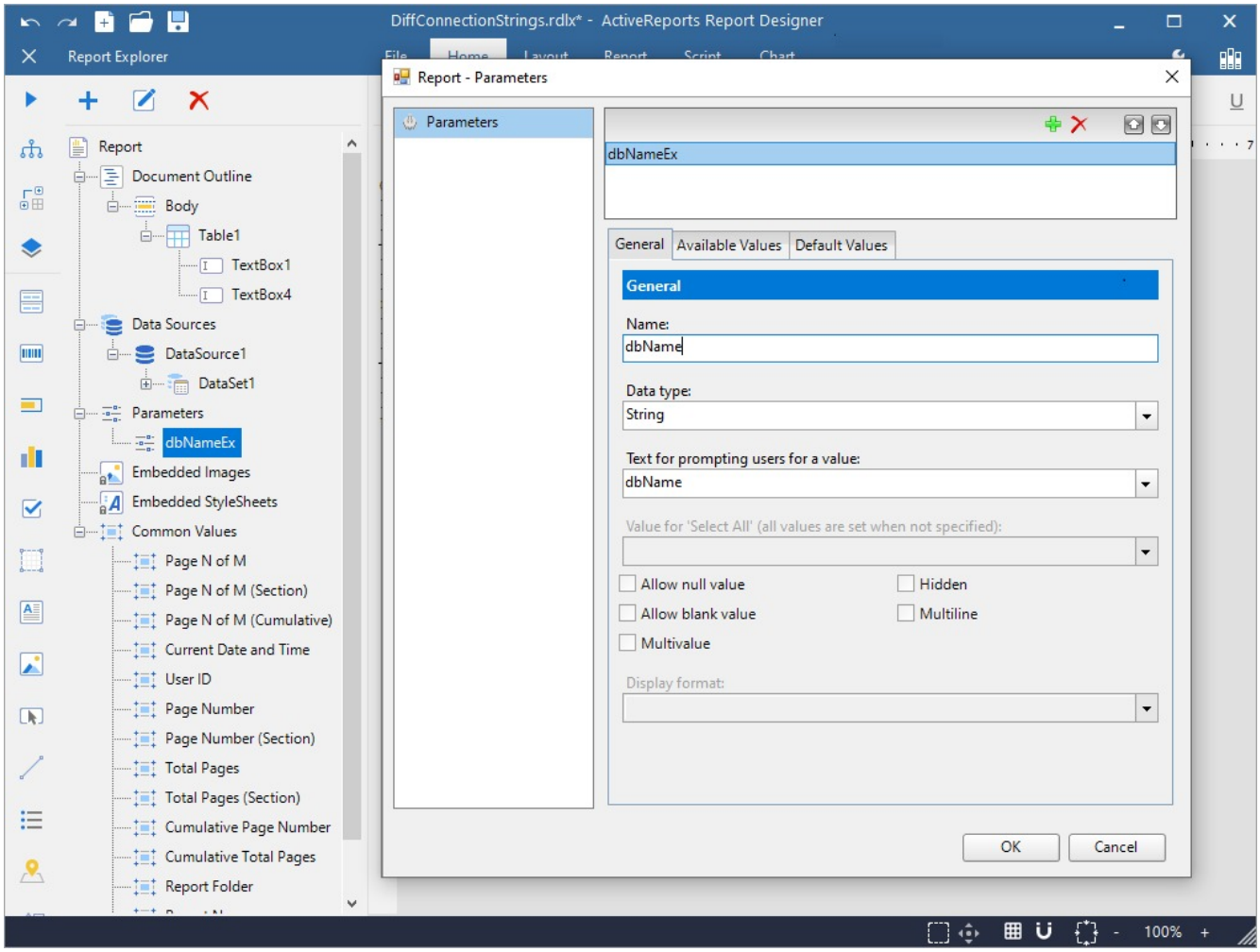
1. Use a Windows or Web application to create a report and create parameters that the report will use.
2. Create these parameters and connection strings, using the code.

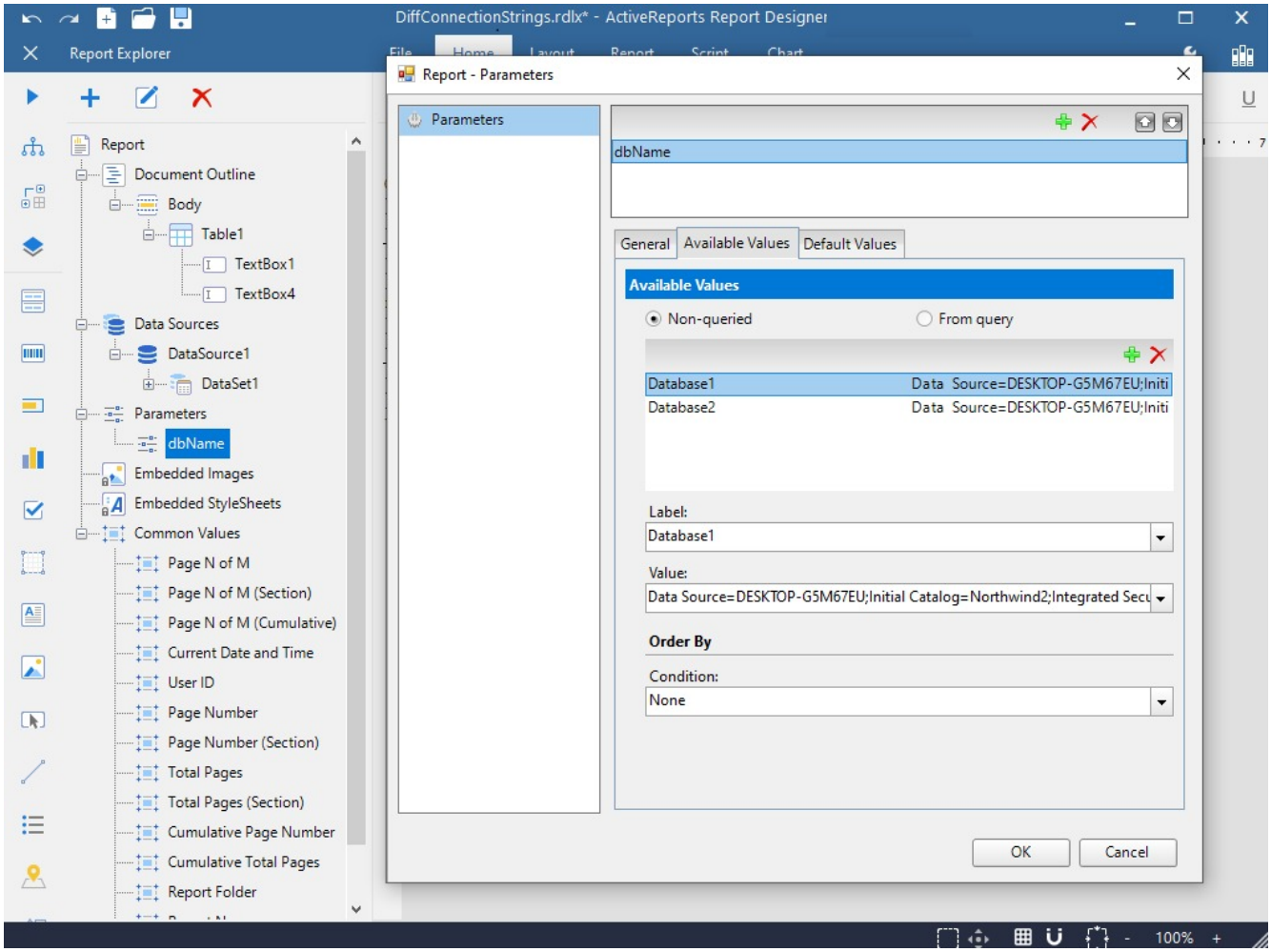
Create a dynamic connection string in the Windows or Web application

Creating a dynamic connection string in a Windows or Web application is a common scenario when interacting with the parameters of your report. Follow these steps to create a dynamic connection string.

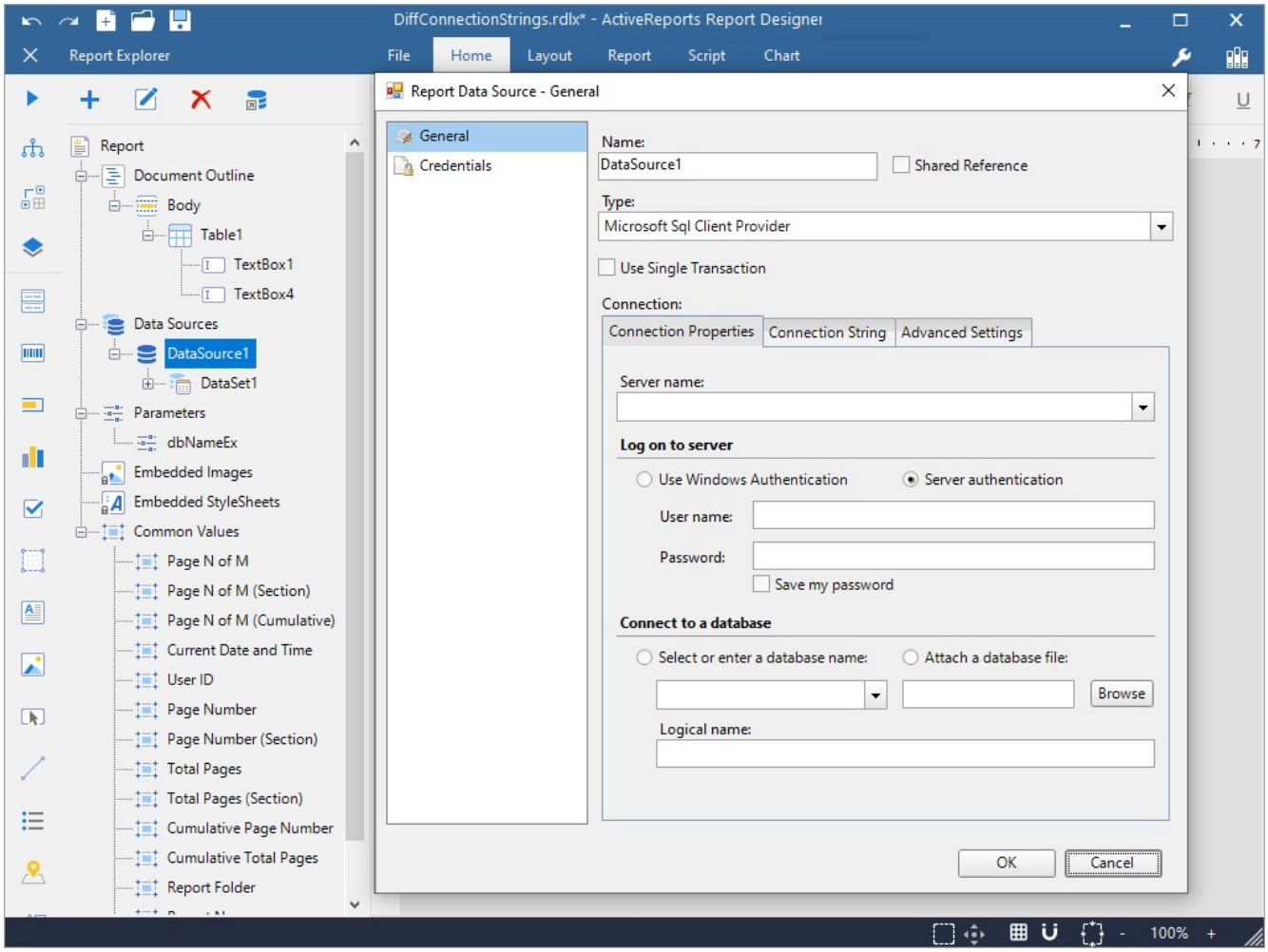
Note: The sample below uses the Windows Forms Designer.

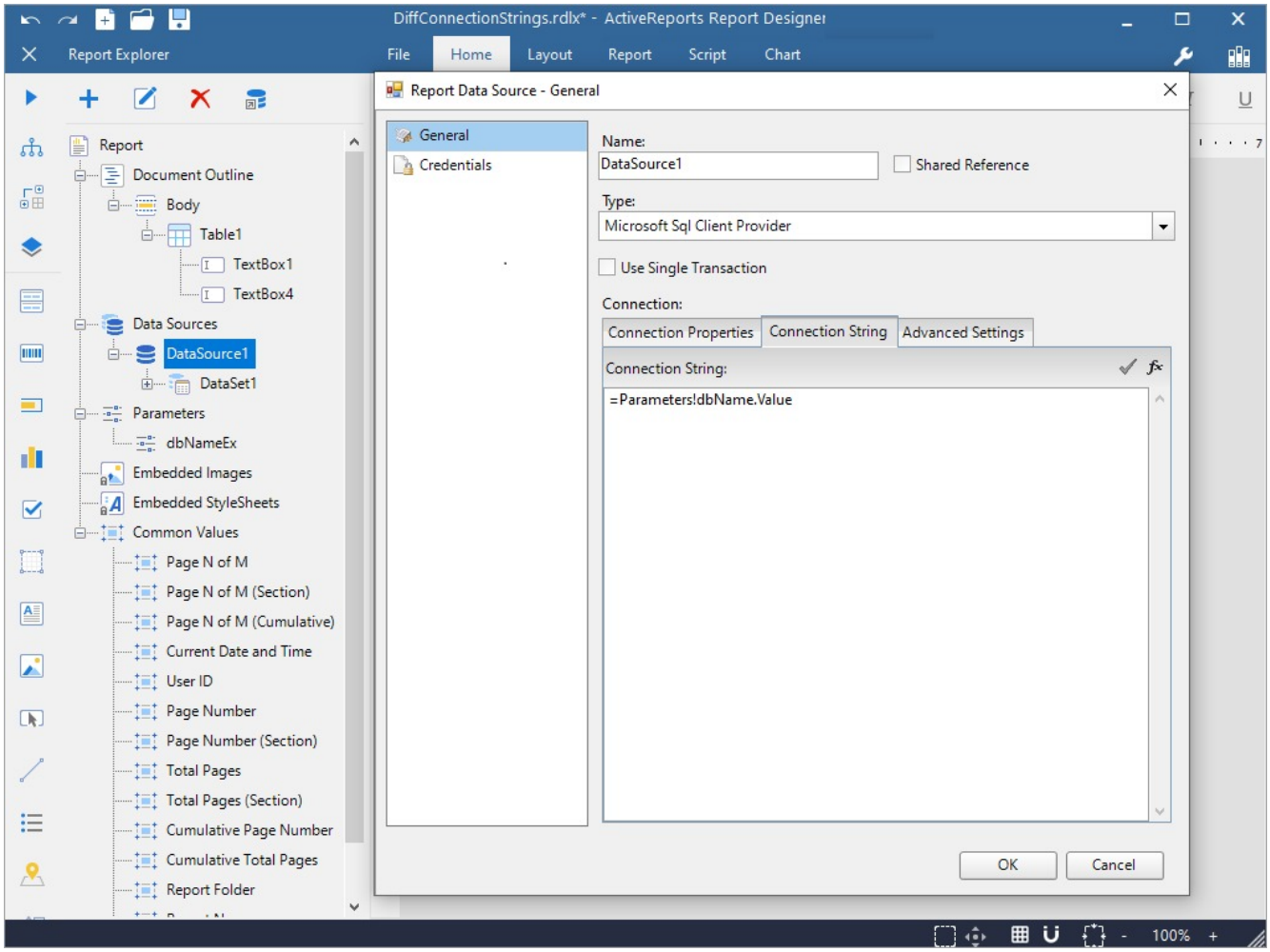
1. Create a report and go to the **Report - Parameters** dialog to specify the report settings.





2. Click **OK** to close the **Report - Parameters** dialog and open the **Report Data Source** dialog. In the opened dialog, select the type of the database connection. The example below uses **Microsoft Sql Client Provider**.





Note: The value `=Parameters!dbName.Value` is the value from the **Name** field, specified in step 1.

By following these steps, you will use the created data source while designing a report in the Windows or Web application and also manage it, using the code as demonstrated below.

```
private PageReport LoadReport()
{
    // Your path to the report
    string path = "";
    // 0 - "Database1" from step 1
    // 1 - "Database2" from step 1
    int databaseIndex = 0;
    // If "true", you must choose which database to use, otherwise the one you specified will be
automatically selected
    bool showDatabaseSelection = false;
    PageReport report = new PageReport(new FileInfo(path));
    ConfigureConnectionString(databaseIndex, report, showDatabaseSelection);
    return report;
}
private void ConfigureConnectionString(int databaseIndex, PageReport report, bool
```

```
showDatabaseSelection=true)
{
    if (!showDatabaseSelection)
    {
        // Select the connection string from the previously created parameter
        string connectionString =
report.Report.ReportParameters[0].ValidValues.ParameterValues[databaseIndex].Value.ToString();
        report.Report.ReportParameters.Clear();
        report.Report.DataSources[0].ConnectionProperties.ConnectString = connectionString;
    }
}
```

The **ConfigureConnectionString** method takes values as parameters, indicating what database to use from a previously created parameter.

Create a dynamic connection string in code

This code example shows how to create a string.

```
private PageReport LoadReport()
{
    // Your path to the report
    string path = "";

    PageReport report = new PageReport(new FileInfo(path));
    // Create dynamic connection strings
    report.Report.ReportParameters.Add(CreateDynamicConnectionString());

    DataSource dataSource = report.Report.DataSources[0]; // Your data source
    if (dataSource.ConnectionProperties.DataProvider == "SQL") // Use any other database
    {
        // Set a dynamic string to connect to the data source
        dataSource.ConnectionProperties.ConnectString = @"=Parameters!dbName.Value";
    }
    return report;
}
private ReportParameter CreateDynamicConnectionString()
{
    string database1 = ""; // Your first connection string
    string database2 = ""; // Your second connection string
    ValidValues validValues = CreateValidValues(database1, database2);
    ReportParameter dynamicString = new ReportParameter
    {
        AllowBlank = true,
        DataType = ReportParameterDataType.String,
        Name = "", // The name of your parameter. It is used in the connection string
        ValidValues = validValues,
        Nullable = false,
        Prompt = ""
    };
```

```
    return dynamicString;
}
private ValidValues CreateValidValues(string firstConnString, string secondConnString)
{
    // The label of your parameter is "Database1"
    ParameterValue firstParameter = CreateParameterValue("Database1", firstConnString);
    // The label of your parameter is "Database2"
    ParameterValue secondParameter = CreateParameterValue("Database2", secondConnString);
    ValidValues validValues = new ValidValues();
    validValues.ParameterValues.Add(firstParameter);
    validValues.ParameterValues.Add(secondParameter);
    return validValues;
}
private ParameterValue CreateParameterValue(string label, string value)
{
    return new ParameterValue()
    {
        Label = label,  // The Label of your parameter value
        Value = value,
    };
}
```

Looking at the code above, you can see that creating a dynamic connection string consists of 2 steps.

1. Creating the dynamic string parameter.

   ```
   report.Report.ReportParameters.Add(CreateDynamicConnectionString());
   ```

   It is important to create a report parameter because it stores information and connection strings. If this is not done or done incorrectly, nothing will work. This procedure is performed with the **CreateDynamicConnectionString**, **CreateValidValues**, and **CreateParameterValue** methods.
2. Applying this parameter to the data source in the connection string.

   ```
   DataSource dataSource = report.Report.DataSources[0]; // Your data source
   if (dataSource.ConnectionProperties.DataProvider == "SQL")
   {
       // Set a dynamic string to connect to the data source
       dataSource.ConnectionProperties.ConnectString = @"=Parameters!dbName.Value";
   }
   ```

   In the example above you will notice that it uses a connection to the SQL provider. However, you can use a different database at this point. See Bind Page/RDLX Report at Run Time for the list of available database providers.
   Note that you need to specify a previously created parameter as a connection string, not the value of the connection string itself.

   ```
   // Set a dynamic string to connect to the data source
   dataSource.ConnectionProperties.ConnectString = @"=Parameters!dbName.Value";
   ```

   For example, a connection string that you can place in a parameter looks like this.

   ```
   Data Source=[data source];Initial Catalog=[initial catalog];Integrated Security=True;Connect
   Timeout=30;Encrypt=False
   ```

If you want to change only a part of the parameter, just change the connection string as follows.

```
// Set a dynamic string to connect to the data source
dataSource.ConnectionProperties.ConnectString = "=\"Data Source=[data source];Initial
Catalog=\" & Parameters!dbName.Value & \";Integrated Security=True;Connect
Timeout=30;Encrypt=False\"";
```

In the **CreateDynamicConnectionString** method, you need to pass the names of the databases, not the full connection strings:

```
string database1 = "example1"; // Your first database name
string database2 = "example2"; // Your second database name
```

Having done this, you will be asked, which database name is substituted in your connection string at the stage of thre report execution.

Save and Load Reports

The topic describes saving and loading a Page/RDLX report.

Save a report as an RDLX file at design time

1. From the Visual Studio **Report** menu, select **Save Layout**.

   > **Note:** The **Report** menu in **Visual Studio 2019** and above is available as submenu under **Extensions**. You should select the **Design View** of the report in the ActiveReports Designer first.

2. In the **Save As** dialog that appears, set the file name and select the location where you want to save it. The file extension is ***.rdlx**.
3. Click the **Save** button to save the report layout and close the dialog.

Load an RDLX file at design time

1. From the Visual Studio **Report** menu, select **Load Layout**.
2. In the **Open** dialog that appears, navigate to the location of the .rdlx file and select it.
3. Click the **Open** button to load the report layout.

Save a report as an RDLX file at run time

Use the **Save** method to save your report layout at run time.

1. Right-click the Windows Form and select **View Code** to see the code view for the Windows form.
2. Add the following code to the Form class to save the report.

The following example shows what the code for the method looks like to save a Page or an RDLX report.

VB# code. Paste INSIDE the Form class

```
Private Sub SaveButton_Click(ByVal sender As Object, ByVal e As EventArgs)
    Dim saveFileDialog As SaveFileDialog = New SaveFileDialog()
    saveFileDialog.Title = "Save Report"
```

```
    saveFileDialog.Filter = "Page Report Files|*.rdlx"
    saveFileDialog.ShowDialog()
    If saveFileDialog.FileName <> "" Then
        pageReport.Save(New FileInfo(saveFileDialog.FileName))
    End If
End Sub
```

C# code. Paste INSIDE the Form class

```
private void SaveButton_Click(object sender, EventArgs e)
{

    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Title = "Save Report";
    saveFileDialog.Filter = "Page Report Files|*.rdlx";
    saveFileDialog.ShowDialog();
    if (saveFileDialog.FileName != "")
        {
            //Save Report File
           pageReport.Save(new FileInfo(saveFileDialog.FileName));
        }
}
```

Merge Multiple Reports

Multiple RDLX and Page Reports can be merged or combined into one report. The **ReportCombiner ('ReportCombiner Class' in the on-line documentation)** class is used for performing merging, which adds the reports as subreports. The reports are merged one after the another, in the order in which they are added. To combine the reports correctly, you should use reports with the same layouts. For **Page** reports, you need to set equal margins for all reports and for **RDLX** reports, you need to set equal margins and width for all reports. See Layout topic for more information.

Having access to **BuildReport ('BuildReport Method' in the on-line documentation)** method, user can use full **PageReport** class functionality. You can add a page break or specify the gap between two reports.

You can use the **BuildReport ('BuildReport Method' in the on-line documentation)** method to utilize all the features of the **PageReport** class. You can also insert page breaks and specify the gap between two reports when merging. By default, the gap of 1 inch is added between the reports.

The following code examples demonstrate merging reports in different scenarios. You need to add following packages to your project:

- **MESCIUS.ActiveReports**
- **MESCIUS.ActiveReports.Viewer.Win**
- **MESCIUS.ActiveReports.Export.Pdf** (for exporting the merged report in PDF format)

Combine Three Reports

Visual Basic.NET. Paste inside Form1_Load event.

```
Dim combiner = New GrapeCity.ActiveReports.ReportsCore.Tools.ReportCombiner()

Dim r1 = New GrapeCity.ActiveReports.PageReport()
r1.Load(New System.IO.FileInfo("c:\temp\Report1.rdlx"))

Dim r2 = New GrapeCity.ActiveReports.PageReport()
r2.Load(New System.IO.FileInfo("c:\temp\Report2.rdlx"))

Dim r3 = New GrapeCity.ActiveReports.PageReport()
r3.Load(New System.IO.FileInfo("c:\temp\Report3.rdlx"))

combiner.SetMargin("0cm")
combiner.DefaultStep = "0cm"
combiner.AddReport(r1)

Dim options = New GrapeCity.ActiveReports.ReportsCore.Tools.LocationOptions
options.Gap = "5in" 'adds a 5 inch gap from the first report. By default this gap is 1 inch.
options.PageBreakBefore = True 'adds a page break.

combiner.AddReport(r2, options)
combiner.AddReport(r3)

'PDF Rendering extension
Dim pdfRe = New GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension()
Dim provider = New GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(New
System.IO.DirectoryInfo("c:\temp\"), "CombinedReport")

Viewer1.LoadDocument(combiner.BuildReport().Document)   'load combined report in viewer
combiner.BuildReport().Document.Render(pdfRe, provider) 'export combined report in PDF format
```

C# code

```
var combiner = new GrapeCity.ActiveReports.ReportsCore.Tools.ReportCombiner();

var r1 = new GrapeCity.ActiveReports.PageReport();
r1.Load(new System.IO.FileInfo(@"c:\temp\Report1.rdlx"));

var r2 = new GrapeCity.ActiveReports.PageReport();
r2.Load(new System.IO.FileInfo(@"c:\temp\Report2.rdlx"));

var r3 = new GrapeCity.ActiveReports.PageReport();
r3.Load(new System.IO.FileInfo(@"c:\temp\Report3.rdlx"));

combiner.SetMargin("0cm");
combiner.DefaultStep = "0cm";
combiner.AddReport(r1);
combiner.AddReport(r2, new LocationOptions() { PageBreakBefore = true, Gap = "5in" }); //adds
```

```
second report after a page break and a 5 inch gap from the first report. By default this gap
is 1 inch.
combiner.AddReport(r3);

//PDF Rendering extension
var pdfRe = new GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension();
var provider = new GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(new
System.IO.DirectoryInfo(@"c:\temp\"), "CombinedReport");

viewer1.LoadDocument(combiner.BuildReport().Document);   //load combined report in viewer
combiner.BuildReport().Document.Render(pdfRe, provider); //export combined report in PDF
format
```

The merged reports can be modified in many ways as described below.

Add reports at particular index

If you want to add a report r4 after the first report r1, use the following code with index '1':

C#

```
combiner.Insert(1, r4, new LocationOptions());
report = combiner.BuildReport();
```

VB.NET

```
combiner.Insert(1, r4, New GrapeCity.ActiveReports.ReportsCore.Tools.LocationOptions())
report = combiner.BuildReport()
```

Add a list of reports

C#

```
combiner.AddRange(new PageReport[] {r1, r2, r3, r4 }, new LocationOptions())
report = combiner.BuildReport();
```

VB.NET

```
Dim reports As IEnumerable(Of GrapeCity.ActiveReports.PageReport) = {r1, r2, r3, r4}
combiner.AddRange(reports, New GrapeCity.ActiveReports.ReportsCore.Tools.LocationOptions())
report = combiner.BuildReport()
```

Delete report(s)

If you want to delete a report in first position, use the following code with index '0':

C#

```
combiner.RemoveAt(0);
report = combiner.BuildReport();
```

VB.NET

```
combiner.RemoveAt(0)
report = combiner.BuildReport()
```

Similarly, if you want to delete report at second position, use index '1'.

If you want to delete all instances of report r2, use the following code:

C#

```
combiner.RemoveAll(r2);
report = combiner.BuildReport();
```

VB.NET

```
combiner.RemoveAll(r2)
report = combiner.BuildReport()
```

Note:

- If the reports being merged are of different sizes, the page size of the first report is applied to all the pages of combined report.
- If the reports being merged are of different paper orientations, the paper orientation of the first report is applied to all the pages of combined report.

Add Code to Layouts Using Script

In a Page Report or an RDLX Report, you can use custom code in your expressions to extend the capabilities of your report. However, for complex functions, or functions you plan to use many times in the report, you also have the facility to embed the code within the report. You can also create and maintain a custom assembly for code that you want to use in multiple reports and refer to its methods in expressions.

Note:

1. You should be aware that scripts allow to execute different code and can be a security gap. Therefore, it is not recommended to use scripts in web/cloud environments.
2. In **WebDesigner**, the Scripts are available for Section reports only.

To turn off scripts in **WinForms Designer**, use the following code:

For WinForms Designer

```
_designer.EnableScripting = false;
```

Page/RDLX reports are designed to avoid scripts. However, some situations may still require using scripts. Page/RDLX reports support Visual Basic scripts, which can be used as expressions - see the following example:

Script in Page/RDLX reports

```
Public Function GetRandom(ByVal seed As Int32, ByVal maxValue As Int32) As Int32
    Return New System.Random(seed).Next(maxValue)
```

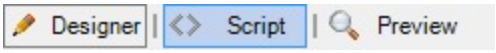
```
End Function
```

Then, you should set a Textbox to a value as in the example below.

Example Title
`=Code.GetRandom(Fields!ID.Value, 10000)`

Embed Code in the Report

Add a Page Report/RDLX report template to your project and in the Visual Studio Integrated Designer that appears, add code like the following in the Script tab.



Call a function from the control's property

This is a simple example of a single method code block:

```
Public Function GetDueDate() as Date
     Return DateTime.Now.AddDays(30)
 End Function
```

This is an expression used to call the method in the code block from the control's property. For example, you can call this function in the **Value** property of the Textbox control:

```
=Code.GetDueDate()
```

Use the custom constant and variable from the control's property

This is a simple example of how to define custom constants and variables in your code block:

```
Public Dim MyVersion As String = "123.456"
Public Dim MyDoubleVersion As Double = 123.456
Public Const MyConst As String = "444"
```

This is an expression to use the custom constant and variable in the code block from the control's property. For example, you can get the value of a variable or a constant in the **Value** property of the Textbox control:

```
=Code.MyVersion
=Code.MyDoubleVersion
=Code.MyConst
```

Call a global collection from the control's property

This is a simple example of a global collection code block where the code block references the report object to access

the report parameter value:

```
Public Function ReturnParam() As String
    Return "param value = " + Report.Parameters!ReportParameter1.value.ToString()
End Function
```

This is an expression used to call a global collection in the code block from the control's property. For example, you can call the global collection in the Value property of the Textbox control:

```
=Code.ReturnParam()
```

Use instance-based Visual Basic .NET code in the form of a code block. You can include multiple methods in your code block, and access those methods from expressions in the control properties.

> **Note:** In a Page Report or an RDLX Report, you can use Visual Basic.NET as the script language. However, you can use both Visual Basic.Net and C# in your script for a Section Report.

Create custom assemblies

You can create custom assemblies in C# or Visual Basic .NET to make code available to multiple reports:

1. Create or find the assembly you wish to use. The assemblies can be found in the installed location: C:\Program Files (x86)\MESCIUS\ActiveReports 18\NuGet\{*Package name*}\lib\net46\{*Assembly*}
2. Make the assembly available to the report engine.
    - If you are embedding the designer or viewer controls in your own application, copy the assembly to the same location as your executable.
    - If you are using the included designer or viewer, copy the assembly into the ActiveReports assembly folder located at ...\MESCIUS\ActiveReports 18 by default.

      > **Note:** To make the assembly available for use in your own application and for use in designing reports for your application, copy it to both locations listed above. Alternatively, you can place the assembly in the Global Assembly Cache (C:\Windows\assembly).
3. Add an assembly reference to the report.
    1. From the Report Menu, choose **Report Properties**.
    2. In the Report dialog that appears, select the **References** and click the **Open** icon above the assembly name list to add your own assembly.
    3. Go to the Class list below the assembly names and under **Class name**, enter the namespace and class name. Similarly, under **Instance name**, enter the name you want to use in your expressions.
4. Access the assembly through expressions.
    - To access static members (member denoted as `public static` in C# assemblies, or as `Public Shared` in Visual Basic assemblies):
      `=Namespace.Class.Member`
    - To access class instances:
      `=Code.`*`InstanceName`*

Create a Custom Report Item

Custom report items (CRIs) are supported in all Viewers and all types of exports. However, they are not currently supported in the WebDesigner component.

> **Note**: Currently, you need to use some internal API to create a CRI, which we are planning to change in the future versions. In case of any problems that may occur during upgrade, contact our Support Team.

With a CRI, you can implement the following tasks:

- Implement simple custom rendering, which may be useful for implementing a simple rendering like a custom SVG image, a custom barcode, etc.
- Implement a custom layout, which may be useful in case of a custom content layout (if size is not defined, etc.).
- Implement custom data processing, which is the most complicated scenario where a CRI requests data from a data set. We recommend that you first check these samples before you start the implementation - CustomChart and Calendar.

To implement a simple CRI, see the RadarChart.cs file in the CustomChart sample to have an idea of what your CRI implementation should look like. If you require access to data, then you should use the **IDataRegion** interface.

To implement the Designer, see the RadarDesigner.cs file in the CustomChart sample. Remember that the DataSet is required only in case of data binding (the **IDataRegion** interface). Otherwise, you can use just a simple property that is passed and obtained from a CRI as **ReportItem.CustomProperties**.

You should also remember to add a configuration file - you can find an example in the CustomChart sample.

Apply Styles through Code

The following steps set a style sheet source and apply style to a **TextBox** control.

1. In Visual Studio, create a new **Page Report Application** or open an existing one.
2. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
3. Add the following code inside the Form_Load event.

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

```
'Path and Name of the loaded PageReport
Dim filePath As String = "C:\SampleReport.rdlx"
Dim pageReport As New GrapeCity.ActiveReports.PageReport(New System.IO.FileInfo(filePath))
Dim reportDocument As New GrapeCity.ActiveReports.Document.PageDocument(pageReport)

' Set the style sheet source and value using external style sheets
reportDocument.PageReport.Report.StyleSheetSource =
GrapeCity.ActiveReports.PageReportModel.StyleSheetSource.External
reportDocument.PageReport.Report.StyleSheetValue = "C:\ExternalStyle.rdlx-styles"

' Set the style sheet source and value using embedded style sheets
reportDocument.PageReport.Report.StyleSheetSource =
GrapeCity.ActiveReports.PageReportModel.StyleSheetSource.Embedded
reportDocument.PageReport.Report.StyleSheetValue = "EmbeddedStylesheet1

' Add a Textbox control and apply style
Dim text As New GrapeCity.ActiveReports.PageReportModel.TextBox()
text.Value = "Sample Text"
text.Style.StyleName = "Style1"
pageReport.Report.Body.ReportItems.Add(text)
viewer1.LoadDocument(reportDocument)
```

C# code. Paste INSIDE the Form Load event.

```
//Path and Name of the loaded PageReport
```

```
string filePath = @"C:\SampleReport.rdlx";
GrapeCity.ActiveReports.PageReport pageReport =
new GrapeCity.ActiveReports.PageReport(new System.IO.FileInfo(filePath));
GrapeCity.ActiveReports.Document.PageDocument reportDocument =
new GrapeCity.ActiveReports.Document.PageDocument(pageReport);

// Set the style sheet source and value using external style sheets
reportDocument.PageReport.Report.StyleSheetSource =
GrapeCity.ActiveReports.PageReportModel.StyleSheetSource.External;
reportDocument.PageReport.Report.StyleSheetValue = @"C:\ExternalStyle.rdlx-styles";

// Set the style sheet source and value using embedded style sheets
reportDocument.PageReport.Report.StyleSheetSource =
GrapeCity.ActiveReports.PageReportModel.StyleSheetSource.Embedded;
reportDocument.PageReport.Report.StyleSheetValue = "EmbeddedStylesheet1";

// Add a Textbox control and apply style
GrapeCity.ActiveReports.PageReportModel.TextBox text =
new GrapeCity.ActiveReports.PageReportModel.TextBox();
text.Value = "Sample Text";
text.Style.StyleName = "Style1";
pageReport.Report.Body.ReportItems.Add(text);
viewer1.LoadDocument(reportDocument);
```

Section Report

Section report is a format that is similar to the RDLX report type with the BandedList data region in the paginated mode. ActiveReports supports two kinds of Section reports:

- **Xml-based Section report (RPX)** is an XML-based format that uses event handlers as restricted scripts. This is your best choice if you can write code scripts.
- **Code-based Section report (CodeDOM)** best meets your needs if you are developer and can support the report along with the application. The Designer is available with Visual Studio only but you can perform many report operations using event handlers. In Visual Studio 2022, only .NET Framework 4.8 and .NET 5.0/.NET 6.0/.NET 7.0 are supported for code-based section reports.

Create a Report or Load an Existing Report

This topic discusses about creating and loading a Section Report (xml or code based) in designer using code. We recommend you to create a new report directly from the standalone designer or you using the built-in project templates from Visual Studio.

Create a Report

Use **NewReport() ('NewReport Method' in the on-line documentation)** method in the **Designer ('Designer Class' in the on-line documentation)** class to initialize the designer with a new report layout, with specified type. Make sure to add **MESCIUS.ActiveReports.Design.Win** package to your project.

C#. Add using statements on the top of Form.cs

```
using GrapeCity.ActiveReports.Design;
```

```
using System.Windows.Forms;
```

C# code. Paste INSIDE the Form Load event.

```
var _designer = new Designer() { Dock = DockStyle.Fill };
_designer.NewReport(DesignerReportType.Section);
Controls.Add(_designer);
```

Bind a Section Report to Data

You can use one of the following ways to bind a section report to data at run time:

- You can bind a section report to a data source.
- You can bind your section report to a dataset and unbound data.

A Section report also allows to locate data in scripts and can help locate credentials for some built-in data sources.

The **LocateCredentials ('LocateCredentials Event' in the on-line documentation)** event can work with the following data source types:

- **SQL ('SqlDBDataSource Class' in the on-line documentation)**
- **OleDB ('OleDBDataSource Class' in the on-line documentation)**
- **ODBC ('OdbcDataSource Class' in the on-line documentation)**

The following topics provide information on the data binding scenarios with Secion reports:

Bind a Section report to a Data Source at Run Time

Bind Data Set and Unbound Data to Section Report at Run Time

Bind a Section Report to Data Source at Run Time

You can bind a section report to data by changing the data source at run time.

Create and Configure OLEDB Data Source

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.

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 connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + "[User
Folder]\Samples18\Data\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
```

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 connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + @"[User
Folder]\Samples18\Data\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;
```

Close the data connection (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.

Visual Basic.NET code. Paste INSIDE the ReportEnd event

```
reader.Close()
conn.Close()
```

Close the data connection (CS)

1. Click in the gray area below rptModifyDS to select the report.
2. Click the events icon in the Properties Panel 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.

C# code. Paste INSIDE the ReportEnd event

```
reader.Close();
conn.Close();
```

See the UnboundData sample for details on how to use the FetchData event to display the report unbound data.

Create and Configure JSON Data Source

You can bind a section report to a JSON data source at run time using code.

1. Double-click in the gray area below **SectionReport1** to create an event-handling method for the **ReportStart** event.
2. Add code to the handler to change the data source at run time.

Visual Basic.NET code. Paste JUST ABOVE the ReportStart event

```
Dim jsonDS As GrapeCity.ActiveReports.Data.JsonDataSource = New
GrapeCity.ActiveReports.Data.JsonDataSource()
jsonDS.ConnectionString =
"jsondoc=https://demodata.mescius.io/northwind/odata/v1/Employees"
jsonDS.JsonPath = "$.value[*]"
Me.DataSource = jsonDS
```

C# code

```
GrapeCity.ActiveReports.Data.JsonDataSource jsonDS = new
GrapeCity.ActiveReports.Data.JsonDataSource();
jsonDS.ConnectionString =
@"jsondoc=https://demodata.mescius.io/northwind/odata/v1/Employees";
jsonDS.JsonPath = "$.value[*]";
this.DataSource = jsonDS;
```

Bind Data Set and Unbound Data to Section Report at Run Time

A Section report allows to specify a data set and unbound data, using a similar API. You need to just specify DataSet/DataTable/DataView/DataRowCollection/DataRow[] or IList as shown in the **DataSource ('DataSource Property' in the on-line documentation)** property. For DataSet, you should specify it as shown in the **DataMember ('DataMember Property' in the on-line documentation)** property.

In addition, a Section report allows to use any unbound data as a data source with the following script logic:

To use the IEnumerable data source

1. Right-click the design surface and select View Code.
2. Add the following code inside the class declaration of the report:

Visual Basic.NET code. Paste inside the class declaration of the report.

```
Private datasource1 As IEnumerator(Of String) = Nothing
Dim list As List(Of String)= Nothing
```

Visual Basic.NET code. 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
```

C# code. Paste inside the class declaration of the report.

```
private IEnumerator<string> datasource = null;
```

C# code. Paste inside the class declaration of the report.

```
private IEnumerable<string> GetIEnumerableData()
{
    for (int i = 1; i <= 10; i++)
    {
        yield return string.Format("TestData_{0}", i.ToString());
    }
}
```

3. On the design surface, right-click the gray area around the design surface to select the report and select Properties.
4. In the Properties window that appears, click the Events icon to view the available events for the report.
5. Double-click the **DataInitialize** event. This creates an event-handling method for the report's DataInitialize event.
6. Add the following code to the handler to add fields to the report's Fields collection.

Visual Basic.NET code. Paste inside the DataInitialize event.

```
Me.Fields.Add("TestField")
Me.list = New List(Of String)
datasource1 = GetIEnumerableData().GetEnumerator()
```

C# code. Paste inside the DataInitialize event.

```
this.Fields.Add("TestField");
datasource = GetIEnumerableData().GetEnumerator();
```

7. Repeat steps 3 and 4 to open the events list in the property window.
8. Double-click the **FetchData** event. This creates an event-handling method for the report's FetchData event.
9. Add code to the handler to retrieve information to populate the report fields.

Visual Basic.NET code. Paste inside the FetchData event.

```
If datasource1.MoveNext() Then     Me.Fields("TestField").Value = datasource1.Current
eArgs.EOF = FalseElse     eArgs.EOF = TrueEnd If
```

C# code. Paste inside the FetchData event.

```
if (datasource.MoveNext())
{
    this.Fields["TestField"].Value = datasource.Current;
    eArgs.EOF = false;
}
else
    eArgs.EOF = true;
```

Tip: In order to view the added data at run time, add controls to your report and assign their **DataField** property to the name of the fields you added in code while creating a field collection.

Save and Load Section Reports

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, add a new blank ActiveReport and load the RPX file onto it.

Save a report as an RPX file at design time

1. From the Visual Studio **Report** menu, select **Save Layout**.

   > **Note:** The **Report** menu in **Visual Studio 2019** and above is available as submenu under **Extensions**. You should select the **Design View** of the report in the ActiveReports Designer first.

2. In the **Save As** dialog that appears, set the file name and select the location where you want to save it. The file extension is ***.rpx**.
3. Click the **Save** button to save the report layout and close the dialog.

> **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.

Load an RPX file at design time

1. From the Visual Studio **Report** menu, select **Load Layout**.
2. In the **Open** dialog that appears, navigate to the location of the .rpx file and select it.
3. Click the **Open** button to load the report layout.

Save a report as an RPX file at run time

Use the **SaveLayout(XmlWriter) ('SaveLayout Method' in the on-line documentation)** method to save your report layout at run time.

> **Note**: 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.

1. Right-click the Windows Form and select **View Code** to see the code view for the Windows form.
2. Add the following code to the Form class to save the report.

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

Visual Basic.NET code. Paste INSIDE the Form class.

```
Dim rpt As New SectionReport1()
 Dim xtw As New System.Xml.XmlTextWriter(Application.StartupPath + "\report.rpx",
Nothing)
 rpt.SaveLayout(xtw)
 xtw.Close()
```

C# code. Paste INSIDE the Form class.

```
SectionReport1 rpt = new SectionReport1();
System.Xml.XmlTextWriter xtw = new System.Xml.XmlTextWriter(Application.StartupPath +
"\\report.rpx", null);
```

```
rpt.SaveLayout(xtw);
xtw.Close();
```

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.

> **Note:** The SaveLayout method uses utf-16 encoding when you save to a stream, and utf-8 encoding when you save to a file.

Load an RPX file into the ActiveReports viewer at run time

Use the **LoadLayout(XmlReader) ('LoadLayout Method' in the on-line documentation)** method to load your report layout at run time.

1. Right-click on the Windows Form and select **View Code** 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.

Visual Basic.NET code. Paste INSIDE the Form class.

```
Dim rpt As New GrapeCity.ActiveReports.SectionReport()
' For the code to work, this report.rpx must be stored in the bin\debug folder of your
project.
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath + "\report.rpx")
rpt.LoadLayout(xtr)
xtr.Close()
Viewer1.Document = rpt.Document
rpt.Run()
```

C# code. Paste INSIDE the Form class.

```
GrapeCity.ActiveReports.SectionReport rpt = new GrapeCity.ActiveReports.SectionReport();
// For the code to work, this report.rpx must be stored in the bin\debug folder of your
project.
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
"\\Sample.rpx");
rpt.LoadLayout(xtr);
xtr.Close();
viewer1.Document = rpt.Document;
rpt.Run();
```

Change Ruler Measurements

To change measurements, you need to call measurement conversion at run-time, for example, the **CmToInch ('CmToInch Method' in the on-line documentation)** method or **InchToCm ('InchToCm Method' in the on-line documentation)** method at run-time. For example, you can use the following code when you are working in centimeters and need to convert a Label's position measurements from centimeters to inches at run-time.

1. On the design surface select the section containing a control like a Label.
2. In the Properties Panel, click the Events button to get a list of report events.
3. Select the **Format** event and double-click to create an event-handling method.
4. Add code like the following to the handler to set the size of the control using centimeters to inches.

Visual Basic.NET code. Paste inside the Format event.

```
Me.Label1.Left = SectionReport1.CmToInch(2)
Me.Label1.Top = SectionReport1.CmToInch(2)
```

C# code. Paste inside the Format event.

```
this.label1.Left = SectionReport1.CmToInch(2);
this.label1.Top = SectionReport1.CmToInch(2);
```

Printer Settings for Section Reports

In a Section Report, you can modify various printer settings or print multiple copies of a report at design time and at run time.

Printer Settings

Section reports depend on default printer settings may require some extra configuration.

C# code. Paste INSIDE the ReportStart event.

```
var rpt = new SectionReport();
rpt.Document.Printer.Name = "";
```

Note: It is recommended to set the printer name as empty if you not sure about target environments.

At design time, you can set up duplex printing, page orientation, collation, and page size in the **Printer Settings** tab of the Report Settings Dialog.

Set up duplex printing in Printer Settings

1. In the Report Explorer, double-click the **Settings** node.
2. In the Report Settings dialog that appears, click **Printer Settings**.
3. On the Printer Settings page, next to **Duplex**, select one of the following options:
   - **Printer Default**: The report uses the default settings of 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.
4. Click **OK** to return to the report.

Use code to set up duplex printing

1. Double-click the gray area below the design surface to create an event-handling method for the report's ReportStart event.
2. Add the following code to the handler to set up duplexing.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Me.PageSettings.Duplex = GrapeCity.ActiveReports.Printing.Duplex.Horizontal
```

To write the code in C#

C# code. Paste INSIDE the ReportStart event.

```
this.PageSettings.Duplex = GrapeCity.ActiveReports.Printing.Duplex.Horizontal;
```

Set page orientation on the Printer Settings page

1. In the Report Explorer, double-click the **Settings** node.
2. In the Report Settings dialog that appears, click **Printer Settings**.
3. On the **Printer Settings** page, in the Orientation section, select either **Default**, **Portrait**, or **Landscape**.
4. Click **OK** to return to the report.

Use code to change page orientation

1. Double-click the gray area below the design surface to create an event-handling method for the report's ReportStart event.
2. Add the following 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 =
GrapeCity.ActiveReports.Document.Section.PageOrientation.Landscape
```

To write the code in C#

C# code. Paste INSIDE the ReportStart event.

```
this.PageSettings.Orientation =
GrapeCity.ActiveReports.Document.Section.PageOrientation.Landscape;
```

Multiple Copies

You can print multiple copies using the Print dialog in the Preview tab or in the Viewer, or you can use code to set the number of copies to print.

Set multiple copies in the print dialog

1. With a report displayed in the viewer or in the preview tab, click **Print**.
2. In the Print dialog that appears, next to **Number of copies**, set the number of copies that you want to print.

Use code to set multiple copies

1. Double-click in the gray area below the design surface to create an event-handling method for the report's ReportStart event.
2. Add the following 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.Printer.Print()`

   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.Printer.Print();`

Insert or Add Report Pages

In a section layout, you can run multiple reports, merge their entire page collection or specific portions and view it as a single report. You can save the document containing merged reports to an RDF file or even export them.

> **Note**: When you combine section reports into one document, you should always make sure that they have the same compatibility mode (CrossPlatform or GDI). Otherwise, you may get incorrect results.

These steps assume that you have already placed a Viewer control on a Windows Form and your Visual Studio project contains two section layout (code based) reports (rptOne and rptTwo). See Windows Forms Viewer for more information.

Add pages from one report to another

To add an entire report to another, use code like the one in the example below to iterate through the entire pages collection of a report and append it to the first report. The **Add ('Add Method' in the on-line documentation)** of the PagesCollection takes one parameter (value), which references a report document page.

1. In the design view of the Form containing the Viewer, double-click the title bar of the Form to create an event-handling method for the **Form_Load** event.

2. Add the following code to the handler to add the entire rptTwo page collection 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 INSIDE the Form Load event.

```
Dim i As Integer
Dim rpt As New rptOne()
rpt.Run()
Dim rpt2 As New rptTwo()
rpt2.Run()
For i = 0 To rpt2.Document.Pages.Count - 1
rpt1.Document.Pages.Add(rpt2.Document.Pages(i))
Next
Viewer1.LoadDocument(rpt1.Document)
```

   To write the code in C#

   C# code. Paste INSIDE the Form Load event.

```
int i;
rptOne rpt1 = new rptOne();
rpt1.Run();
rptTwo rpt2 = new rptTwo();
rpt2.Run();
for(i = 0; i < rpt2.Document.Pages.Count; i++)
{
rpt1.Document.Pages.Add(rpt2.Document.Pages[i]);
}
viewer1.LoadDocument(rpt1.Document);
```

Add a range of pages from one report to another

To add a range of pages from one report to another, use the **AddRange ('AddRange Method' in the on-line documentation)**. This method has two overloads, each with one parameter. The first overload takes an array of page objects which you can use to append only the specified pages from the second report onto the first (as in the example below).

1. In the design view of the Form containing the Viewer, double-click the title bar of the Form to create an event-handling method for the **Form_Load** event.
2. Add the following code to the handler to use the AddRange() method to add pages from 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 rpt1 As New rptOne()
rpt1.Run()
```

```
Dim rpt2 As New rptTwo()
rpt2.Run()
rpt1.Document.Pages.AddRange(New GrapeCity.ActiveReports.Document.Section.Page()
{rpt2.Document.Pages(1), rpt2.Document.Pages(2)})
Viewer1.LoadDocument(rpt1.Document)
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
rptOne rpt1 = new rptOne();
rpt1.Run();
rptTwo rpt2 = new rptTwo();
rpt2.Run();
rpt1.Document.Pages.AddRange(new GrapeCity.ActiveReports.Document.Section.Page[]
{rpt2.Document.Pages[0],rpt2.Document.Pages[1]} );
viewer1.LoadDocument(rpt1.Document);
```

Insert pages from one report into another

To insert pages from one report to another, use the **Insert ('Insert Method' in the on-line documentation)** that takes two parameters, an index, which determines where to insert the pages in the main report, and a value which references the report page to insert.

1. In the design view of the Form containing the Viewer, double-click the title bar of the Form to create an event-handling method for the **Form_Load** event.
2. Add the following code to the handler to insert page 1 of rptTwo at the beginning of rptOne.

   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 rpt1 As New rptOne()
rpt1.Run()
Dim rpt2 As New rptTwo()
rpt2.Run()
rpt1.Document.Pages.Insert(0, rpt2.Document.Pages(0))
Viewer1.LoadDocument(rpt1.Document)
```

To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
rptOne rpt1 = new rptOne();
rpt1.Run();
rptTwo rpt2 = new rptTwo();
rpt2.Run();
rpt1.Document.Pages.Insert(0, rpt2.Document.Pages[0]);
viewer1.LoadDocument(rpt1.Document);
```

Insert a new page at a specific report location

To insert a new blank page at a specific location in the report, use the **InsertNew ('InsertNew Method' in the on-line documentation)** which takes one parameter, **index**, which specifies the page after which you want to insert a new blank page.

1. In the design view of the viewer form, double-click the title bar of the Form to create an event-handling method for the **Form_Load** event.
2. Add the following code to the handler to insert a blank page at the beginning of rptOne.

   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 rpt1 As New rptOne()
rpt1.Run()
rpt1.Document.Pages.InsertNew(0)
Viewer1.Document = rpt1.Document
```

   To write the code in C#

C# code. Paste INSIDE the Form Load event.

```
rptOne rpt1 = new rptOne();
rpt1.Run();
rpt1.Document.Pages.InsertNew(0);
viewer1.Document = rpt1.Document;
```

Save and Load RDF Files

ActiveReports allows reports to be saved in 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. You can save a report to an RDF file and load it into the viewer control.

Section report can be stored as an RDF document using the **SectionDocument ('SectionDocument Class' in the on-line documentation)** class. It is the intermediate format and can be used in following scenarios:

- to access data and parameters and export later as described in this topic
- to work with annotations, see Add and Save Annotations.
- to merge reports, see Insert or Add Report Pages.

Save a report as a static RDF file

1. Double-click the title bar of the Windows Form to create an event-handling method for the Form_Load event.
2. Add the following code to the handler to save the report.

   To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form_Load event.

```
Dim rpt As New YourReportName()
rpt.Run()
rpt.Document.Save(Application.StartupPath + \NewRDF.RDF)
```

To write the code in C#

C# code. Paste INSIDE the Form_Load event.

```
YourReportName rpt = new YourReportName();
rpt.Run();
rpt.Document.Save(Application.StartupPath + \\NewRDF.RDF);
```

Load a saved RDF file into the ActiveReports viewer

1. Double-click the title bar of the Windows Form to create an event-handling method for the Form_Load event.
2. Add the following code to the handler to load the saved report.

To write the code in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form_Load event.

```
Viewer1.Document.Load("Location of the .RDF File")
```

To write the code in C#

C# code. Paste INSIDE the Form_Load event.

```
viewer1.Document.Load(@"Location of the .RDF File");
```

Note: The Windows Form Viewer can display RDF files made with any version of ActiveReports, including COM versions.

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 the following 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 YourReportName()
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();
YourReportName rpt = new YourReportName();
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);
```

Add and Save Annotations

In a Section Report, you can save a report containing annotations along with the report data into an RDF file. You can also add annotations at run time. The following steps demonstrate how to accomplish these tasks in code.

These steps assume that you have already added a Section Report (code based) template in a Visual Studio project.

Save annotations

The following example shows how to add a **Save Annotated Report** button to the viewer and save a report with annotations in RDF.

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. See Save and Load RDF Files 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");
```

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 GrapeCity.ActiveReports.Document.Section.Annotations
```

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

```
Dim rpt As New SectionReport1
'Run the report first.
rpt.Run()

'Assign the viewer.
Me.Viewer1.LoadDocument(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 GrapeCity.ActiveReports.Document.Section.Annotations;
```

C# code. Paste INSIDE the Form Load event.

```
SectionReport1 rpt = new SectionReport1();
//Run the report first.
rpt.Run();

//Assign the viewer
this.viewer1.LoadDocument(rpt.Document);
```

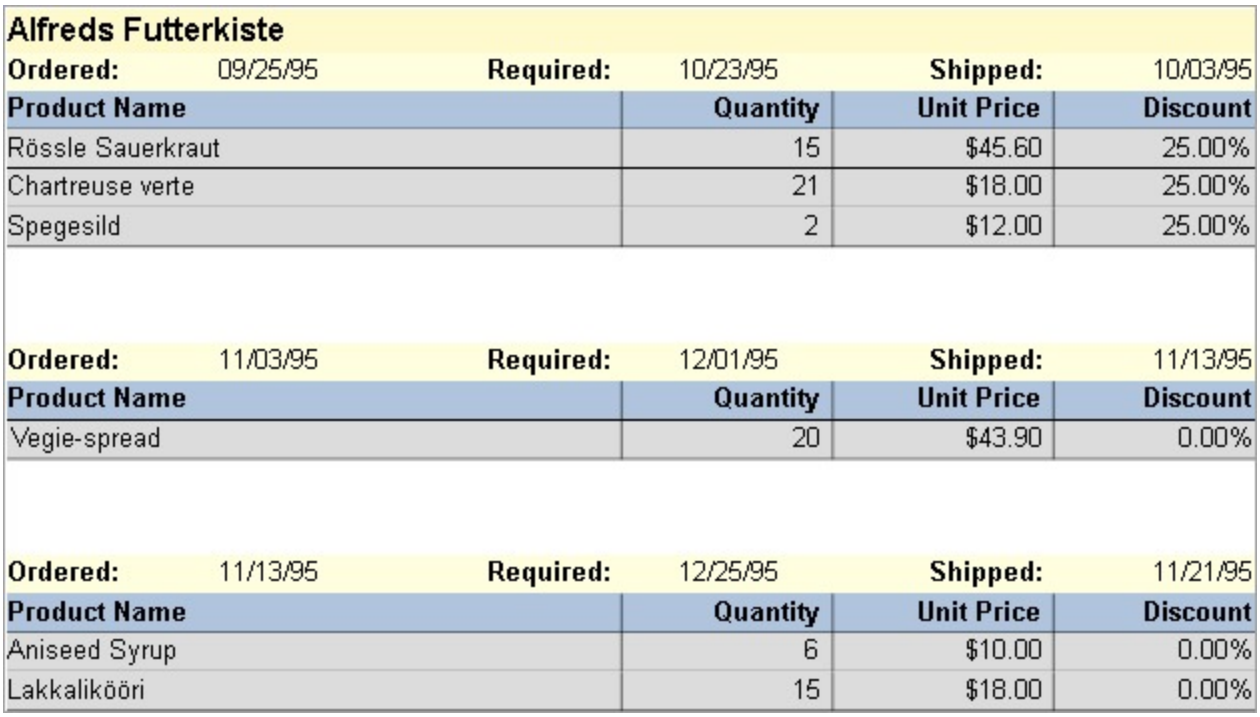
```
//Create an annotation and assign property values.
AnnotationCircle circle = new AnnotationCircle();
circle.Color = System.Drawing.Color.GreenYellow;
circle.Border.Color = System.Drawing.Color.Chartreuse;

//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");
```

Work with Subreports

This tutorial illustrates creating a subreport using scripts.
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.

> **Note:** We are using Visual Studio 2019 and will be connecting to Microsoft Jet 4.0 OLE DB Provider. Since Visual Studio 2022 runs in 64 bit, it does not show Microsoft Jet 4.0 OLE DB provider; you should use one of the compatible providers (e.g. Microsoft.ACE.OLEDB.12.0).
>
> This walkthrough uses the Northwind database. The NWIND.mdb file can be downloaded from GitHub: ..\Samples18\Data\NWIND.mdb.

When you have finished this walkthrough, you will have main report 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	25.00%
Chartreuse verte			21	$18.00	25.00%
Spegesild			2	$12.00	25.00%
Ordered:	11/03/95	**Required:**	12/01/95	**Shipped:**	11/13/95
Product Name			**Quantity**	**Unit Price**	**Discount**
Vegie-spread			20	$43.90	0.00%
Ordered:	11/13/95	**Required:**	12/25/95	**Shipped:**	11/21/95
Product Name			**Quantity**	**Unit Price**	**Discount**
Aniseed Syrup			6	$10.00	0.00%
Lakkalikööri			15	$18.00	0.00%

Create Project and the Main Report

1. Create a new Visual Studio project.
2. Select the **ActiveReports 18 Section Report Application (xml-based)** and click **Next**.
3. Fill the **Project name**, **Location**, and **Framework** fields and click **Create**.
4. Rename the default report SectionReport1.rpx as 'rptMain'.
5. Double-click the rpx file to open the Section Report in the designer.

OR

1. In the existing Visual Studio project, from the **Project** menu, select **Add** > **New Item**.
2. In the **Add New Item** dialog, select **ActiveReports 18 Section Report (xml-based)** and in the Name field, rename the file as 'rptMain'.
3. Click the **Add** button to open a new Section Report in the designer.

Connect the Main Report to Data

1. In the Report Data Source dialog, on the **OLE DB** tab, next to Connection String, click the Build button.
2. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button to move to the Connection tab.
   Note:
3. Click the ellipsis (...) button to browse to your database, for example the NWind.mdb sample database. Click **Open** once you have selected the appropriate database path.
4. Click the **Test Connection** button to see if you have successfully connected to the database.
5. Click **OK** to close the Data Link Properties window and return to the Report Data Source dialog. Notice that the Connection String field gets filled automatically.
6. In the **Query** field on the **OLE DB** tab, enter the following SQL query.

```
SQL Query
```

```
SELECT * FROM Orders INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID
ORDER BY CompanyName, OrderDate
```

7. Click **OK** to save the data source and return to the report design surface.

Add a Subreport

1. From the **Project** menu, select **Add New Item**.
2. In the Add New Item dialog that appears, select **ActiveReports 18 Section report (xml-based)** and in the Name field, rename the file as **rptSub**.
3. Click the **Add** button to open a new Section Report in the designer.
4. Right-click the PageHeader or PageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
5. 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.

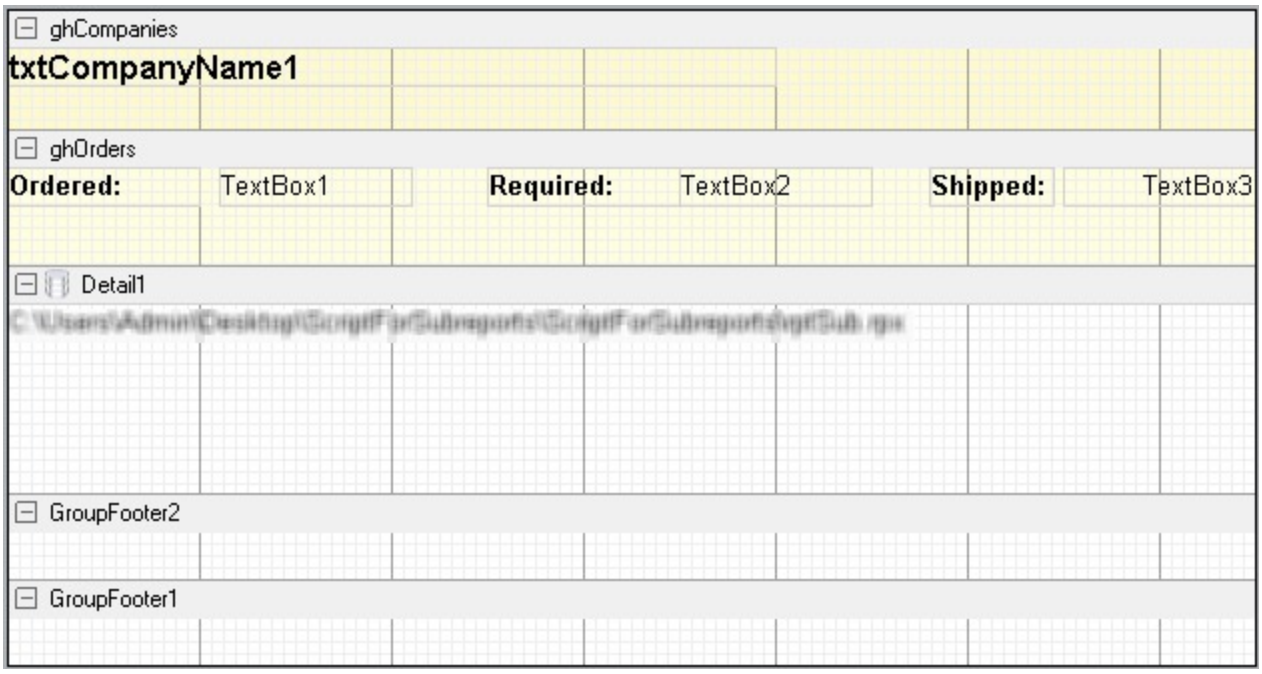
Connect the Subreport to Data

1. In the Report Data Source dialog, on the **OLE DB** tab, next to Connection String, click the Build button.
2. In the Data Link Properties window that appears, select **Microsoft Jet 4.0 OLE DB Provider** and click the **Next** button to move to the Connection tab.
3. Click the ellipsis (...) button to browse to your database, for example the NWind.mdb sample database. Click **Open** once you have selected the appropriate database path.
4. Click the **Test Connection** button to see if you have successfully connected to the database.
5. Click **OK** to close the Data Link Properties window and return to the Report Data Source dialog. Notice that the Connection String field gets filled automatically.
6. In the **Query** field on the **OLE DB** tab, enter the following SQL query.

SQL Query
`SELECT * FROM [order details] inner join products on [order details].productid = products.productid`

7. Click **OK** to save the data source and return to the report design surface.

Create Layout for Main Report



1. Right-click 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 Panel, make the following changes to the group header.

Property Name	Property Value
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 **CompanyName** field onto ghCompanies and in the Properties window, set the properties as follows.

Property Name	Property Value
Size	4, 0.2 in
Location	0, 0 in
Font Bold	True
Font Size	12

4. Right-click the design surface of rptMain and select **Insert** then **Group Header/Footer** to add the second group header and footer sections to the report.
5. In the Properties Panel, make the following changes to the second group header.

Property Name	Property Value
Name	ghOrders

BackColor	LightYellow
CanShrink	True
DataField	OrderDate
GroupKeepTogether	All
KeepTogether	True

6. From the toolbox, drag three TextBox controls onto **ghOrders** and set the properties for each control as follows.

TextBox1

Property Name	Property Value
DataField	OrderDate
OutputFormat	MM/dd/yy

TextBox2

Property Name	Property Value
DataField	RequiredDate
OutputFormat	MM/dd/yy

TextBox3

Property Name	Property Value
DataField	ShippedDate
OutputFormat	MM/dd/yy
Alignment	Right

7. From the toolbox, drag three Label controls onto **ghOrders** and set the properties for each control as follows.

Label1

Property Name	Property Value
Text	Ordered:
Font	Bold:True

Label2

Property Name	Property Value
Text	Required:
Font	Bold:True

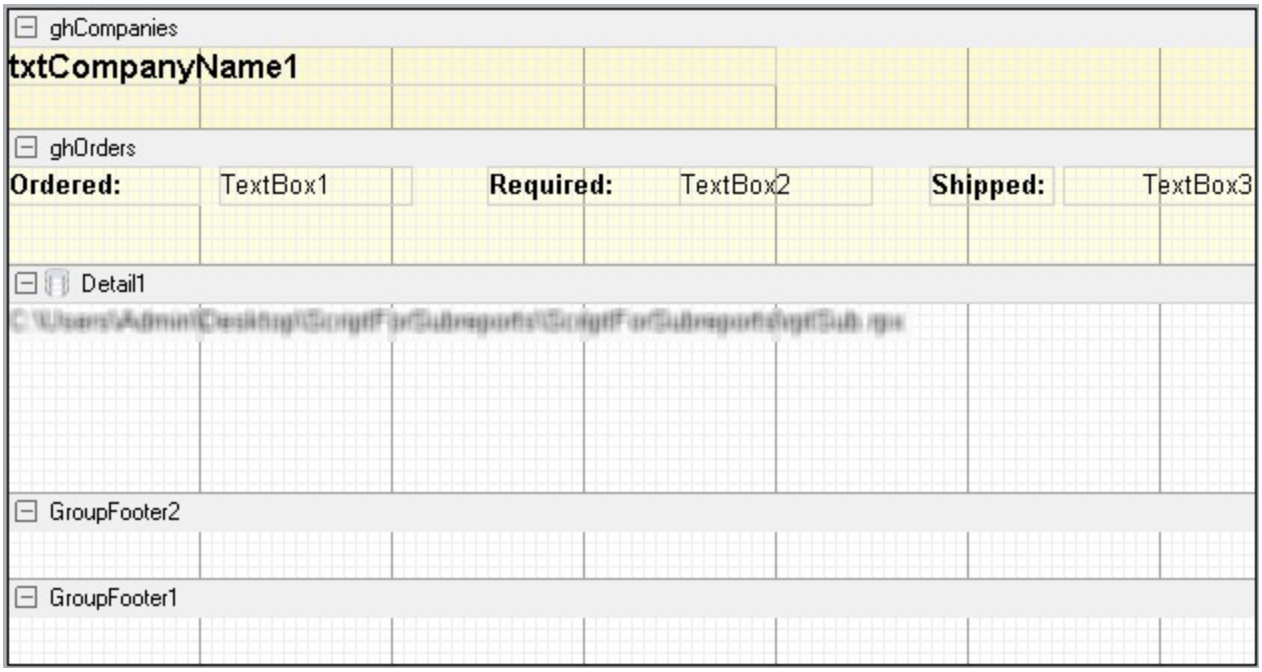
Label3

Property Name	Property Value
Text	Shipped:
Font	Bold:True

8. Select the Detail section and in the Properties window, set the **CanShrink** property to **True**.
9. From the toolbox, drag the Subreport control onto the Detail section and in the Properties window, set the properties as follows.

Property Name	Property Value
ReportName	***full project path***\rptSub.rpx
Name	SubReport1

Create a Layout for the SubReport



1. Right-click the design surface of rptSub 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.

Property Name	Property Value
Name	ghOrderDetails
BackColor	LightSteelBlue
CanShrink	True
DataField	OrderID

3. From the toolbox, drag four label controls to **ghOrderDetails** and set the properties for each label as follows.

Label1

Property Name	Property Value
Text	Product Name
Font	Bold:True
Alignment	Left

Label2

Property Name	Property Value
Text	Quantity
Font	Bold:True
Alignment	Right

Label3

Property Name	Property Value
Text	Unit Price
Font	Bold:True
Alignment	Right

Label4

Property Name	Property Value
Text	Discount
Font	Bold:True
Alignment	Right

4. Click the **Detail** section and in the Properties window, set the following properties.

Property Name	Property Value
BackColor	Gainsboro
CanShrink	True

5. From the toolbox, drag four TextBox controls onto the Detail section and set the properties as follows.

TextBox1

Property Name	Property Value
DataField	ProductName
Alignment	Left

TextBox2

Property Name	Property Value
DataField	Quantity
Alignment	Right

TextBox3

Property Name	Property Value
DataField	Products.UnitPrice
Alignment	Right
OutputFormat	Currency

TextBox4

Property Name	Property Value
DataField	Discount
Alignment	Right
OutputFormat	Percentage

Embed script in the main report (rptMain)

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.

Visual Basic.NET script. Paste in the script editor window.

```
Dim rptSub As GrapeCity.ActiveReports.SectionReport
Sub ActiveReport_ReportStart
  'Create a new instance of the generic report
  rptSub = new GrapeCity.ActiveReports.SectionReport()
  'Load the rpx file into the generic report
  rptSub.LoadLayout(me.SubReport1.ReportName)
  'Connect data to the main report
  Dim connString As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=[User
Folder]\Samples18\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 GrapeCity.ActiveReports.Data.OleDBDataSource()
  ds.ConnectionString = connString
  ds.SQL = sqlString
  rpt.DataSource = ds
End Sub
Sub Detail_Format
```

```
  Dim rptSubCtl As GrapeCity.ActiveReports.SubReport = me.SubReport1
  Dim childDataSource As New GrapeCity.ActiveReports.Data.OleDBDataSource()
  childDataSource.ConnectionString = CType(rpt.DataSource,
GrapeCity.ActiveReports.Data.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
```

C# code. Paste in the script editor window.

```
GrapeCity.ActiveReports.SectionReport rptSub;
public void Detail_Format()
{
  GrapeCity.ActiveReports.SectionReportModel.SubReport rptSubCtl = this.SubReport1;
  GrapeCity.ActiveReports.Data.OleDBDataSource childDataSource = new
GrapeCity.ActiveReports.Data.OleDBDataSource();
  childDataSource.ConnectionString = ((GrapeCity.ActiveReports.Data.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 GrapeCity.ActiveReports.SectionReport();
  //Load the rpx file into the generic report
  rptSub.LoadLayout(this.SubReport1.ReportName);
  //Connect data to the main report
  string connString = @"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=[User
Folder]\Samples18\Data\NWIND.mdb;Persist Security Info=False";
  string sqlString = "Select * from orders inner join customers on orders.customerid =
customers.customerid order by CompanyName,OrderDate";
  GrapeCity.ActiveReports.Data.OleDBDataSource ds = new
GrapeCity.ActiveReports.Data.OleDBDataSource();
  ds.ConnectionString = connString;
  ds.SQL = sqlString;
  rpt.DataSource = ds;
}
```

Preview report

Click the preview tab to view the report at design time.

OR
Open the report in the Viewer. See Save and Load Section Reports for further information on how to load the xml-based Section Report onto the viewer.

Code-based Section reports

This section covers the following features that are specific to Code-based Section Reports.

Inherit a Report Template

Design Code-based Sections Reports in .NET Core

Load an RTF or HTML File in RichTextBox at Runtime

Use Custom Controls on Reports

Add Code to Layouts Using Script

Subreport in Section Reports

Create Mail Merge with RichTextBox

Overlaying in Reports (Letterhead)

Inherit a Report Template

In a section layout, you can create a base report as a template from which other reports can inherit. This behavior is similar to creating a master report and is available in a Section Report (code-based) layout.

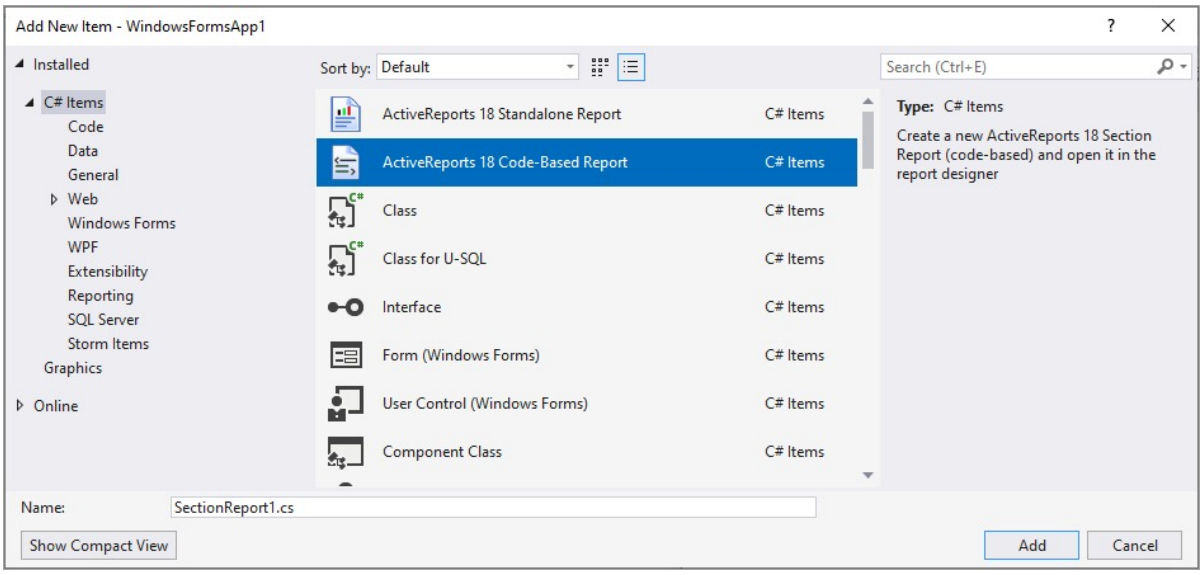
Inheriting reports is useful when multiple reports share common features, such as identical page headers and footers. Instead of recreating the look every time, create template headers and footers once and use inheritance to apply them to other reports.

Use the following instructions to create a base report and inherit it in other reports.

> ⚠ **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 you change the duplicate names.

Create a base report

1. From the Visual Studio **File** menu, select **New**, then **Project**.
2. In the **Create New Project** dialog that appears, select Windows Forms App (.NET Framework) and click **Next**.
3. In the **Configure your new project** dialog, type a name for your project, set **Framework** to .NET Framework 4.7 and click **Create.**
4. Add a code-based Section Report by selecting **ActiveReports 18 Code-Based Report** in **Project** > **Add New Item** and naming it **rptLetterhead.**

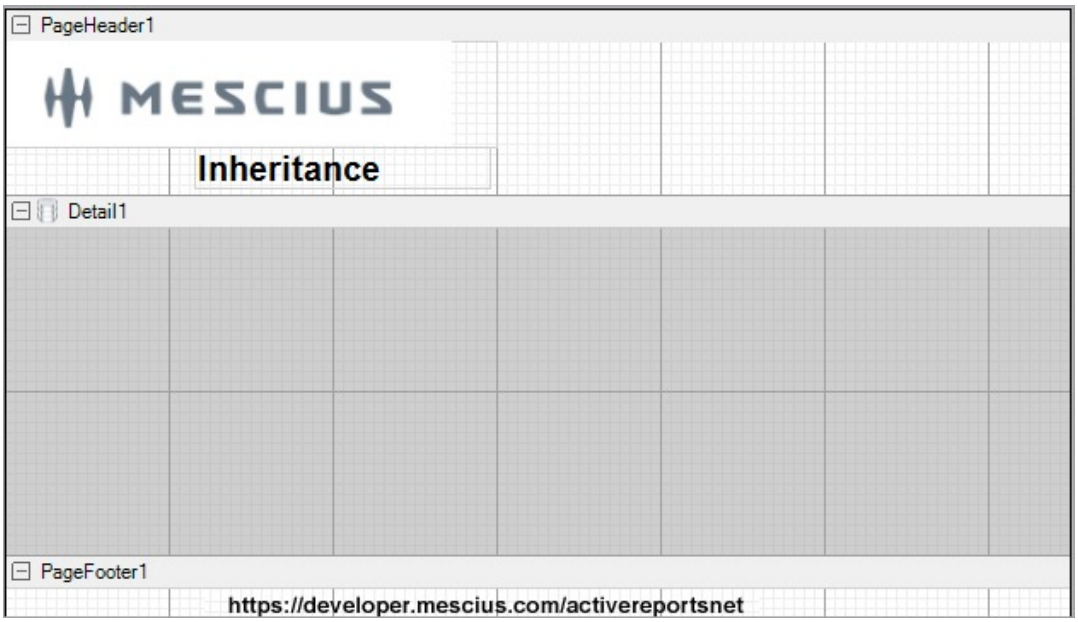


5. In the report template that appears, add the following controls from the Visual Studio toolbox to the indicated section of rptLetterhead and set the properties.

Control	Section	Location	Size	Miscellaneous
Picture	PageHeader	0, 0 in	3, 0.65 in	Image = (click ellipsis and navigate to the location of your image file) PictureAlignment = TopLeft
Label	PageHeader	1.16, 0.65 in	1.8, 0.25 in	Text = Inheritance Font = Arial, 15pt, style=Bold
Label	PageFooter	0, 0 in	6.5, 0.19 in	Text = https://developer.mescius.com/activereportsnet HyperLink = https://developer.mescius.com/activereportsnet Font/Bold = True Alignment = Center

6. Right-click the gray area below the design surface and choose properties, to open the Properties window.
7. In the Properties window, set the **MasterReport** property to **True**. Setting the MasterReport property to True locks the Detail section.

> ⚠ **Caution:** Do not set the **MasterReport** property to **True** until you have finished designing or making changes to the report. Setting this property to True triggers major changes in the designer file of the report.



You can use the Page Header and Page Footer sections to design the base report. When you create reports that inherit the layout from this base report, only the detail section is available for editing.

Inherit layout from a base report

These steps assume that you have already added another Section Report (code-based) template. This report functions like a content report where you can create the layout of the Detail section.

1. In a Visual Studio project, add a Section Report by selecting **ActiveReports 18 Code-Based Report** in **Project** > **Add New Item** and naming it **rptLetter**.
2. In the Solution Explorer, right-click the new report and select the **View Code** option to open the code behind of the report.
3. In the code view, modify the inheritance statement as shown below. The content report inherits from the base report instead of **GrapeCity.ActiveReports.SectionReport**.

   > ⚠ **Caution**: The existing report layout in the content report is lost once you inherit the base report. Even if you change it back to GrapeCity.ActiveReports.SectionReport, the original layout in content report will not be retrieved.

   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
   ```

4. Close the reports and from the Build menu on the Visual Studio menu bar, select **Rebuild**. When you reopen the report, the inherited sections and controls are disabled.

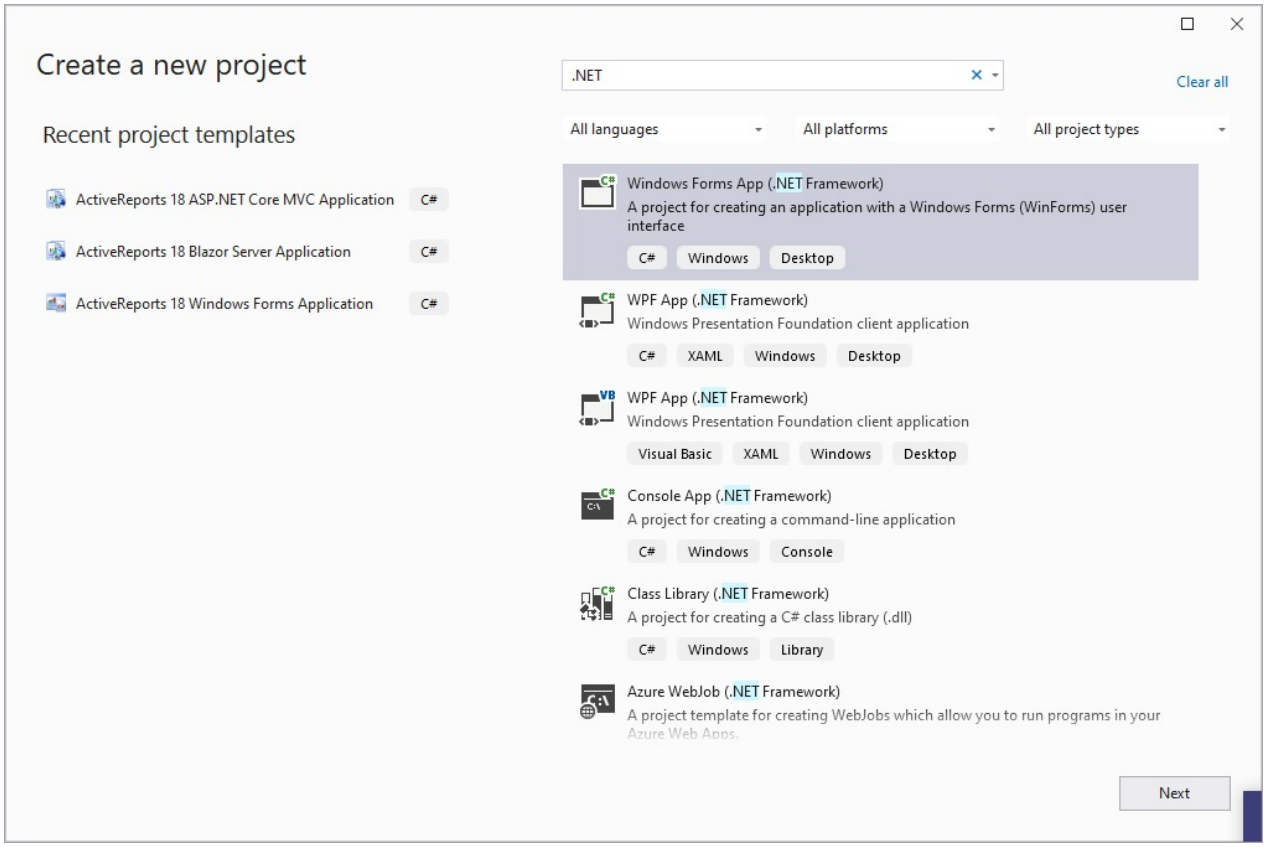
   > **Note:** To apply further changes from the base report to the content report, you might have to rebuild the project again.

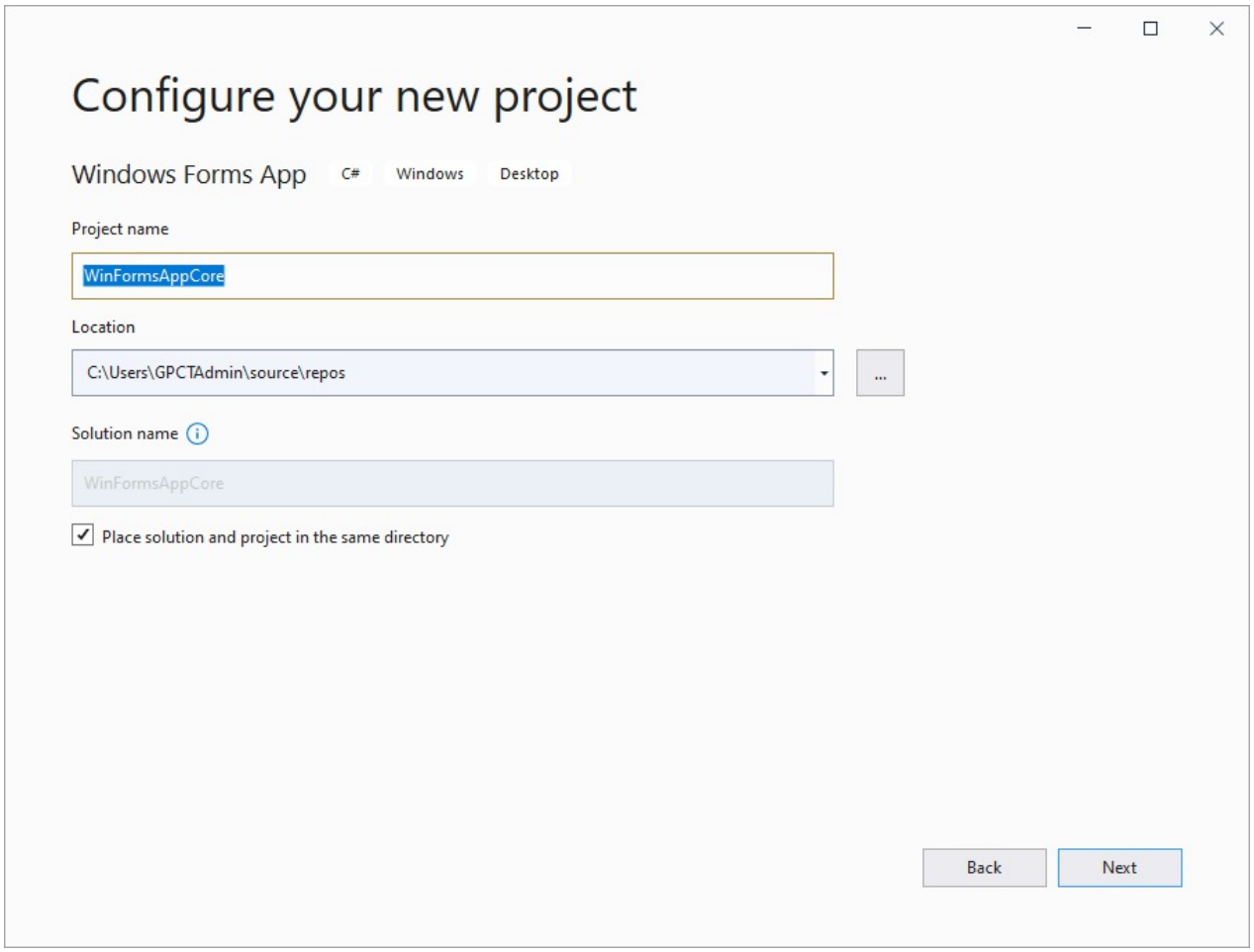
Design Code-based Section Reports in .NET Core

The existing limitation in .NET Core 3.1/.NET 5.0/.NET 6.0/.NET 7.0 does not allow the ActiveReports Integrated Designer to be used for designing Code-based Section Reports in WinForms applications in Visual Studio.
As workaround, you need to use Visual Studio's option to link report files from .NET Core project in the .NET Framework project, and use the .NET Framework WinForms Designer.

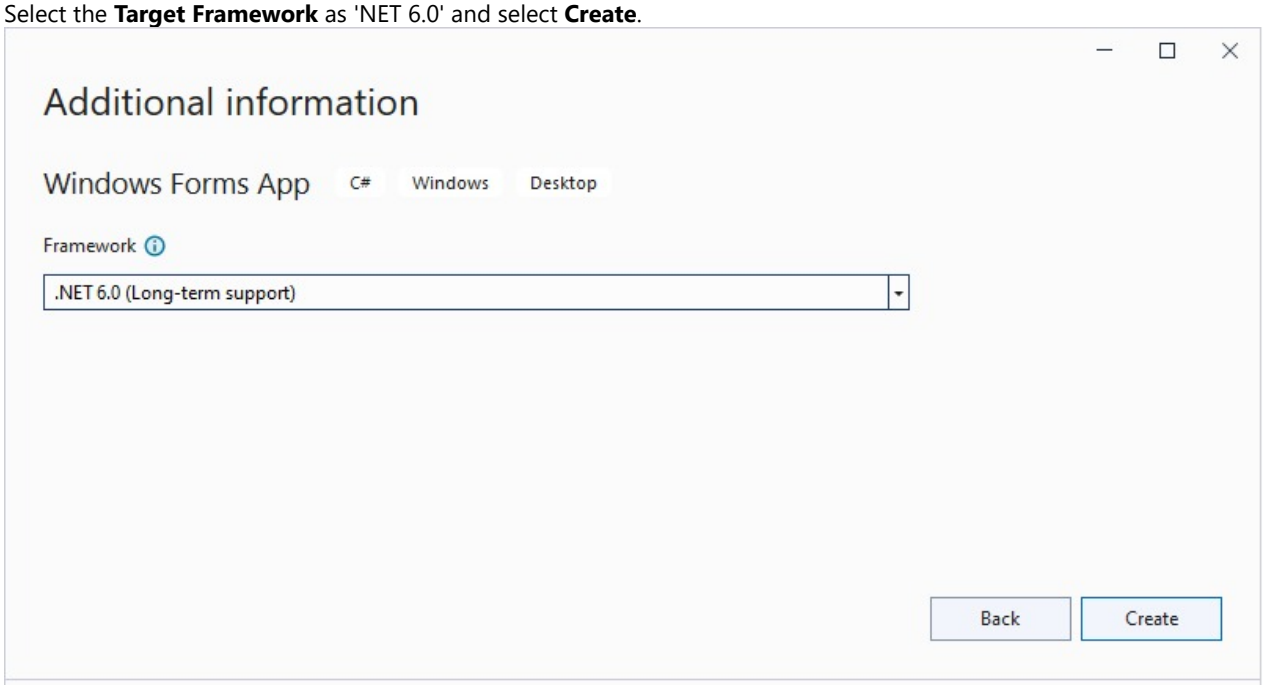
The steps to enable design-time report creation in .NET Core project for Code-Based Section Reports are as follows:

1. **Create a new Windows Forms .NET Core project.**

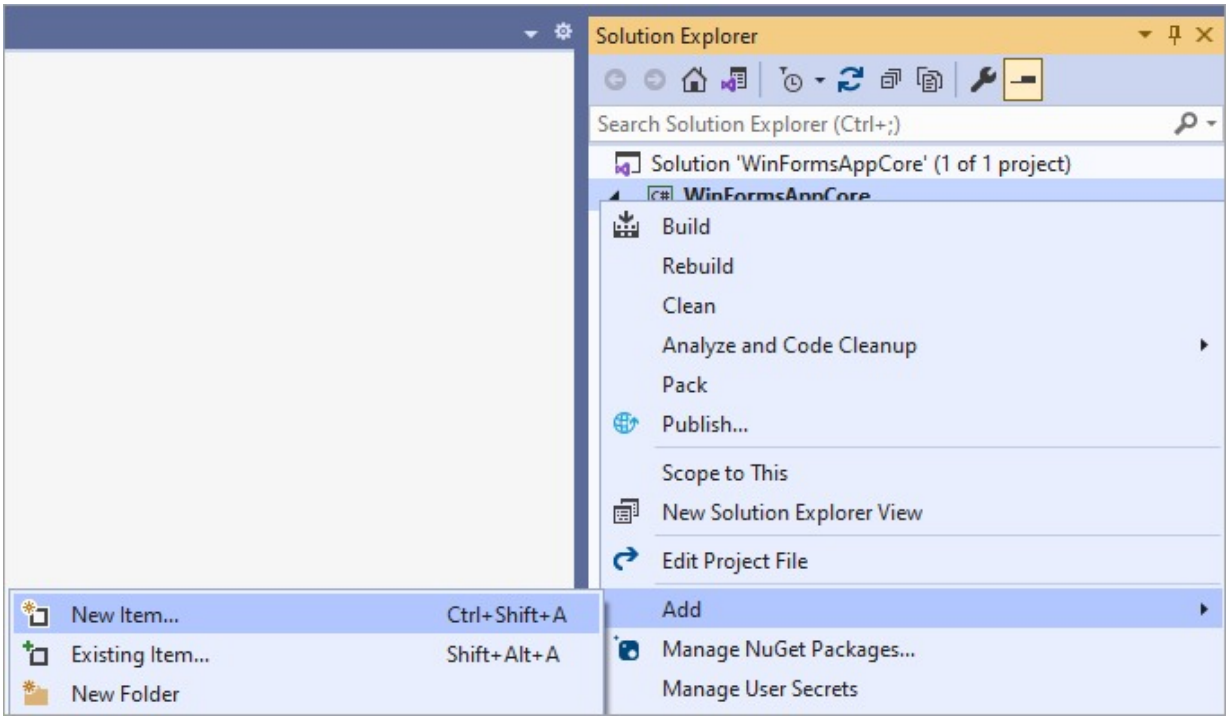




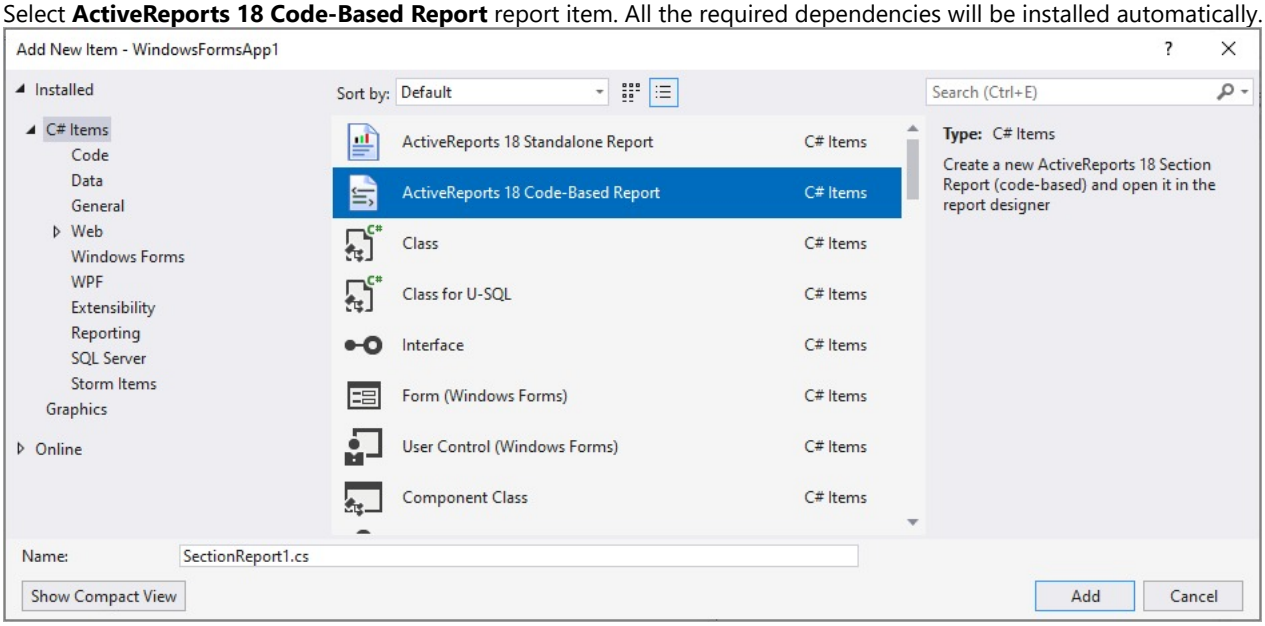
2. Select the **Target Framework** as 'NET 6.0' and select **Create**.



3. Add the code-based Section Report item to the .NET Core project. To do so, right-click the project and go to **Add > New Item**.

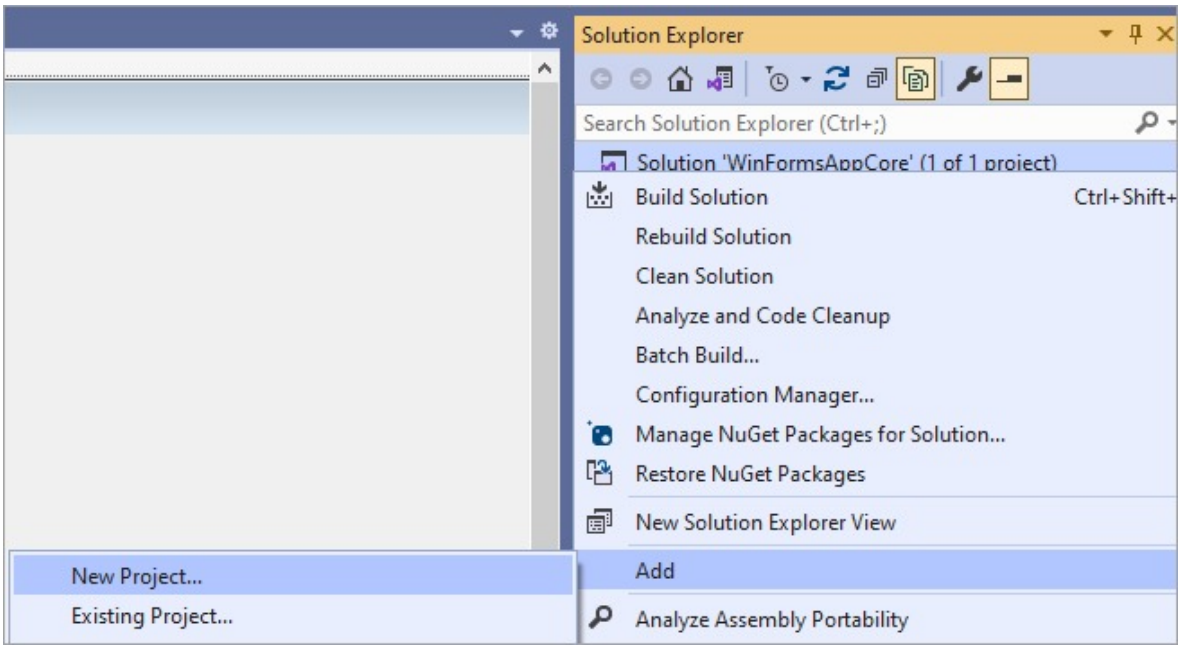


4. Select **ActiveReports 18 Code-Based Report** report item. All the required dependencies will be installed automatically.

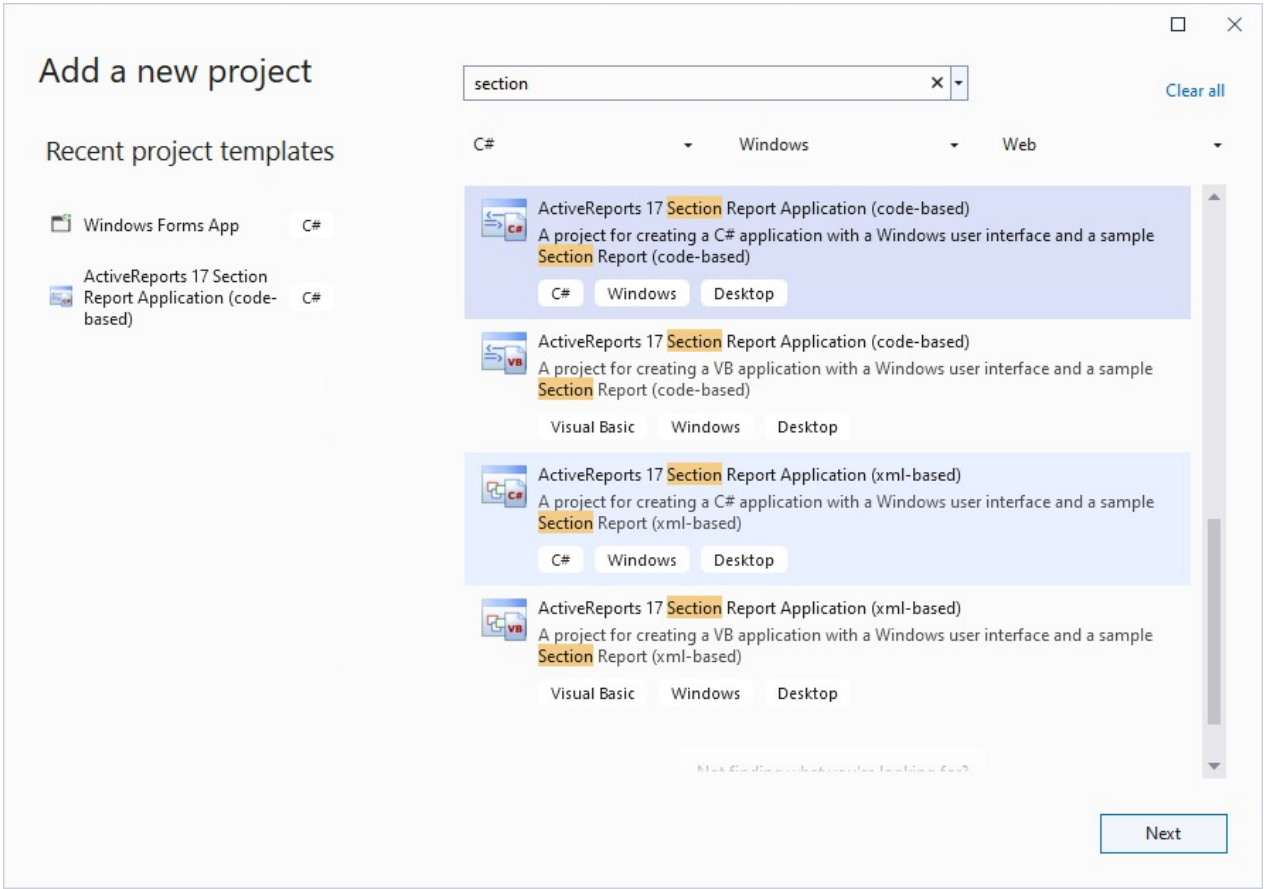


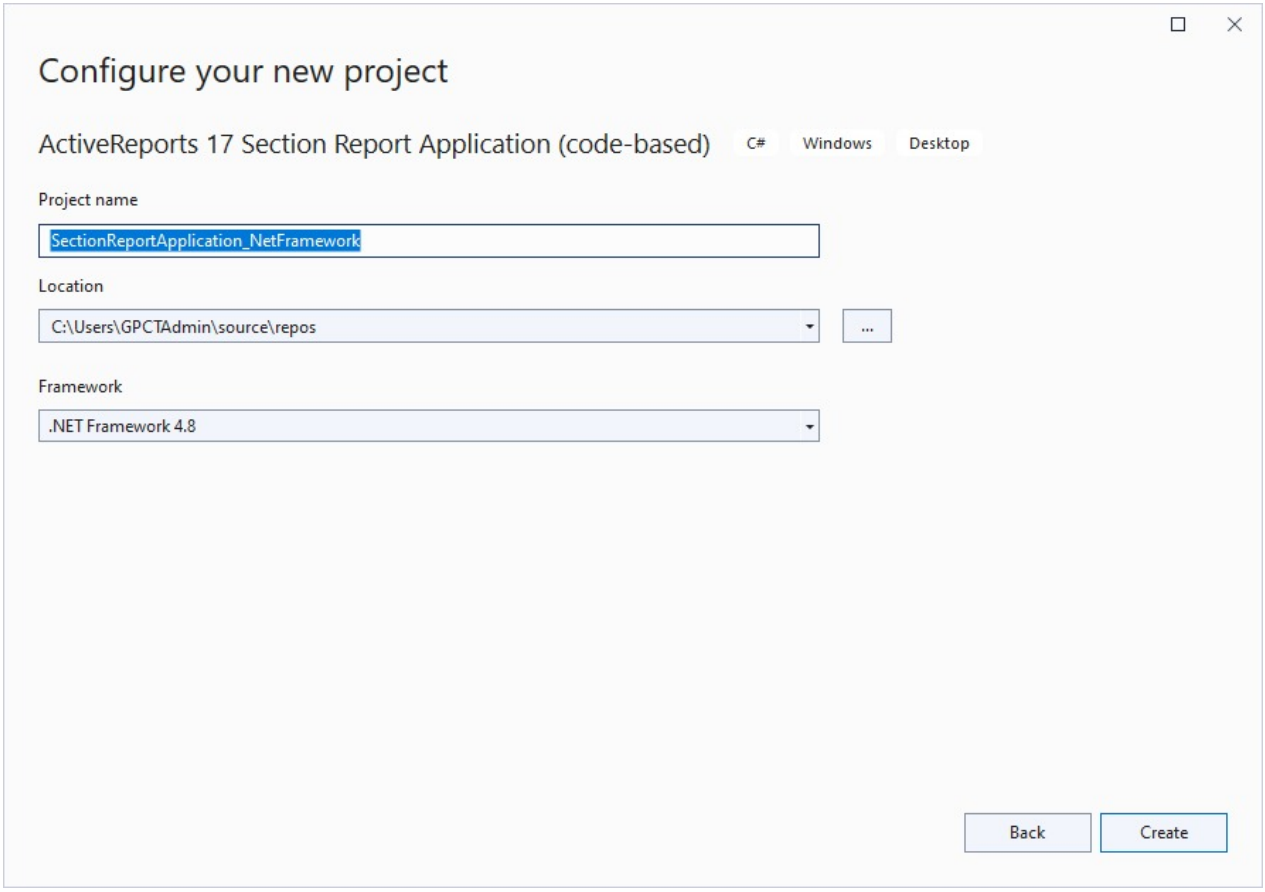
5. **Add a new .NET Framework project using built-in Section Report (code-based) template.**

    1. In the Solution Explorer, right-click the solution node and go to **Add** > **New Project**.

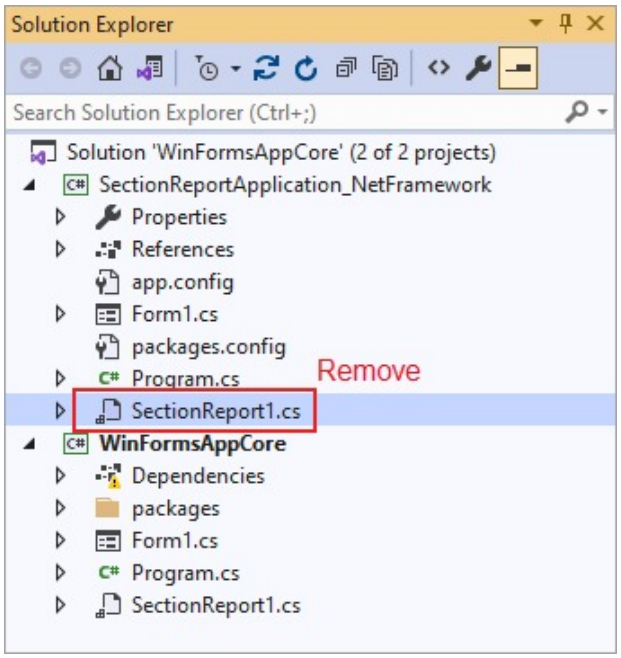


2. Select **ActiveReports 18 Section report Application (code-based)** template and configure the project for the target .NET Framework version.



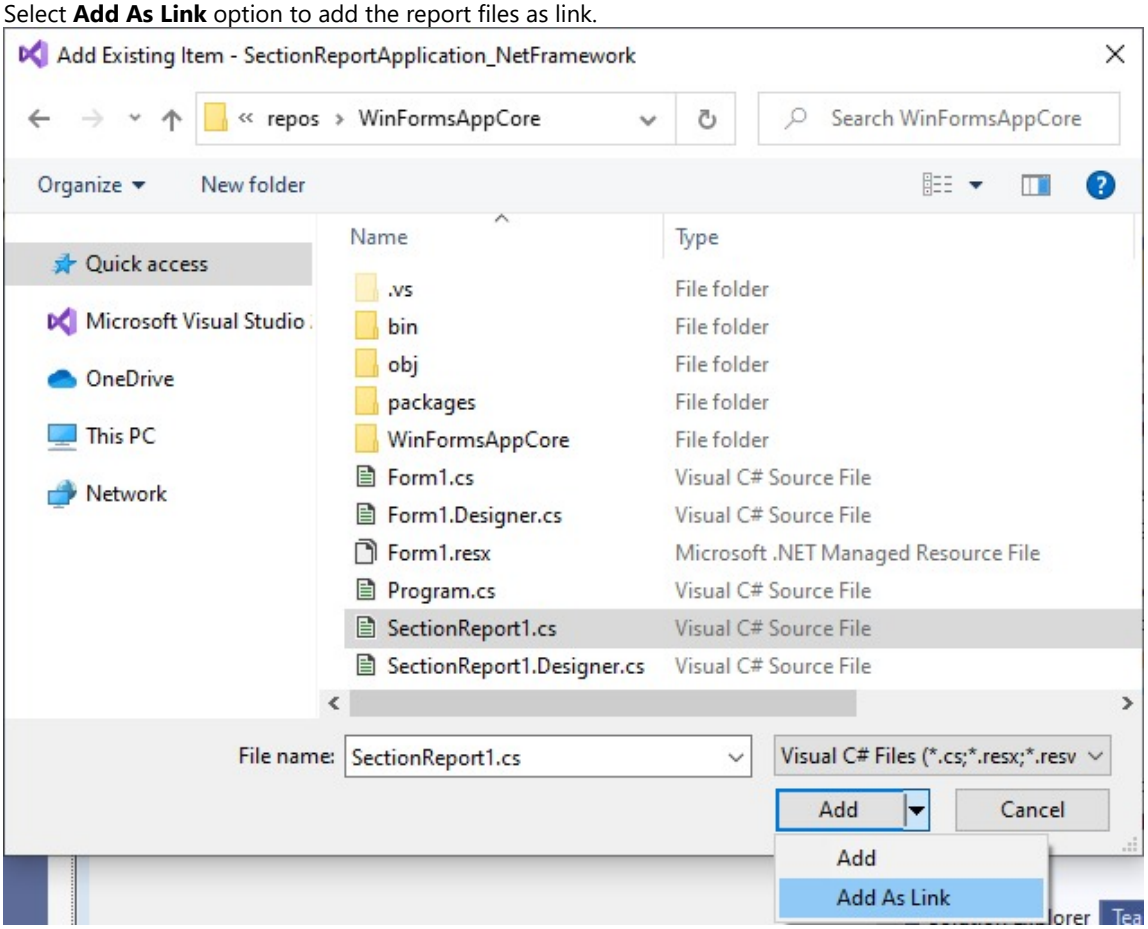


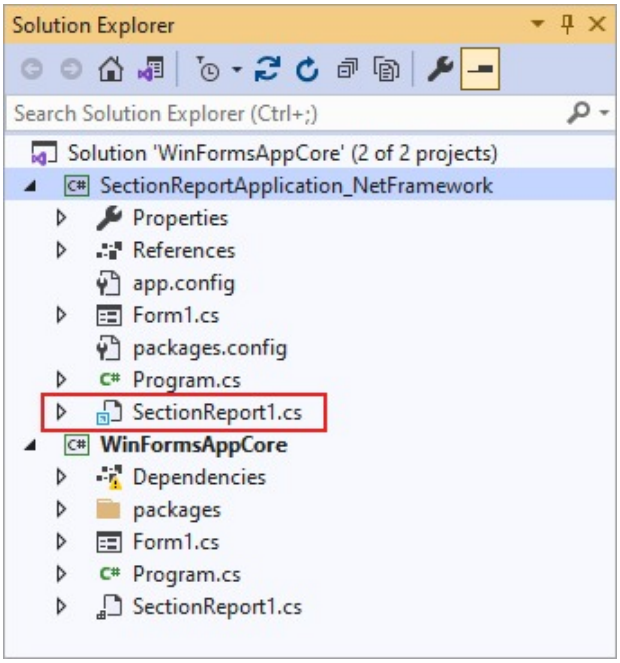
6. Remove the code-based report (SectionReport1.cs) from the .NET Framework project, which was added automatically in the previous step.



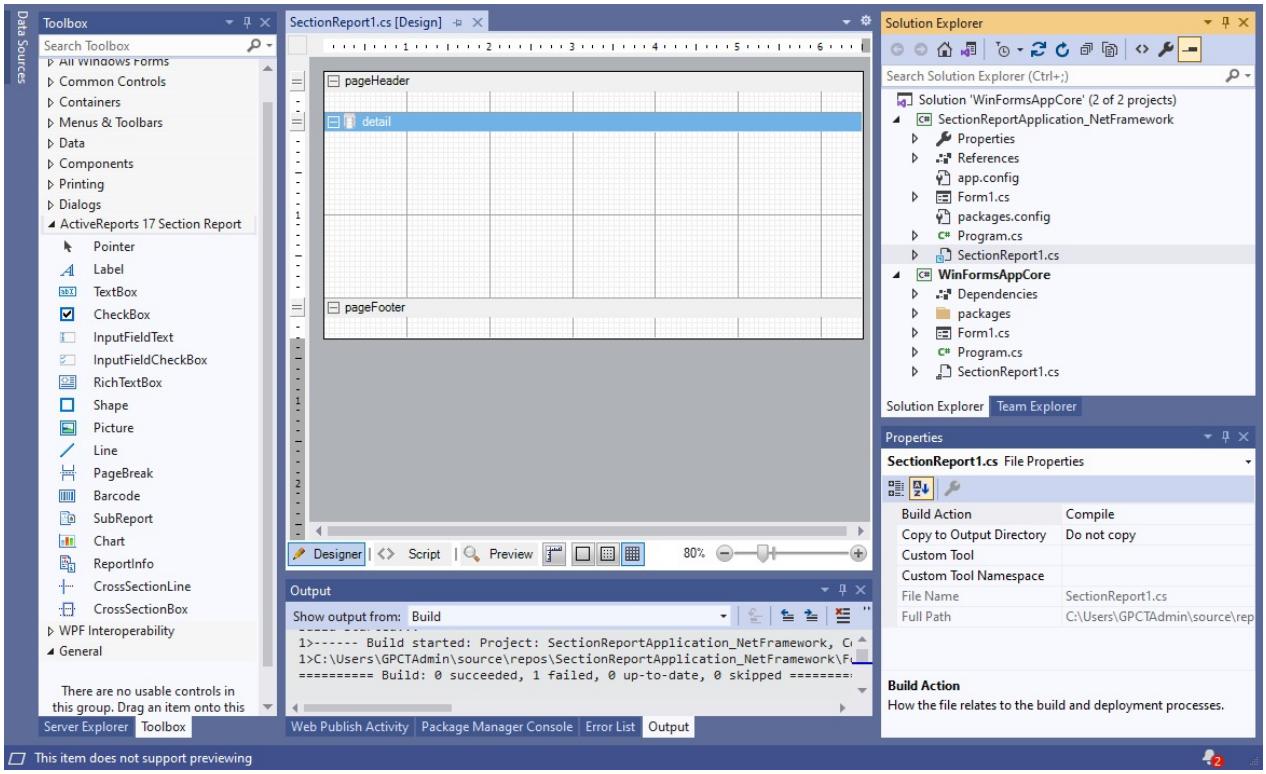
7. **Add report from the .NET Core project to the .NET Framework project as link.**

1. In the Solution Explorer, right-click the .NET Framework project, go to **Add** > **Add Existing Item**.
2. Navigate to the .NET Core project and select report file: 'SectionReport1.cs'.
3. Select **Add As Link** option to add the report files as link.





8. Double-click the linked SectionReport1.cs (in .Net Framework project) to open the ActiveReports Integrated Designer for the report.



9. Now design the report. You will observe that the modification of a report in .NET Framework project leads to modification of the report in the .NET Core project.

Load an RTF or HTML File in RichTextBox at Runtime

In a section layout, you can load an RTF or an HTML file into the RichTextBox control both at design time and at run time. Following is a step-by-step process that helps you load these files into the RichTextBox control at run-time.

These steps assume that you have already added a Section Report (code based) template in a Visual Studio project and have placed a RichTextBox control inside its detail section.

> ⚠ **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 already in use by that first iteration and returns "file in use" errors when that section is processed again.

Write an RTF file to load into a RichTextBox control

1. Open WordPad, and paste a formatted text into it, for example:

   [Paste the following section 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

- Familia Arquibaldo
- Wellington Improtadora
- Que Delícia
- Tradição Hipermercados
- Ricardo Adocicados
- Hanari Carnes
- Queen Cozinha
- Comércio Mineiro
- Gourmet Lanchonetes

- -

2. Save the file as **sample.rtf**.

Note: The RichTextBox control is limited in its support for advanced RTF features such as the ones supported by Microsoft Word. In general, the features supported by WordPad are supported in this control.

Load an RTF file into a RichTextBox control

Note: The RichTextBox control has limited support for advanced RTF features such as the ones supported by Microsoft Word. Therefore, use a WordPad for obtaining best results.

These steps assume that the RTF file (for example, sample.rtf) to load has been saved in the bin/debug directory of your project.

1. Right-click the report and select View Code to open the code view.
2. Add an Imports (Visual Basic.NET) or using (C#) statement at the top of the code view for the **GrapeCity.ActiveReports.SectionReportModel ('GrapeCity.ActiveReports.SectionReportModel Namespace' in the on-line documentation)** namespace.
3. In the design view, double-click the detail section of the report to create an event-handling method for the Detail Format event.
4. Add the following code to the handler to load the RTF file into the RichTextBox control.

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

Visual Basic.NET code. Paste INSIDE the Detail1_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)
```

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);
```

Write an HTML file to load into a RichTextBox control

1. Open a Notepad, and paste an HTML code into it, for example:

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>Brazi1>
<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>
</body>
</html>
```

2. Save the file as **sample.html**.

Load an HTML file into the RichTextBox control

These steps assume that the HTML file (for example, sample.html) to load has been saved in the bin/debug directory of your project.

1. Right-click the report and select View Code to open the code view.
2. Add an Imports (Visual Basic.NET) or using (C#) statement at the top of the code view for the **GrapeCity.ActiveReports.SectionReportModel ('GrapeCity.ActiveReports.SectionReportModel Namespace' in the on-line documentation)** namespace.
3. In the design view, double-click the detail section of the report to create an event-handling method for the Detail Format event.
4. Add code to the handler to load the HTML file into the RichText control.

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

Visual Basic.NET code. Paste INSIDE the Detail1_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)
```

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.richTextBox1.Load(streamHTML, RichTextType.Html);
```

Use Custom Controls on Reports

In a Section Report, ActiveReports allows you to drop a third party control onto the report design surface where it is recognized as a custom control. You can access its properties using type casting.

In the following steps, we use hidden textbox controls to populate a Visual Studio TreeView control. These steps assume that you have already added a code-based Section Report template in a Visual Studio project.

Add the TreeView control to a report

1. From the Visual Studio toolbox **Common Controls** tab, drag and drop a **TreeView** control onto the detail section of a report.

2. Notice that in the Properties window, the control is called **CustomControl1**.

Add data and hidden TextBox controls to the report

1. Connect the report to the JSON data Orders.
2. From the Report Explorer, drag and drop the following fields onto the detail section of the report:
   - ShipCountry
   - ShipCity
   - CustomerId
   - EmployeeId
3. On the design surface, select all four TextBox controls, and in the Properties window, change their **Visible** property to **False**.

Create a function to add nodes to the TreeView control

1. Right-click the design surface and select **View Code** to see the code view for the report.
2. Add the following code inside the report class to add a function to the report for adding nodes to the TreeView control.

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

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#. Paste INSIDE the report class.

```
private TreeNode AddNodeToTreeView(TreeNodeCollection colNodes, string sText)
{
    TreeNode objTreeNode;
    objTreeNode = new TreeNode(sText);
    colNodes.Add(objTreeNode);
    return objTreeNode;
}
```

Access the TreeView control properties in code and assign data

1. On the report design surface, double-click the detail section to create an event-handling method for the **Detail_Format** event.
2. Add the following code to the handler to access the **TreeView** properties and assign data from the hidden TextBox controls.

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

To write the code in Visual Basic.NET

```
'Type cast the custom control as a TreeView
Dim TreeView1 As New TreeView
TreeView1 = 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(TreeView1.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(TreeView1.Nodes, Me.txtCustomerId1.Text)
AddNodeToTreeView(objCustomerTreeNode.Nodes, Me.txtEmployeeId1.Text)
objCustomerTreeNode.Expand()
```

To write the code in Visual Basic.NET

```
//Type cast the custom control as a TreeView TreeView TreeView1 = new
TreeView();TreeView1 = (TreeView)this.customControl1.Control;
//Create a tree node TreeNode objCountryTreeNode;//Assign the text from a hidden textbox
to the node objCountryTreeNode = AddNodeToTreeView(TreeView1.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(TreeView1.Nodes,
this.txtCustomerId1.Text);AddNodeToTreeView(objCustomerTreeNode.Nodes,
this.txtEmployeeId1.Text);objCustomerTreeNode.Expand();
```

Load the report in the Viewer, change the Form Load event

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

```
Dim ar = New SectionReport1()
 ar.Run(False)
 viewer1.LoadDocument(ar.Document)
```

C# code. Paste INSIDE the Form Load event

```
var ar = new SectionReport1();
 ar.Run(false);
 viewer1.LoadDocument(ar.Document);
```

Create Custom Chart Control

This is an internal control that allows using third-party WinForms controls inside a Section report (mainly, a code-based

Section report). The main target area of this control is using scripts.

CustomControl is not supported in:

- Section reports in the CrossPlatform compatibility mode
- WebDesigner

Using a CustomControl to Create Chart Control

1. In Microsoft Visual Studio 2022 (version 17.0 or above), create a Windows Forms App (.NET Framework) project and in **Project > Add New Item..**, select ActiveReports 18 Code-Based Report template and click **Add**.
2. Add reference to System.Windows.Forms.DataVisualization.
3. From the Visual Studio toolbox, drag and drop the **Data > Chart** item to the design surface.
4. Add the following code to the code-based report constructor after the InitializeComponent method:

VB code. Add this code after the InitializeComponent method

```
Dim chart = CType(Me.customControl1.Control,
System.Windows.Forms.DataVisualization.Charting.Chart)
Dim chartArea = New System.Windows.Forms.DataVisualization.Charting.ChartArea("Main")
chart.ChartAreas.Add(chartArea)
Dim seriesColumns = New
System.Windows.Forms.DataVisualization.Charting.Series("Columns")
For Each val In New Single() {6, 14, 3, 4, 2, 2, 1, 3}
    seriesColumns.Points.Add(Convert.ToDouble(val))
Next
seriesColumns.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Radar
chart.Series.Add(seriesColumns)
```

C# code. Add this code after the InitializeComponent method

```
var chart =
(System.Windows.Forms.DataVisualization.Charting.Chart)this.customControl1.Control;
var chartArea = new System.Windows.Forms.DataVisualization.Charting.ChartArea("Main");

chart.ChartAreas.Add(chartArea);
var seriesColumns = new
System.Windows.Forms.DataVisualization.Charting.Series("Columns");
foreach (var val in new float[] { 6, 14, 3, 4, 2, 2, 1, 3 })

    seriesColumns.Points.Add(Convert.ToDouble(val));
seriesColumns.ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Radar;
chart.Series.Add(seriesColumns);
```

Add Code to Layouts Using Script

In a section report, you can use script to access controls, functions in a class, namespaces, etc. You can also create

classes inside the script to call methods or add code to a report's script from a Windows Form. The following sections illustrate simple scripting scenarios with examples.

These steps assume that you have already added a Section Report (code based) template in a Visual Studio project.

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 **Detail**. 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.

To access a textbox in the detail section in VB.NET script

Visual Basic.NET script. Paste INSIDE the Detail Format event.

```
Me.TextBox1.Text = "Hello"
```

Or

Visual Basic.NET script. 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. Paste INSIDE the Detail Format event.

```
this.textBox1.Text = "Hello";
```

Or

C# script. Paste INSIDE the Detail Format event.

```
((TextBox)rpt.Sections["detail"].Controls["TextBox1"]).Text = "Hello";
```

Give a script access to functions in a class in your project

Using the **AddNamedItem** method, you can 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 saving it in the RPX file.

1. In the Code View of the Form, add a class to your project named **clsMyItem**.

To add a class in Visual Basic.NET

Visual Basic.NET code.

```
Public Class clsMyItem
End Class
```

To add a class in C#

C# code.

```
public partial class 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. Go to the design view of the report and double-click the gray area around the design surface to create an event-handling method for the **ReportStart** event.
4. Add the following code to the handler:

To access the class in Visual Basic.NET

Visual Basic.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());
```

5. From the Visual Studio toolbox, drag and drop a TextBox control onto the detail section of the report.
6. Go to the script tab and drop down the **Object** list to select **Detail**. This populates the Event drop-down list with section events.
7. Drop down the **Event** list and select **Format**. This creates script stubs for the event.
8. Add the following script to the event to access a control on the report and populate it using the named item.

To access the control in VB.NET script

VB.NET script. Paste INSIDE the Detail Format event.

```
Me.TextBox1.Text = myItem.getMyItem()
```

Or

VB.NET script. Paste INSIDE the Detail Format event.

```
CType(rpt.Sections("Detail1").Controls("TextBox1"), TextBox).Text = myItem.getMyItem()
```

To access the control in C# script

C# script. Paste INSIDE the Detail Format event.

```
this.textBox1.Text = myItem.getMyItem();
```

Or

C# script. Paste INSIDE the Detail Format event.

```
((TextBox)rpt.Sections["detail"].Controls["textBox1"]).Text = myItem.getMyItem();
```

9. Go to the preview tab to view the result.

Access namespaces

By using the **AddScriptReference** method, you can gain access to .NET or custom namespaces. 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 a namespace in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form code. Replace *YourReportName* with the name of your report.

```
Private Sub runReport()
      Dim rpt as new YourReportName
      rpt.AddScriptReference("System.Data.dll")
      rpt.Run()
End Sub
```

To access a namespace in C#

C# code. Paste INSIDE the Form code. Replace *YourReportName* with the name of your report.

```
private void runReport()
{
      YourReportName rpt = new YourReportName;
      rpt.AddScriptReference("System.Data.dll");
      rpt.Run();
}
```

Note: If you are using the custom assemblies, they must have Strong Name and registered to GAC folder.

If you want to use custom assemblies in the Script section of the Designer, then you need to add the assembly reference to a report before loading it into the report designer. To do this, follow these steps:

1. Create a custom assembly with the strong name and register it to GAC.
2. Create a new sample with a section report (XML-based).

3. Add the report designer to the form.
4. Add the following code to the **Form_Load** event.

To access a namespace in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the Form code. Replace *YourReportName* with the name of your report.

```
Dim rpt As SectionReport = New SectionReport()
Dim xtr As System.Xml.XmlTextReader = New
System.Xml.XmlTextReader("..\..\SectionReport2.rpx")
   rpt.LoadLayout(xtr)
   xtr.Close()
   rpt.AddScriptReference("ClassLibrary1.dll")
   designer1.Report = rpt
```

To access a namespace in C#

C# code. Paste INSIDE the Form code. Replace *YourReportName* with the name of your report.

```
SectionReport rpt = new SectionReport();
System.Xml.XmlTextReader xtr = new
System.Xml.XmlTextReader(@"..\..\SectionReport2.rpx");
rpt.LoadLayout(xtr);
xtr.Close();
rpt.AddScriptReference(@"ClassLibrary1.dll");
designer1.Report = rpt;
```

5. Run the sample.
6. Open the **Script** section of the designer.

You can now use the custom assemblies reference in the Script section of the Desginer.

Add code to a report's script from a Windows Form

Using the **AddCode** method in the Code View of the Form, you can add 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 in the RPX file.

1. Go to the Code View of your report and add a public function like the following:

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
```

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;
}
```

2. In the design view of your report double-click the gray area around the design surface to create an event-handling method for the **ReportStart** event.
3. Add the following code to the handler:

To access the class in Visual Basic.NET

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
Me.AddCode(addThisCode())
```

To access the class in C#

C# code. Paste INSIDE the ReportStart event.

```
this.AddCode(addThisCode());
```

4. Go to the script tab and drop down the **Object** list to select **Detail**. 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 the following script to the event:

To write the script in Visual Basic.NET

VB.NET script. Paste INSIDE the Detail1_Format event.

```
Me.TextBox1.Text = ShowACMessage()
```

Or

VB.NET script. Paste INSIDE the Detail1_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();
```

Or

C# script. Paste INSIDE the detail_Format event.

```
((TextBox)rpt.Sections["detail"].Controls["textBox1"]).Text = ShowACMessage();
```

Create classes inside the script to call methods

If the script requires a method to be called, you can construct a class inside the script.

1. Go to the script tab and add the following code at the top:

   To create a class inside the script in VB.NET script

   VB.NET script. Paste INSIDE the script tab.

   ```
   Public Class MyFuncs
      Public  Sub New()
      End Sub
      Public Function ShowMyString() As String
         Return "This is my string"
      End Function
   End Class
   ```

   To create a class inside the script in C#

   C# script. Paste INSIDE the script tab.

   ```
   public class MyFuncs
   {
      public MyFuncs()
      {
      }
      public string ShowMyString()
      {
         return "This is my string";
      }
   }
   ```

2. On the script tab, now drop down the Object list and select **Detail**. This populates the Event drop-down list with section events.
3. Drop down the Event list and select **Format**. This creates script stubs for the event.
4. Add the following script to the event:

   To create a class inside the script in VB.NET script

   VB.NET script. Paste INSIDE the Detail1_Format event.

   ```
   Dim f As MyFuncs =  New MyFuncs()
   Me.TextBox1.Text = f.ShowMyString
   ```

   Or

   VB.NET script. Paste INSIDE the Detail1_Format event.

   ```
   Dim f As MyFuncs =  New MyFuncs()
   CType(rpt.Sections("Detail1").Controls("TextBox1"), TextBox).Text = f.ShowMyString
   ```

To create a class inside the script in C#

C# script. Paste INSIDE the detail_Format event.

```
MyFuncs f = new MyFuncs();
this.textBox1.Text = f.ShowMyString();
```

Or

C# script. Paste INSIDE the detail_Format event.

```
MyFuncs f = new MyFuncs();
((TextBox)rpt.Sections["detail"].Controls["textBox1"]).Text = f.ShowMyString();
```

Note: Use the examples with the "this" (C#) and "Me"(Visual Basic.NET) keywords, as they are recommended rather than the ones with "rpt".

Subreport in Section Reports

This section discusses some common scenarios of using the subreports with code-based section reports.

Embed Subreports

Subreports with XML Data

Subreports with JSON Data

Subreports with Run-Time Data Sources

Embed Subreports

To embed a subreport into a parent report, you add two reports (one parent and one child report) to a Visual Studio project, and from the ActiveReports 18 Section Report toolbox, drag the **SubReport** control onto the parent report. The following steps take you through the process of adding a subreport in a Section Report.

These steps assume that you have already added a **Section Report (code-based)** template in a Visual Studio project.

Add code to create an instance of the child report in the parent report

1. Double-click the gray area around the parent report to create an event-handling method for the **ReportStart** event.
2. Add code like the following to the handler to create a new instance of the child report.

VB Code:

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()
```

C# Code:

C# code. Paste JUST ABOVE the ReportStart event.

```
private rptYourChildReportName rpt;
```

C# code. Paste INSIDE the ReportStart event.

```
rpt = new rptYourChildReportName();
```

⚠ **Caution**: It is recommended that you 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, using a lot of memory.

Add code to display the child report in a subreport control on a parent report

1. Add the SubReport control onto the design surface of the parent report.
2. Double-click the detail section of the report to create a detail_Format event.
3. Add code like the following to the handler to display a report in the SubReport control.

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

```
Me.SubReport1.Report = rpt
```

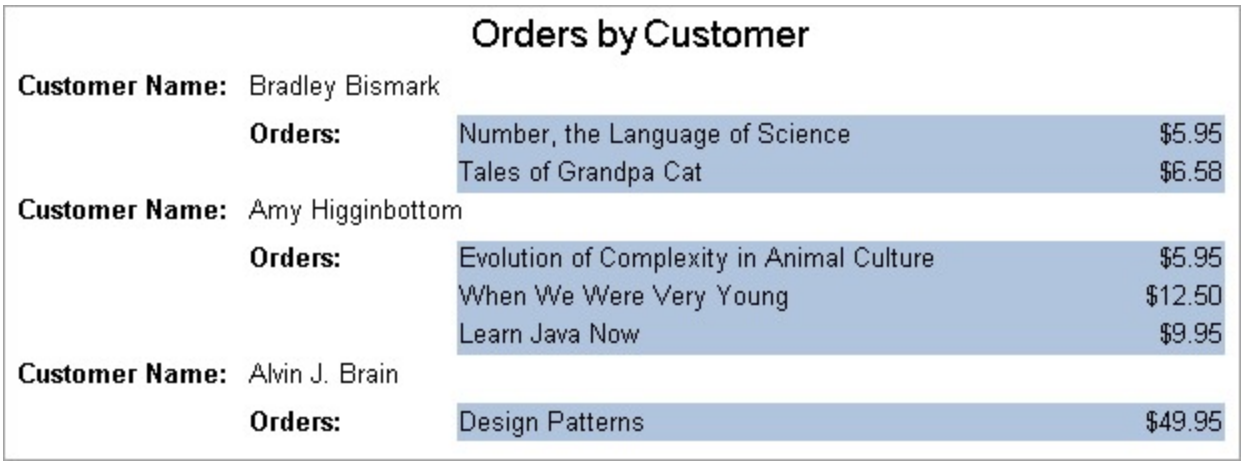
C# code. Paste INSIDE the Format event.

```
this.subReport1.Report = rpt;
```

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.

When you complete this walkthrough you get a layout that looks similar to the following.



Create Report

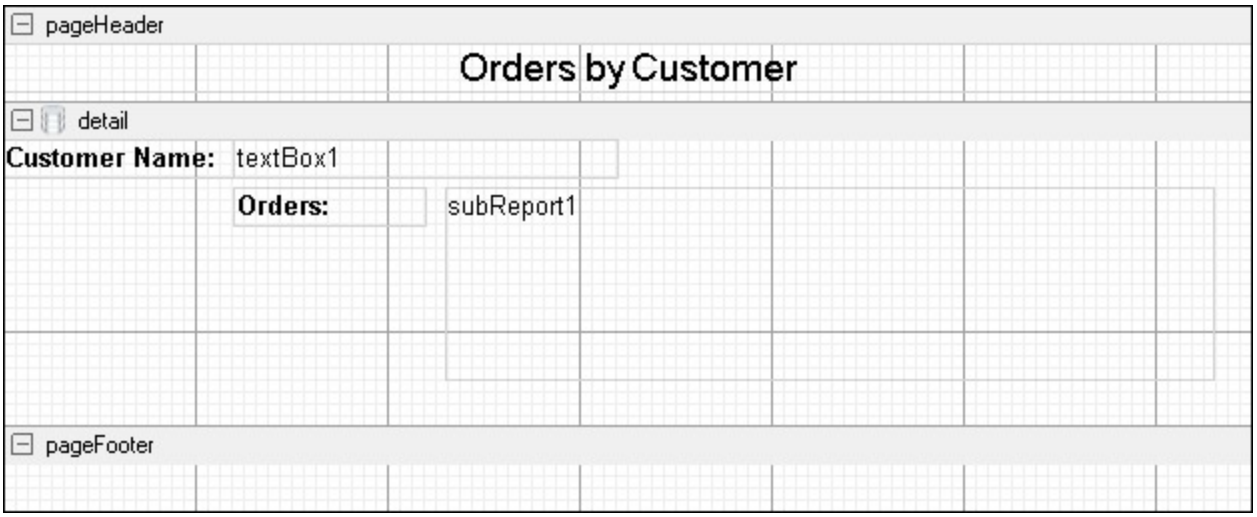
1. In Visual Studio, create a new Windows Forms App (.NET Framework) project and click **Next**.
2. In the **Configure your new project** dialog, type a name for your project, set **Framework** to .NET Framework 4.7 and click **Create.**
3. From the **Project** menu, select **Add New Item**.
4. In the Add New Item dialog that appears, select **ActiveReports 18 Code-Based Report** and in the Name field, rename the file as **rptMain**.
5. Click the **Add** button to open a new Section Report in the designer.
6. From the **Project** menu, select **Add New Item**.
7. In the Add New Item dialog that appears, select **ActiveReports 18 Code-Based Report** and in the Name field, rename the file as **rptSub**.
8. Click the **Add** button to open a second new Section Report in the designer.

Connect the Parent Report (rptMain) to a Data Source

1. On the detail section band, click the Data Source Icon.

2. In the Report Data Source dialog, on the **XML** tab, click the ellipsis (...) button next to File URL field.
3. In the **Open File** window that appears, navigate to **Customer.xml** and click the **Open** button.
4. In the **Recordset Pattern** field, enter `//CUSTOMER`.
5. Click **OK** to save the data source and return to the report design surface.

Design Report Layout for the Parent Report (rptMain)



1. On the design surface, select the pageHeader section and in the Properties panel, set the **Height** property to **0.3**.
2. On the design surface, select the grey area outside the report and in the Properties panel, set the **PrintWidth** property to **6.5**.
3. On the design surface, select the detail section and in the Properties panel, set the **CanShrink** property to **True** to eliminate white space.
4. From the toolbox, drag the Label control onto the pageHeader section and in the Properties panel, set the **Text** property to 'Orders by Customer'.
5. From the toolbox, drag the controls onto the detail section and in the Properties panel, set the properties of each control as follows.

6. From the toolbox, drag the following controls onto the detail section and in the Properties panel, set the properties of each control as follows.
    - **TextBox1**
      DataField: NAME
    - **Label1**
      Text: Customer Name:
    - **Label2**
      Text: Orders:
    - **Subreport**

Design Report Layout for the Child Report (rptSub)

1. On the design surface, select the **detail** section and in the Properties panel, set the **CanShrink** property to 'True' and **BackColor** to 'LightSteelBlue'.

    > **Tip**: Even if you do not want colors in your finished reports, using background colors on subreports can help in troubleshooting layout issues.

2. On the design surface, right-click the **pageHeader** or **pageFooter** section and select **Delete**.

    > **Note:** Subreports do not render these sections, so deleting them saves processing time.

3. From the toolbox, drag the following controls to the detail section and in the Properties panel, set the properties as follows.

Property Name	Property Value
TextBox1	
DataField	TITLE
Name	txtTitle
TextBox2	
DataField	PRICE
Name	txtPrice
OutputFormat	$#,##0.00 (or select Currency in the dialog)

Add Code to Create a new instance of the Child Report (rptSub)

> **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.

VB Code:

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()
```

C# Code:

1. Click in the gray area below **rptMain** to select it.
2. Click the events icon in the Properties panel 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();
```

Add Code to Pass a subset of the Parent Report's data to the Child Report

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 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

   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 GrapeCity.ActiveReports.Data.XMLDataSource
xmlDS.NodeList = CType(CType(Me.DataSource,
GrapeCity.ActiveReports.Data.XMLDataSource).Field("ORDER/ITEM", True),
System.Xml.XmlNodeList)
rpt.DataSource = xmlDS
SubReport1.Report = rpt
```

   C# code. Paste INSIDE the Format event.

```
GrapeCity.ActiveReports.Data.XMLDataSource xmlDS = new
GrapeCity.ActiveReports.Data.XMLDataSource();
```

```
xmlDS.NodeList = (System.Xml.XmlNodeList)((GrapeCity.ActiveReports.Data.XMLDataSource)
this.DataSource).Field("ORDER/ITEM", true);
rpt.DataSource = xmlDS;
subReport1.Report = rpt;
```

Preview the report

Click the preview tab to view the report at design time.

Subreports with JSON Data

Using JSON data requires some setup that is different from other types of data. This walkthrough illustrates how to set up a subreport bound to the JSON data source in the parent report.

When you complete this walkthrough you get a layout that looks similar to the following at design time and at run time.

Product Details

Category Name:	Beverages		
	Products:	Chai	$18.00
		Chang	$19.00
		Guaraná Fantástica	$4.50
		Sasquatch Ale	$14.00
		Steeleye Stout	$18.00
		Côte de Blaye	$263.50
		Chartreuse verte	$18.00
		Ipoh Coffee	$46.00
		Laughing Lumberjack Lager	$14.00
		Outback Lager	$15.00
		Rhönbräu Klosterbier	$7.75
		Lakkalikööri	$18.00
Category Name:	Condiments		
	Products:	Aniseed Syrup	$10.00
		Chef Anton's Cajun Seasoning	$22.00
		Chef Anton's Gumbo Mix	$21.35
		Grandma's Boysenberry Spread	$25.00
		Northwoods Cranberry Sauce	$40.00
		Genen Shouyu	$15.50
		Gula Malacca	$19.45
		Sirop d'érable	$28.50
		Vegie-spread	$43.90
		Louisiana Fiery Hot Pepper Sauce	$21.05
		Louisiana Hot Spiced Okra	$17.00
		Original Frankfurter grüne Soße	$13.00

Create Report

1. In Visual Studio, create a new Windows Forms App (.NET Framework) project and click **Next**.
2. In the **Configure your new project** dialog, type a name for your project, set **Framework** to .NET Framework 4.7 and click **Create.**
3. From the **Project** menu, select **Add New Item**.
4. In the Add New Item dialog that appears, select **ActiveReports 18 Code-Based Report** and in the Name field, rename the file as **rptMain**.
5. Click the **Add** button to open a new Section Report in the designer.
6. From the **Project** menu, select **Add New Item**.
7. In the Add New Item dialog that appears, select **ActiveReports 18 Code-Based Report** and in the Name field, rename the file as **rptSub**.
8. Click the **Add** button to open a second new Section Report in the designer.

Connect the Parent Report (rptMain) to a Data Source

1. On the detail section band, click the Data Source Icon.

2. In the Report Data Source dialog, select the **JSON** tab.
3. Click the **Build** button next to the Connection String section to open the **Configure JSON Data Source** dialog box.
4. Specify the following JSON file URL in the **Data Path** field of the dialog box.

JSON File URL
`https://demodata.mescius.io/northwind/odata/v1/Categories?$expand=Products`

The Connection String section displays the generated connection string as shown below.

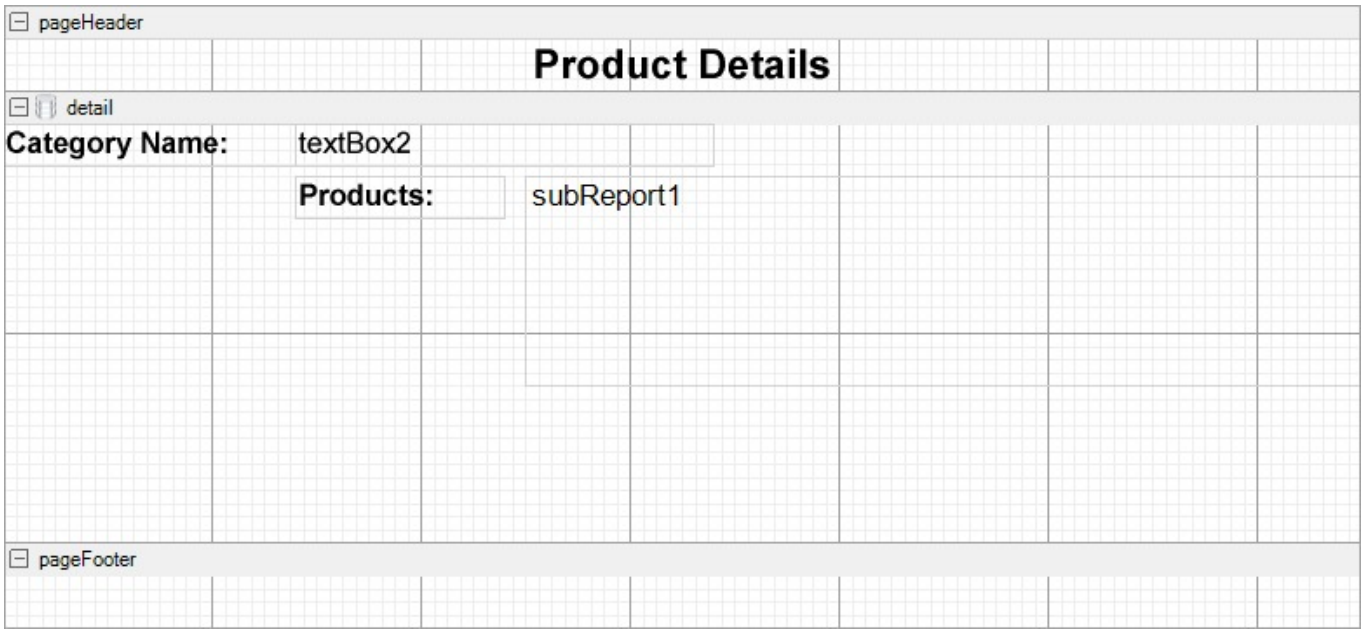
Connection String
`jsondoc=https://demodata.mescius.io/northwind/odata/v1/Categories?$expand=Products`

5. Click **OK** to proceed further.
6. In the **JSON Path** field, enter the below query to fetch the required data from the JSON path defined above.

Query
`$.value[*]`

7. Click **OK** to save the data source and return to the report design surface.

Design Report Layout for the Parent Report (rptMain)



1. On the design surface, select the pageHeader section and in the Properties window, set the **Height** property to **0.3**.
2. On the design surface, select the detail section and in the Properties window, set the **CanShrink** property to **True** to eliminate white space.
3. From the toolbox, drag the Label control onto the **pageHeader** section and in the Properties window, set the **Text** property to 'Product Details'.
4. From the toolbox, drag the following controls onto the detail section and in the Properties panel, set the properties of each control as follows.
    - **TextBox1**
      DataField: CategoryName
    - **Label1**
      Text: Category Name:
    - **Label2**
      Text: Products:
    - **Subreport**

Design Report Layout for the Child Report (rptSub)

1. On the design surface, select the **detail** section and in the Properties panel, set the BackColor to 'White Smoke'

    > **Tip**: Even if you do not want colors in your finished reports, using background colors on subreports can help in troubleshooting layout issues.

2. On the design surface, right-click the pageHeader or pageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
3. From the toolbox, drag the following controls to the **detail** section and in the Properties panel, set the properties as follows.

Property Name	Property Value
TextBox1	
DataField	ProductName
Name	txtProductName
TextBox2	

DataField	UnitPrice
Name	txtUnitPrice
OutputFormat	$#,##0.00 (or select Currency in the dialog)

Add Code to Create a new instance of the Child Report (rptSub)

> ⚠ **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.

VB Code:

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()
```

C# Code:

1. Click in the gray area below **rptMain** to select it.
2. Click the events icon in the Properties panel 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();
```

Add Code to Pass a subset of the Parent Report's data to the Child Report

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 JSONDataSource.
- Type cast the new data source as rptMain's data source.
- Specify a valid JSONPath expression.
- Pass the new data source to the subreport
- Display rptSub in the subreport control

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

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

```
Dim jsonDS As New GrapeCity.ActiveReports.Data.JsonDataSource
jsonDS.JsonData = (CType(Me.DataSource,
GrapeCity.ActiveReports.Data.JsonDataSource)).Field("Products").ToString()
jsonDS.JsonPath = "$.[*]"
rpt.DataSource = jsonDS
SubReport1.Report = rpt
```

C# code. Paste INSIDE the Format event.

```
GrapeCity.ActiveReports.Data.JsonDataSource jsonDS = new
GrapeCity.ActiveReports.Data.JsonDataSource();
jsonDS.JsonData =
((GrapeCity.ActiveReports.Data.JsonDataSource)this.DataSource).Field("Products").ToString();
jsonDS.JsonPath = "$.[*]";
rpt.DataSource = jsonDS;
subReport1.Report = rpt;
```

Preview the report

Click the preview tab to view the report at design time.

Subreports with Run-Time Data Sources

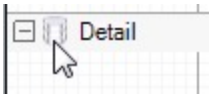
ActiveReports allows Section Reports to contain any number of child reports using the Subreport control. Child reports, or subreports, are executed each time the parent section (i.e. the section in which the Subreport control is placed) is processed. This tutorial illustrates how to modify the subreport record source from the data in the parent report to retrieve the correct information.

Create Report

1. In Visual Studio, create a new Windows Forms App (.NET Framework) project and click **Next**.
2. In the **Configure your new project** dialog, type a name for your project, set **Framework** to .NET Framework 4.7 and click **Create.**
3. From the **Project** menu, select **Add New Item**.
4. In the Add New Item dialog that appears, select **ActiveReports 18 Code-Based Report** and in the Name field, rename the file as **rptMain**.
5. Click the **Add** button to open a new Section Report in the designer.
6. From the **Project** menu, select **Add New Item**.
7. In the Add New Item dialog that appears, select **ActiveReports 18 Code-Based Report** and in the Name field, rename the file as **rptSub**.
8. Click the **Add** button to open a second new Section Report in the designer.

Connect the Parent Report (rptMain) to a Data Source

1. On the detail section band, click the Data Source Icon.

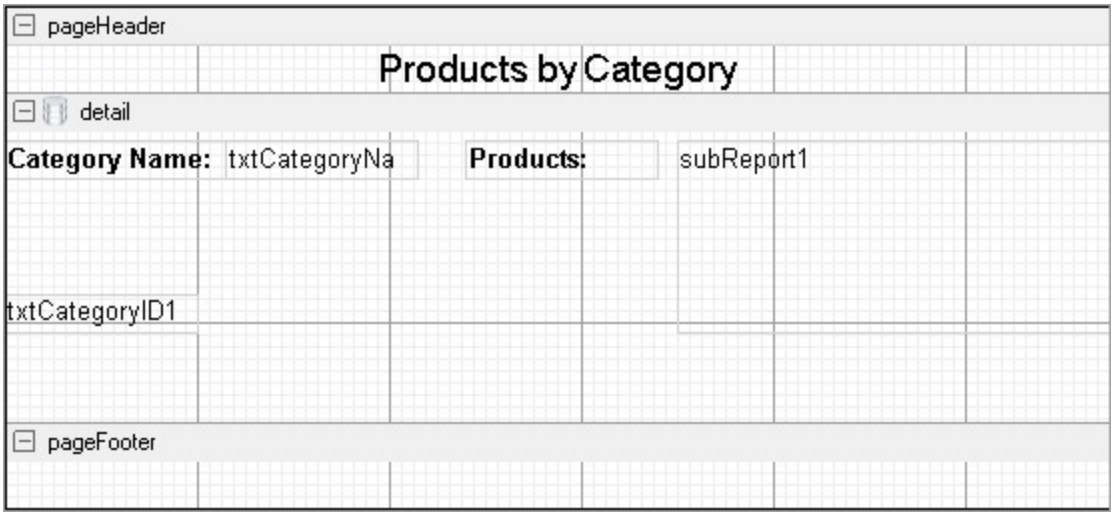


2. In the Report Data Source dialog that appears, from the **OLE DB** tab, create a data source connection.
3. Once the connection string field is populated, in the Query field, enter the following SQL query.

SQL Query
`SELECT * FROM Categories`

4. Click **OK** to save the data source and return to the report design surface.

Design Report Layout for the Parent Report (rptMain)



1. In the Report Explorer, select the report and in the Properties window, set the **PrintWidth** property to **5.75**.
2. On the design surface, select the detail section and in the Properties window, set the **CanShrink** property to **True** to eliminate white space.
3. From the toolbox, drag a Label control onto the pageHeader section and in the Properties window, set the properties as follows.

Property Name	Property Value
Name	lblProductsbyCategory
Text	Products by Category

4. From the toolbox, drag the following controls onto the detail section and in the Properties window, set the properties as follows.

Property Name	Property Value
TextBox1	
Name	txtCategoryID1
Data Field	CategoryID
Visible	True
TextBox2	
Name	txtCategoryName1
Data Field	CategoryName
Label1	
Name	lblCategoryName
Text	CategoryName:
Label2	
Name	lblProducts

Text	Products:
Subreport	
Name	SubReport1

Design Report Layout for the Child Report (rptSub)

1. On the design surface, select the detail section and in the Properties window, set the following properties.

Property Name	Property Value
CanShrink	True
BackColor	AliceBlue

> **Tip**: Even if you do not want colors in your finished reports, using background colors on subreports can help in troubleshooting layout issues.

2. On the design surface, right-click the pageHeader or pageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
3. From the toolbox, drag a TextBox control to the detail section and in the Properties window, set the following properties.

Property Name	Property Value
DataField	ProductName
Name	txtProductName
Text	Product Name

Add Code to Create an Instance of the Subreport

Design Report Layout for the Child Report (rptSub)

1. On the design surface, select the **detail** section section and in the Properties panel, set the BackColor to 'White Smoke'

> **Tip**: Even if you do not want colors in your finished reports, using background colors on subreports can help in troubleshooting layout issues.

2. On the design surface, right-click the pageHeader or pageFooter section and select **Delete**. Subreports do not render these sections, so deleting them saves processing time.
3. From the toolbox, drag the following controls to the **detail** section and in the Properties panel, set the properties as follows.

Property Name	Property Value
TextBox1	
DataField	ProductName
Name	txtProductName
TextBox2	

DataField	UnitPrice
Name	txtUnitPrice
OutputFormat	$#,##0.00 (or select Currency in the dialog)

Add Code to Create a new instance of the Child Report (rptSub)

> ⚠ **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.

VB Code:

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.

```
Private rpt As rptSub
Private childDataSource As New GrapeCity.ActiveReports.Data.OleDBDataSource()
```

Visual Basic.NET code. Paste INSIDE the ReportStart event.

```
rpt = New rptSub()
```

C# Code:

1. Click in the gray area below **rptMain** to select it.
2. Click the events icon in the Properties panel 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;
private GrapeCity.ActiveReports.Data.OleDBDataSource childDataSource = new
GrapeCity.ActiveReports.Data.OleDBDataSource();
```

C# code. Paste INSIDE the ReportStart event.

```
rpt = new rptSub();
```

Add Code to Assign a Data Source for the Child Report {rptSub)

1. Back in design view of the Parent report (rptMain), 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

    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,
GrapeCity.ActiveReports.Data.OleDBDataSource).ConnectionString
childDataSource.SQL = "SELECT * FROM Products WHERE CategoryID = " +
Me.txtCategoryID1.Value.ToString
rpt.DataSource = childDataSource
SubReport1.Report = rpt
```

    C# code. Paste INSIDE the Format event.

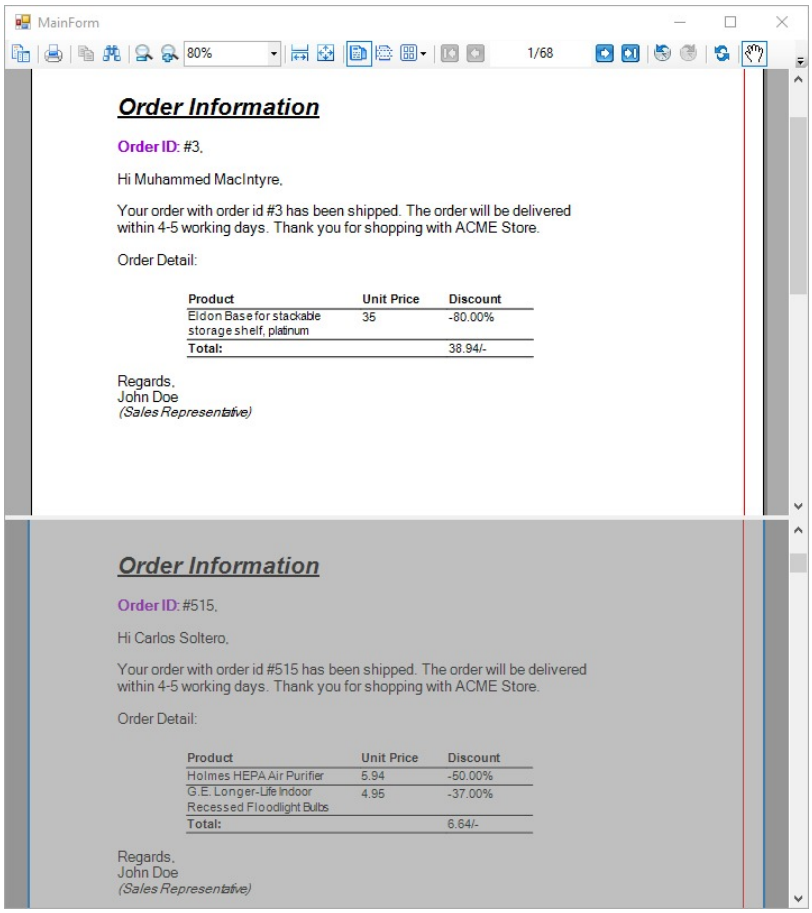
```
childDataSource.ConnectionString =
((GrapeCity.ActiveReports.Data.OleDBDataSource)this.DataSource).ConnectionString;
childDataSource.SQL = "SELECT * FROM Products WHERE CategoryID = " +
this.txtCategoryID1.Value.ToString();
rpt.DataSource = childDataSource;
SubReport1.Report = rpt;
```

Preview the report

Click the preview tab to view the report at design time.

Create Mail Merge with RichTextBox

The RichText control can contain field place holders that can be replaced with values (merged) at run time. Let's create a mail merge report that uses the **RichTextBox** control. When you complete this walkthrough you get a layout that looks similar to the .



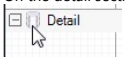
Create a Report

1. In Visual Studio, create a new Windows Forms App (.NET Framework) project and click **Next**.
2. In the **Configure your new project** dialog, type a name for your project, set **Framework** to .NET Framework 4.7 and click **Create**.
3. Select **ActiveReports 18 Code-Based Report** and click **Next**.
4. Enter a **Project name**, select a **Framework** and click **Create**.
5. Rename **SectionReport1.cs** and in the Name field, rename the file as **OrderLetter**.
6. Double-click **OrderLetter.cs** to open the report in the Section Report designer.

Bind Report to Data

Connect to a Data Source

1. On the detail section band, click the Data Source Icon.



2. In the **Report Data Source** dialog that appears, select the **CSV** tab to connect to a CSV data source. Let us connect to MyOrders.csv data available on GitHub. See CSV Provider page for details on binding with CSV data. The connection string for our data is as shown:

Connection String
`Path=data\\MyOrders.csv;Locale=en-IN;TextQualifier=";ColumnsSeparator=,;RowsSeparator=\r\n;Columns=ID,Product,Customer,OrderNumber,Stock,Total,UnitPrice,City,ProductLine,Discount;HasHeaders=True`

3. Click **OK** to save the changes and return to the report design surface. The dataset is also automatically added.

Design Report

1. On the design surface of the report, right-click and select **Insert**, then **Group Header/Footer** to add group header and footer sections.
2. On the design surface, select the grey area outside the report and in the Properties window, set the **PrintWidth** property to **6.5**.
3. Select the group header and in the Properties window, set the properties as follows.

Property Name	Property Value
DataField	OrderNumber
KeepTogether	True

4. On the design surface of the report, select the group footer section and in the Properties window, set the following properties.

Property Name	Property Value
KeepTogether	True
NewPage	After

5. On the design surface of the report, select the pageHeader section and in the Properties panel, set the following properties.

Property Name	Property Value
BackColor	Coral

6. From the toolbox, drag the Label control to the pageHeader section and in the Properties panel, set the properties as follows.

Property Name	Property Value
Location	0, 0 in
Text	Company
Font > Size	20
Font > Bold	True

7. From the toolbox, drag three **TextBox** controls to the **groupHeader** section and in the Properties panel, set the properties as follows.

Property Name	Property Value
TextBox1	
Location	0.8, 1.8 in
Text	Product
Font > Bold	True
TextBox2	
Location	2.80, 1.8 in
Text	Unit Price
Font > Bold	True
TextBox3	
Location	3.80, 1.8 in
Text	Discount
Font > Bold	True

8. In the **Report Explorer**, expand the **Fields** node, then the **Bound** > **Document** node. Drag the **Product** field onto the detail section and in the Properties panel, set the following properties:

Property Name	Property Value
Product field	
Location	0.8, 0 in
Name	textProduct1
UnitPrice field	
Location	2.8, 0 in
Name	txtUnitPrice1
Discount field	
Location	3.8, 0 in
Name	txtDiscount1

9. In the Report Explorer, expand the **Fields** node, then the **Bound** >**Document** node. Drag the **Total** field onto the **groupFooter** section and in the Properties panel, set the following properties.

Property Name	Property Value
Location	3.8, 0 in
Name	txtTotal

10. From the toolbox, drag the **TextBox** control to the **groupFooter** section and in the Properties panel, set the properties as follows.

Property Name	Property Value
Location	0.8, 0 in
Text	Total:
Font > Bold	True

11. From the tool bar drag two **RichTextBox** controls to the **groupHeader** and set the following properties:

Property Name	Property Value
RichTextBox1	
Location	0, 0 in
AutoReplaceFields	True
Name	richTextBox1
RichTextBox2	
Location	0, 0.35 in
AutoReplaceFields	True
Name	richTextBox2

Add fields to the RichText control

1. Double-click **richTextBox1** control box, delete the default text, and add the following text to the richTextBox1 control box:

Paste into the richTextBox1 control
`OrderOrder ID: #[!OrderNumber],`

```
Hi [!Customer],

Your order with order id #[!OrderNumber] has been shipped. The order will be delivered within 4-5 working days. Thank you for shopping with ACME Store.

Order Detail:
```

2. Similarly, add the following text to the richTextBox2 control box:

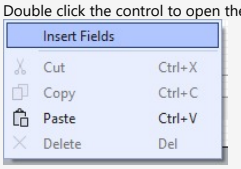
Paste into the richTextBox2 control

```
Regards,
John Doe
(Sales Representative)
```

3. Arrange the text and fields within the control as you would in any text editor.

Note: You can add field to the RichTextBox control by

1. Double click the control to open the edit mode.



2. Right-click and select Insert Field.
3. Enter the field name and click OK.

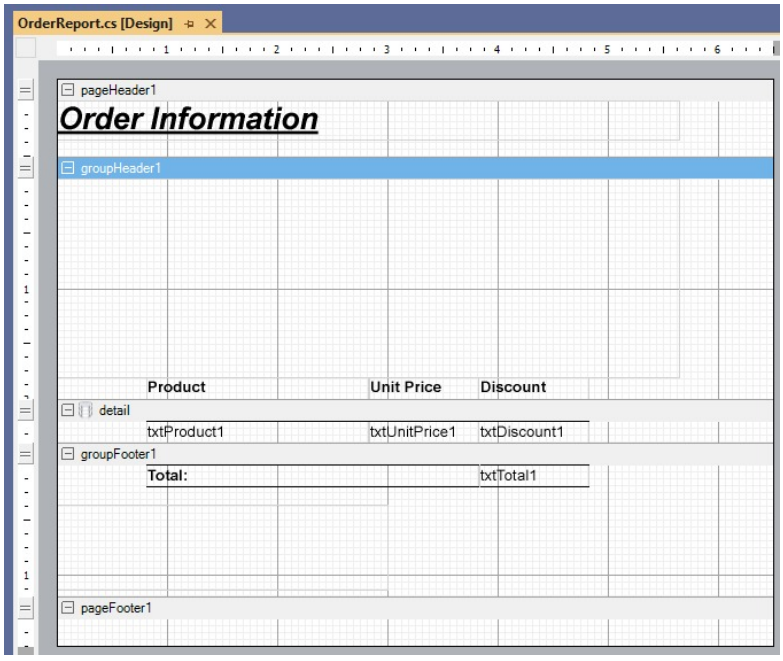
Add code to update RichText fields

1. On the design surface, double click the **detail** section band.
2. Double-click in the group header section of the report to create an event-handling method for the group header's **Format** event.
3. Add code to the handler to:
   - Replace the OrderNumber field in the RichText control with the current OrderNumber field value.
   - Replace the Customer field with the current Customer field value

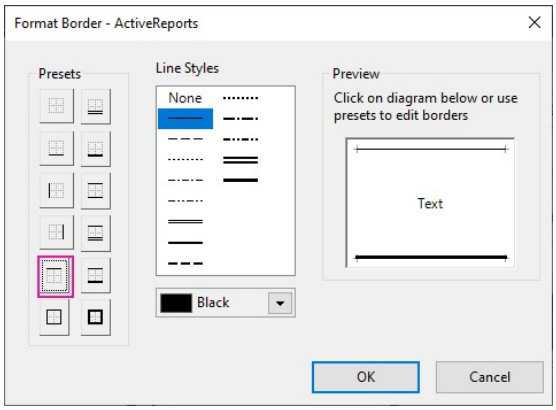
Example Title

```
//Use the OrderNumber Field Value
    richTextBox1.ReplaceField("OrderNumber", Fields["OrderNumber"].Value.ToString());
//Use the Customer Field Value
    richTextBox1.ReplaceField("Customer", Fields["Customer"].Value.ToString());
```

Improve the appearance of the report and preview.



1. Select all the textBoxes in the **detail** section then, right-click on the text box and go to **Format Border** and select the 5th preset in the first column to have a border line on top of the textBox as shown in the image below:



2. Select all the textBoxes in the **groupFooter** section then, right-click on the text box and go to **Format Border** and select the 4th preset in the 1st column also, select the 2nd preset in the 1st column to have two border lines on the textbox controls(top and bottom) as shown in the image below:

Overlaying in Reports (Letterhead)

ActiveReports allows you to overlay static report formats over data reports.

These steps demonstrate how to overlay an ActiveReports report, displaying customers orders by country, with a static letterhead report.

> **Note:** The report connects to NWIND.db that can be downloaded from GitHub.

The final report will look as shown.

Mescius

Customers in Argentina

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

Customers in Austria

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

Create a report

1. Open Visual Studio to create a new project.
2. Select **ActiveReports 18 Section Report (code-based)**, click Next.
3. In the **Project name** field, rename **SectionReport1.cs** as **rptData**.
4. Click **Create**.
5. Double-click **rptData.cs** to open the report in the Section Report designer.

Bind Report to Data

1. As you create a new report, the **Report Data Source** dialog appears for you to configure the report data connection. You can also access this dialog by clicking the **DataSource Icon** in the Detail section band.

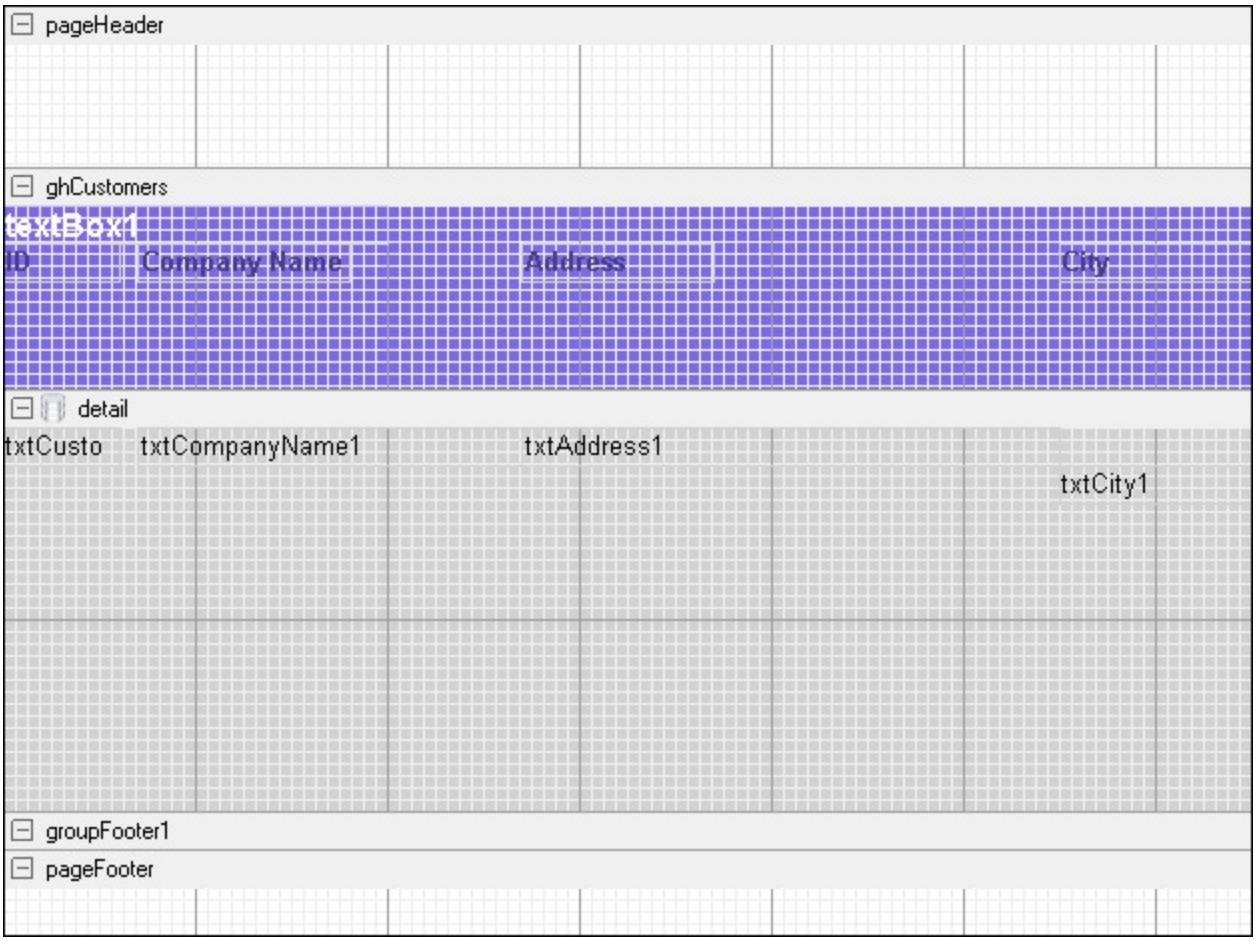
2. Choose **Custom** tab > **SQLite Provider** in the Report Data Source dialog, and bind the report to Sqlite data using the following connection string and query.

Connection String
`data source=c:\data\NWIND.db`

Query
`Select * from Customers ORDER BY Country`

3. Click **OK** to close the Report Data Source dialog and return to the report design surface.

Design Report Layout (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.)
2. On the design surface, select the grey area outside the report and in the Properties window, set the **PrintWidth** property to **6.5**.
3. Right-click the report and select **Insert > GroupHeader/Footer** to add group header and group footer sections.
4. Select the group header and in the Properties window, set the properties as follows.

Property Name	Property Value
Name	ghCustomers

BackColor	MediumSlateBlue
CanShrink	True
DataField	Country
GroupKeepTogether	FirstDetail
KeepTogether	True

5. From the toolbox, drag the following controls to **ghCustomers** and in the Properties window, set the properties as follows.

TextBox1

Property Name	Property Value
DataField	="Customers in " + Country (DataField)
Size	2, 0.2 in
Location	0, 0 in
Font Bold	True
ForeColor	White
Font Size	12

Label1

Property Name	Property Value
Text	ID
Size	0.6, 0.2 in
Location	0, 0.2 in
Font Bold	True
ForeColor	DarkSlateBlue

Label2

Property Name	Property Value
Text	Company Name
Size	1.1, 0.2 in
Location	0.7, 0.2 in
Font Bold	True
ForeColor	DarkSlateBlue

Label3

Property Name	Property Value
Text	Address
Size	1, 0.2 in
Location	2.7, 0.2 in
Font Bold	True
ForeColor	DarkSlateBlue

Label4

Property Name	Property Value
Text	City
Size	1, 0.2 in
Location	5.5, 0.2 in
Font Bold	True
ForeColor	DarkSlateBlue

6. Click the **Detail** section and in the Properties window, set the properties as follows.

Property Name	Property Value
BackColor	LightGray
CanShrink	True

7. From the toolbox, drag four TextBox controls onto the **Detail** section and set the properties of each textbox as follows.

TextBox1

Property Name	Property Value
DataField	CustomerID
Size	0.6, 0.2 in
Location	0, 0 in

TextBox2

Property Name	Property Value
DataField	CompanyName
Size	2, 0.2 in
Location	0.7, 0 in

TextBox3

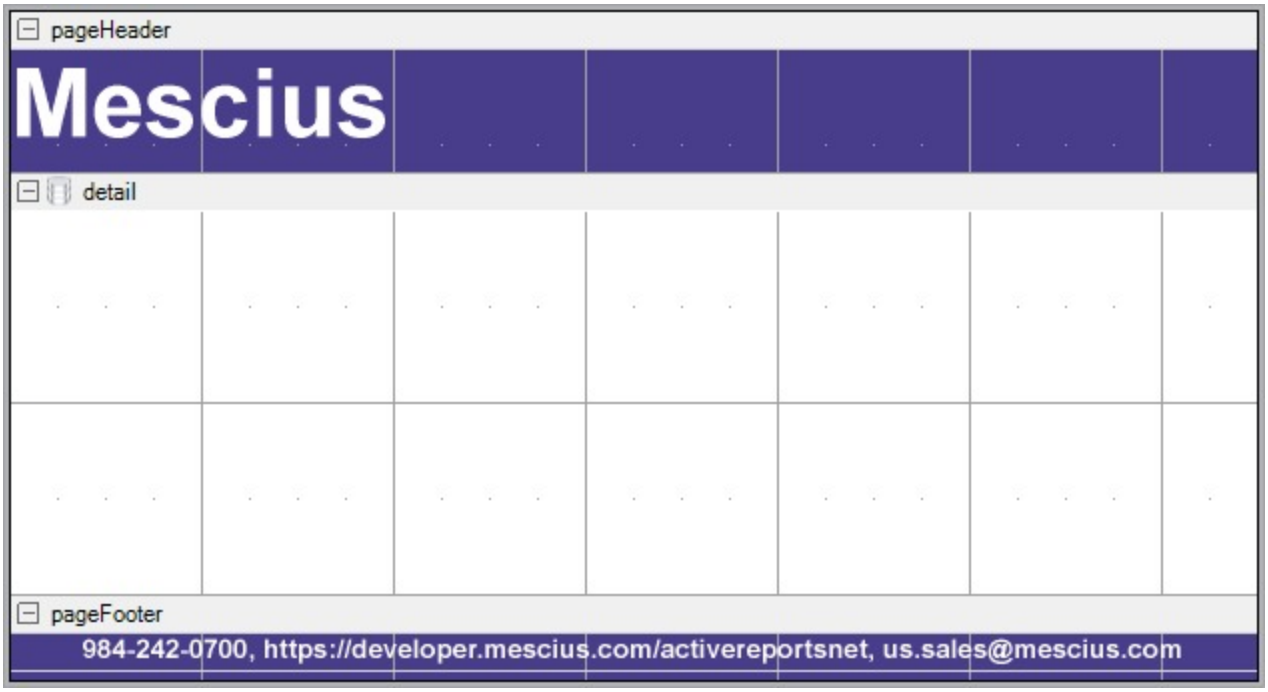
Property Name	Property Value
DataField	Address
Size	2.8, 0.2 in
Location	2.7, 0 in

TextBox4

Property Name	Property Value
DataField	City
Size	1, 0.2 in
Location	5.5, 0 in

8. Select the group footer and in the Properties window, set the **Height** property to **0**.

Create a report and design Report Layout (rptLetterhead)



1. Go to **Project** in the Visual Studio menu and select **Add New Item...**
2. Select **ActiveReports 18 Section Report (code-based)**.
3. In the **Project name** field, rename **SectionReport2.cs** as **rptLetterhead**.
4. Click **Create.**
5. Double-click **rptLetterhead.cs** to open the report in the Section Report designer.
6. Select the **Page Header** and in the Properties window, set the properties as follows.

Property Name	Property Value
BackColor	DarkSlateBlue
Height	0.65

7. From the toolbox, drag a Label control onto the **Page Header** and in the Properties window, set the properties as follows.

   Label1

Property Name	Property Value
Size	6.5, 0.65 in
Location	0, 0 in
Font Size	36
Font Bold	True
ForeColor	White
Text	Mescius

8. Select the **Page Footer** and in the Properties window, set the **BackColor** property to **DarkSlateBlue**.
9. From the toolbox, drag a Label control onto the Page Footer and in the Properties window, set the properties as follows.

Property Name	Property Value
Size	6.5, 0.2 in
Location	0, 0 in
Alignment	Center
Font Bold	True
ForeColor	White
Text	984-242-0700, https://developer.mescius.com/activereportsnet, activereports.sales@mescius.com

Add Code to Overlay Data Report Pages with Letterhead Report

1. Double-click the title bar of the Windows Form to create an event-handling method for the Form_Load event.
2. Add the following code to the handler to set the viewer to display the rptData report document an to overlay rptLetterhead on rptData.

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

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

C# code. Paste INSIDE the Form Load event.

```
rptData rpt = new rptData();
   rpt.Run();
   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]);
   }
   viewer1.Document = rpt.Document;
```

3. Improve the appearance of the report and preview.

Optimize Section Reports

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 slows down the processing time, this allows the document to be cached to disk instead of loading the whole report in memory. The PDF export also detects this setting and exports the cached report. Please note that only the PDF export is affected by the CacheToDisk property; other exports may run out of memory with very large reports. By default, CacheToDisk uses IsolatedStorage, which requires IsolatedStorageFilePermission.
  It is recommended that you use the CacheToDiskLocation property to specify the physical path instead of using isolated storage so that you do not run into the size limit.
- **Summary**: 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.

Speed Considerations

- **Image:** 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.
- **SELECT *:** Using the SELECT * statement is only recommended when you actually want to use all of the data returned by this statement. Contact your database administrator for other methods to speed up your queries.

Printing Considerations

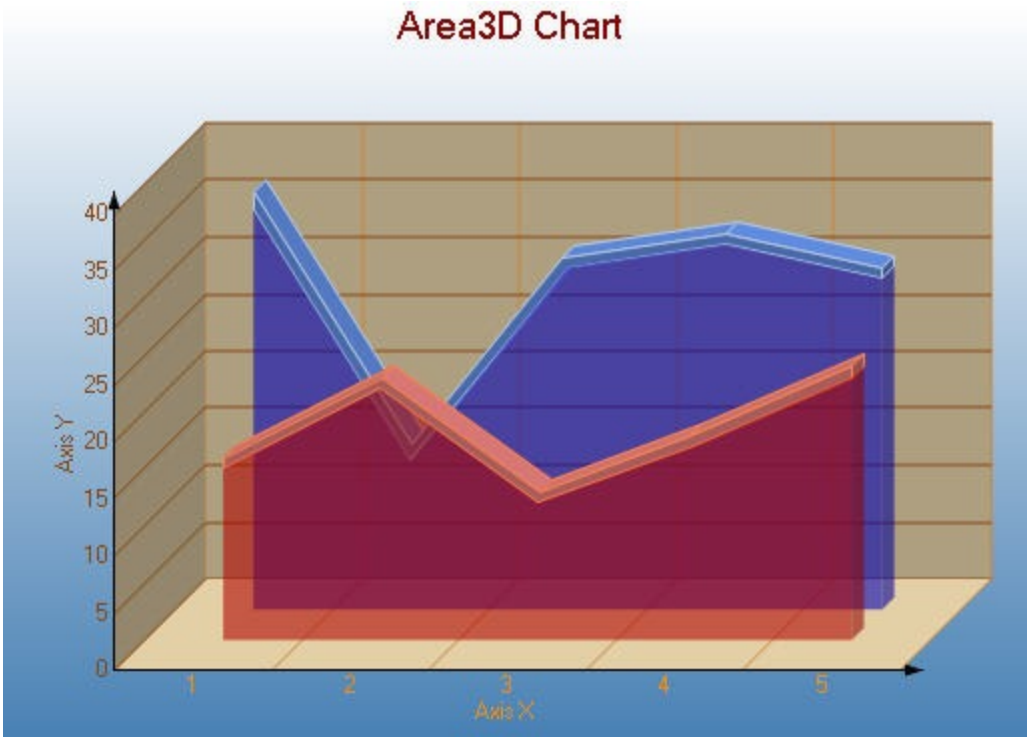
- **Virtual Printer:** We recommend use of virtual printer in case report generate environment differs from the environment in which the report is viewed or printed like in the case of Web applications.
- **Line:** Please be careful as Line control has been used to draw borders within a report. An issue has been observed that spool size is increased (when comparing with Solid) during printing in case LineStyle property is set to an any value e.g. Dash or Dot etc. other than Solid that is the default value. As a result, time is taken for spooling by the report thus hindering the performance while printing. Same issue is observed when rendering a line using GDI+ in PrintDocument of .NET Framework.

Custom Properties in Chart

The topic explains how to set the custom properties in Section Reports charts. 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 Panel of Visual Studio and in the DataPoint Collection dialog box via the Points (Collection) property in the Series dialog.

Note that 2D Area Charts do not have any custom properties.

3D Area Chart



Custom Properties

The **LineBackdrop** property gets or sets the backdrop information for the 3D line.
The **Thickness** property gets or sets the thickness of the 3D line.
The **Width** property gets or sets the width of the 3D line.

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.

Visual Basic

```
Me.ChartControl1.Series(0).Properties("LineBackdrop") = New
GrapeCity.ActiveReports.Chart.Graphics.Backdrop(Color.Red, CType(150, Byte))
Me.ChartControl1.Series(0).Properties("Thickness") = 5.0F
Me.ChartControl1.Series(0).Properties("Width") = 30.0F
```

C#

```
this.ChartControl1.Series[0].Properties["LineBackdrop"] = new
GrapeCity.ActiveReports.Chart.Graphics.Backdrop(System.Drawing.Color.Red, ((System.Byte)
(150)));
this.ChartControl1.Series[0].Properties["Thickness"] = 5f;
this.ChartControl1.Series[0].Properties["Width"] = 30f;
```

Similarly, the for **Stacked Area Chart** and **Stacked Area 100% Chart,** the **Width** property is the custom property that gets or sets the width of the 3D stacked area.

2D Bar Charts

Simple Bar Chart

The **Gap** property 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.

Visual Basic

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

C#

```
this.ChartControl1.Series[0].Properties["Gap"] = 50f;
```

Gantt Chart

The **Gap** property 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.

Visual Basic

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

C#

```
this.ChartControl1.Series[0].Properties["Gap"] = 50f;
```

Horizontal Bar Chart

The **Gap** property 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.

Visual Basic

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

C#

```
this.ChartControl1.Series[0].Properties["Gap"] = 65f;
```

Stacked Bar Chart

The **Gap** property 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.

Visual Basic

```
Me.ChartControl1.Series(0).Properties("Gap") = 100.0F
```

C#

```
this.ChartControl1.Series[0].Properties["Gap"] = 100f;
```

Stacked Bar Chart 100%

The **Gap** property 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 StackedBAR110Pct chart.

Visual Basic

```
Me.ChartControl1.Series(0).Properties("Gap") = 100.0F
```

C#

```
this.ChartControl1.Series[0].Properties["Gap"] = 100f;
```

3D Bar Chart

Simple Bar Chart

The **BarTopPercent** property gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
The **BarType** property gets or sets the type of bars that is displayed.Use BarType enumeration value.
The **Gap** property gets or sets the space between the bars of each X axis value.
The **RotationAngle** propertygets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
The **VertexNumber** property 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.

Gantt Chart

The **BarTopPercent** property gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
The **BarType** property gets or sets the type of bars that are displayed. Use BarType enumeration value.
The **Gap** property gets or sets the space between the bars of each X axis value.
The **VertexNumber** property 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.

Horizontal Bar Chart

The **BarTopPercent** property gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
The **BarType** propertygets or sets the type of bars that is displayed. Use BarType enumeration value.
The **Gap** property gets or sets the space between the bars of each X axis value.
The **RotationAngle** property gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
The **VertexNumber** property 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 horizontal 3D bar chart as shown above.

Visual Basic

```
Me.ChartControl1.Series(0).Properties("BarTopPercent") = 80.0F
 Me.ChartControl1.Series(0).Properties("BarType") =
GrapeCity.ActiveReports.Chart.BarType.Custom
 Me.ChartControl1.Series(0).Properties("Gap") = 65.0F
 Me.ChartControl1.Series(0).Properties("PointBarDepth") = 100.0F
 Me.ChartControl1.Series(0).Properties("RotationAngle") = 0.0F
```

```
 Me.ChartControl1.Series(0).Properties("VertexNumber") = 6
```

C#

```
this.ChartControl1.Series[0].Properties["BarTopPercent"] = 80f;
 this.ChartControl1.Series[0].Properties["BarType"] =
GrapeCity.ActiveReports.Chart.BarType.Custom;
 this.ChartControl1.Series[0].Properties["Gap"] = 65f;
 this.ChartControl1.Series[0].Properties["PointBarDepth"] = 100.0f;
 this.ChartControl1.Series[0].Properties["RotationAngle"] = 0f;
 this.ChartControl1.Series[0].Properties["VertexNumber"] = 6;
```

Stacked Bar Chart

The **BarTopPercent** property gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
The **BarType** property gets or sets the type of bars that are displayed. Use BarType enumeration value.
The **Gap** property gets or sets the space between the bars of each X axis value.
The **VertexNumber** property 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 StackedBar3D chart.

Visual Basic

```
Me.ChartControl1.Series(0).Properties("BarTopPercent") = 80.0F
 Me.ChartControl1.Series(0).Properties("BarType") =
GrapeCity.ActiveReports.Chart.BarType.Custom
 Me.ChartControl1.Series(0).Properties("Gap") = 65.0F
 Me.ChartControl1.Series(0).Properties("VertexNumber") = 6
```

C#

```
this.ChartControl1.Series[0].Properties["BarTopPercent"] = 100f;
 this.ChartControl1.Series[0].Properties["BarType"] =
GrapeCity.ActiveReports.Chart.BarType.Custom;
 this.ChartControl1.Series[0].Properties["Gap"] = 65f;
 this.ChartControl1.Series[0].Properties["VertexNumber"] = 6
```

Bar/Cylinder Chart

The **BarTopPercent** property gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
The **BarType** property gets or sets the type of bars that is displayed. Use BarType enumeration value.
The **Gap** property gets or sets the space between the bars of each X axis value.
The **RotationAngle** property gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
The **VertexNumber** property 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.

Bar/Pyramid Chart

The **BarTopPercent** property gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.

The **BarType** property gets or sets the type of bars that is displayed. Use BarType enumeration value.
The **Gap** property gets or sets the space between the bars of each X axis value.
The **RotationAngle** property gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
The **VertexNumber** property 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.

Clustered Bar Chart

The **BarTopPercent** property gets or sets the percentage of the top of the bar that is shown for Cone or Custom BarTypes.
The **BarType** property gets or sets the type of bars that are displayed. Use BarType enumeration value.
The **Gap** property gets or sets the space between the bars of each X axis value.
The **RotationAngle** property gets or sets the starting horizontal angle for custom 3D bar shapes. Can only be used with the Custom BarType.
The **VertexNumber** property 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.

VB

```
' set the custom properties for series 1.
Me.ChartControl1.Series(0).Properties("BarTopPercent") = 50.0F
Me.ChartControl1.Series(0).Properties("BarType") =
GrapeCity.ActiveReports.Chart.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") =
GrapeCity.ActiveReports.Chart.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
```

C#

```
// set the custom properties for series 1.
this.ChartControl1.Series[0].Properties["BarTopPercent"] = 50f;
this.ChartControl1.Series[0].Properties["BarType"] =
GrapeCity.ActiveReports.Chart.BarType.Custom;
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"] =
GrapeCity.ActiveReports.Chart.BarType.Custom;
this.ChartControl1.Series[1].Properties["Gap"] = 300f;
this.ChartControl1.Series[1].Properties["RotationAngle"] = 90f;
this.ChartControl1.Series[1].Properties["VertexNumber"] = 3;
```

2D Line Chart

Bezier Chart

Custom properties are as follows:

The **Line** property gets or sets the line element. Used to set color, thickness and shape of a line.
The **Tension** property gets or sets the tension of the curved lines.

Bezier XY

Custom properties are as follows:

The **Line** property gets or sets the line element. Used to set color, thickness and shape of a line.
The **Tension** property gets or sets the tension of the curved lines.

Simple 2D Line Chart and Line XY Chart

Custom properties are as follows:

The **Line** property gets or sets the line element. Used to set color, thickness and shape of a line.
The **LineJoin** property sets the type of join to draw when two lines connect.

Pie and Doughnut Charts

Doughnut

The **Clockwise** property gets or sets a value indicating whether to display the data in clockwise order.
The **ExplodeFactor** property gets or sets the amount of separation between data point values.
The **HoleSize** property gets or sets the inner radius of the chart.
The **OutsideLabels** property gets or sets a value indicating whether the data point labels appear outside the chart.
The **StartAngle** property gets or sets the horizontal start angle for the series.

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

Visual Basic

```
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("Radius") = 2.0F
Me.ChartControl1.Series(0).Properties("StartAngle") = 0.0F
```

C#

```
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["Radius"] = 2.0f;
this.ChartControl1.Series[0].Properties["StartAngle"] = 0f;
```

Funnel

The **CalloutLine** property 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.
The **FunnelStyle** property gets or sets the Y value for the series points to the width or height of the funnel. The default value is YIsHeight.
The **MinPointHeight** property gets or sets the minimum height allowed for a data point in the funnel chart. The height is measured in relative coordinates.
The **NeckHeight** property 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.
The **NeckWidth** property 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.
The **OutsideLabels** property gets or sets a value indicating whether the labels are placed outside of the funnel chart. The default value is True.
The **OutsideLabelsPlacement** property 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.
The **PointGapPct** property 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.

C#

```
using GrapeCity.ActiveReports.Chart;
using GrapeCity.ActiveReports.Chart.Graphics;
```

C#

```
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;
```

Pyramid

The **CalloutLine** property 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.
The **MinPointHeight** property gets or sets the minimum height allowed for a data point in the pyramid chart. The height is measured in relative coordinates.
The **OutsideLabels** property gets or sets a value indicating whether the labels are placed outside of the pyramid chart. The default value is True.
The **OutsideLabelsPlacement** property 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.

The **PointGapPct** property 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.

3D Pie and Doughnut charts

Doughnut Chart

The **Clockwise** property gets or sets a value indicating whether to display the data in clockwise order.
The **ExplodeFactor** property 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 to the data point instead of the series.
The **HoleSize** property 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.
The **OutsideLabels** property gets or sets a value indicating whether the data point labels appear outside the chart.
The **Radius** property gets or sets the size of the doughnut within the chart area.
The **StartAngle** property 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.

Visual Basic

```
Me.ChartControl1.Series(0).Properties("ExplodeFactor") = 0.0F
Me.ChartControl1.Series(0).Properties("HoleSize") = 0.33F
Me.ChartControl1.Series(0).Properties("OutsideLabels") = False
Me.ChartControl1.Series(0).Properties("Radius") = 2.0F
Me.ChartControl1.Series(0).Properties("StartAngle") = 50.0F
```

C#

```
this.chartControl1.Series[0].Properties["ExplodeFactor"] = 0f;
this.chartControl1.Series[0].Properties["HoleSize"] = .33f;
this.chartControl1.Series[0].Properties["OutsideLabels"] = false;
this.chartControl1.Series[0].Properties["StartAngle"] = 50f;
```

Funnel Chart

The **BaseStyle** property gets or sets a circular or square base drawing style for the 3D funnel chart.
The **CalloutLine** property 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.
The **FunnelStyle** property gets or sets the Y value for the series points to the width or height of the funnel. The default value is YIsHeight.
The **MinPointHeight** property gets or sets the minimum height allowed for a data point in the funnel chart. The height is measured in relative coordinates.
The **NeckHeight** property 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.
The **NeckWidth** property 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.
The **OutsideLabels** property gets or sets a value indicating whether the labels are placed outside of the funnel chart.

The default value is True.
The **OutsideLabelsPlacement** property 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.
The **PointGapPct** property 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.
The **RotationAngle** property 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.

Visual Basic

```
Imports GrapeCity.ActiveReports.Chart
Imports GrapeCity.ActiveReports.Chart.Graphics
```

Visual Basic

```
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
```

C#

```
using GrapeCity.ActiveReports.Chart;
using GrapeCity.ActiveReports.Chart.Graphics;
```

C#

```
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;
```

Pyramid Chart

The **BaseStyle** property gets or sets a circular or square base drawing style for the 3D pyramid chart.
The **CalloutLine** property 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.
The **MinPointHeight** property gets or sets the minimum height allowed for a data point in the pyramid chart. The height is measured in relative coordinates.
The **OutsideLabels** property gets or sets a value indicating whether the labels are placed outside of the pyramid chart. The default value is True.
The **OutsideLabelsPlacement** property 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.
The **PointGapPct** property 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.
The **RotationAngle** property 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.

Visual Basic

```
Imports GrapeCity.ActiveReports.Chart
Imports GrapeCity.ActiveReports.Chart.Graphics
```

Visual Basic

```
With Me.ChartControl1.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
```

C#

```
using GrapeCity.ActiveReports.Chart;
using GrapeCity.ActiveReports.Chart.Graphics;
```

C#

```
this.ChartControl1.Series[0].Properties["BaseStyle"] = BaseStyle.SquareBase;
this.ChartControl1.Series[0].Properties["MinPointHeight"] = 10f;
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;
```

Candle Stick Chart

The **BodyDownswingBackdrop** property 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.
The **BodyUpswingBackdrop** property 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.
The **BodyWidth** property gets or sets the width of the rectangle used to show upswing or downswing.
The **WickLine** property 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 Pyramid chart.

Visual Basic

```
Imports GrapeCity.ActiveReports.Chart.Graphics
```

Visual Basic

```
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
```

C#

```
using GrapeCity.ActiveReports.Chart.Graphics;
```

C#

```
this.ChartControl1.Series[0].Properties["BodyDownswingBackdrop"]= new Chart.Graphics.Backdrop
(Color.FromArgb(255, 192, 255));
this.ChartControl1.Series[0].Properties["BodyUpswingBackdrop"] = new Chart.Graphics.Backdrop
(Color.FromArgb(192, 192, 255));
this.ChartControl1.Series(0).Properties("WickLine") = new Chart.Graphics.Line(Color.Indigo);
this.ChartControl1.Series[0].Properties["BodyWidth"] = 7f;
```

HiLo Chart

The HiloLine property 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.

Visual Basic

```
Me.ChartControl1.Series(0).Properties("HiloLine") = New
GrapeCity.ActiveReports.Chart.Graphics.Line(Color.DeepSkyBlue, 4)
```

C#

```
this.ChartControl1.Series[0].Properties["HiloLine"] = new
GrapeCity.ActiveReports.Chart.Graphics.Line(Color.DeepSkyBlue, 4);
```

Point and Figure Chart

The **BoxSize** property gets or sets the amount a price must change in order to create another X or O.
The **DownswingLine** property gets or sets the style and color settings for the downswing O's.
The **ReversalAmount** property gets or sets the amount that a price must shift in order for a new column to be added.
The **UpswingLine** property gets or sets the style and color settings for the upswing X's.

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

Visual Basic

```
Imports GrapeCity.ActiveReports.Chart.Graphics
```

Visual Basic

```
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
```

C#

```
using GrapeCity.ActiveReports.Chart.Graphics;
```

C#

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

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

Visual Basic

```
Imports GrapeCity.ActiveReports.Chart.Graphics
```

Visual Basic

```
With Me.ChartControl1.Series(0)
    .Properties("BodyDownswingBackdrop") = New Backdrop(Color.BlueViolet)
    .Properties("BodyUpswingBackdrop") = New Backdrop(Color.Navy)
```

```
    .Properties("BoxSize") = 3.0F
End With
```

C#

```
using GrapeCity.ActiveReports.Chart.Graphics;
```

C#

```
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;
```

Kagi Chart

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

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

Visual Basic

```
Imports GrapeCity.ActiveReports.Chart.Graphics
```

Visual Basic

```
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
```

C#

```
using GrapeCity.ActiveReports.Chart.Graphics;
```

C#

```
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;
```

Stock Chart

The **CloseLine** property gets or sets the information for the close value line.
The **HiLoLine** property gets or sets the line information for the HiLo line.
The **OpenLine** property property gets or sets the information for the open value line.
The **TickLen** property property gets or sets the tick length for the close value and open value lines.

Stock Close Only Chart

The **CloseLine** property gets or sets the information for the close value line.
The **HiLoLine** property gets or sets the line information for the HiLo line.
The **OpenLine** property property gets or sets the information for the open value line.
The **TickLen** property property gets or sets the tick length for the close value and open value lines.

Stock Open Only Chart

The **CloseLine** property gets or sets the information for the close value line.
The **HiLoLine** property gets or sets the line information for the HiLo line.
The **OpenLine** property property gets or sets the information for the open value line.
The **TickLen** property property gets or sets the tick length for the close value and open value lines.

Three Line Break Chart

The **BodyDownswingBackdrop** property gets or sets the style and color settings for the downswing boxes.
The **BodyUpswingBackdrop** property gets or sets the style and color settings for the upswing boxes.
The **NewLineBreak** property 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.

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

Visual Basic

```
Imports GrapeCity.ActiveReports.Chart.Graphics
```

Visual Basic

```
With Me.ChartControl1.Series(0)
    .Properties("BodyDownswingBackdrop") = New Backdrop(Color.Red)
    .Properties("BodyUpswingBackdrop") = New Backdrop(Color.Black)
    .Properties("NewLineBreak") = 3
End With
```

C#

```
using GrapeCity.ActiveReports.Chart.Graphics;
```

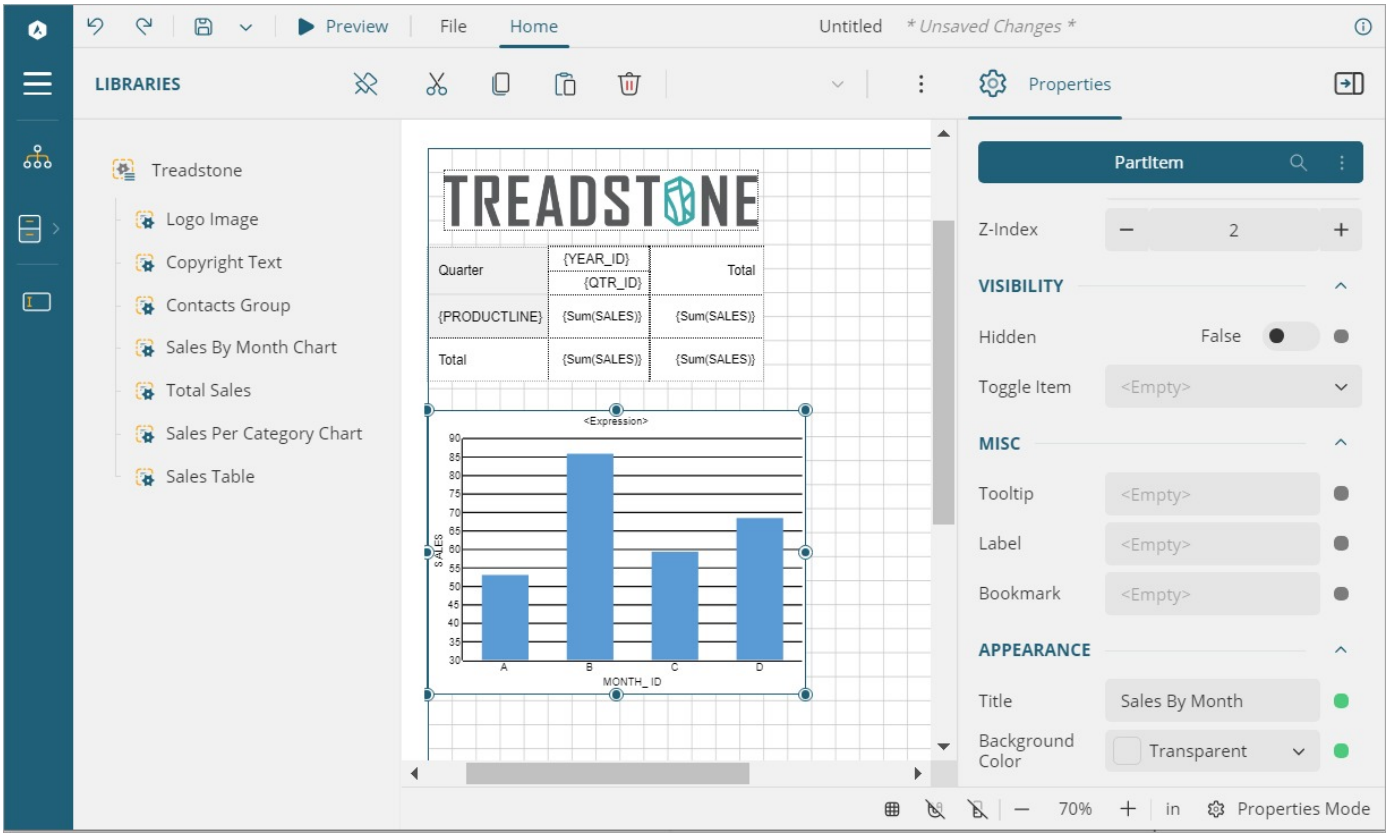
C#

```
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;
```

Report Parts

Introduction

ActiveReports.NET includes the web designer component that you can integrate into your web-application to allow end-users to create and modify reports. The web designer covers most of the functionality of the VS.NET integrated designer and the standalone report designer; it allows configuring data binding, adding report items, and setting their properties. However, end-users of your application might not have extensive knowledge of data binding, report items, and surrounding topics. To streamline the user experience, ActiveReports.NET offers the Report Parts feature. Report Parts are basically pre-configured report items that end-users could drag-and-drop into a report body and customize further. The available customization properties are also defined in a report part.



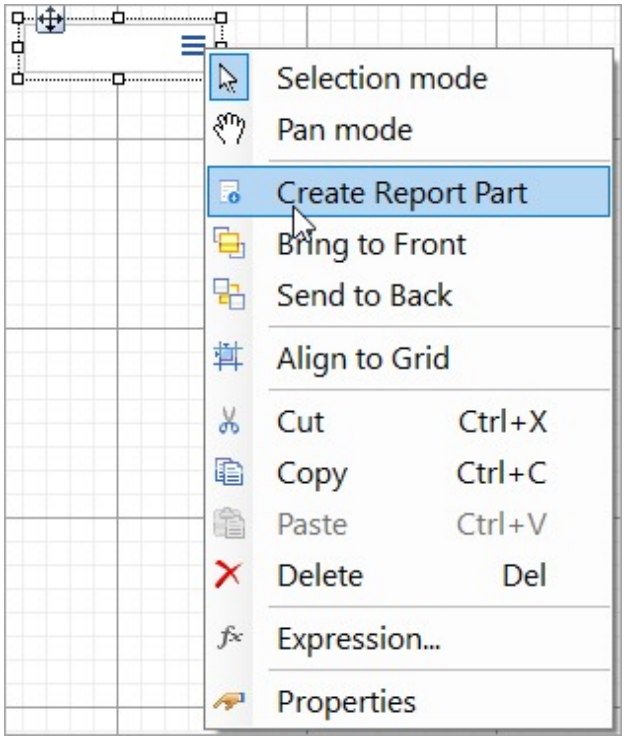
Report Libraries

A Report Library is a report that contains one or more Report Parts. You can build multiple report libraries for different user groups. If an end-user adds a report part to a target report, then it includes a link to the corresponding report library. You can then modify report parts definitions within a report library, and modifications will be automatically picked up for all target reports that use these report parts. Note that the data sources, datasets, and parameters are copied to a target report. Hence, modifying these items in the report library won't immediately affect the target reports.

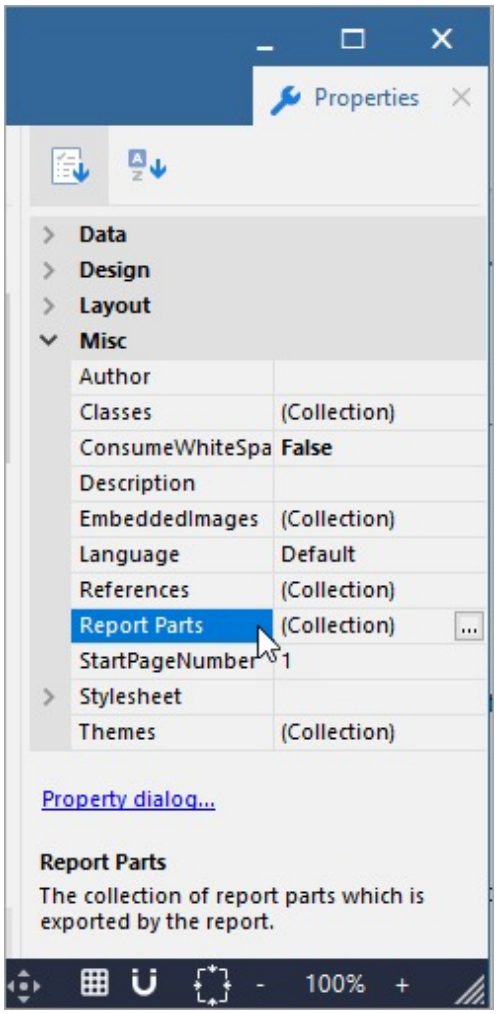
Creating Report Libraries

To create a report library, you can create a new or open an existing RDLX Report in the standalone designer application, or in the VS.NET integrated designer. Once you decided which report items should be exposed as report parts, you can initialize a new report part using the following methods:

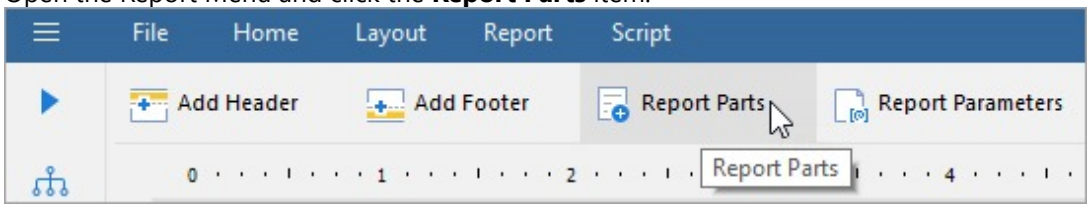
- Select one or more report items, right-click the selection, and in the context menu, choose the **Create Report Part** menu.



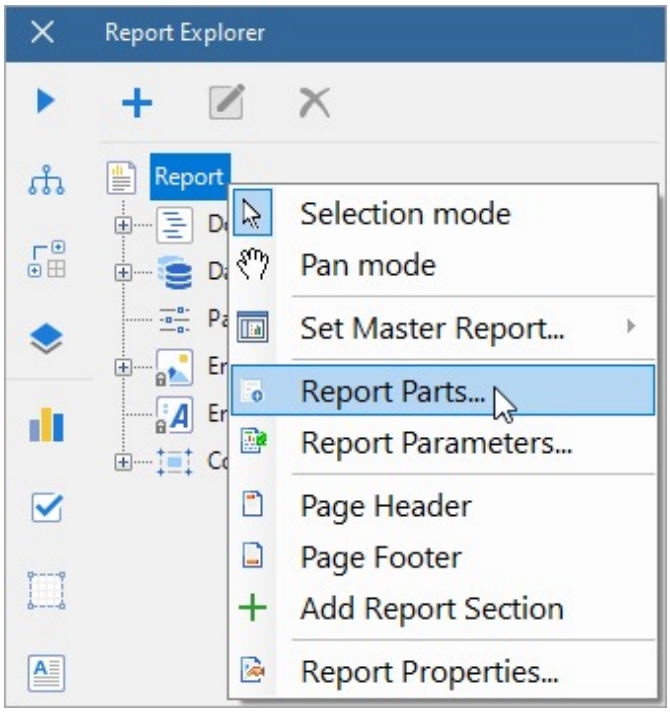
- Select the Report object by clicking anywhere outside the report body, and open the **Report Parts** collection from the Properties Panel.



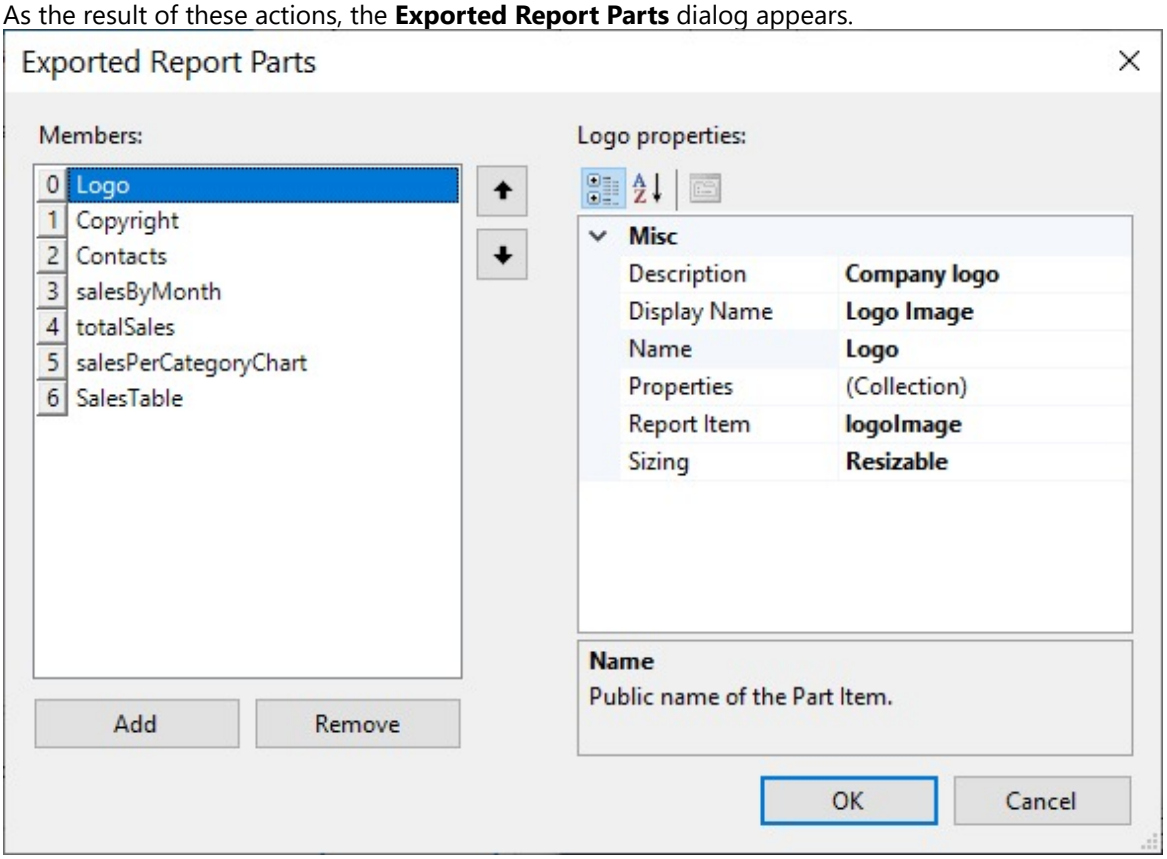
- Open the Report Menu and click the **Report Parts** item.



- Right-click the Report node in the Report Explorer and select the **Report Parts...** item.



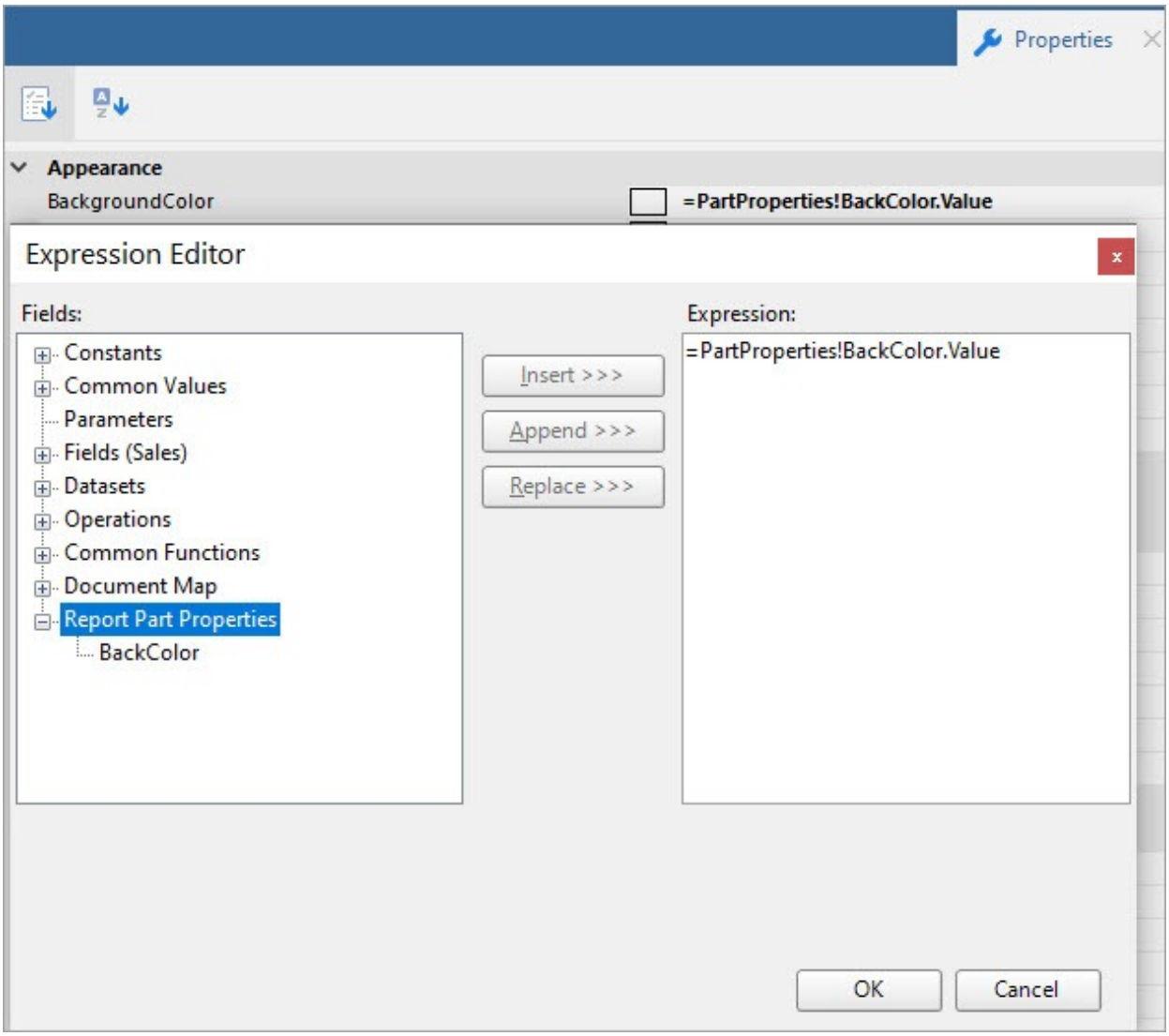
As the result of these actions, the **Exported Report Parts** dialog appears.



If you open the Export Report Parts dialog via the context menu of an existing report item, the new report part linked to that report item will be automatically created. In all other cases, you can add report parts manually. Each report part exposes the following properties:

- **Description** - the description of a report part that will be visible for end-users. You can use this property to provide additional information for them.
- **Display name** - the name of a report part that will be visible for end-users.
- **Name** - the unique, internal name of a report part. Note that it should be a CLI-compliant identification.
- **Properties** - A collection of custom properties that should be available for end-users to customize a report part. The configuration of a custom property includes:
  - **Category** - the category under which a custom property appears in the property grid of the web designer. You can group custom properties by using categories.
  - **Default value** - the initial value of a custom property.
  - **Description** - the description of a custom property that will be visible for end-users in the web designer property grid.
  - **Display Name** - the name of a custom property that will be useful for end-users in the web designer property grid.
  - **Name** - the unique, internal name of a custom property. Note that it should be a CLI-compliant identification.
  - **Property Type** - identifies the editor of a custom property that will be available for end-users. For example, if you want to allow end-users to modify the background color of a report item, you can set the type of the corresponding custom property to 'Color' and the color editor will appear in the web designer property grid.
- **Report Item** - the link to the report item that is represented by a report part.
- **Sizing** - Identifies whether end-users could resize a report part after adding it to a report.

After defining a custom property, you can refer to its value by using the **PartProperties** collection. For example, if you have defined the 'BackColor' custom property in a report part, you can set the Background Color of a report item to the `=PartProperties.BackColor.Value` expression. The custom properties for all report parts are available via the expression editor.



Integrating Report Libraries

After creating a report library and saving it (it is just an RDLX report), you can save it within the resources of a web-application that uses the web designer. The recommended way is to keep the report libraries the same way you keep other reports – in the file system or a database. Then you can list the report libraries that should be available for end-users via the ReportPartsSettings API. Here is the sample of code:

Index.html

```
<html lang="en">
<head>
    @*get these files from the installation location for the package:
\node_modules\@mescius\activereportsnet-designer\dist*@
    <link rel="stylesheet" href="web-designer.css" />
    <script src="web-designer.js"></script>
```

```
</head>
<body>
    <div id="web-designer-host">
    </div>
<script>
        GrapeCity.ActiveReports.Designer.create('#web-designer-host', {
            rdlx: {
                reportParts: {
                    enabled: true,
                    libraries: [{
                        name: 'Sales',
                        path: 'Libraries/Lib_Sales.rdlx'
                    },
                    {
                        name: 'Marketing',
                        path: 'Libraries/Lib_Marketing.rdlx'
                    }]
                }
            }
        });
    </script>
</body>
</html>
```

After running the application, the web designer will have the Libraries' panel that lists available libraries and the report parts. End-users will be able to drag-and-drop the report parts, modify the custom properties. Upon previewing a report, the report parts will be rendered according to its definition with the customization applied.

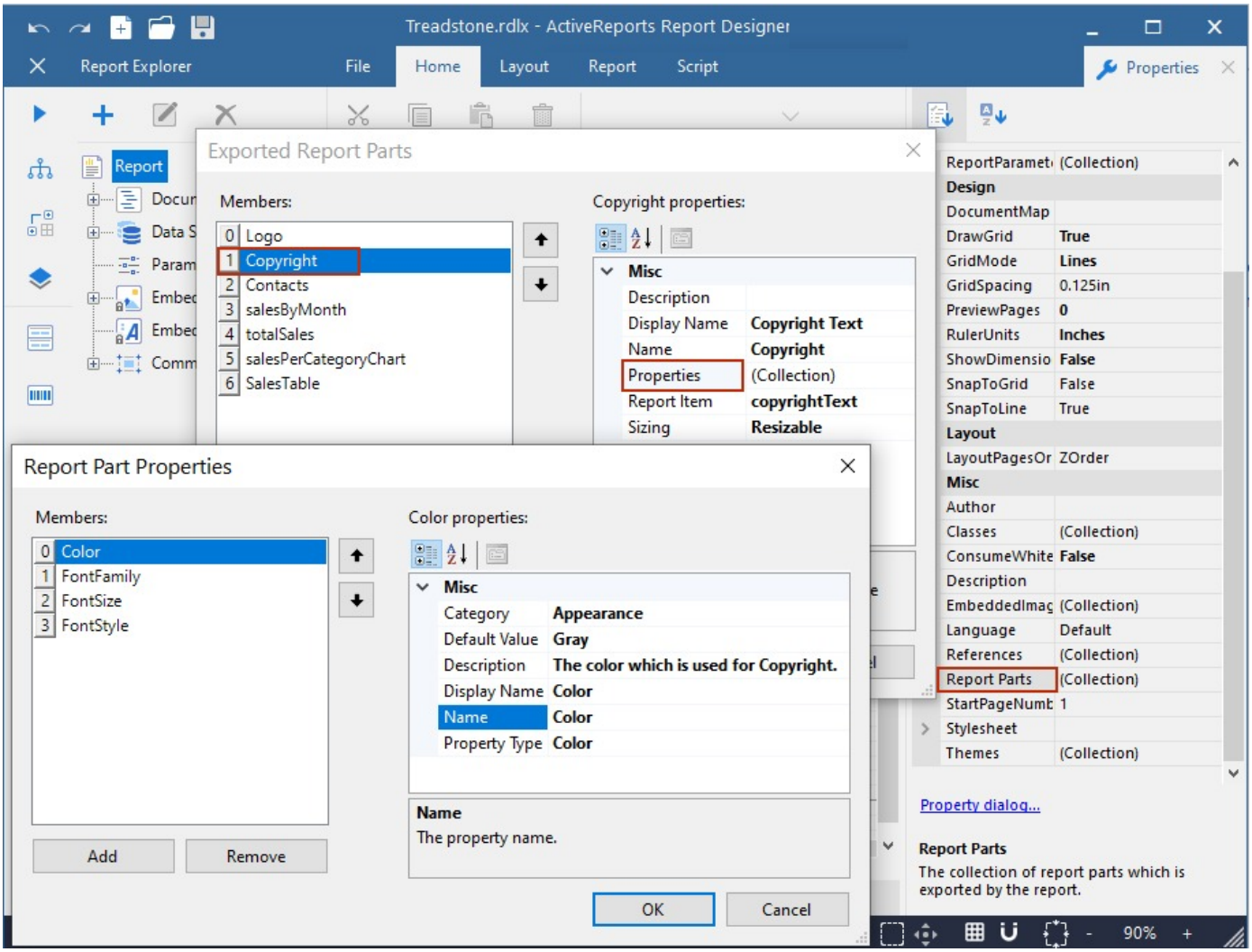
Report Parts Sample

You can download the example of report parts usage from the ActiveReports GitHub repository.

Upon running the sample, you can see a customized web designer with almost all standard UI removed and the report libraries panel available and opened by default.

To observe how a library is created, go to the **File** menu > **libraries** and open 'Treadstone.rdlx' report. This is a multi-section report with three sections Cards, Tables, and Charts. All report parts that are available in this library can be viewed from the **Report Parts** property of the report.

The following image shows the report parts present in 'Treadstone' library in the **Exported Report Parts** dialog and the properties set for one of the report part in the **Report Part Properties** dialog, in desktop designer.



The 'Treadstone' library contains report parts as shown in the table below. The table describes the custom property of each report part, the original report item linked to the report part, and the expression used for referencing the report part properties into the report item's property.

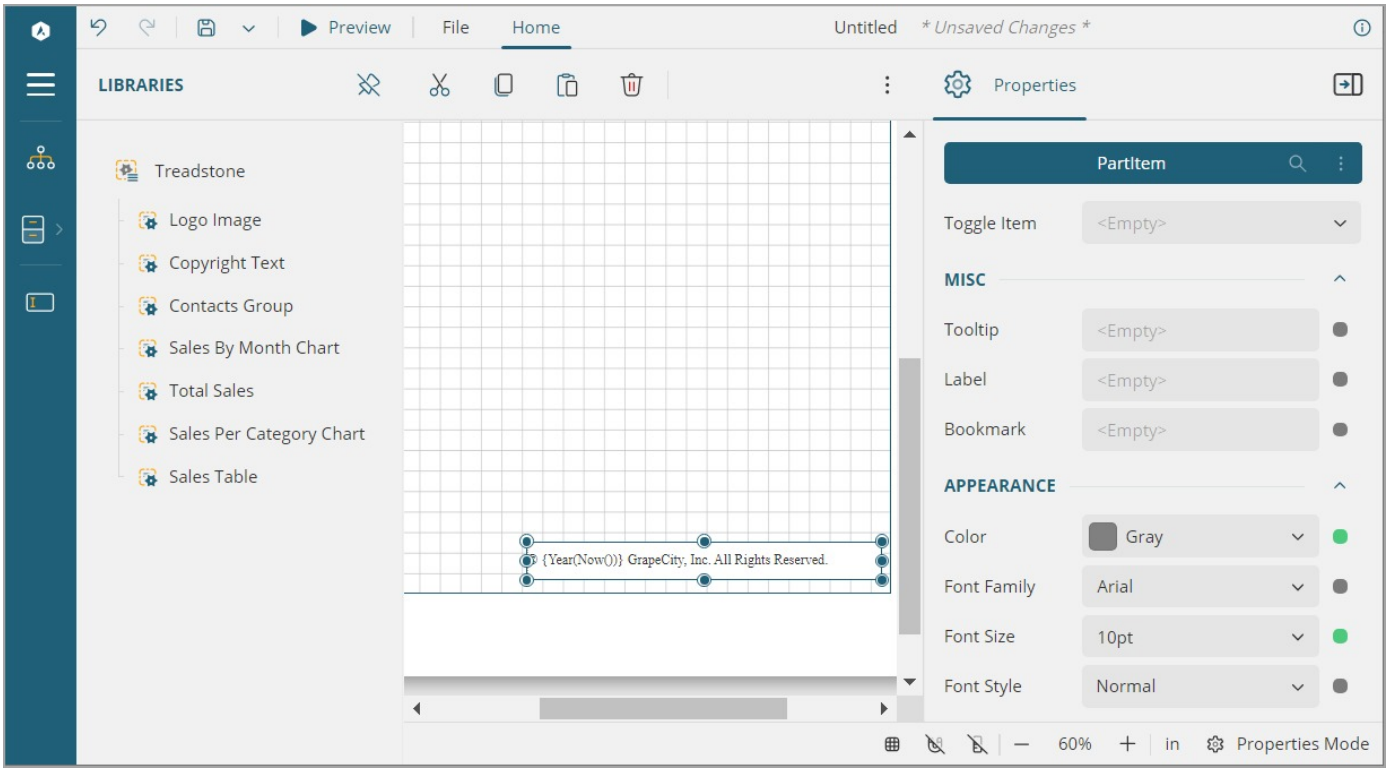
Report Part	Custom properties of the Report Part*	Linked Original Report Item	Corresponding Report Item property and the Expression (in WebDesigner)
Logo	BorderColor	Image	Border > Color > Default {PartProperties!BorderColor.Value}
	BorderStyle		Border > Style > Default {PartProperties!BorderStyle.Value}
	BorderWidth		Border > Width > Default {PartProperties!BorderWidth.Value}
Copyright	Color	TextBox	Text > Color {PartProperties!Color.Value}

	FontFamily		Text > Font Family {PartProperties!FontFamily.Value}
	FontSize		Text > Font Size {PartProperties!FontSize.Value}
	FontStyle		Text > Font Style {PartProperties!FontStyle.Value}
Contacts	Color	Container. The properties are applied on the text boxes it contains.	Text > Color {PartProperties!Color.Value}
	FontFamily		Text > Font Family {PartProperties!FontFamily.Value}
	FontSize		Text > Font Size {PartProperties!FontSize.Value}
	FontStyle		Text > Font Style {PartProperties!FontStyle.Value}
salesByMonth	Title	Chart	Select Chart's Header > Caption {PartProperties!Title.Value}
	BackgroundColor		Background >Color {PartProperties!BackgroundColor.Value}
totalSales	Caption	FormattedText	The corresponding expressions are used in the 'Html' property as shown. `<html><body style='margin:0px'><center><p style='color:{PartProperties!ValueColor.Value};font-family:"{PartProperties!ValueFont.Value}";font-size:{PartProperties!ValueFontSize.Value};font-style:{PartProperties!ValueFontStyle.Value}'>{FormatCurrency(Sum(SALES))}</p><p style='color:{PartProperties!CaptionColor.Value};font-family:"{PartProperties!CaptionFont.Value}";font-size:{PartProperties!CaptionFontSize.Value};font-style:{PartProperties!CaptionFontStyle.Value}'>{PartProperties!Caption.Value}</p></center></body></html>`
	ValueFont		
	ValueFontSize		
	ValueFontStyle		
salesPerCategoryChart	Title	Chart	Select Chart's Header > Caption {PartProperties!Title.Value}
	BackgroundColor		Background > Color

			{PartProperties!BackgroundColor.Value}
SalesTable	HeaderTextColor	Tablix	Select Table Header's TextBox: Text > Color {PartProperties!HeaderTextColor.Value}
	HeaderBgColor		Background > Color {PartProperties!HeaderBgColor.Value}
	HeaderBorderColor		Border > Color > Default {PartProperties!HeaderBorderColor.Value}

*For complete list of custom properties in each report part, check the sample.

Try drag-and-dropping report parts from the Libraries panel to the report body, and modifying the custom properties, and previewing the result.



Create Applications

This section provides some essential guidelines on how to create simple ActiveReports applications.

We recommend that you first check our ActiveReports samples.

.NET Viewer Application

You have an option of previewing the report as you create it in a Visual Studio project at design time. With the in-built

Viewers for Windows Forms and Web, you can view your report in any application based on these platforms in a separate viewer control. You will learn all the available report viewing options in .NET Viewer application in this section of the documentation.

WPF Viewer

To add Viewer Control to your WPF application,

1. In Visual Studio 2022, create a new WPF Application or open an existing one.
2. Install NuGet package for 'MESCIUS.ActiveReports.Viewer.Wpf'. For more information on installing NuGet packages, see Manage ActiveReports Dependencies topic.
3. Open **XAML Designer** (MainWindow.xaml) to verify the **Viewer** control showing up in the Visual Studio Toolbox under 'WPF Viewer' tab.
   Make sure that the XAML Hot Reload is enabled. See Troubleshooting for more information.
4. From the Toolbox ActiveReports 18 tab, drag the **Viewer** control and drop it on the design view of MainWindow.xaml.

   > **Note:** Dragging the **Viewer** control to the design view of MainWindow.xaml automatically adds the corresponding reference to the licenses.licx file.

On using the Page, RDLX, or Section report template, a new ActiveReports 18 tab is automatically added in the toolbox with the controls in sync with the references.

Load Reports

To load and display a report using the Viewer control in a WPF application,

1. Create a new a WPF application in Visual Studio.
2. From the Visual Studio toolbox, drag the **Viewer** control onto your XAML Form.
3. Set the viewer's **Dock** property to **Fill** to show the complete Viewer control on the Form.
4. In the Solution Explorer, select the report you have created and from the Properties window, set **Copy to Output Directory** to **Copy Always**.

   > **Note:** If Page/RDLX report is selected from dropdown of Source property (containing relative path), 'Copy to Output Directory' for selected report should be set as 'Copy always/Copy if newer' otherwise error "Could not find file ... " appears on loading WPF Viewer.

   > **Caution:** In WPF Viewer control, previewing code-based Section reports using **Source ('Source Property' in the on-line documentation)** and **SourceFormat ('SourceFormat Property' in the on-line documentation)** properties is not supported.

5. On MainWindow.xaml, with the viewer selected, go to the Properties window and double click the Loaded event.
6. In the MainWindow code view that appears, add code like the following to the viewer1_loaded event to bind the report to the viewer. This code shows an .rdlx report being loaded, but you can use an .rpx report as well.

Visual Basic.NET code. Paste INSIDE the viewer1_Loaded event in MainWindow.xaml.vb.
`Viewer1.LoadDocument("rptSingleLayout.rdlx")`

C# code. Paste INSIDE the viewer1_Loaded event in MainWindow.xaml.cs.

```
viewer1.LoadDocument("rptSingleLayout.rdlx");
```

Note:

- Refer to the **LoadDocument ('LoadDocument Method' in the on-line documentation)** method to see other ways to load a report in WPF Viewer.
- We can set report for WPFViewer directly on XAML file and load the report in WPF Viewer using **Source ('Source Property' in the on-line documentation)** and **SourceFormat ('SourceFormat Property' in the on-line documentation)** properties.
- To avoid evaluation banners appearing at run time, license your ActiveReports WPF Application project. You can find information on licensing in License Your ActiveReports.

Customize the WPF Viewer

Let us see how we can customize different elements of WPF Viewer.

Add the Customization Template to the WPF Project

The ActiveReports WPF Viewer is a customizable control. You can easily change the look of the WPF Viewer and its elements, such as the error panel, search panel, sidebar and toolbar by modifying properties in the default WPF Viewer template (DefaultWPFiewerTemplates.xaml).

Once you copy the template, you can modify the default style and apply it to your viewer by creating the desired style and setting its Template property.

Note that while modifying the template, do not remove the required template parts. The required parts are usually marked with the prefix "PART_".

The template allows you to customize the appearance of viewer and take advantage of XAML-based styling.

1. Open your WPF project.
2. In Solution Explorer, select the *YourProjectName* node.
3. On the Visual Studio **Project** menu, click **Add Existing Item**.
4. In the dialog that appears, locate and select DefaultWPFViewerTemplates.xaml and click **OK**. You can find DefaultWPFViewerTemplates.xaml at *C:\Program Files\MESCIUS\ActiveReports 18\Deployment\WPF\Templates* folder (on a **64-bit Windows operating system**, this file is located in *C:\Program Files (x86)\MESCIUS\ActiveReports 18\Deployment\WPF\Templates*.
5. On MainWindow.xaml before the opening <Grid> tag, add the following code.

Paste to the XAML view of MainWindow.xaml before the opening <Grid> tag

```
<Window.Resources>
<ResourceDictionary Source="DefaultWPFViewerTemplates.xaml" />
</Window.Resources>
```

Customize the WPF Viewer Sidebar

Modify Thumbnails

1. In Solution Explorer, double-click DefaultWPFViewerTemplates.xaml.

2. In the file that opens, search for **"thumbnails tab"**.
3. In the **GroupBox Header** property of <!-- thumbnails tab -->, remove "{Binding Sidebar.ThumbnailsPane.Text}" and type **"THUMBNAILS"**.

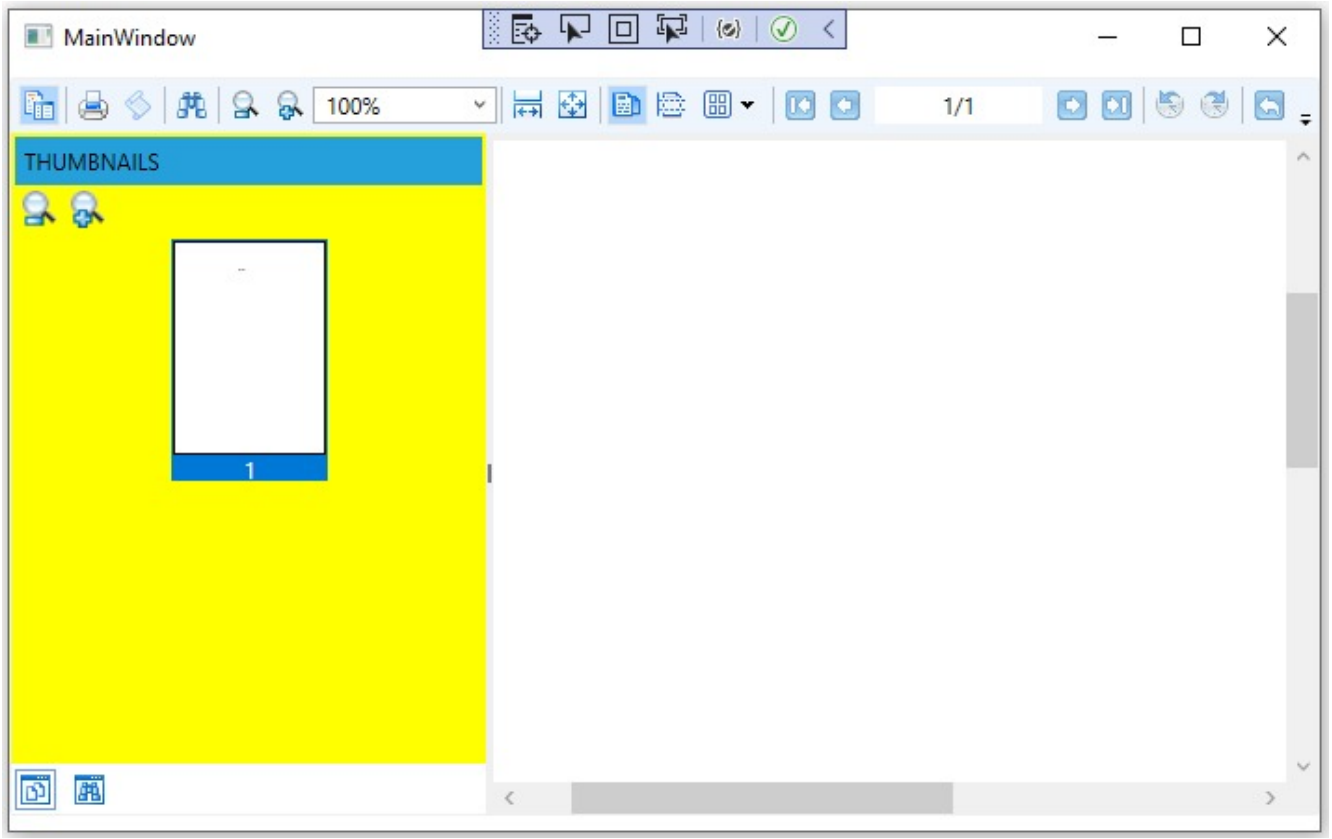
```
<!-- thumbnails tab -->
<TabItem Visibility="{Binding Sidebar.ThumbnailPanelVisible, Converter={StaticResource VisibilityConverter}}" IsEnable
  <GroupBox Header="THUMBNAILS" AutomationProperties.AutomationId="PageThumbnailsTab">
    <View:ThumbnailPanel x:Name="ThumbnailPanel" DataContext="{Binding Sidebar.ThumbnailsPane}" Grid.Row="1" />
  </GroupBox>
  <TabItem.Header>
    <Image Style="{StaticResource TabButtonImage}" AutomationProperties.AutomationId="ThumbnailsTabIconButton" Source=
  </TabItem.Header>
</TabItem>
```

Modify Background Color of the Sidebar

4. Search for **"TabControl x:Name="Sidebar"**.
5. Add **Background="Yellow"** after TabControl x:Name="Sidebar".

```
<!-- sidebar -->
<TabControl x:Name="Sidebar"  Background="Yellow" DockPanel.Dock="Left"
  <TabControl.Resources>
    <Style TargetType="TabItem">
      <Setter Property="Template">
        <Setter.Value>
          <ControlTemplate TargetType="TabItem">
```

6. Press **F5** to see the customized viewer sidebar.



Add a customized button to the WPF Viewer toolbar

1. In Solution Explorer, select the *YourProjectName* node.
2. On the Visual Studio Project menu, select **Add New Item**.
3. In the **Add New Item** dialog that appears, select **Class**, rename it to **MyCommand** and click **Add**.
4. In the MyCommand.cs/vb that opens, add the following code to implement a command.

Visual Basic.NET code. Add to MyCommand.vb

```
Implements ICommand
Public Function CanExecute(ByVal parameter As Object) As Boolean Implements
System.Windows.Input.ICommand.CanExecute
    Return True
End Function
Public Event CanExecuteChanged(ByVal sender As Object, ByVal e As System.EventArgs) Implements
System.Windows.Input.ICommand.CanExecuteChanged

Public Sub Execute(ByVal parameter As Object) Implements System.Windows.Input.ICommand.Execute
    MessageBox.Show("MESCIUS is the world's largest component vendor.", "About Us",
MessageBoxButton.OK)
End Sub
```

C# code. Add after the statement using System.Text

```
using System.Windows.Input;
using System.Windows;
```

C# code. Add after the statement using System.Text

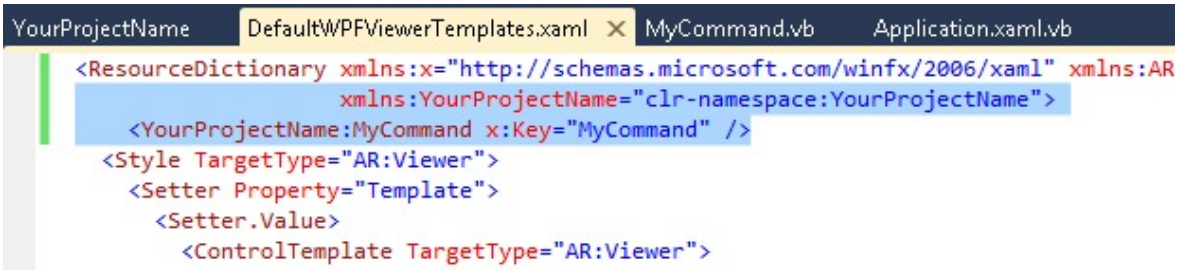
```
public class MyCommand : ICommand
    {
        public bool CanExecute(object parameter)
        {
            return true;
        }

        public void Execute(object parameter)
        {
            MessageBox.Show("MESCIUS is the world's largest component vendor.", "About Us",
MessageBoxButton.OK);
        }

        public event EventHandler CanExecuteChanged;
    }
```

5. In Solution Explorer, double-click DefaultWpfViewerTemplates.xaml.
6. In the file that opens, add the following code.

XML code. Add to DefaultWpfViewerTemplates.xaml

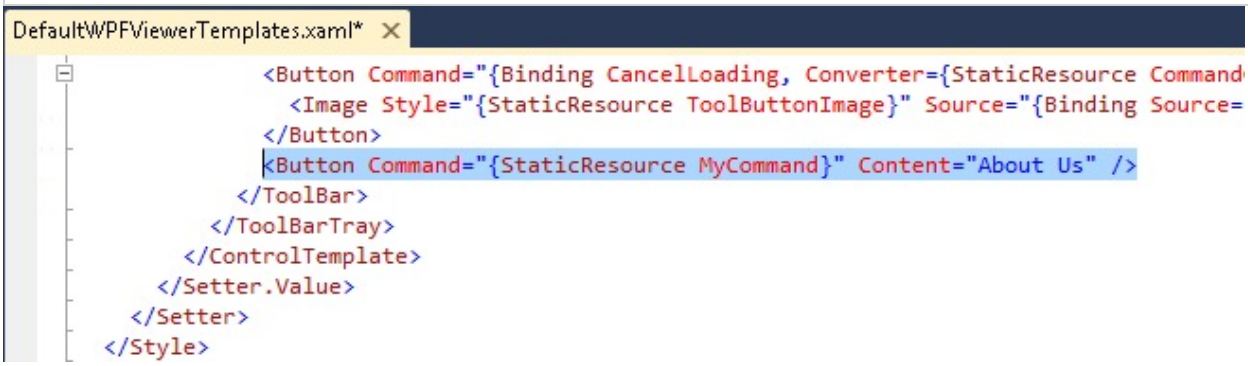
```
<ResourceDictionary>
 ...
xmlns:YourProjectName="clr-namespace:YourProjectName">
<YourProjectName:MyCommand x:Key="MyCommand" />
...
```

```
</ResourceDictionary>
```



7. In the same file, add the following code to add a button.

XML code. Add to DefaultWpfViewerTemplates.xaml before the closing Toolbar tag	Copy Code
`<Button Command="{StaticResource MyCommand}" Content="About Us" />`	

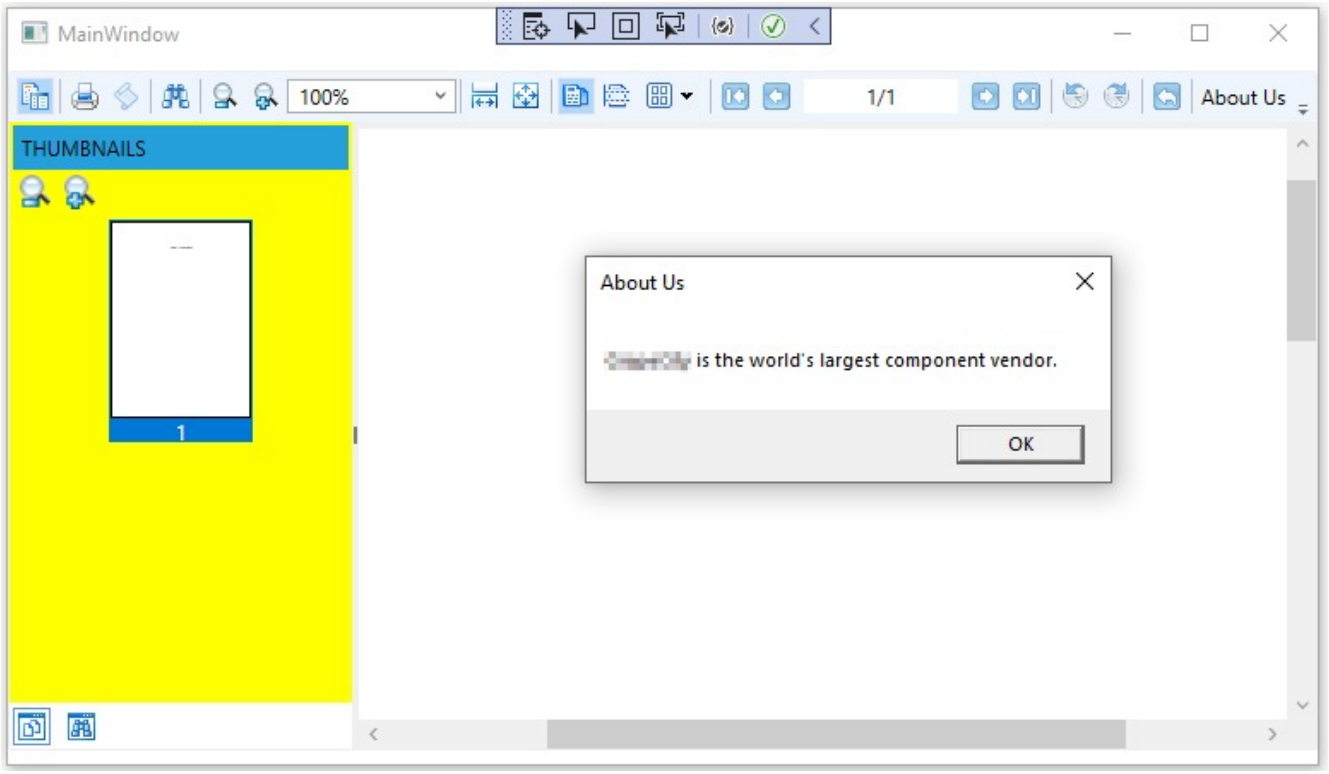


8. Press F5 to see the new customized button **About Us** in the Viewer toolbar.

Remove the Refresh button from the WPF Viewer toolbar

1. In Solution Explorer, double-click DefaultWpfViewerTemplates.xaml.
2. In the file that opens, search for **"<!--Refresh btn-->"**.
3. Replace the existing content in **Visiblity="..."** with the following.

XML code. Add to DefaultWpfViewerTemplates.xaml	Copy Code
`<Button Command=... Visibility="Collapsed">`	



Windows Forms Viewer

On using the Page, RDLX, or Section report template, the Viewer control is already available on the Windows Form. The control is also available on the Toolbox under ActiveReports 18 tab, with the controls in sync with the references.

To add Viewer control to your WinForms Application,

1. In Visual Studio 2022, create a new Windows Forms App or open an existing one.
2. Install NuGet package for 'MESCIUS.ActiveReports.Viewer.Win'. For more information on installing NuGet packages, see Manage ActiveReports Dependencies topic.
3. Open **WinForms Designer** (Form1.cs or Form1.vb) to verify the **Viewer** control showing up in the Visual Studio Toolbox under 'ActiveReports 18' tab.
   To show the Visual Studio Toolbox, navigate to **View** > **Toolbox**.

Load Reports

To load and display a Page, RDLX, and Section report output using the Viewer control in the Windows Forms application.

1. Create a new Windows Forms application in Visual Studio.
2. From the Visual Studio toolbox, drag the **Viewer** control onto your Windows Form Designer.
3. Set the viewer's **Dock** property to **Fill** to show the complete Viewer control on the Form.
4. Double-click the title bar of the Form to create an event-handling method for the Form_Load event.
5. In the Form_Load event, add the code as follows to run the report and display it in the viewer.

Load Page/RDLX report

C#

```
string file_name = @"..\..\ReportName.rdlx";
GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(new System.IO.FileInfo(file_name));
GrapeCity.ActiveReports.Document.PageDocument pageDocument = new
GrapeCity.ActiveReports.Document.PageDocument(pageReport);
viewer1.LoadDocument(pageDocument);
```

VB.NET

```
Dim file_name As String = "..\..\ReportName.rdlx"
Dim pageReport As New GrapeCity.ActiveReports.PageReport(New
System.IO.FileInfo(file_name))
Dim pageDocument As New GrapeCity.ActiveReports.Document.PageDocument(pageReport)
Viewer1.LoadDocument(pageDocument)
```

Load Code-based Section Report

C#

```
SectionReport1 sectionReport = new SectionReport1();
viewer1.LoadDocument(sectionReport);
```

VB.NET

```
Dim sectionReport As New SectionReport1()
Viewer1.LoadDocument(sectionReport)
```

Load Xml-based Section Report

C#

```
GrapeCity.ActiveReports.SectionReport sectionReport = new
GrapeCity.ActiveReports.SectionReport();
System.Xml.XmlTextReader xtr = new
System.Xml.XmlTextReader(@"..\..\SectionReport1.rpx");
sectionReport.LoadLayout(xtr);
xtr.Close();
viewer1.LoadDocument(sectionReport);
```

VB.NET

```
Dim sectionReport As New GrapeCity.ActiveReports.SectionReport()
Dim xtr As New System.Xml.XmlTextReader("..\..\SectionReport1.rpx")
```

```
sectionReport.LoadLayout(xtr)
xtr.Close()
Viewer1.LoadDocument(sectionReport)
```

Caution: Refresh button gets disabled when you load a section report in the Viewer control through any of the following:

- **Document Property (on-line documentation)**
- **LoadDocument(SectionDocument) Method (on-line documentation)**
- **LoadDocument(String) Method (on-line documentation)**

Customize the WinForms Viewer Control

There are a number of ways in which you can customize the Viewer control to make it a perfect fit for your Windows application. You can add and remove buttons from the toolbars, add and remove menu items, create custom dialogs, and call these dialogs from custom click events.

Customize Viewer Toolbar

You can use the methods listed in the System.Windows.Forms.ToolStripItemCollection documentation on MSDN to customize each ToolStrip.

ToolStrip

The **ToolStrip** contains the following **ToolStripItems** by index number.

• 0 Toggle sidebar • 1 Separator • 2 Print • 3 Galley mode • 4 Separator • 5 Copy • 6 Find • 7 Separator • 8 Zoom out • 9 Zoom In	• 10 Current Zoom • 11 Separator • 12 Fit width • 13 Fit page • 14 Separator • 15 Single page • 16 Continuous mode • 17 Multipage mode • 18 Separator • 19 First page	• 20 Previous page • 21 Current • 22 Next page • 23 Last page • 24 Separator • 25 History back • 26 History forward • 27 Separator • 28 Back to parent • 29 Separator	• 30 Refresh • 31 Cancel button • 32 Separator • 33 Pan mode • 34 Copy select • 35 Snapshot • 36 Separator • 37 Annotations

You can access these ToolStripItems by index with the **Insert** and **RemoveAt** methods. Other methods, including **Add** and **AddRange**, are described in the System.Windows.Forms.ToolStripItemCollection documentation on MSDN.

ToolStrip Implementation

When you add a new item to a ToolStrip, you need to add an **ItemClicked** event handler and an **ItemClicked** event for the ToolStrip with the new item. At run time, when a user clicks the new ToolStrip item, they raise the ItemClicked event of the ToolStrip containing the item

Add the event handler to the **Load** event of the Form that contains the Viewer control, and use the IntelliSense Generate Method Stub feature to create the related event. For example, of the code to create an event hander, as explained in the below section.

You can modify both viewer's mouse mode and touch mode toolbars by accessing the index number corresponding to the toolbar button and set custom commands. For example, in the following sample codes we show customization specific to mouse mode toolbar and touch mode toolbar.

Create a Viewer Form and Load Report in Viewer

See Load Reports topic to create a WinForms application and preview a report in the Viewer.

Customize the Mouse Mode Toolbar

Lets display a custom print button as shown in the following image on the WinForms Viewer toolbar (mouse mode):

1. Add a second Windows Form (Form2.cs) 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 go to the Form Load event.
5. Add the following code to the Form Load event to remove the default print button and add your own.

C#

```
//Remove the original print button.
viewer1.Toolbar.ToolStrip.Items.RemoveAt(2);
//Remove the extra separator.
viewer1.Toolbar.ToolStrip.Items.RemoveAt(1);
//Add a new button to the end of the tool strip with the caption "Print."
ToolStripButton tsbPrint = new ToolStripButton("Print");
viewer1.Toolbar.ToolStrip.Items.Add(tsbPrint);
//Create a click event handler for the button.
tsbPrint.Click += new EventHandler(tsbPrint_Click);
```

VB.NET

```
'Remove the print button.
Viewer1.Toolbar.ToolStrip.Items.RemoveAt(2)
'Remove the extra separator.
Viewer1.Toolbar.ToolStrip.Items.RemoveAt(1)
'Add a new button to the end of the tool strip with the caption "Print."
Dim tsbPrint As New ToolStripButton("Print")
Viewer1.Toolbar.ToolStrip.Items.Add(tsbPrint)
'Create a click event handler for the button.
AddHandler tsbPrint.Click, AddressOf tsbPrint_Click
```

6. Add the following code to the Form class below the Load event to display 'frmPrintDlg' when a user clicks the custom print button.

C#

```
//Call the custom dialog from the new button's click event.
```

```
private void tsbPrint_Click(object sender, EventArgs e)
{
    this.CustomPrint();
}
//Call the custom print dialog.
private void CustomPrint()
{
    frmPrintDlg _printForm = new frmPrintDlg();
    _printForm.ShowDialog(this);
}
```

VB.NET

```
'Call the custom dialog from the new button's click event.
Private Sub tsbPrint_Click(sender As Object, e As EventArgs)
    Me.CustomPrint()
End Sub
'Call the custom print dialog.
Private Sub CustomPrint()
    Dim _printForm As New frmPrintDlg()
    _printForm.ShowDialog(Me)
End Sub
```

7. Press **F5** to run the project and click on the print button of main viewer to view custom print dialog.

Customize the Touch Mode Toolbar

1. Double-click in the title bar of the Viewer form to create a Form Load event.
2. Add the following code to add a custom Zoom Out button.

C#

```
viewer1.TouchMode = GrapeCity.Viewer.Common.Model.TouchMode.True;

var zoomOutButton = viewer1.TouchModeToolbar.ToolStrip.Items[8];
zoomOutButton.Visible = true;
```

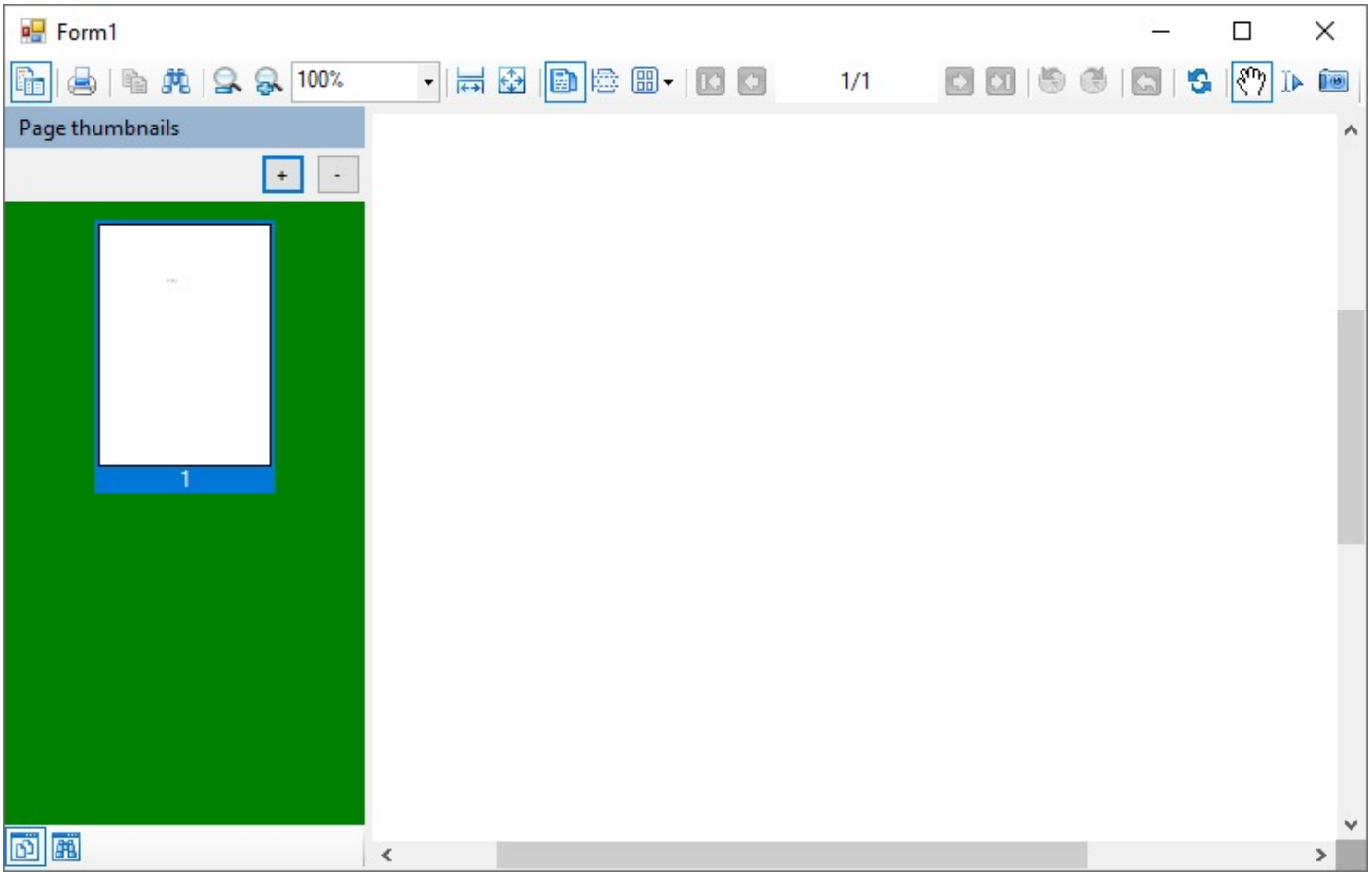
VB.NET

```
viewer1.TouchMode = GrapeCity.Viewer.Common.Model.TouchMode.[True]

Dim zoomOutButton = viewer1.TouchModeToolbar.ToolStrip.Items(8)
zoomOutButton.Visible = true
```

⚠ **Caution:** Any customization done in mouse mode does not apply to the touch mode and vice versa.

Customize Viewer Sidebar

The following code customizes the background of Thumbnail in Sidebar of the Viewer.



In the Viewer Form Load event, add the code as follows:

C#

```
private void Form1_Load(object sender, EventArgs e)
        {
            Control o = GetControlByName(viewer1, "thumbnailViewer1");
            o.BackColor = System.Drawing.Color.Green;

            //write code to load report
        }
        public Control GetControlByName(Control ParentCntl, string NameToSearch)
        {
            if (ParentCntl.Name == NameToSearch)
                return ParentCntl;
            foreach (Control ChildCntl in ParentCntl.Controls)
            {
                Control ResultCntl = GetControlByName(ChildCntl, NameToSearch);
                if (ResultCntl != null)
                    return ResultCntl;
```

```
            }
            return null;
        }
```

VB.NET

```
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Dim o As Control = GetControlByName(Viewer1, "thumbnailViewer1")
        o.BackColor = System.Drawing.Color.Green

        'write code to load report
    End Sub
    Public Function GetControlByName(ByVal ParentCntl As Control, ByVal NameToSearch As
String) As Control
        If ParentCntl.Name = NameToSearch Then Return ParentCntl
        For Each ChildCntl As Control In ParentCntl.Controls
            Dim ResultCntl As Control = GetControlByName(ChildCntl, NameToSearch)
            If ResultCntl IsNot Nothing Then Return ResultCntl
        Next
        Return Nothing
    End Function
```

Js Viewer Application

The Js Viewer is a Javascript component that you can easily customize and use in web applications to preview all types of reports - Page, Rdl, and Section reports. The Js Viewer works on modern web application frameworks - ASP.NET MVC, ASP.NET Core MVC, and major JavaScript Frameworks such as Angular, React, and Vue.js. The Viewer supports all the major browsers as well as rendering on mobile devices.

This section covers some most popular scenarios of modern Web applications using different frameworks:

ASP.NET MVC Core Integration

Integration to Angular Application

Integration to React Application

Integration to VueJS Application

For special situations, we recommend that you read these topics:

Js Viewer ASP.NET Core Middleware

Load Reports

Switch Between Render Formats

Update Security Token in Report Service

Caching Reports

Prevent Cross-Site Scripting Attacks

Customize UI

Customize Page View

Animation

View Reports from Different Domains using CORS

Predefined Export Settings

Js Viewer API

> **Note**: For some reports, you may need to configure a fonts factory or a resource locator. For details, see Custom Resource Locator and Custom Font Resolver.

Js Viewer ASP.NET Core Middleware

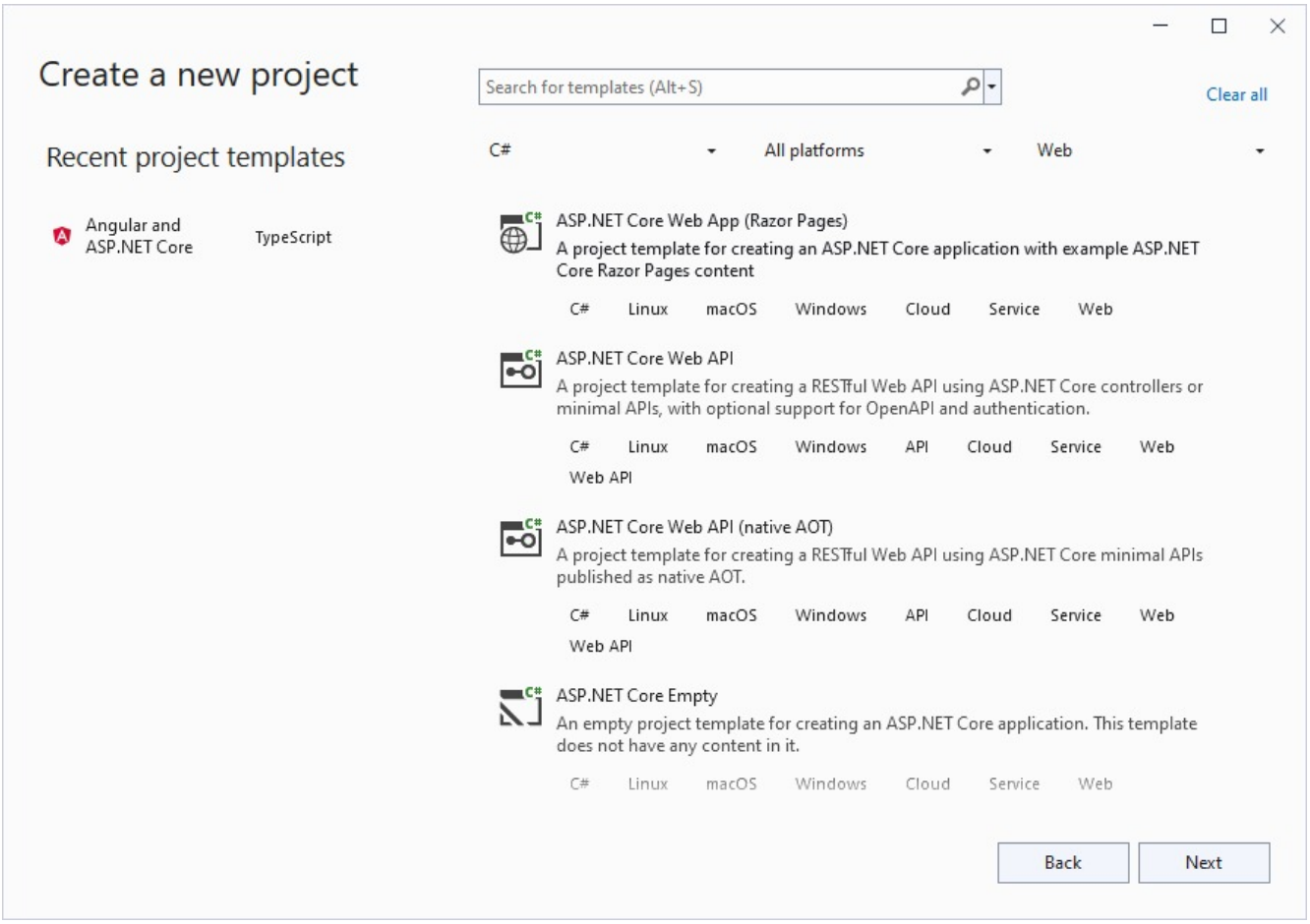
ASP.NET Core middleware is software to handle HTTP requests and responses. Each middleware component serves a specific purpose, one authenticates a user, and the other handles static files like Javascript or CSS files. These middleware components together set up a request processing pipeline.

The default code developed by the **ASP.NET Core Web App** template sets up the request processing pipeline for the application using a set of middlewares - **UseDeveloperExceptionPage()** and **UseStaticFiles()** of the IApplicationBuilder interface. The middlewares are executed in the order in which they are added to the pipeline.

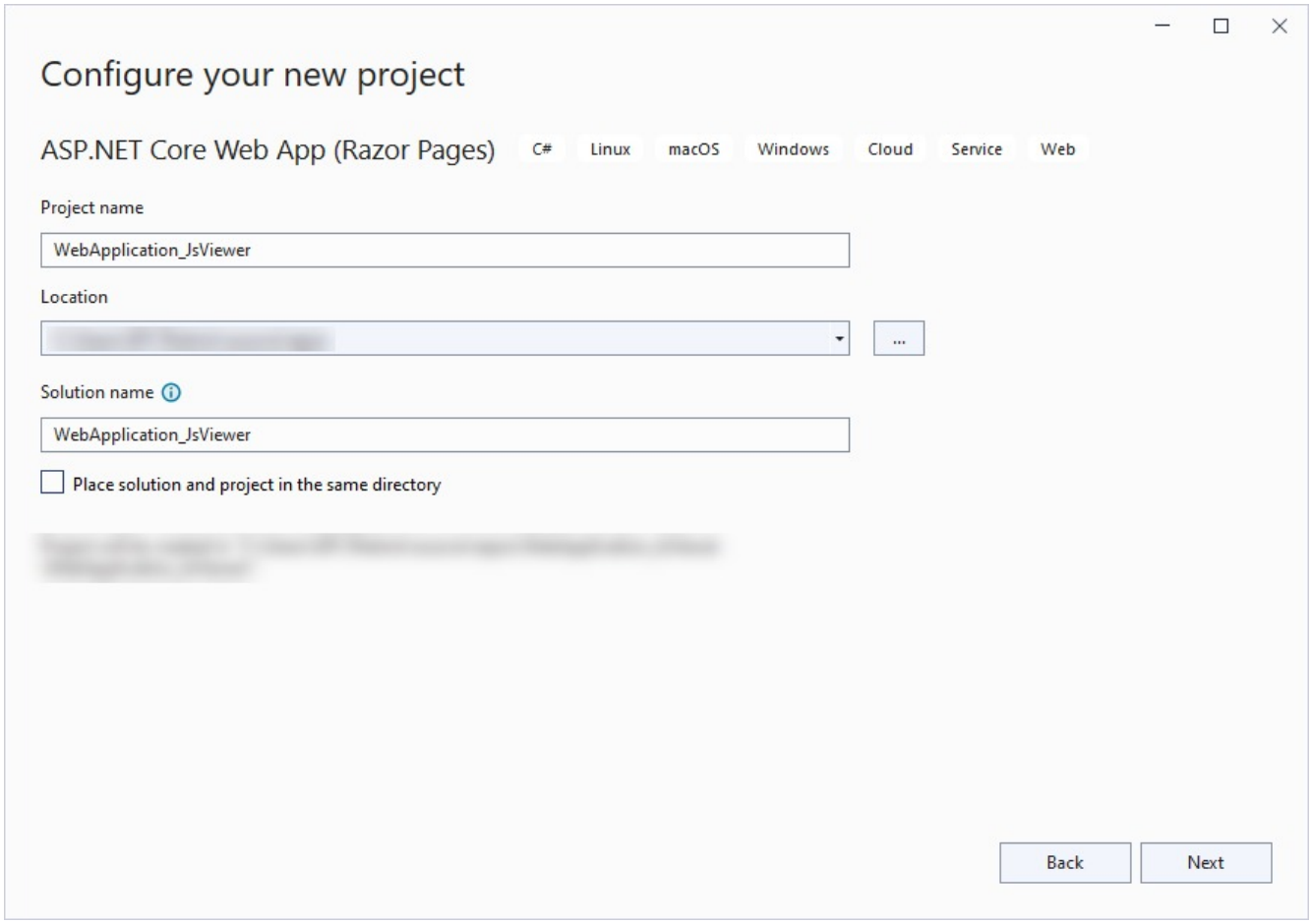
To provide access to a report output from the browser, you need to configure ActiveReports JSViewer in ASP.NET Core middleware. It is done by adding the **UseReportViewer()** middleware, which configures middleware for ActiveReports API and handlers.

Create an ASP.NET Core Project

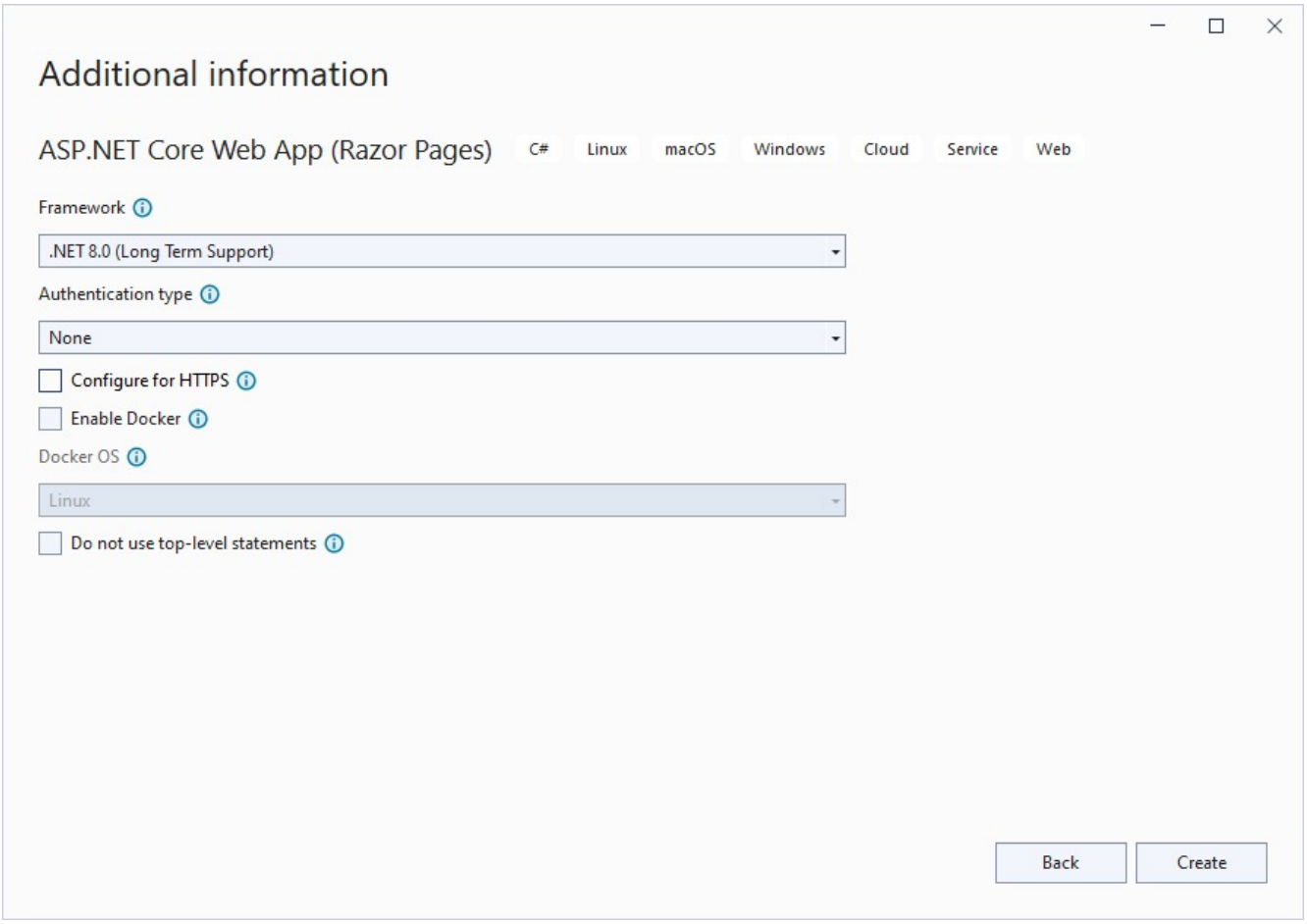
1. Open **Microsoft Visual Studio 2022** and create a new **ASP.NET Core Web App** project which includes example Razor Pages.



2. Type a name for your project and click **Next**.



3. Fill-in the additional info like 'Framework' as .NET 8.0.



4. Uncheck the 'Configure for HTTPS' checkbox instruction and click **Create**.

   Configure ActiveReports in ASP.NET Core Middleware

5. Right-click the project in the **Solution Explorer** and select **Manage Nuget Packages**.
6. Add the following package to the project.

   `MESCIUS.ActiveReports.Aspnetcore.Viewer`

7. In the **License Acceptance** dialog that appears, click **I Accept**.
8. Add a new folder called 'Reports' in application's root and place the report you want to display in the viewer, in this folder.
9. Make sure to set the **Build Action** property of the report to 'Embedded Resource'.
10. Modify the content for the **Program.cs** file as follows to enable the application to use ActiveReports:

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using System.Reflection;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddRazorPages();
```

```
var app = builder.Build();
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");

    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseStaticFiles();
// Configure middleware for ActiveReports API and handlers.
app.UseReportViewer(settings =>
{
    settings.UseEmbeddedTemplates("WebApplication_JsViewer.Reports",
System.Reflection.Assembly.GetAssembly(GetType()));
});
app.UseRouting();
app.UseAuthorization();
app.MapRazorPages();
app.Run();
```

Make sure that the correct namespace is provided in the first argument of **UseEmbeddedTemplates** for the report.

Note: Instead of 'UseEmbeddedTemplates', you can use either 'UseFileStore' or 'UseCustomStore' method calls.

- 'UseEmbeddedTemplates' stores reports as resources in dlls.
- 'UseFileStore' stores reports in the file system.
- 'UseCustomStore' allows you to store reports in any user-defined location, like a custom database or any other type of location.

ASP.NET MVC Core Integration

Let us create an ASP.NET Core application using ActiveReports and render the reports in the Js Viewer.

1. Create an ASP.NET Core Web Application which includes example Razor Pages and configure ActiveReports in ASP.NET Core Middleware as illustrated in Js Viewer ASP.NET Middleware topic.
2. Add Js Viewer to the application.
    1. Open the **Tools** menu > **NuGet Package Manager** > **Package Manager Console** and run the following command:
    `npm install @mescius/activereportsnet-viewer`
    2. Copy the 'jsViewer.min.js' and 'jsViewer.min.css' files installed in the **node_modules** folder to the **wwwroot\js** and **wwwroot\css** folder in the application, respectively.
3. Replace the complete code in the 'Index.cshtml' with the following content:

Index.cshtml

```
@page
@model IndexModel
```

```
@{
    ViewData["Title"] = "Home page";
}
<!DOCTYPE html>
<html lang="en">
<head>
    <title>ActiveReports JSViewer</title>
    <link rel="stylesheet" href="css/jsViewer.min.css" />
</head>
<body>
    <div id="viewer-id" style="width: 100%; height: 100%;">
    </div>
    <!--reference to js file-->
    <script src="js/jsViewer.min.js"></script>
    <script type="text/javascript">
        GrapeCity.ActiveReports.JSViewer.create({
            element: '#viewer-id',
            reportService: {
                url: 'api/reporting',
            },
            reportID: 'DemoReport.rdlx',
            settings: {
                zoomType: 'FitPage'
            }
        });
    </script>
</body>
</html>
```

4. Modify the 'Startup.cs' class by adding the Reporting service in the 'Startup.cs' class as follows:

Startup.cs

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            else
            {
                app.UseExceptionHandler("/Error");
                // The default HSTS value is 30 days. You may want to change this for
production scenarios, see https://aka.ms/aspnetcore-hsts.
                app.UseHsts();
            }
            app.UseHttpsRedirection();
            app.UseStaticFiles();
```

```
            app.UseReportViewer(settings =>
            {
                settings.UseFileStore(new System.IO.DirectoryInfo(env.ContentRootPath +
@"\Reports\"));
            });
            app.UseRouting();
            app.UseAuthorization();
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapRazorPages();
            });
        }
```

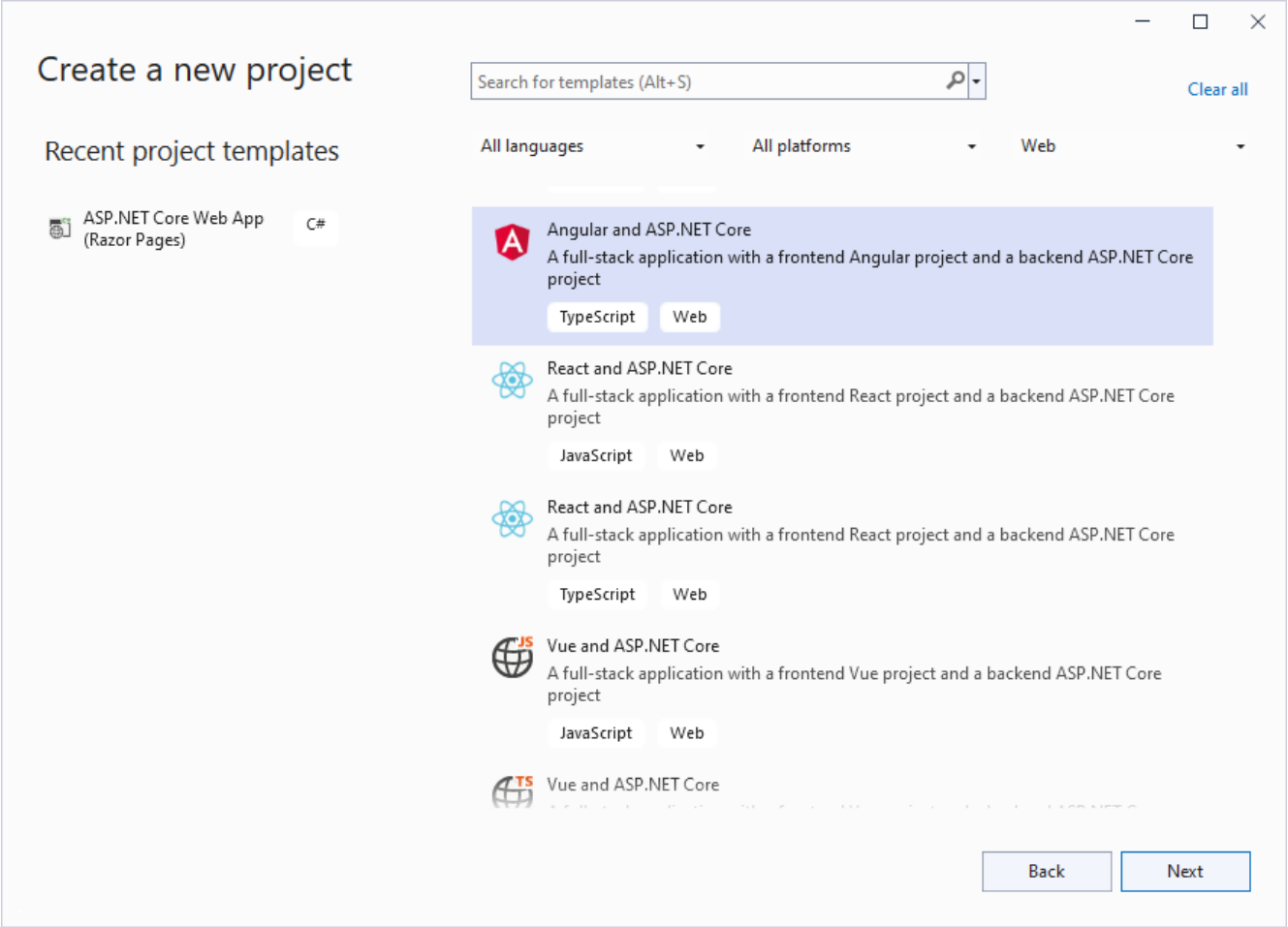
5. Run the application.

Integration to Angular Application

This page explains how you can embed the Js Viewer component in your Angular application. To run the Angular Application Server, you will require the node.js JavaScript runtime and Angular CLI. Use the following command in the terminal or command prompt to install the **Angular CLI**:

```
npm install -g @angular/cli
```

1. Open **Microsoft Visual Studio 2022** and create a new **Angular and ASP.NET Core** project.



2. Type a name for your project and click **Next**.
3. Select the **Framework** to a latest version and uncheck other options.
4. Right-click the '.Server' project in the **Solution Explorer** and select **Manage NuGet Packages**.
5. Add the following packages to the project.

   `MESCIUS.ActiveReports.Aspnetcore.Viewer`

6. Create 'resources' folder in your sample project root; you can put your existing reports, themes, and images in this folder.
7. Make sure to set the **Build** Action property of the resources to 'Embedded Resource'.
8. Open 'Program.cs' file and update the file to include the 'using' statements at the top, and specify the resource folder and add services to container, so that the complete file looks like below.

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddReportViewer();
builder.Services.AddControllers();
```

```
var app = builder.Build();
var ResourcesRootDirectory =
    new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"));
app.UseRouting();
app.UseDefaultFiles();
app.UseStaticFiles();
app.UseRouting();
app.UseReportViewer(config => config.UseFileStore(ResourcesRootDirectory));

// Configure the HTTP request pipeline.
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.MapFallbackToFile("/index.html");
app.Run();
```

9. In the '.client' project, open 'package.json' file and add the following packages under 'dependencies':
   `"@mescius/activereportsnet-viewer": "^18.x.x"`

10. Open the '.client' project in the command prompt or terminal window and run the following command to install the npm packages.
    `npm install`

    The designer and viewer files/folders will be downloaded in your current directory:
    .\node_modules\@mescius\activereportsnet-viewer\dist.
11. Expand the 'src\app' folder in the '.client' project, open 'app.component.ts' file, and replace the existing code with the following code to initialize the Designer instance.

app.component.ts

```
import { Component, OnInit } from '@angular/core';
import { createViewer } from '@mescius/activereportsnet-viewer';
import '@mescius/activereportsnet-viewer/dist/jsViewer.min.css';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent implements OnInit {
  title = "app";

  ngOnInit() {
    this.viewer = createViewer({
      element: '#viewer-host'
    });
    this.viewer.openReport("DemoReport.rdlx");
  }
  }
```

12. Open the 'app.component.html' file and replace the existing content with the following markup for hosting the element.

app.component.html

```
<body>
  <div id="viewer-host"></div>
</body>
```

13. Open the '\src\styles.css' file and add the following content.

styles.css

```
#viewer-host {
  width: 100%;
  height: 100vh;
}
```

14. Open 'proxy.conf.js' file and update the 'context' section of code as follows.

proxy.conf.js

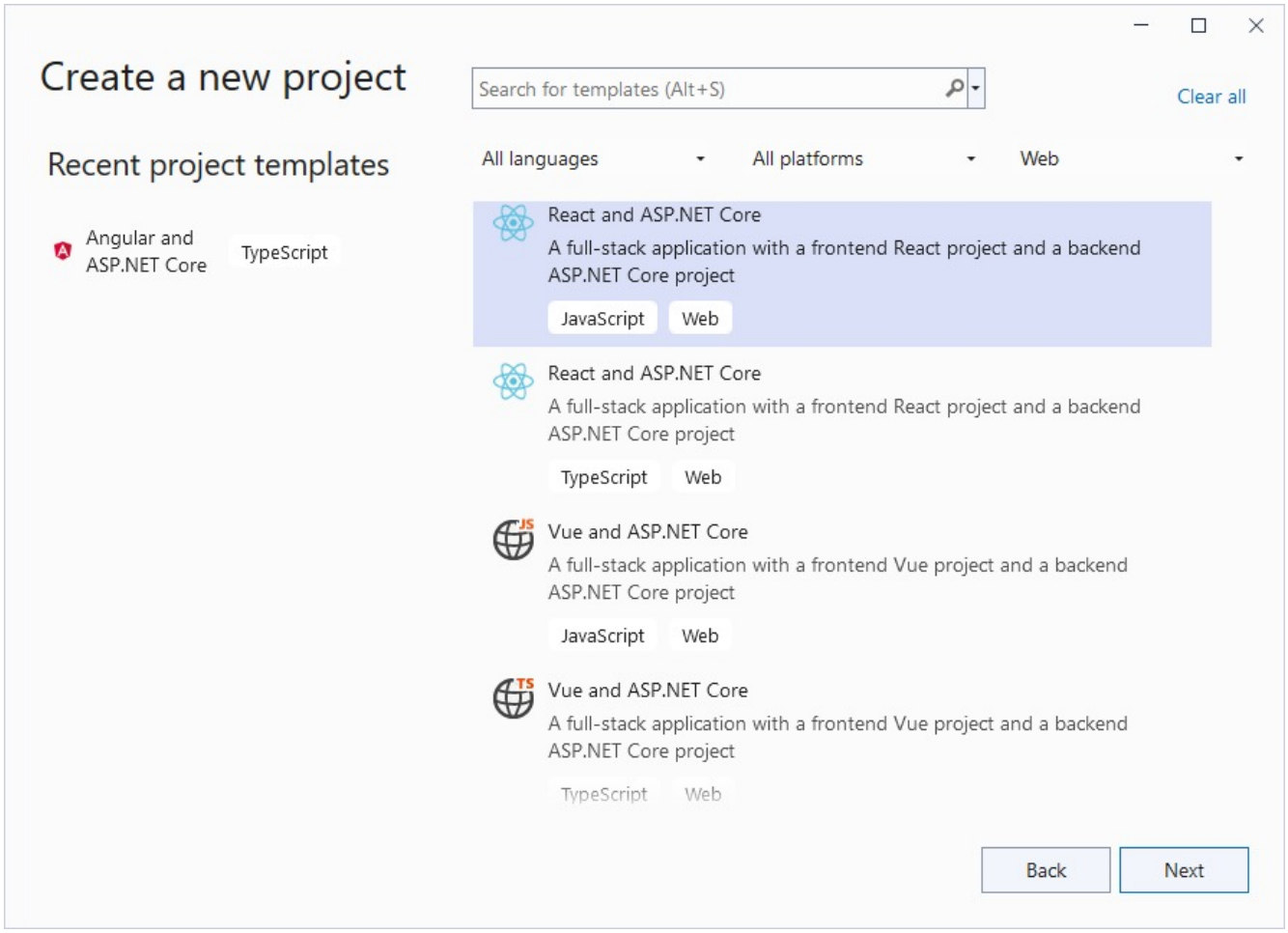
```
context: [
      "/weatherforecast",
      "/api"
   ],
```

15. Press **Ctrl** + **Shift** + **B** to build your application and then press **F5** to run it.
    On running the application, you can find the report placed in the resource folder by navigating to **File** menu > **Open**.

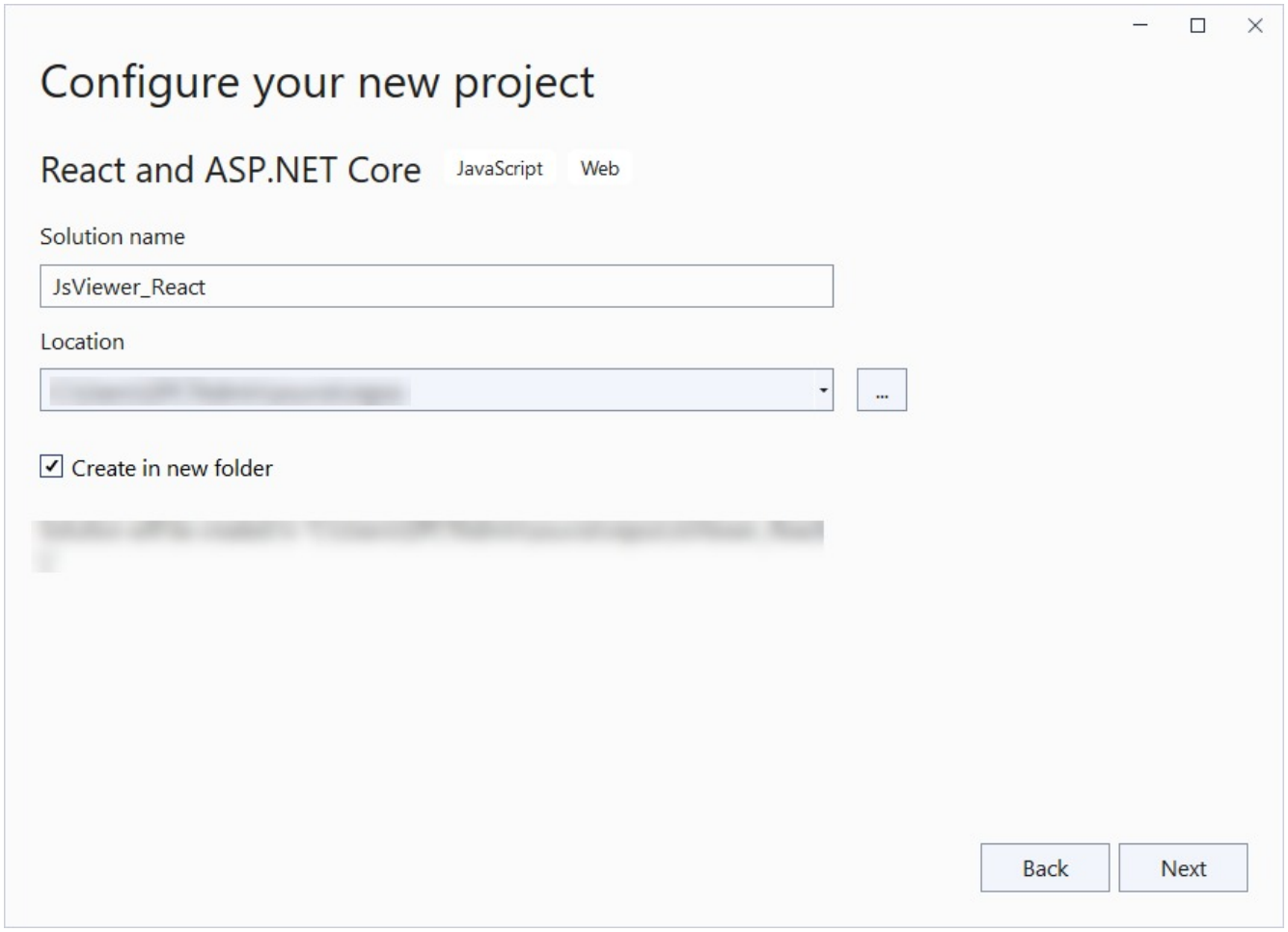
Integration to React Application

This page explains how you can embed the Js Viewer component in your React application (ASP.NET Core). To run the React Application Server, you will require the node.js JavaScript runtime.

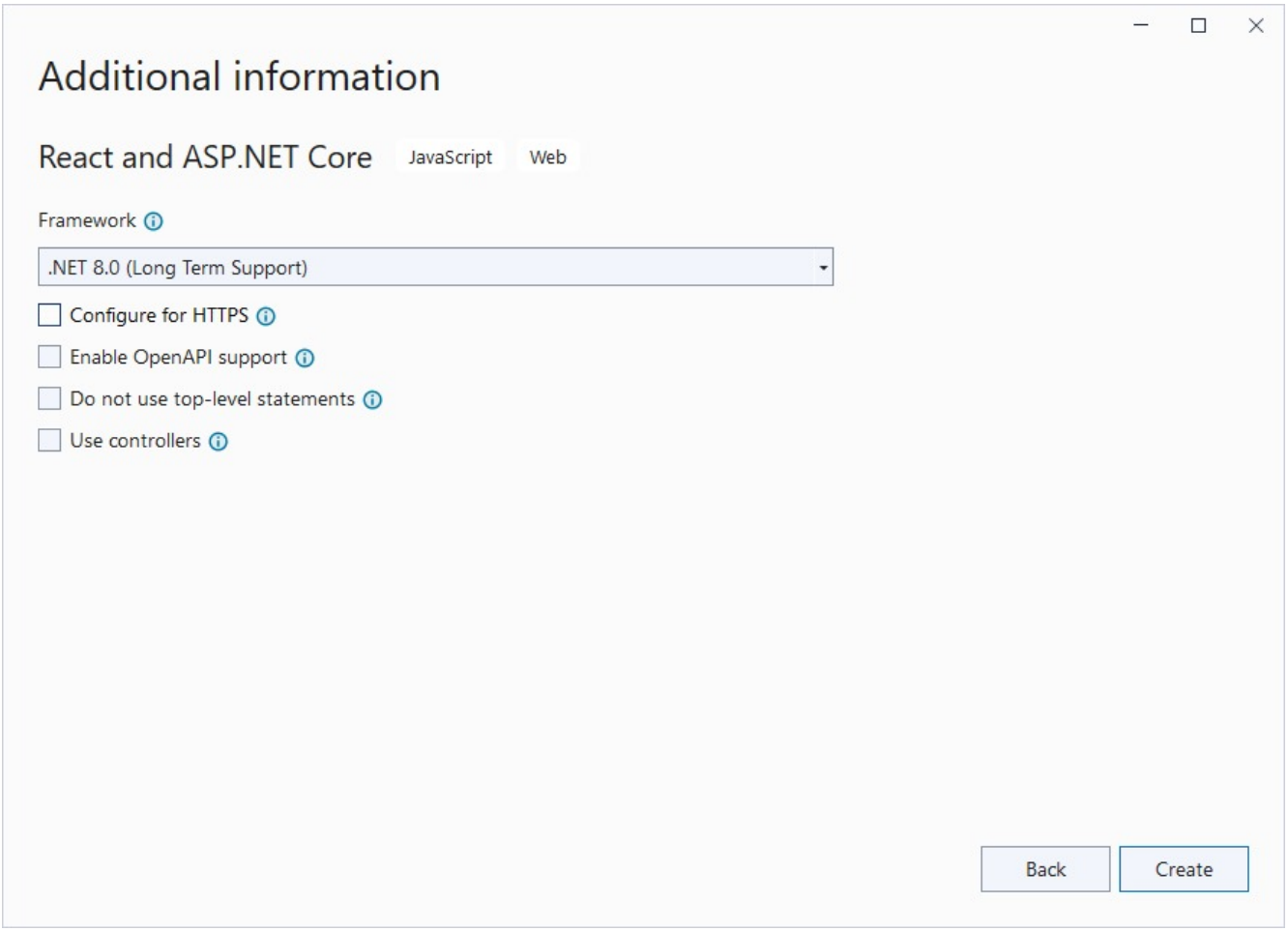
1. Open **Microsoft Visual Studio 2022** and create a new **React and ASP.NET Core** project.



2. Type a name for your project and click **Next**.



3. Select the **Framework** to a latest version and uncheck other options.



4. Right-click the project in the **Solution Explorer** and select **Manage NuGet Packages**.
5. Add the following package to the project.

   `MESCIUS.ActiveReports.Aspnetcore.Viewer`

6. Create 'resources' folder in your sample project root; you can put your existing reports, themes, and images in this folder.
7. Make sure to set the **Build** Action property of the resources to 'Embedded Resource'.
8. Open 'Program.cs' file and update the file to include the 'using' statements at the top, and specify the resource folder, and add services to container, so that the complete file looks like below.

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddReportViewer();
builder.Services.AddControllers();
var app = builder.Build();
var ResourcesRootDirectory =
    new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"));
app.UseReportViewer(config => config.UseFileStore(ResourcesRootDirectory));
```

```
app.UseDefaultFiles();
app.UseStaticFiles();
// Configure the HTTP request pipeline.
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.MapFallbackToFile("/index.html");
app.Run();
```

9. In the '.client' project, open 'package.json' file and add the following package under 'dependencies':

   ```
   "@mescius/activereportsnet-viewer": "^18.x.x"
   ```

10. Open the '.client' project in the command prompt or terminal window and run the following command to install the npm packages.

    ```
    npm install
    ```

    The viewer files/folders will be downloaded in your current directory: .\node_modules\@mescius\activereportsnet-viewer\dist.

11. Open 'App.css' to update the 'root' selector, and add style for the 'viewer-host' element as follows.

    App.css

    ```
    #root {
        height: 100%;
        width: 100%;
    }
    tr:nth-child(even) {
        background: #F2F2F2;
    }
    tr:nth-child(odd) {
        background: #FFF;
    }
    th, td {
        padding-left: 1rem;
        padding-right: 1rem;
    }
    .viewer-host {
        height: 100vh;
        width: 100%;
    }
    ```

12. Open 'App.jsx' file and replace the existing code with the following code.

    App.jsx

    ```
    import { useEffect } from 'react';
    import './App.css';
    import { createViewer } from '@mescius/activereportsnet-viewer';
    import "@mescius/activereportsnet-viewer/dist/jsViewer.min.css";
    function App() {
        useEffect(() => {
            const viewer = createViewer({
                element: '#viewer-host',
                reportID: "DemoReport.rdlx"
    ```

```
        });
        return () => {
            viewer.destroy();
        };
    }, []);
    return (
        <div id="viewer-host" className="viewer-host" />
    );
}
export default App;
```

13. Open 'vite.config.js' file and update the 'proxy' setting as follows.

vite.config.js

```
proxy: {
    '/api':{
        target: 'http://localhost:5267',
        secure: false
    }
}
```

14. Open 'main.jsx' file and remove `import React from 'react'` statement if using React 17 or higher, and the outer statements `<React.StrictMode>` and `</React.StrictMode>` to disable strict mode. The final main.jsx is as shown.

main.jsx

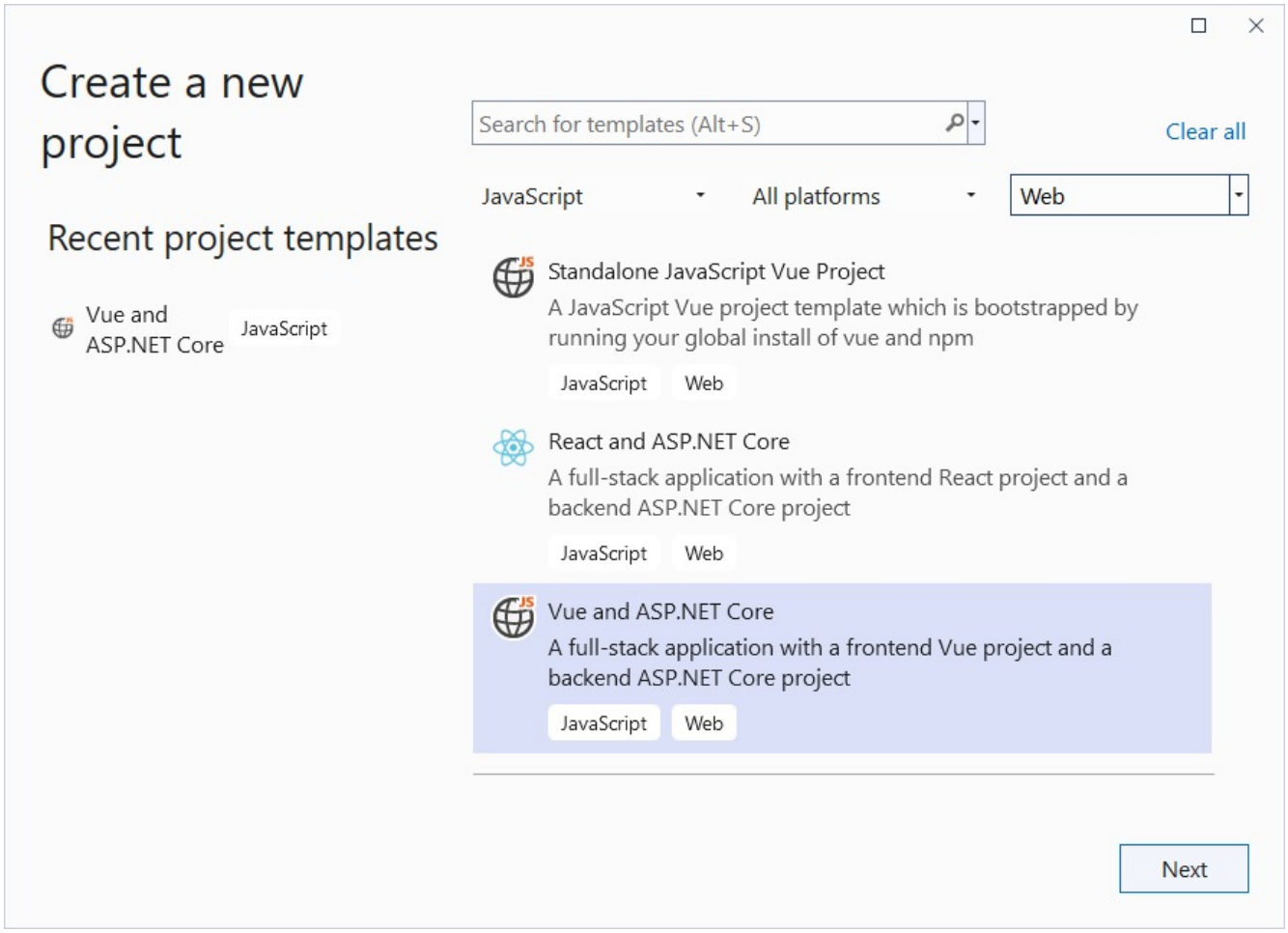
```
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
ReactDOM.createRoot(document.getElementById('root')).render(
    <App />
)
```

15. Build the application and then press **F5** to run it.

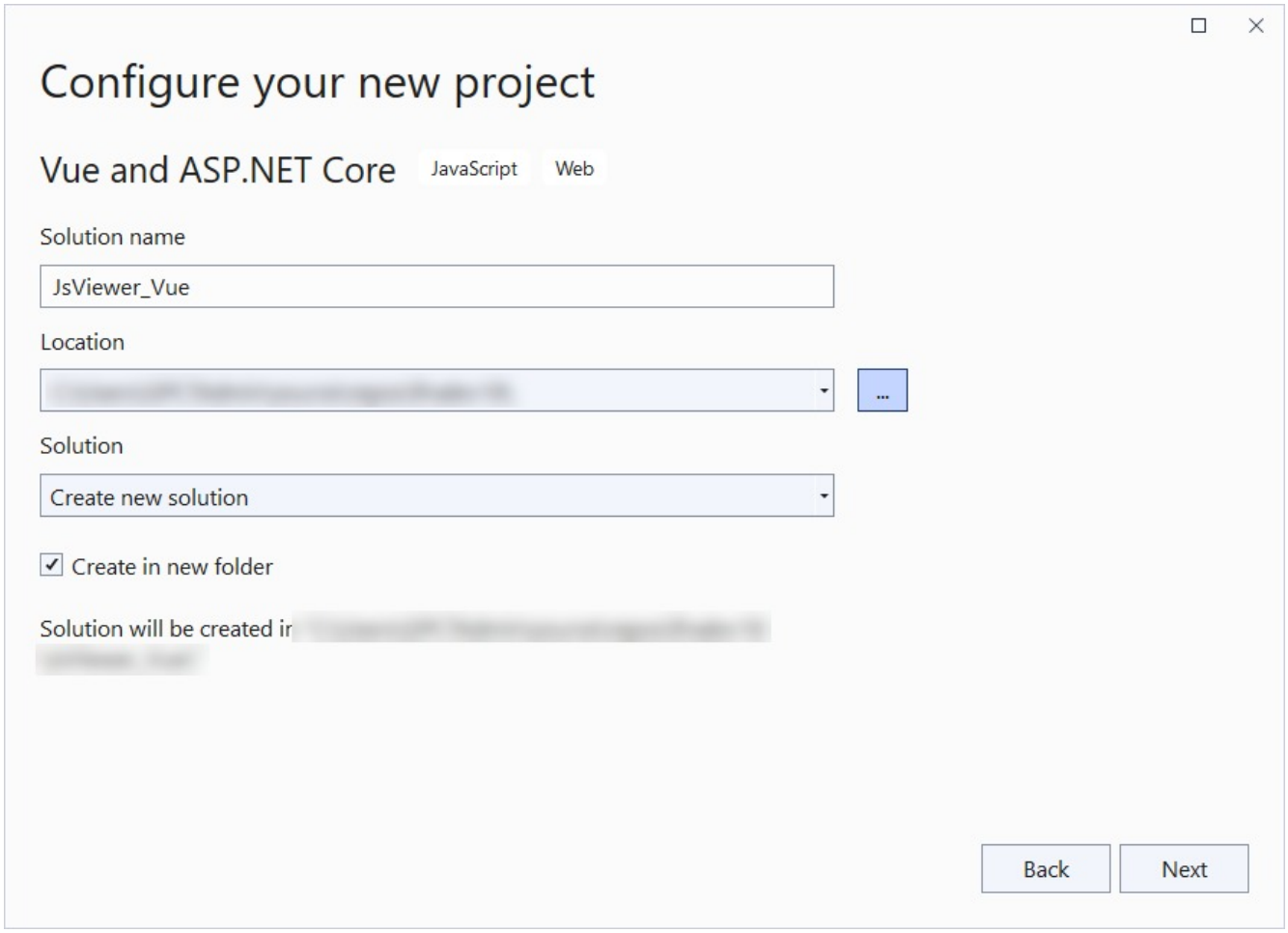
Integration to VueJS Application

This page explains how you can embed the Js Viewer component in your VueJS application (ASP.NET Core). To run the Vue Application Server, you will require the node.js JavaScript runtime.

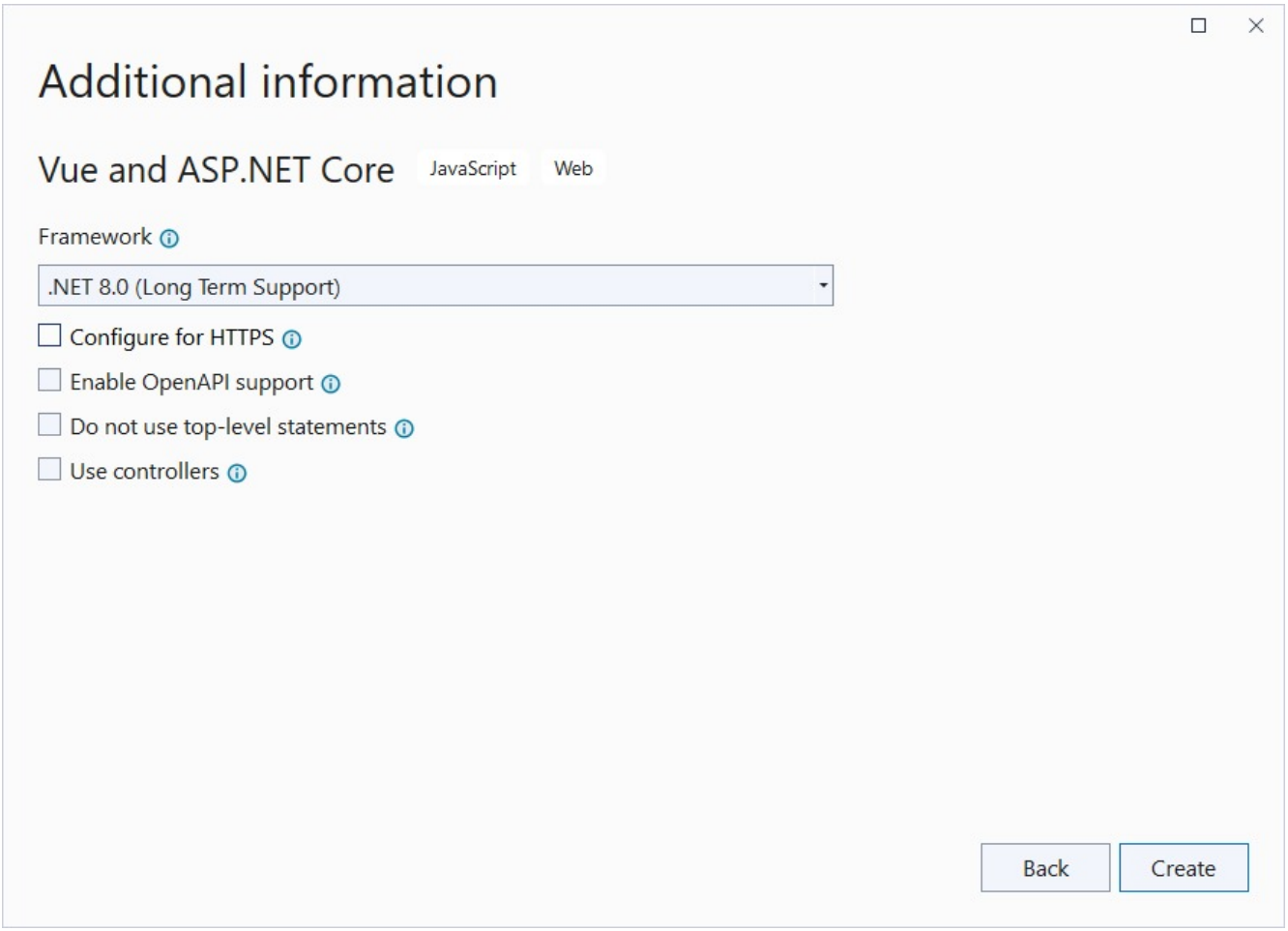
1. Open **Microsoft Visual Studio 2022** and create a new **Vue and ASP.NET Core** project.



2. Type a name for your project and click **Next**.



3. Select the **Framework** to a latest version and uncheck other options.



4. Right-click the project in the **Solution Explorer** and select **Manage NuGet Packages**.
5. Add the following package to the project.

   `MESCIUS.ActiveReports.Aspnetcore.Viewer`

6. Create 'resources' folder in your sample project root; you can put your existing reports, themes, and images in this folder.
7. Make sure to set the **Build** Action property of the resources to 'Embedded Resource'.
8. Open 'Program.cs' file and update the file to include the 'using' statements at the top, and specify the resource folder, and add services to container, so that the complete file looks like below.

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddReportViewer();
builder.Services.AddControllers();
var app = builder.Build();
app.UseCors(options => options
    .WithOrigins("http://localhost:5173")
    .AllowAnyHeader()
```

```
    .AllowAnyMethod()
    .AllowCredentials()
);
var ResourcesRootDirectory =
    new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"));
app.UseReportViewer(config => {
    config.UseFileStore(ResourcesRootDirectory);
});
app.UseDefaultFiles();
app.UseStaticFiles();
app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();
app.MapFallbackToFile("/index.html");
app.Run();
```

9. In the '.client' project, open 'package.json' file and add the following package under 'dependencies':

   `"@mescius/activereportsnet-viewer": "^18.x.x"`

10. Open the '.client' project in the command prompt or terminal window and run the following command to install the npm packages.

    `npm install`

    The viewer files/folders will be downloaded in your current directory: .\node_modules\@mescius\activereportsnet-viewer\dist.

11. Open 'App.vue' inside the 'src' folder and replace its default content with the following code.

App.vue

```
<template>
    <div id="viewer-host">
    </div>
</template>
<script>
    import { createViewer } from '@mescius/activereportsnet-viewer';
    import "@mescius/activereportsnet-viewer/dist/jsViewer.min.css";
    export default {
        name: 'app',
        mounted() {
            let serverUrl = 'http://localhost:5151';

            this.viewer = createViewer({
                element: '#viewer-host',
                reportService: {
                    url: serverUrl + '/api/reporting'
                }
            });
            this.viewer.openReport("DemoReport.rdlx");
        }
    }
</script>
<style>
```

```
    #viewer-host {
        background-color: #e5e5e5;
        height: 100vh;
        float: right;
        width: 100%;
    }
</style>
```

12. Open 'vite.config.js' file and update the 'server' setting as follows.

vite.config.js

```
server: {
     proxy: {
         '/api': {
             target: 'http://localhost:5001/' // Update the target to use HTTP
         }
     },
     port: 5173,
     https: false // Set https to false to run on HTTP
 }
```

13. To disable browser launch, set the 'launchBrowser' to 'false' in 'launchSettings.json' (in .Server project\Properties).
14. Run the application by pressing **F5**.

Load Reports

This topic provides you with the code examples for loading a report in Js Viewer in different JavaScript frameworks.

JS

Index.cshtml

```
var viewer = new GrapeCity.ActiveReports.JSViewer.create({
    element: '#viewer-host'
});
viewer.openReport("DemoReport.rdlx");
```

Angular

app.component.ts

```
import { Component, OnInit } from '@angular/core';
import { createViewer } from '@mescius/activereportsnet-viewer';
import '@mescius/ar-viewer/dist/jsViewer.min.css';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html',
   styleUrl: './app.component.css'
 })
 export class AppComponent implements OnInit {
   title = "app";
```

```
  ngOnInit() {
    this.viewer = createViewer({
      element: '#viewer-host'
    });
    this.viewer.openReport("DemoReport.rdlx");
  }
  }
```

React

App.jsx

```
import { useEffect } from 'react';
 import './App.css';
 import { createViewer } from '@mescius/activereportsnet-viewer';
 import "@mescius/ar-viewer/dist/jsViewer.min.css";
function App() {
    useEffect(() => {
         const viewer = createViewer({
             element: '#viewer-host',
             reportID: "DemoReport.rdlx"
         });
        return () => {
             viewer.destroy();
         };
     }, []);
    return (
         <div id="viewer-host" className="viewer-host" />
     );
 }
export default App;
```

VueJS

App.vue

```
<script>
     import { createViewer } from '@mescius/activereportsnet-viewer';
     import "@mescius/ar-viewer/dist/jsViewer.min.css";
     export default {
         name: 'app',
         mounted() {
             let serverUrl = 'http://localhost:5151';

             this.viewer = createViewer({
                 element: '#viewer-host',
                 reportService: {
                     url: serverUrl + '/api/reporting'
                 }
```

```
            });
            this.viewer.openReport("DemoReport.rdlx");
        }
    }
</script>
<style>
    #viewer-host {
        background-color: #e5e5e5;
        height: 100vh;
        float: right;
        width: 100%;
    }
</style>
```

To load the most updated version of the report in the viewer, use the **Refresh** button.

Switch Between Render Formats

It is required to specify the render format in Js Viewer in the following scenarios:

- Rounding issues in HTML output. It is not always possible to fix this in HTML, especially in Galley mode.
- The output is not WYSIWYG sometimes.

The Js Viewer supports three render formats - 'auto' (default), 'html', and 'svg'. For a Page or an RDLX report, set the value to 'html'. For a Section report, set the value to 'svg'

To switch the mode, you should specify the renderFormat setting through the JS Viewer API:

Index.cshtml

```
viewer = GrapeCity.ActiveReports.JSViewer.create({
    element: '#viewer-host',
    renderFormat: 'svg'
});
```

Update Security Token in Report Service

The Js Viewer allows to specify a security token during creation. However, sometimes you may need to change a security token at run time. To do so, you should use the onRequest handler as demonstrated in the example below:

Index.cshtml

```
const viewer = GrapeCity.ActiveReports.JSViewer.create({
  element: '#root',
  reportService: {
    onRequest: (init) => alert(init.credentials),
  },
});
```

Caching Reports

Caching reports is a great technique to improve the performance of report rendering and is extremely beneficial for large reports with an infrequent change of data, like periodic reports that generate monthly or weekly.

The Js Viewer backends read a report as many times as required. However, sometimes you may need to cache the report as demonstrated in the example below.

Index.cshtml

```
var cache = new MemoryCache(new MemoryCacheOptions { SizeLimit = 10000 });
settings.UseCustomStore(reportId =>
    cache.GetOrCreate(reportId, entry =>
    {
        entry.SetSlidingExpiration(TimeSpan.FromSeconds(30));
        return new PageReport(new FileInfo(Path.Combine(@"C:\Reports", reportId)));
    });
```

Prevent Cross-Site Scripting Attacks

ActiveReports allows pre-processing of all links from reports. To prevent possible attacks and if you do not trust report authors, we recommend that you add processing of hyperlinks as demonstrated in the code example below:

Startup.cs

```
app.UseReportViewer(settings => {
  settings.OnHyperlink = link =>
  {
    if (!Uri.TryCreate(link, UriKind.RelativeOrAbsolute, out Uri uri))
      return string.Empty;
    if (uri.IsAbsoluteUri)
    {
      if (uri.Scheme.ToLowerInvariant() == "javascript")
        return string.Empty;
      return uri.AbsoluteUri;
    }
    return uri.ToString();
  }
  ...
```

Customize UI

The Js Viewer API lets developers completely overwrite the toolbar's default user interface and the viewer component's sidebar.

Setting the toolbar layout

Js Viewer has three built-in layouts for the toolbar:

- **desktop** - the default layout
- **full-screen** - displayed when the full-screen mode of the Viewer is on
- **mobile** - displayed on narrow screens

By default, each of these layouts contains the following items.

Internal ID	Description
$navigation	Go to the 1st page, go to the previous page, page number/page total, go to the next page, go to the last page buttons
$split	Separator
$refresh	Refresh report button
$history	Go to the parent report, go back in history, go forward in history buttons
$zoom	Zoom mode drop-down
$fullscreen	Toggle full-screen mode button
$print	Print button
$singlepagemode	Switch to the single page mode button
$continuouspagemode	Switch to the continuous page mode button
$galleymode	Switch to the galley page mode button

You can use the **layout** method to display only specified items in the toolbar for each layout mode. Here is the example of using this approach in an Angular application for showing 'Zoom', 'Toggle Full Screen', and 'Print' items for the regular layout - 'Toggle Full Screen' and 'Print' items for the full-screen layout, and displaying 'Navigation' item only for the mobile layout.

app.component.ts

```
import { Component, AfterViewInit } from '@angular/core';
import { createViewer} from '@mescius/activereportsnet-viewer';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent implements AfterViewInit {
  title = "app";

  ngAfterViewInit() {
    this.viewer = createViewer({
      element: '#viewer-host'
    });
    this.viewer.toolbar.desktop.layout(["$zoom", "$split", "$fullscreen", "$print"]);
    this.viewer.toolbar.fullscreen.layout(["$fullscreen", "$print"]);
```

```
    this.viewer.toolbar.mobile.layout(["$navigation"]);
    this.viewer.openReport("DemoReport.rdlx");
  }
}
```

Similarly, you can customize the layout of the viewer for React, Vue, and Pure JavaScript applications.

Adding and removing a toolbar item

The **addItem** and **removeItem** methods of all three layout modes of the **viewer.toolbar** property can be used to add and remove custom elements in the toolbar. Below is an example that inserts the 'About' button in the toolbar for Angular application. You can use the same approach in React, Vue, and ASP.NET MVC applications.

app.component.ts

```
import { Component, AfterViewInit } from '@angular/core';
 import { createViewer} from '@mescius/activereportsnet-viewer';
@Component({
    selector: 'app-root',
    templateUrl: './app.component.html'
  })
  export class AppComponent implements AfterViewInit {
    title = "app";
  ngAfterViewInit() {
      this.viewer = createViewer({
        element: '#viewer-host'
      });
    var self = this;
    var options = { filename: "DemoReport" };
    var pdfExportButton = {
        key: '$pdfExportButtonKey',
        iconCssClass: 'mdi mdi-file-pdf',
        text: "Pdf",
        enabled: true,
        action: function (item) {
          console.log('Export to PDF function works here');
          self.viewer.export('Pdf', null, true, options);
        },
        onUpdate: function (arg, item) {
          console.log('Something in viewer was updated, check/update button state here');
        }
      };
     this.viewer.toolbar.desktop.addItem(pdfExportButton);
    this.viewer.openReport("DemoReport.rdlx");
    }
  }
```

Replacing toolbar and sidebar

Finally, the default toolbar and sidebar components of the Js Viewer can be hidden using the toolbar.toggle and sidebar.toggle methods. This approach can help if you decide to use custom UI to invoke viewer functions by using the public API. Here is an example of customizing Js Viewer toolbar and sidebar in an Angular application.

app.component.ts

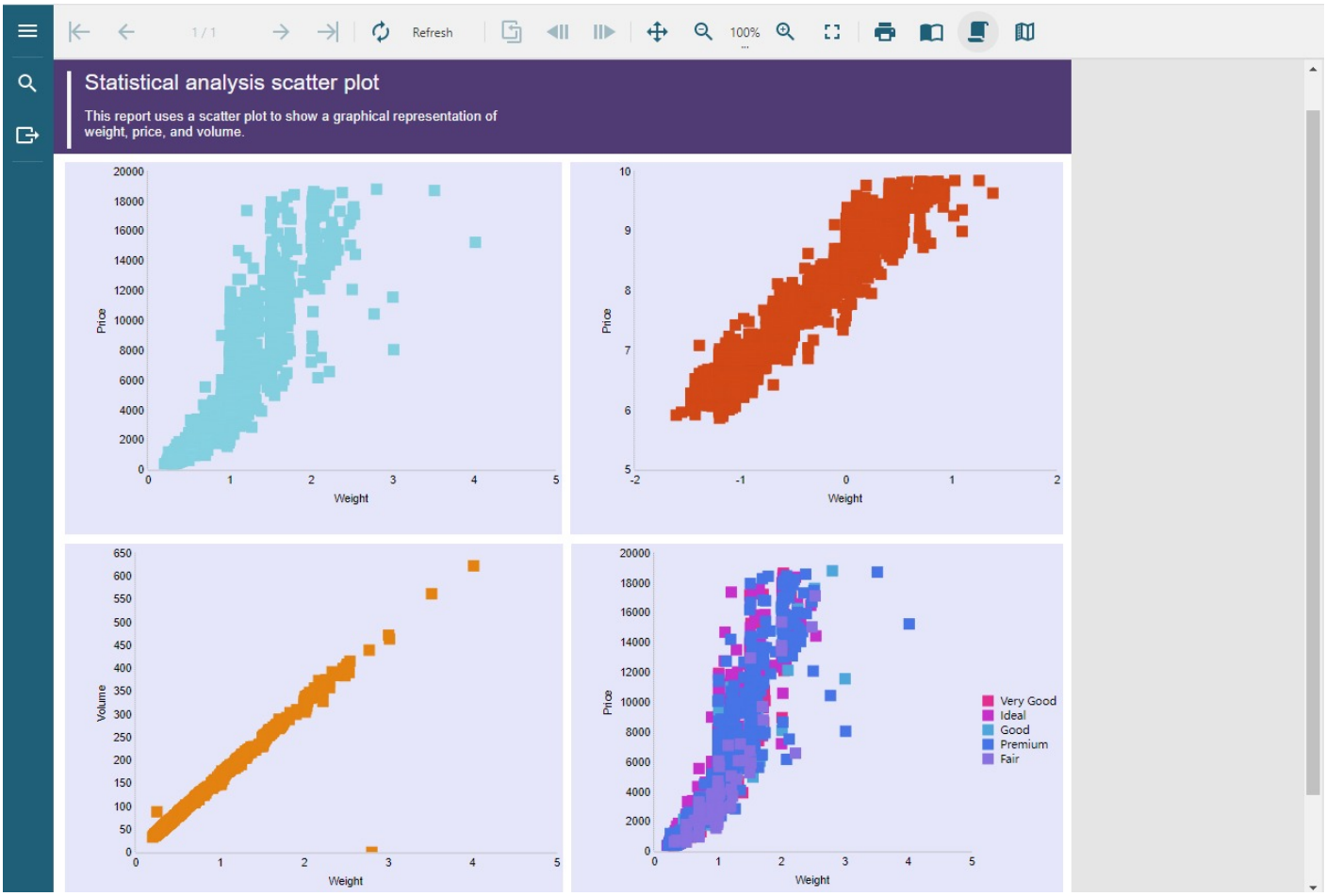
```
import { Component, AfterViewInit } from '@angular/core';
import { createViewer} from '@mescius/activereportsnet-viewer';
@Component({
   selector: 'app-root',
   templateUrl: './app.component.html'
 })
 export class AppComponent implements AfterViewInit {
   title = "app";
  ngAfterViewInit() {
    this.viewer = new createViewer({
       element: '#viewer-host'
     });
    this.viewer.toolbar.toggle(false);
    this.viewer.sidebar.toggle(false);
    this.viewer.openReport("DemoReport.rdlx");
   }
```

Customize Page View

You can set the horizontal alignment of the report on previewing a report on JS Viewer, and also set the report to appear as a page or a web page element using the Js Viewer API.

Horizontal Alignment

The horizontal alignment of the report page relative to the viewer frame can be set to center, left, or right. The following image shows the report page aligned to left.



The code to obtain the above preview is as shown below.

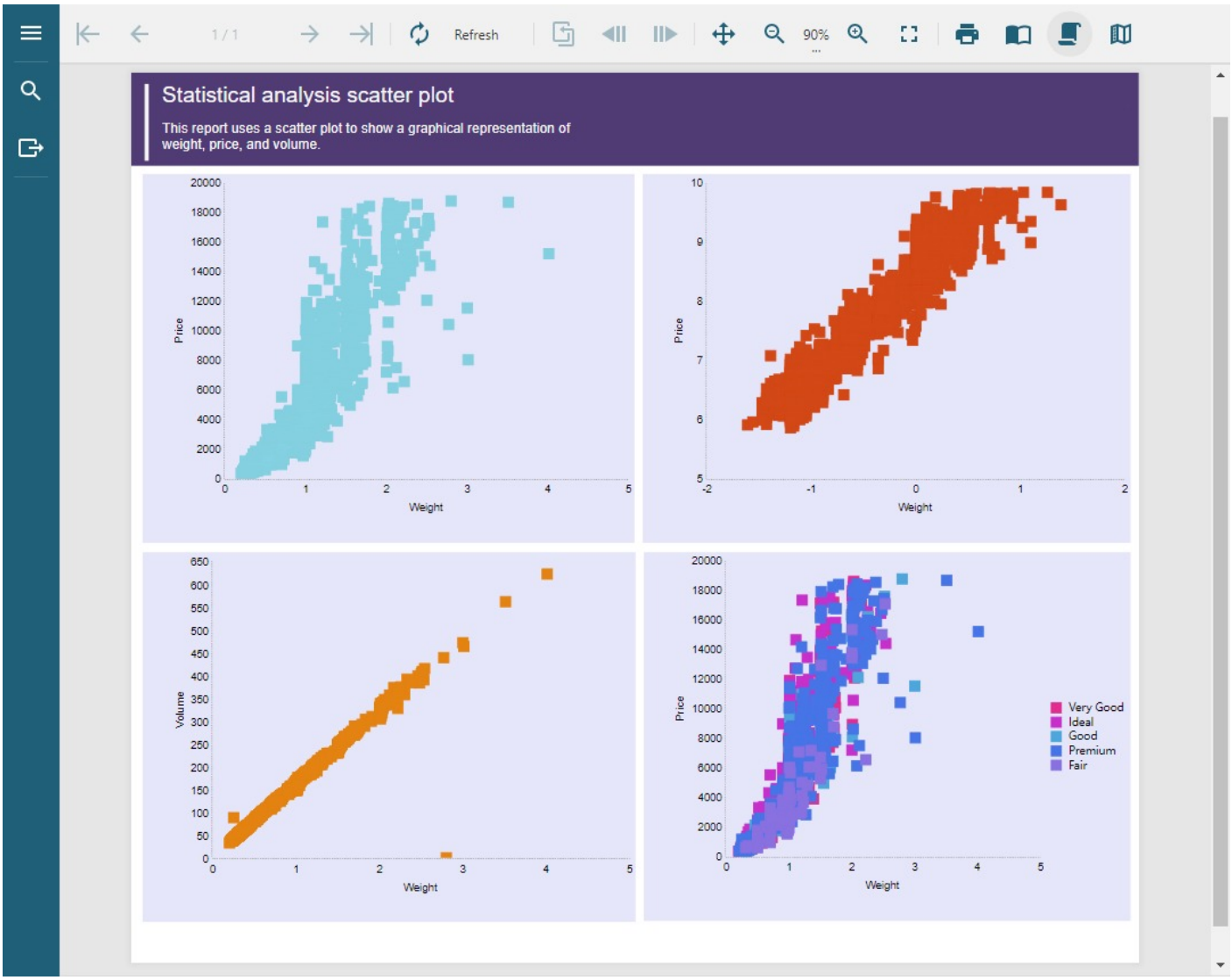
index.html

```
pageView: {
     horizontalAlignment: "left",
     viewMode: "standard"
          }
```

View Modes

View Mode as 'standard'

The report is displayed as a page.



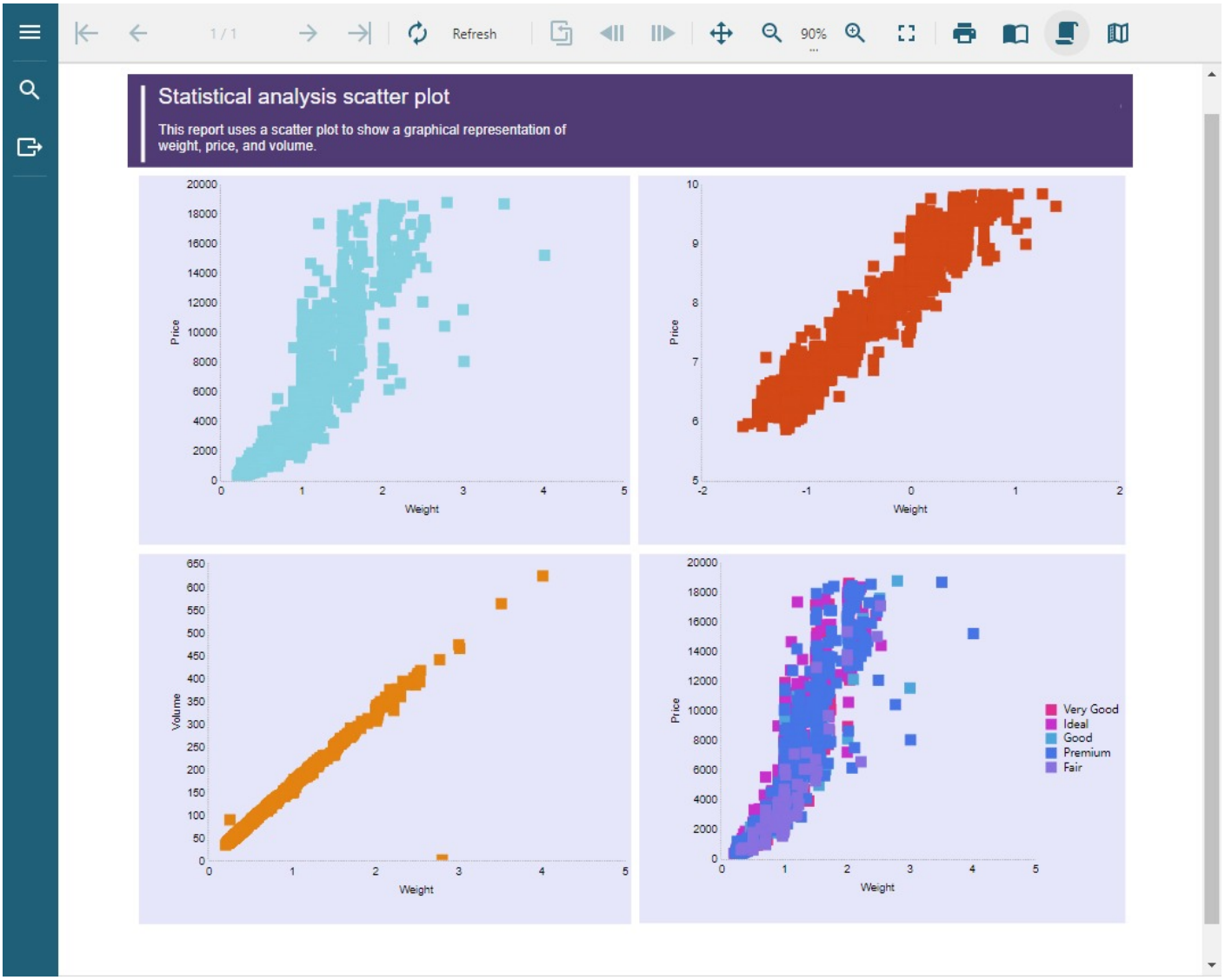
The code to obtain the above preview is as shown below.

index.html

```
pageView: {
     horizontalAlignment: "center",
     viewMode: "standard"
          }
```

View Mode as 'noPaper'

The report is displayed as an element of a web page. In this mode, the viewer's background color is set to transparent.



The code to obtain the above preview is as shown below.

index.html

```
pageView: {
     horizontalAlignment: "center",
     viewMode: "noPaper"
          }
```

Complete Code Implementation

The following code snippet shows the complete code for customizing the page view in JS Viewer.

index.html

```
<!DOCTYPE html>
<html lang="en">
```

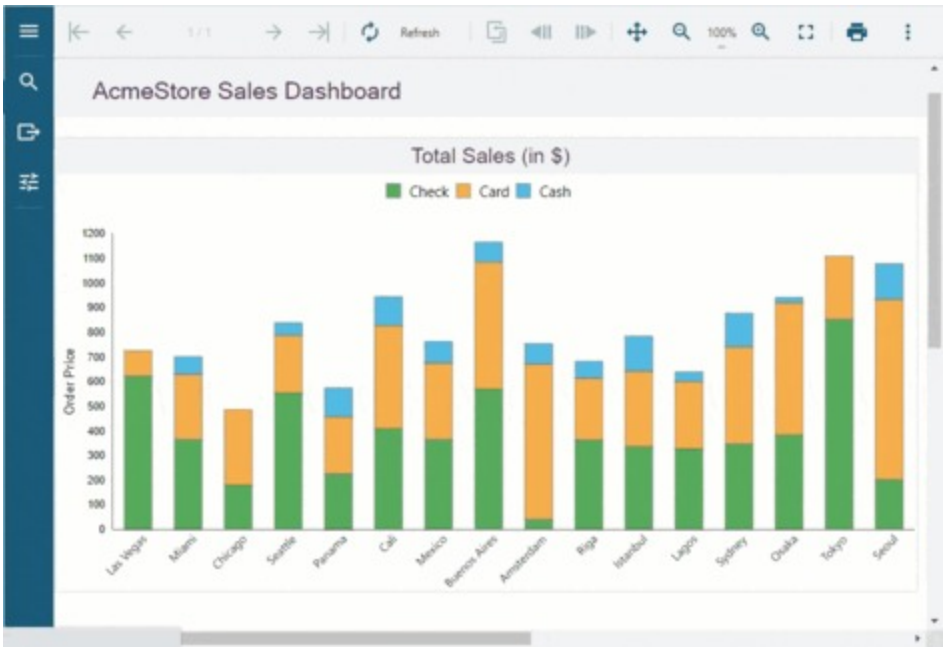
```
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                displayMode: "Continuous",
                pageView: {
                    horizontalAlignment: "center",
                    viewMode: "standard"
                }
            });

            console.dir(viewer)
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

Animation

The Js Viewer (and WebViewer control) provides the API for users to add animations to charts and tables. You can enable the animation feature on charts when the chart is loaded and when the user hovers over a chart or a table.

> **Note:** For the chart animations, you need to add the **jsViewer.chart.min.js** and **jsVIewer.chart.min.css** files from the NPM packages to your project and link them on your page.

Load Animation on Charts



Load animation occurs when the chart first loads, or when the chart is forced to reload. To enable load animation in charts, use the following code:

index.html

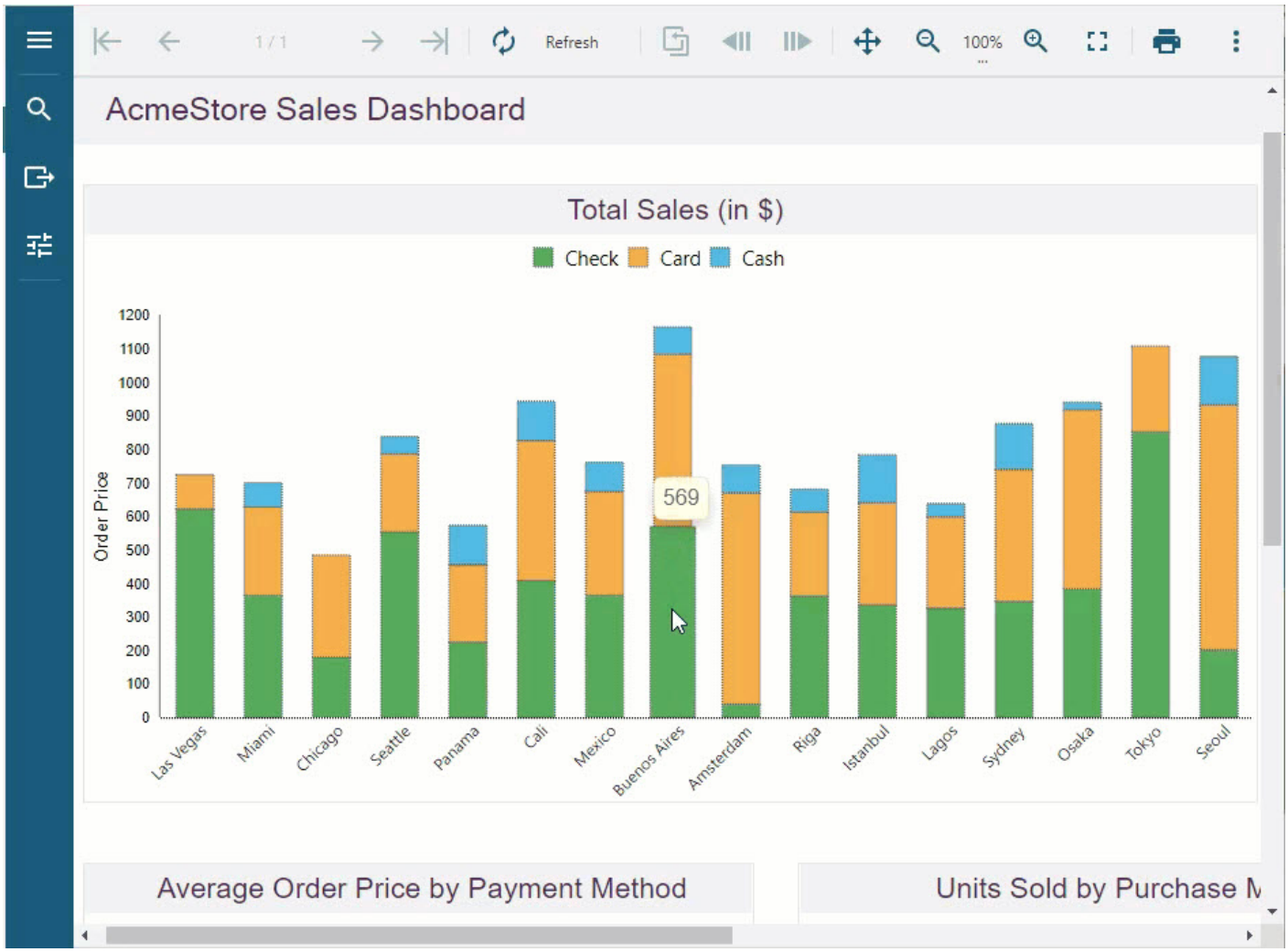
```
animation: {
    loadChart: {
        enabled: true
               }
          }
```

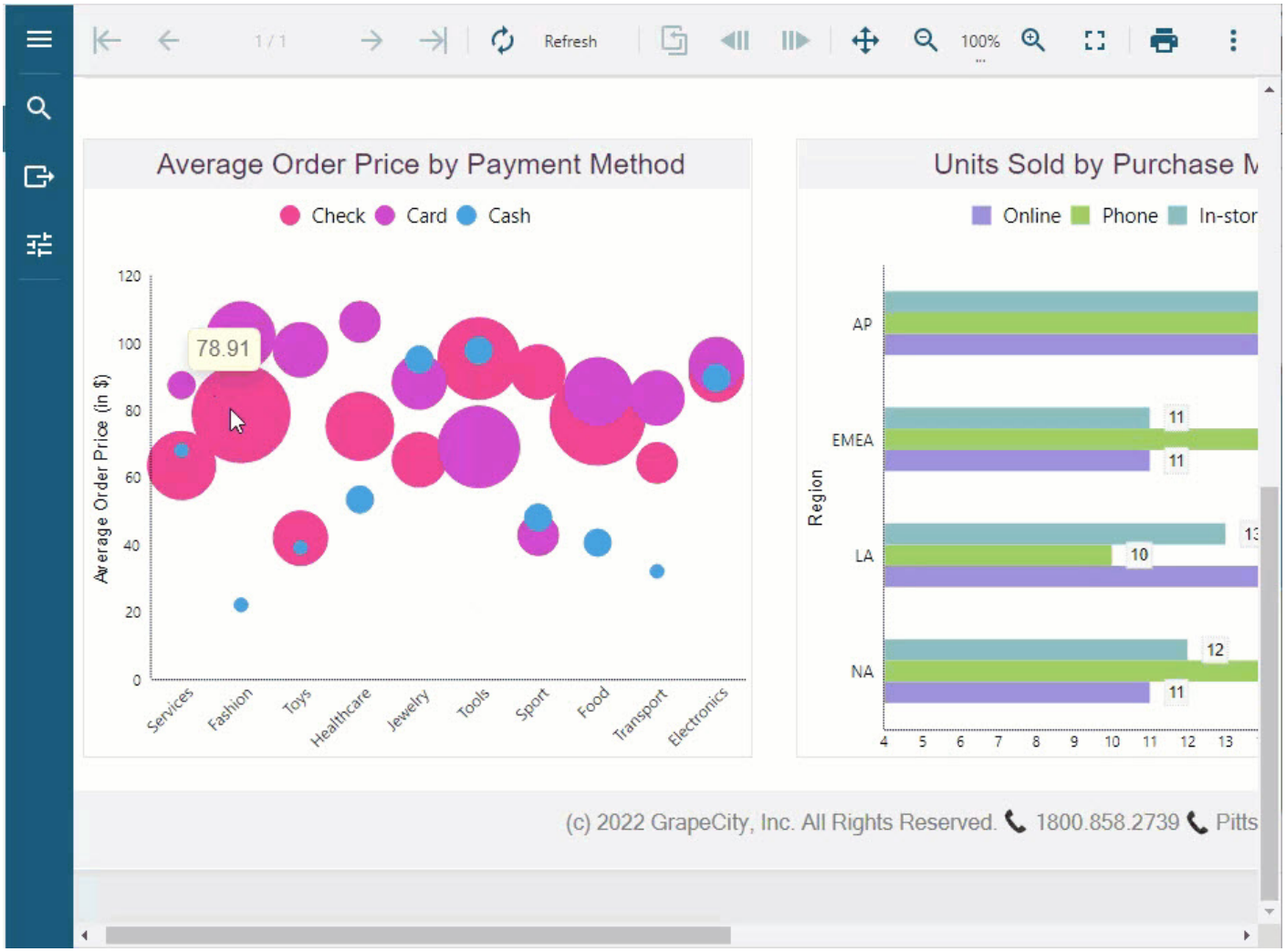
Hover Animation

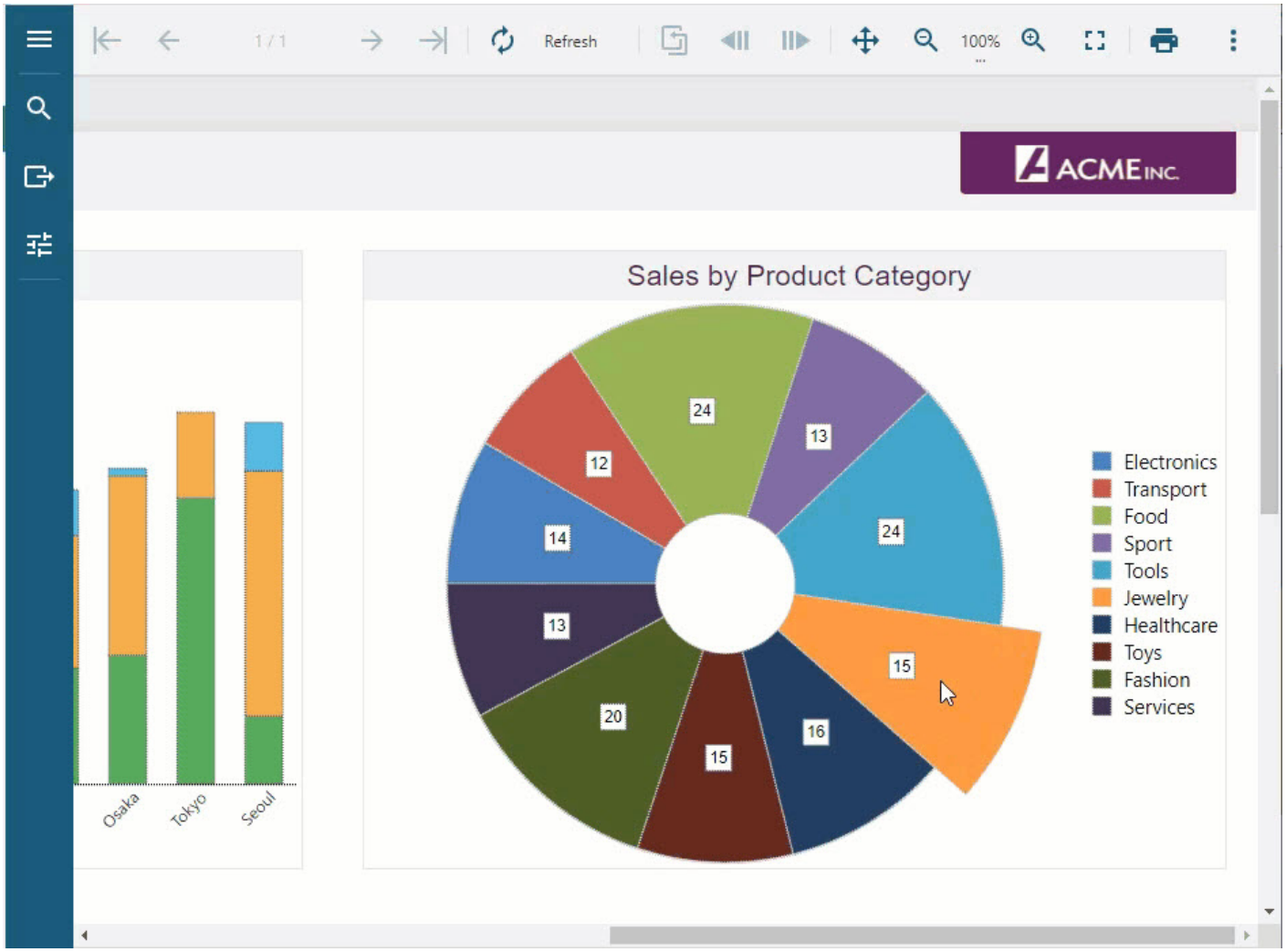
Hover animation occurs when the user hovers the mouse over a data point.

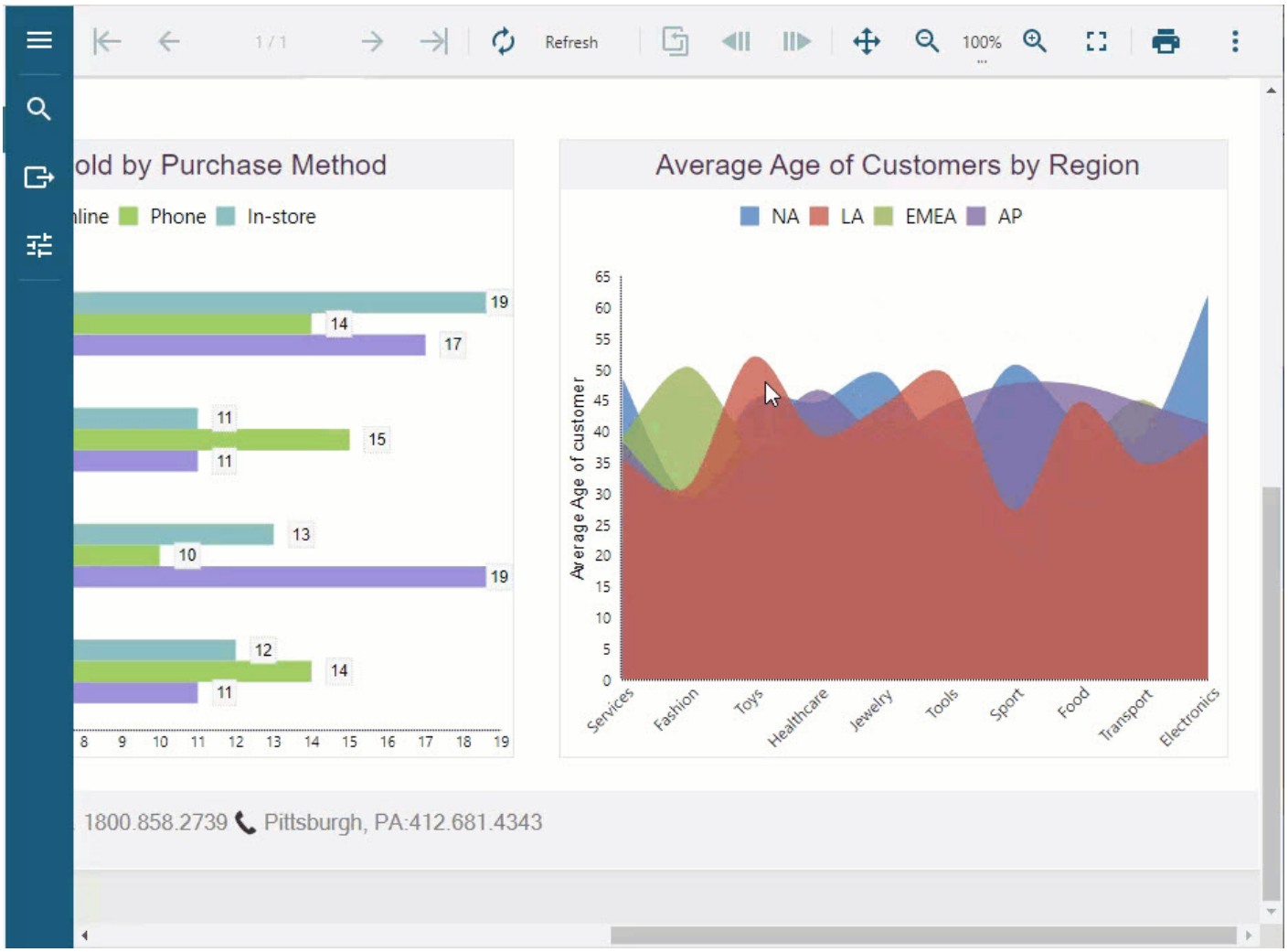
Hover Animation on Charts

The hover animation in the case of charts scales the data point. The data point is enlarged when the user hovers the mouse over the data and is shrunk back to the original size when the user hovers away from the data point. Depending on the Tooltip template set on the chart while designing a chart, the hover labels can also be displayed over the data points.







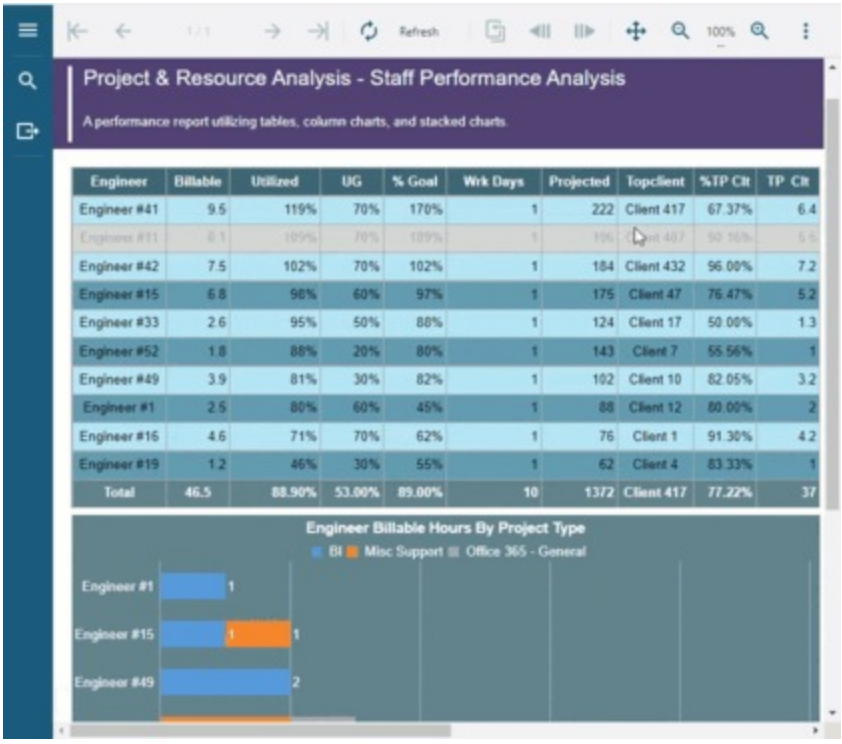


index.html

```
animation: {
    hoverChart: {
        enabled: true
                },
           }
```

Hover Animation on Tables

The hover animation in the case of tables highlights the table row by coloring the row background and the text on the hover.



index.html

```
animation: {
    hoverTable: {
        enabled: true,
        backgroundColor: 'LightGray',
        textColor: 'DarkGray'
                }
            }
```

Complete Code Implementation

The complete working code is provided below for implementing the chart load, and the chart and table hover animations.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="jsViewer.chart.min.css" rel="stylesheet" />
```

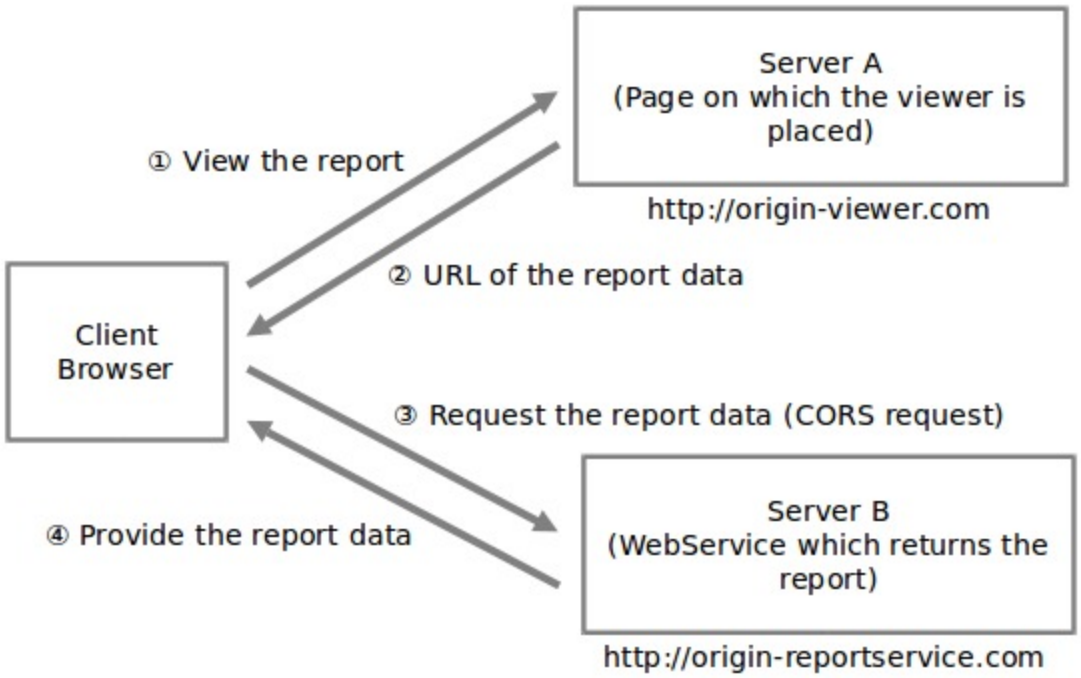
```
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript" src="jsViewer.chart.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                displayMode: "Continuous",
                pageView: {
                    horizontalAlignment: "center",
                    viewMode: "standard" // default : noPaper
                },
                animation: {
                    hoverTable: {
                        enabled: true,
                        backgroundColor: 'LightGray',
                        textColor: 'DarkGray'
                    },
                    loadChart: {
                        enabled: true
                    },
                    hoverChart: {
                        enabled: true
                    }
                }
            });

            console.dir(viewer)
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

⚠ **Important:** The plot rules do not work on charts with animation. If you want the plot rules to apply on a chart, you must disable the animation properties.

View Reports from Different Domains using CORS

Cross-Origin Resource Sharing is a technology for the web that provides async web operations to directly access reports from different domains. CORS works by adding a special header to responses from a server to the client. If a response contains the Access-Control-Allow-Origin header, then you can directly access the reports from another

domain.



The following viewers require CORS:

1. Js Viewer
2. HTMLViewer (WebViewer control)
3. RawHTML (WebViewer control)

.NET Framework

If Server is an ASP.NET application, then following markup should be added to the web.config file:

web.config

```
<customHeaders>
     <add name="Access-Control-Allow-Origin" value="http://localhost:44362" />
     <add name="Access-Control-Allow-Methods" value="GET, POST, OPTIONS" />
     <add name="Access-Control-Allow-Credentials" value="true"/>
     <add name="Access-Control-Allow-Headers" value="pragma,cache-control,expires,content-
type"/>
     <add name="Access-Control-Expose-Headers" value="Content-Disposition"/>
 </customHeaders>
```

Replace the "http://localhost:44362" with the actual client url.

.NET Core

If Server is an ASP.NET Core application,

1. Add the following code to the **Startup.ConfigureServices** method in **Startup.cs** file:

Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services
        .AddLogging(config =>
        {
            config.ClearProviders();
            if (Environment.GetEnvironmentVariable("ASPNETCORE_ENVIRONMENT") ==
Environments.Development)
            {
                config.AddConsole();
            }
        })
        .AddCors(options =>
        {
            options.AddPolicy("AllowAll", builder =>
            {
                builder.SetIsOriginAllowed(origin => new Uri(origin).Host == "XXXX")
                .AllowCredentials()
                .AllowAnyMethod()
                .AllowAnyHeader()
                .WithExposedHeaders("Content-Disposition");
            });
        })
        .AddReportViewer()
        .AddMvc(option => option.EnableEndpointRouting = false);
}
```

2. Add the following code to the **Startup.Configure** method in **Startup.cs** file:

Startup.cs

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    app.UseCors("AllowAll");
    app.UseReportViewer(settings =>
    {
        settings.UseEmbeddedTemplates(EmbeddedReportsPrefix, Assembly.GetAssembly(GetType()));
        settings.UseCompression = true;
    });
    app.UseMvc();
}
```

Note: If you get error 404 or 500 on the report preview, please make sure that your browser supports CORS.

Predefined Export Settings

The Export dialog with the export settings is build-in into the Js Viewer UI. You can easy configure the settings for a report reader to use as predefined export settings.

Enable/Disable Exports

Not all exports are available by default, some exports are hidden. To enable/disable exports, you should use the availableExports setting as demonstrated below:

index.html

```
const viewer = GrapeCity.ActiveReports.JSViewer.create({
    element: '#root',
    availableExports: ['Pdf', 'XlsxData'],
    // other properties
});
```

The following exports are available.

- WYSIWYG exports
  - Pdf - available for Page/RDLX and Section reports.
  - Tiff - available for Page/RDLX and Section reports.
- Excel exports
  - Xls - available for Page/RDLX and Section reports.
  - Xlsx - available for Page/RDLX and Section reports.
  - XlsxData - available for Page/RDLX reports. This export is hidden by default because it exports only tabular data.
- Word exports
  - Doc - available for Page/RDLX reports.
  - Docx - available for Page/RDLX reports.
  - Rtf - available for Section reports.
  - Mht - available for Page/RDLX and Section reports.
- Text exports
  - Csv - available for Page/RDLX reports.
  - CsvData - available for Page/RDLX reports. This export is hidden by default because it exports only tabular data.
  - TXT - available for Section reports.
  - JSON - available for Page/RDLX reports.
  - XML - available for Page/RDLX reports.

- TextPrint - available for Page/RDLX reports. This export is hidden by default because it is a replacement for a special generic/text only printer driver from Windows.

Pre-initialized Settings

To specify export settings, you should use the defaultExportSettings as demonstrated below:

index.html

```
const viewer = GrapeCity.ActiveReports.JSViewer.create({
    element: '#root',
    defaultExportSettings: {
    xls: {
      enableToggles: {
        value: true,
        visible: false
      },
      sheetName: {
        value: 'report'
      }
    }
  },
    // other properties
});
```

Customizing Export Settings

Excel 2003 (.xls)

Name	Type	Description
Both rdlx and rpx		
MultiSheet	Boolean	Indicate whether to generate single-sheet or multi-sheet Excel document. Default for rpx is 'false' and rdlx is 'true'.
SheetName	String	The name of the Excel sheet. Default is 'Sheet'.
UseDefaultPalette	Boolean	Indicate whether to export the Excel document with Excel default palette. **Remark:** Setting this value to true, the application will use the color which is in default palette and is closest to pre-defined custom color of the control's fore color and back color. Default is 'false'.

Orientation	`Default|Portrait|Landscape`	The orientation of the Excel document pages, to be printed in portrait or landscape.
PaperSize	`Default|Letter|LetterSmall|Tabloid|Ledger|Legal|Statement|Executive|A3|A4|A4Small|A5|B4|B5`	Size of the Excel document.
Password	String	The password required to open the Excel document.
ProtectedBy	String	User Name responsible to password protect the Excel document.
ReadOnlyRecommended	Boolean	Indicate if the Excel document was saved as read only recommended. Default is 'false'.
WritePassword	String	The password required to edit the document.
FileName	String	Name of the Excel document.
Only rpx		
FileFormat	`Xls97Plus|Xls95`	The file format version the exported Excel document should support. Default is 'Xls97Plus'.
AutoRowHeight	Boolean	Indicate whether to set the row heights in the Excel document according to the content in the rows. Choose carefully since it may affect pagination. Default is 'false'.
DisplayGridLines	Boolean	Indicate whether to display grid lines in the Excel document. Default is 'true'.
Pagination	Boolean	Indicate if pagination should be used for the resulting Excel document. Default is 'true'.
Only rdlx		
EnableToggles	Boolean	Indicate whether to export toggles from table details or groups to collapsible rows. Default is 'false'. This setting is applicable when Pagination is set to 'false'.
LayoutMode	`Galley|Paginated`	The layout mode to use for the exported Excel document.
RightToLeft	Boolean	Indicate whether to show the mirror image of sheets, that is, if sheets should be exported right to left. Note that content is not mirrored. Default is 'false'.

Customizing .xls Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {

                    xls: {
                        /* only rdlx */
                        EnableToggles: { value: true },
                        LayoutMode: { value: 'Galley' },
                        RightToLeft: { value: true },
                        /* only rpx */
                        FileFormat: { value: 'Xls95' },
                        AutoRowHeight: { value: true },
                        DisplayGridLines: { value: true },
                        Pagination: { value: false },
                        /* both (rdlx and rpx) */
                        MultiSheet: { value: false },
                        SheetName: { value: 'Sheet_xls' },
                        UseDefaultPalette: { value: true },
                        Orientation: { value: 'Landscape' },
                        PaperSize: { value: 'A5' },
                        Password: { value: '123456' },
                        ProtectedBy: { value: 'USER' },
                        ReadOnlyRecommended: { value: true },
                        WritePassword: { value: '123456' },
                        FileName: { value: "ar" }
                    },
                }
            });
            viewer.openReport("DemoReport.rdlx");
```

```
        }
    </script>
</body>
</html>
```

Excel (.xlsx)

Name	Type	Description
Both rdlx and rpx		
MultiSheet	Boolean	Indicate whether to generate single-sheet or multi-sheet Excel document. Default for rpx is 'false' and rdlx is 'true'.
SheetName	String	The name of the Excel sheet. Default is 'Sheet'.
UseDefaultPalette	Boolean	Indicate whether to export the Excel document with Excel default palette. **Remark:** Setting this value to true, application will use the color which is in default palette and is closest to the predefined custom color of the control's fore color and back color. Default is 'false'.
OutputFormat	`Transitional|Strict`	
UseCompression	Boolean	Indicate whether to use compression on exporting with Xlsx file format. Default is 'true'.
Orientation	`Default|Portrait|Landscape`	The orientation of the Excel document pages, to be printed in portrait or landscape.
PaperSize	`Default|Letter|LetterSmall|Tabloid|Ledger|Legal|Statement|Executive|A3|A4|A4Small|A5|B4|B5`	Size of the Excel document.
Password	String	The password required to open the Excel document.
ProtectedBy	String	User Name responsible to password protect the Excel document.
ReadOnlyRecommended	Boolean	Indicate if the Excel document was saved as read-only recommended. Default is 'false'.
WritePassword	String	The password required to edit the document.
FileName	String	The name of the Excel document.

Only rpx		
AutoRowHeight	Boolean	Indicate whether to set the row heights in the Excel document according to the content in the rows. Choose carefully since it may affect pagination. Default is 'false'
DisplayGridLines	Boolean	Indicate whether to display grid lines in the Excel document. Default is 'true'.
Pagination	Boolean	Indicate if pagination should be used for resulting Excel document. Default is 'true'.
OpenXmlStandard	`Transitional (default)|Strict`	The level of Open XML document conformance on exporting in Xlsx file format. The Excel document generated using Strict cannot be viewed on iOS devices.
Only rdlx		
EnableToggles	Boolean	Indicate whether to export toggles from table details or groups to collapsible rows. Default is 'false'.
LayoutMode	`Galley|Paginated`	The layout mode to use for the exported Excel document.
RightToLeft	Boolean	Show sheets right to left to show mirror image of sheets. Note that content is not mirrored. Default is 'false'.

Customizing .xlsx Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
```

```
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {

                  xlsx: {
                      /* only rdlx */
                      EnableToggles: { value: true },
                      LayoutMode: { value: 'Galley' },
                      RightToLeft: { value: true },
                      OutputFormat: { value: 'Strict' },
                      /* only rpx */
                      AutoRowHeight: { value: true },
                      DisplayGridLines: { value: false },
                      Pagination: { value: false },
                      OpenXmlStandard: { value: 'Strict' },
                      /* both (rdlx and rpx) */
                      MultiSheet: { value: false },
                      SheetName: { value: 'Sheet_xls' },
                      UseDefaultPalette: { value: true },
                      OutputFormat: { value: 'Strict' },
                      UseCompression: { value: false },
                      Orientation: { value: 'Landscape' },
                      PaperSize: { value: 'A5' },
                      Password: { value: '123456' },
                      ProtectedBy: { value: 'USER' },
                      ReadOnlyRecommended: { value: true },
                      WritePassword: { value: '123456' },
                      FileName: { value: "ar" }
                  },
                }
            });
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

Word 2003 (.doc)

Export to Word 2003 is available for Page and RDLX reports only.

Name	Type	Description
Author	String	The name of the author of the Word document.

BaseHref	String	The Base Url for the relative hyperlinks used in the Word document.
Generator	String	The identity of the generator of the Word document.
PageHeight	Integer	The height of the page of the Word document.
PageWidth	Integer	The width of the page of the Word document
Title	String	The title of the Word document.
FileName	String	The name of the Word document.

Customizing .doc Export Settings

index.html

```
<!DOCTYPE html>
 <html lang="en">
<head>
     <meta charset="utf-8">
     <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
     <meta name="theme-color" content="#000000">
     <title>JS Viewer</title>
     <link href="jsViewer.min.css" rel="stylesheet">
     <link href="index.css" rel="stylesheet">
 </head>
<body onload="loadViewer()">
     <div style="width: 100%">
         <div id="viewerContainer"></div>
     </div>
     <script type="text/javascript" src="jsViewer.min.js"></script>
     <script type="text/javascript">
         let viewer;
        function loadViewer() {
             viewer = GrapeCity.ActiveReports.JSViewer.create({
                 element: '#viewerContainer',
                defaultExportSettings: {

                  doc: {
                      Author: { value: 'USER', visible: true },
                      BaseHref: { value: 'www.com' },
                      Generator: { value: 'Created by USER' },
                      PageHeight: { value: '10' },
                      PageWidth: { value: '5' },
                      Title: { value: 'TITLE_01' },
                      FileName: { value: 'ar_doc', visible: true }
                  },
```

```
                }
            });
             viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

Word (.docx)

Export to Word is available for Page and RDLX reports only.

Name	Type	Description
Author	String	The name of the author of the Word document that appears in Word document properties.
CompanyName	String	The name of the company to appear in the Word document properties.
DocumentCompatibilityVersion	`Word2007|Word2010|Word2013 (default)`	The version in which the Word document should open.
DpiX	Integer	The horizontal resolution of the custom report items. Default is '96'.
DpiY	Integer	The vertical resolution of the custom report items. Default is '96'.
Title	String	The title of the Word document that appears in the Word document properties.
TOCAutoUpdate	Boolean	Indicate whether macros are inserted in the document to automatically update the TOC control on opening the Word document. Default is 'false'.
Orientation	`Default|Portrait|Landscape`	The orientation of the Word document.
PaperSize	`Default|Letter|LetterSmall|Tabloid|Ledger|Legal| Statement|Executive|A3|A4|A4Small|A5|B4|B5`	The size of the Word document.
Password	String	The password required to open the document.
ReadOnlyRecommended	Boolean	Indicate if the Word document was saved as read only recommended. Default is 'false'.
WritePassword	String	The password required to edit the

		Word document.
FileName	String	The name of the Word document.

Customizing .docx Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                    docx: {
                        Author: { value: 'USER' },
                        CompanyName: { value: 'USER_COMPANY' },
                        DocumentCompatibilityVersion: { value: 'Word2010' },
                        DpiX: { value: 80 },
                        DpiY: { value: 80 },
                        Title: { value: 'TITLE_01' },
                        TOCAutoUpdate: { value: true },
                        Orientation: { value: 'Portrait' },
                        PaperSize: { value: 'B4' },
                        Password: { value: '123456' },
                        ReadOnlyRecommended: { value: true },
                        WritePassword: { value: '123456' },
                        FileName: { value: 'ar_doc' }
                    },
                }
            });
            viewer.openReport("DemoReport.rdlx");
        }
```

```
    </script>
</body>
</html>
```

PDF (.pdf)

Name	Type	Description
Both rdlx and rpx		
Title	String	The title of the PDF document that appears in the document meta data.
Author	String	The name of the author of the PDF document that appears in the document meta data.
Subject	String	The subject of the PDF document displayed that appears in the document meta data.
Keywords	String	The keywords associated with the document that appear as the document meta data.
Application	String	The string value for the 'Application' field that appears in the document meta data.
EmbedFonts	`Partial` *`(default)`*`|All|None`	Indicate how the fonts used in the report should be embedded in the PDF document.
Version	`Pdf12|Pdf13|Pdf14` *`(default)`*`|Pdf15|Pdf16|Pdf17|Pdf20|PdfA1a|PdfA1b|PdfA2a|PdfA2b|PdfA2u|PdfA3a|PdfA3b|PdfA3u|PdfUA1`	The version of the PDF format the exported document is saved in.
UserPassword	String	The password required to open the document.
OwnerPassword	String	The owner password to be entered in the PDF reader that permits full access to the document regardless of the specified user permissions
Encrypt	Boolean	The value indicating whether the document is encrypted or not. Default is 'false'.
FileName	String	The name of the PDF document.
Only rpx		
ConvertMetaToPng	Boolean	Indicate whether Windows metafiles are converted to PNG files in the exported PDF document. Default is 'false'.

ExportBookmarks	Boolean	Indicate whether bookmarks are exported to the PDF document. Default is 'true'.
ImageInterpolation	None *(default)*| Auto	The image interpolation value.
ImageQuality	Medium *(default)*|Lowest|Highest	The quality used for any images that are converted by ActiveReports. Note that if a JPG image is used in the report, it is written directly to PDF without any conversion. Other image formats may incur a conversion, which this value will affect.

Customizing .pdf Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                    pdf: {
                        /*only for rpx*/
                        ConvertMetaToPng: { value: true },
                        ExportBookmarks: { value: false },
                        ImageInterpolation: { value: 'Auto' },
                        ImageQuality: { value: 'Highest' },
                        /*both (rdlx and rpx)*/
                        Title: { value: 'Document' },
                        Author: { value: 'USER' },
                        Subject: { value: 'PDF' },
                        Keywords: { value: 'PDF export' },
```

```
                        Application: { value: 'AR' },
                        EmbedFonts: { value: 'All' },
                        Version: { value: 'Pdf15' },
                        UserPassword: { value: 'user_pwd' },
                        OwnerPassword: { value: 'owner_pwd' },
                        Encrypt: { value: true },
                        FileName: { value: 'ar_pdf', visible: true }
                    },
                }
            });
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

CSV (.csv)

Export to CSV is available for Page and RDLX reports only.

Name	Type	Description
ColumnsDelimiter	*, (default)*|.	The text to be placed between fields in data row.
Encoding	Unicode (UTF-8) *(default)*|US-ASCII|Unicode	The encoding schema for the output.
NoHeader	Boolean	Indicate whether CSV Header should be omitted. Default is 'false'.
QuotationMode	Auto quote *(default)*|Always quote	The quotations to be added on the exported values, whether to quote only simple values or all exported values.
QuotationSymbol	" *(default)*|\	The quotation symbol or the qualifier character to put around the results.
RowsDelimiter	\r\n *(default)*|\\r\\n\\r\\n	The string to be placed between data rows.
DateTimeFormat	String	Default format for date time values, for example 'MM/dd/yyyy H:mm'.
NumericFormat	String	Default format for numeric values, for example '0.####'.
FileName	String	The name of the CSV document.

Customizing .csv Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                    csv: {
                        ColumnsDelimiter: { value: '.' },
                        Encoding: { value: 'Unicode' },
                        NoHeader: { value: true },
                        QuotationMode: { value: "Always quote", visible: true },
                        QuotationSymbol: { value: '\'' },
                        RowsDelimiter: { value: '\\r\\n\\r\\n' },
                        DateTimeFormat: { value: 'MM/dd/yyyy H:mm' },
                        NumericFormat: { value: 'money' },
                        FileName: { value: 'ar_csv' }
                    },
                }
            });
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

JSON (.json)

Export to JSON is available for Page and RDLX reports only.

Name	Type	Description
Formatted	Boolean	Indicate if the file name should be formatted with tabs and spaces for readability. Default is 'true'.

QuotePropertyNames	Boolean	Indicate if the property names should be enclosed in quotation marks. Default is 'false'.
FileName	String	The name of the JSON document.

Customizing .json Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                    json: {
                        Formatted: { value: false },
                        QuotePropertyNames: { value: true },
                        FileName: { value: 'ar_json' }
                    },
                }
            });
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

XML (.xml)

Export to XML is available for Page and RDLX reports only.

Name	Type	Description
Encoding	UTF-8 (default)|ASCII|Unicode	The encoding schema for the output.
FileName	String	The name of the XML document.

Customizing .xml Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                    xml: {
                        Encoding: { value: 'Unicode' },
                        FileName: { value: 'ar_xml' }
                    },
                }
            });
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

Tagged Image (.tiff)

Name	Type	Description
Both rpx and rdlx		

Dither	Boolean	The encoding schema for the output. Default is 'false'.
DpiX	Integer	The horizontal resolution of the rendered images. Default is '200'.
DpiY	Integer	The vertical resolution of the rendered images. Default is '196'.
FileName	String	The name of the TIFF image.
Only rpx		
Pagination	Boolean	Indicate if pagination should be used, that is, whether to render the entire report as a single image or each page of the report as separate images. Default is 'true'.
CompressionScheme	`Ccitt3 (default)| None | Rle | Ccitt4 | Lzw`	The type of compression scheme to use for the reports rendered as TIFF images.
Only rdlx		
Compression	`Ccitt3 (default)| None | Rle | Ccitt4 | Lzw`	The type of compression to use for the reports rendered as TIFF images.

Customizing .tiff Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                  tiff: {
```

```
                    /* only for rpx */
                    Pagination: { value: false },
                    CompressionScheme: { value: 'None' },
                    /* only for rdlx */
                    Compression: { value: 'None' },
                    /*both (rdlx and rpx)*/
                    Dither: { value: true },
                    DpiX: { value: 150 },
                    DpiY: { value: 140 },
                    FileName: { value: 'ar_tiff' }
                },
            }
        });
        viewer.openReport("DemoReport.rdlx");
    }
</script>
</body>
</html>
```

RTF (.rtf)

Export to RTF is available for Section Report only.

Name	Type	Description
EnableShapes	Boolean	Indicate whether to export the Shapes and Lines to RTF format. You will require Microsoft Word to view them correctly. Default is 'false'.
Pagination	Boolean	Indicate if the property names should be enclosed in quotation marks. Default is 'true'.
FileName	String	The name of the RTF document.

Customizing .rtf Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
```

```
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                    rtf: {
                        EnableShapes: { value: true },
                        Pagination: { value: false },
                        FileName: {
                            value: 'ar_mht'
                        }
                    },
                }
            });
            viewer.openReport("Invoice.rpx");
        }
    </script>
</body>
</html>
```

Web Archive (.mht)

Name	Type	Description
Both rpx and rdlx		
FileName	String	The name of the MHT document.
Only rpx		
BookmarkStyle	`Html (default) | None`	Specify Html to generate bookmarks if the report has the bookmarks.
CharacterSet	`UnicodeUtf8 (default) | Big5 | EucJp | HzGb2312 | Ibm850 | Iso2022Jp | Iso8859_1 | Iso8859_2 | Iso8859_5 | Iso8859_6 | Koi8r | ShiftJis`	The character set encoding to use for the outputted HTML pages. This property only takes effect if the IncludeHtmlHeader property is set to 'true'

	`| UnicodeUtf16`	
CreateFramesetPage	Boolean	Indicate whether 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. Default is 'false'.
IncludeHtmlHeader	Boolean	Indicate whether to embed the HTML output in another HTML document. Otherwise, the HTML output includes the usual HTML, HEAD, and BODY elements. Default is 'true'.
IncludePageMargins	Boolean	Indicate whether to include the report margins in the HTML output. Default is 'false'.
MultiPage	Boolean	Indicate whether to generate multiple HTML pages for the document. Default is 'false'.
OutputType	`DynamicHtml (default) | LegacyHtml`	The type of HTML output.
Pagination		Indicate whether to use pagination in the HTML output. Default is 'true'.
RemoveVerticalSpace		Indicate whether to remove empty vertical space from the output. Default is 'false'.
Title		The tile used in the HEAD of the HTML pages.
Only rdlx		
Fragment		Indicate whether to return only the contents inside the body tags so that you can embed it on the Web page. False returns the full HTML text.
OutputTOC		Indicate if the table of contents should be included in the output if it is available in the report. Default is 'true'.
LinkTarget	`_blank (default) | [window_name]: _self | _parent | _top`	The value for a target for hyperlinks contained inside the report. A value of • _blank opens a new window • _self opens the same window • _top opens the new page to load in the full body of the window • _parent similar to '_top' but refers to the immediate parent of a frame in case of nested frames
Mode	`Paginated (default) | Galley`	The layout mode to use for the exported document. • Paginated renders each page as a section inside the HTML document with Page headers and footers. • Galley renders one page with a single page header and footer.

Customizing .mht Export Settings

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
         let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
                  mht: {
                      /* only for rdlx */
                      Fragment: { value: true },
                      OutputTOC: { value: false },
                      LinkTarget: { value: '_top' },
                      Mode: { value: 'Galley' },
                      /* only for rpx */
                      BookmarkStyle: { value: 'None' },
                      CharacterSet: { value: 'Big5' },
                      CreateFramesetPage: { value: true },
                      IncludeHtmlHeader: { value: false },
                      IncludePageMargins: { value: true },
                      MultiPage: { value: true },
                      OutputType: { value: 'LegacyHtml' },
                      Pagination: { value: false },
                      RemoveVerticalSpace: { value: true },
                      Title: { value: 'TITLE_01' },
                      /*both (rdlx and rpx)*/
                      FileName: { value: 'ar_mht' }
                  },
                 }
            });
```

```
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

Hide Export Settings

To hide an export setting, you need to specify the **visible** property as 'false' along with the **value** property.

For example. the following code hides all the .doc export settings.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <link rel='shortcut icon' type='image/x-icon' href='favicon.ico' />
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <title>JS Viewer</title>
    <link href="jsViewer.min.css" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
</head>
<body onload="loadViewer()">
    <div style="width: 100%">
        <div id="viewerContainer"></div>
    </div>
    <script type="text/javascript" src="jsViewer.min.js"></script>
    <script type="text/javascript">
        let viewer;
        function loadViewer() {
            viewer = GrapeCity.ActiveReports.JSViewer.create({
                element: '#viewerContainer',
                defaultExportSettings: {
/*hide all doc export settings*/
                  doc: {
                      Author: { value: 'USER', visible: false },
                      BaseHref: { value: 'www.com', visible: false },
                      Generator: { value: 'Created by USER', visible: false },
                      PageHeight: { value: '10', visible: false },
                      PageWidth: { value: '5', visible: false },
                      Title: { value: 'TITLE_01', visible: false },
                      FileName: { value: 'ar_doc', visible: false }
                  },
                }
```

```
            });
            viewer.openReport("DemoReport.rdlx");
        }
    </script>
</body>
</html>
```

Js Viewer API

Check the online helpfile for Universal Module Definition (UMD) and ECMAScript Module (ESM) API documentation.

ASP.NET WebViewer Application

You can use the ASP.NET WebForms Viewer control with .NET Framework 4.6.2 - 4.8.1. only.

The WebViewer control that is licensed with the Professional Edition allows you to quickly display reports in Web applications. Once you drop the control onto a Web Form, you can look in the Visual Studio Properties grid and select the **ViewerType** that you want to use.

The WebViewer control supports the following types:

- **HTMLViewer** (default): Provides a scrollable view of a single page of the report at a time. Downloads only HTML and java script 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

In a WebViewer, an RDLX report can be rendered in two modes - Paginated and Galley. Using galley mode, you can view the contents of the RDLX report in a single and scrollable page. You can set Galley mode through UI of the WebViewer or through code by setting **RenderMode** property to **Galley**.

You should use the **ReportName ('ReportName Property' in the on-line documentation)** property to specify a report for the viewer to display. Also, you must add the **api/reporting/* HTTP handler** to your 'web.config' file and corresponding **UseReportViewer** call to 'Global.asax' for this ViewerType to work properly.

> **Note**: The WebViewer and JSViewer are supported only in the **Integrated pipeline mode**. You will get PlatformNotSupportedException on using these Viewers in Classic pipeline mode.

Use the WebViewer control

1. In Visual Studio, create a new ASP.NET Web Forms Application.
2. To install NuGet package for **MESCIUS.ActiveReports.Web**, go to **Tools > NuGet Package Manager > Manage NuGet Packages for Solution...**, browse for the package and click **Install**.
3. In Solution Explorer, right-click the project and select **Add > New Item**.
4. Select WebForm and click **Add**.
5. Go to the **Design** tab of the newly added WebForm and drag and drop the WebViewer control to the WebForm designer.

> **Note**: if you get an error on adding the WebViewer control, you should install or upgrade the Microsoft.CodeDom.Providers.DotNetCompilerPlatform NuGet package. See Troubleshooting for details.

Preview Code-Based Section Reports in WebViewer control

You need to update the **Global.asax** file as follows:

Global.asax.cs

```
public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            this.UseReporting(settings =>
            {
                settings.UseFileStore(new DirectoryInfo(Server.MapPath("~")));
                settings.UseCompression = true;
                settings.UseCustomStore(GetReport);
            });
        }
        public object GetReport(string reportName = "SectionReport")
        {
            SectionReport1 rpt = new SectionReport1();
            return rpt;
        }
    }
```

Global.asax.vb

```
Public Class _Global
    Inherits System.Web.HttpApplication
    Protected Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
        Me.UseReporting(Sub(settings)
                            settings.UseFileStore(New DirectoryInfo(Server.MapPath("~")))
                            settings.UseCompression = True
                            settings.UseCustomStore(AddressOf GetReport)
                        End Sub)
    End Sub
    Public Function GetReport(ByVal Optional reportName As String = "SectionReport") As Object
        Dim rpt As SectionReport1 = New SectionReport1()
        Return rpt
    End Function
End Class
```

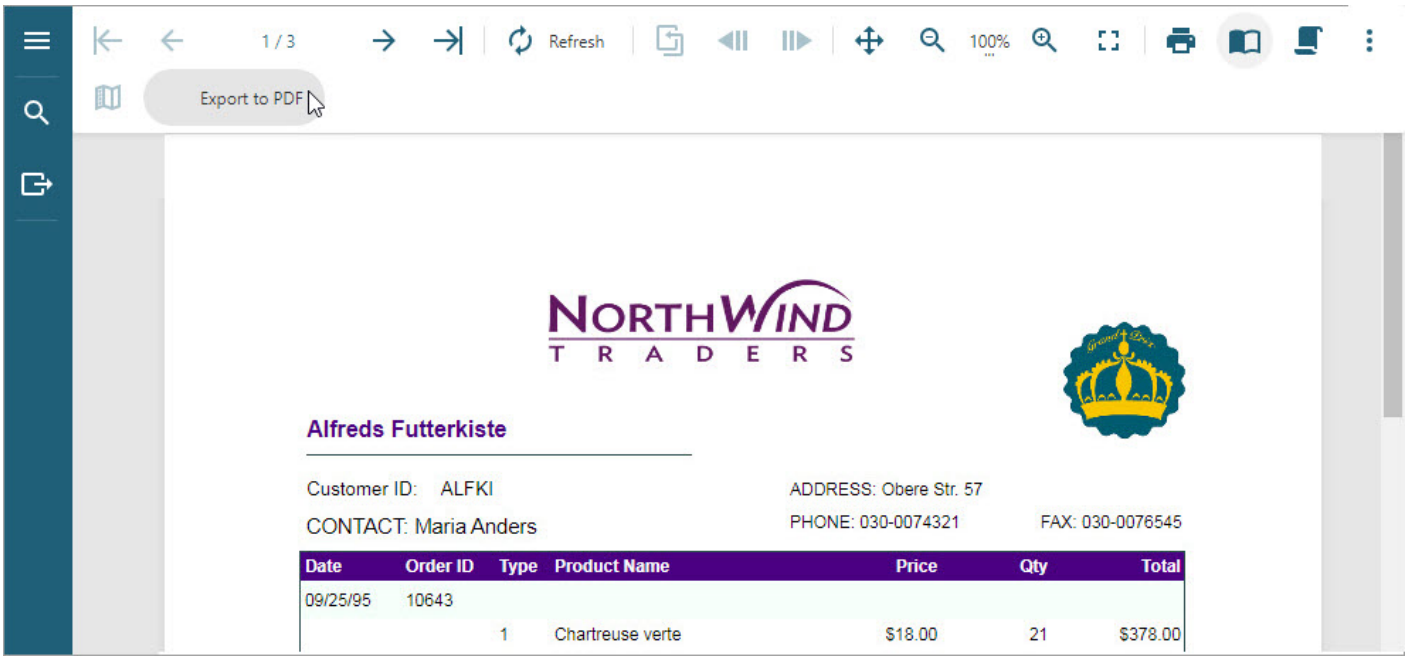
> **Note**: Instead of 'UseEmbeddedTemplates', you can use either 'UseFileStore' or 'UseCustomStore' method calls.
>
> - 'UseEmbeddedTemplates' stores reports as resources in dlls.
> - 'UseFileStore' stores reports in the file system.
> - 'UseCustomStore' allows you to store reports in any user-defined location, like a custom database or any

other type of location.

Customizing the WebViewer UI

You can customize the WebViewer interface using JQuery methods. WebViewer control adds JQuery library in page scripts. Use the code in this walkthrough to add a button on the toolbar and add a client side PDF export implementation.

When you complete this walkthrough you get a WebViewer that looks similar to the following at run time.



Load an ActiveReport to the Web application

1. Create a new Visual Studio **ASP.NET Web Forms** application.
2. Install **MESCIUS.ActiveReports.Web** package. Go to **Tools > NuGet Package Manager > Manage NuGet Packages for Solution...**, browse for the package and click **Install**.
3. In Solution Explorer, right-click the project and select **Add > New Item**.
4. Select WebForm and click **Add**.
5. Go to the **Design** view of the newly added WebForm.aspx and drag and drop the WebViewer control to the WebForm designer. The default viewer type is **HTMLViewer**.
6. Load a report in the WebViewer by setting the **ReportName** property.

> **Note**: You may load any report, section or page in the HTMLViewer viewer type of WebViewer. See ASP.NET WebViewer Application for information on loading a report.

Add the jQuery library to the Web application project

In the Source view of the **WebForm.aspx** file, add the following code.

Add this code after the <head> tag

```
<script src="https://code.jquery.com/jquery-2.1.4.min.js"></script>
```

Access the WebViewer view model

The HTML WebViewer is created using the MVVM pattern that provides a view model which does not depend on the UI. The code can access the Viewer's view model and bind the custom UI to it by using well-documented properties and methods. For MVVM support, the code can use knockout.js which is the standard MVVM library for JavaScript. Knockout.js provides declarative bindings, observable objects and collections in HTML markup.

Follow the steps below to access the ViewModel.

1. In the Source view of the **WebForm.aspx** file, add a <script> tag.
2. Add the following Javascript code for document's **Onload** event handler and WebViewer's **Loaded** event handler that gets fired when the UI is rendered on the Html Page:

   Paste the code into .aspx source

   ```
   <script>
   function viewer_loaded()
     {
     };
   function document_onload()
     {
     };
   </script>
   ...
   <body onload="document_onload()">
   ```

3. Add the following Javascript code inside the **viewer_loaded** event handler to access WebViewer's view model:

   Paste the code into .aspx source

   ```
   function viewer_loaded()
   {
      var viewModel = GetWebViewer('ArWebViewerDiv_WebViewer1');
   };
   ```

4. Add the following Javascript code inside the **document_onload** event handler to bind WebViewer's Loaded event to client side viewer_loaded event:

   Paste the code into .aspx source

   ```
   function document_onload()
   {
       $('#WebViewer1').ready(viewer_loaded);
   };
   ```

Add a button to the WebViewer toolbar

In the Source view of the WebForms.aspx file, add the following Javascript code inside the **viewer_loaded** event handler to access the WebViewer toolbar. Lets add the custom button in the toolbar - an export button and add PDF export functionality to it.

Paste the code into .aspx source

```
function viewer_loaded()
```

```
{
    var viewModel = GetWebViewer('ArWebViewerDiv_WebViewer1');
    var pdfExportButton = {
        key: '$pdfExportButtonKey',
        text: 'To PDF',
    iconCssClass: 'mdi mdi-file-pdf',
        enabled: true,
        action: function (item) {
            console.log('Export to PDF function works here');
        },
        onUpdate: function (arg, item) {
            console.log(The Viewer UI was updated, check/update button state here');
        }
    };
    viewModel.toolbar.desktop.addItem(pdfExportButton);
};
```

The complete code for WebForm1.aspx in the Source view is as shown:

Paste the code into .aspx source

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>

<!DOCTYPE html>
<html xmlns="https://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body onload="document_onload()">
    <form id="form1" runat="server">
        <div>
            <ActiveReportsWeb:WebViewer ID="WebViewer1" runat="server" height="466px"
width="667px" ReportName="AllCustomers.rdlx">
            </ActiveReportsWeb:WebViewer>
        </div>

    </form>
    <script src="https://code.jquery.com/jquery-2.1.4.min.js">
    </script>
    <script>
        function viewer_loaded() {
            var viewModel = GetWebViewer('ArWebViewerDiv_WebViewer1');

            var pdfExportButton = {
                key: '$pdfExportButtonKey',
```

```
                text: 'To PDF',
                iconCssClass: 'mdi mdi-file-pdf',
                enabled: true,
                action: function (item) {
                    console.log('Export to PDF function works here');
                },
                onUpdate: function (arg, item) {
                    console.log('Something in viewer was updated, check/update button state
here');
                }
            };

            viewModel.toolbar.desktop.addItem(pdfExportButton);
        };

        function document_onload() {
            $('#WebViewer1').ready(viewer_loaded);
        };
    </script>
</body>
</html>
```

To remove a button from the viewer's UI

Paste the code into .aspx source

```
function viewer_loaded()
{
    var viewModel = GetWebViewer('ArWebViewerDiv_WebViewer1');
    ...

    viewModel.toolbar.desktop.removeItem($newButtonKey); //(key of the button)
};
```

Note:

- Replace 'WebViewer1' in the code snippets above, with the actual ID of the WebViewer control in your application.
- When trying to get access of a WebViewer using GetWebViewer method, we should add 'ArWebViewerDiv_' prefix before the WebViewer ID.
- In case you provide report name which contains special symbols (like backslash '\'), e.g. webViewer.ReportName="Folder\Report.rdlx", you need to update the web.config file to allow such characters. Otherwise, "Report not found" error occurs. Please see Troubleshooting on resolving this issue.
- If your report has parameters, you can choose to set the parameters pane position to the top of the viewer using the **ParametersPanelLocation** to 'Top'.

Customize Export Settings

The export settings in the export panel of WebViewer (HTML) can now be preset via code as well as from the Properties panel in the design view of the viewer. The topic provides a complete description of the properties and the code sample of each of these. Along with presetting the export panel, the export settings can also be selectively opted to be hidden from the panel.

Excel 2003 (.xls)

Name	Type	Description
Both rdlx and rpx		
MultiSheet	Boolean	Indicate whether to generate single-sheet or multi-sheet Excel document. Default for rpx is 'False' and rdlx is 'True'.
SheetName	String	The name of the Excel sheet. Default is 'Sheet'.
UseDefaultPalette	Boolean	Indicate whether to export the Excel document with Excel default palette. **Remark:** Setting this value to 'True', the application will use the color which is in default palette and is closest to pre-defined custom color of the control's fore color and back color. Default is 'False'.
Orientation	`Default|Portrait|Landscape`	The orientation of the Excel document pages, to be printed in portrait or landscape.
PaperSize	`Default|Letter|LetterSmall|Tabloid|Ledger|Legal|Statement|Executive|A3|A4|A4Small|A5|B4|B5`	Size of the Excel document.
Password	String	The password required to open the Excel document.
ProtectedBy	String	User Name responsible to password protect the Excel document.
ReadOnlyRecommended	Boolean	Indicate if the Excel document was saved as read only recommended. Default is 'False'.
WritePassword	String	The password required to edit the document.
FileName	String	Name of the Excel document.
Only rpx		
FileFormat	`Xls97Plus|Xls95`	The file format version the exported Excel document should support. Default is 'Xls97Plus'.
AutoRowHeight	Boolean	Indicate whether to set the row heights in the

		Excel document according to the content in the rows. Choose carefully since it may affect pagination. Default is 'False'.
DisplayGridLines	Boolean	Indicate whether to display grid lines in the Excel document. Default is 'True'.
Pagination	Boolean	Indicate if pagination should be used for the resulting Excel document. Default is 'True'.
Only rdlx		
EnableToggles	Boolean	Indicate whether to export toggles from table details or groups to collapsible rows. Default is 'False'.
LayoutMode	`Galley *(default)*|Paginated`	The layout mode to use for the exported Excel document.
RightToLeft	Boolean	Indicate whether to show the mirror image of sheets, that is, if sheets should be exported right to left. Note that content is not mirrored. Default is 'False'.

Customizing .xls Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication4.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <XlsExportSettings>
                    <%-- only rdlx --%>
                    <FileFormat Value="Xls97Plus" />
                    <EnableToggles Value="True" />
                    <LayoutMode Value="Galley" />
                    <%-- only rpx --%>
```

```
                    <%-- both (rdlx and rpx) --%>
                    <MultiSheet Value="False" />
                    <SheetName Value="Sheet_xls" />
                    <UseDefaultPalette Value="True" />
                    <Orientation Value="Landscape" />
                    <PaperSize Value="A5" />
                    <AutoRowHeight Value="True" />
                    <DisplayGridLines Value="True" />
                    <Pagination Value="False" />
                    <RightToLeft Value="False" />
                    <Password Value="123456" />
                    <ProtectedBy Value="USER" />
                    <ReadOnlyRecommended Value="True" />
                    <WritePassword Value="1234567" />
                    <FileName Value="ar" />
                </XlsExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

Excel (.xlsx)

Name	Type	Description
Both rdlx and rpx		
MultiSheet	Boolean	Indicate whether to generate single-sheet or multi-sheet Excel document. Default for rpx is 'False' and rdlx is 'True'.
SheetName	String	The name of the Excel sheet. Default is 'Sheet'.
UseDefaultPalette	Boolean	Indicate whether to export the Excel document with Excel default palette. **Remark:** Setting this value to 'True', application will use the color which is in default palette and is closest to the predefined custom color of the control's fore color and back color. Default is 'False'.
OutputFormat	`Transitional|Strict`	
UseCompression	Boolean	Indicate whether to use compression on exporting with Xlsx file format. Default is

		'True'.
Orientation	`Default|Portrait|Landscape`	The orientation of the Excel document pages, to be printed in portrait or landscape.
PaperSize	`Default|Letter|LetterSmall|Tabloid|Ledger|Legal|Statement|Executive|A3|A4|A4Small|A5|B4|B5`	Size of the Excel document.
Password	String	The password required to open the Excel document.
ProtectedBy	String	User Name responsible to password protect the Excel document.
ReadOnlyRecommended	Boolean	Indicate if the Excel document was saved as read-only recommended. Default is 'False'.
WritePassword	String	The password required to edit the document.
FileName	String	The name of the Excel document.
Only rpx		
AutoRowHeight	Boolean	Indicate whether to set the row heights in the Excel document according to the content in the rows. Choose carefully since it may affect pagination. Default is 'False'
DisplayGridLines	Boolean	Indicate whether to display grid lines in the Excel document. Default is 'True'.
Pagination	Boolean	Indicate if pagination should be used for resulting Excel document. Default is 'True'.
OpenXmlStandard	`Transitional` *(default)*`|Strict`	The level of Open XML document conformance on exporting in Xlsx file format. The Excel document generated using Strict cannot be viewed on iOS devices.
Only rdlx		
EnableToggles	Boolean	Indicate whether to export toggles from table details or groups to collapsible rows. Default is 'False'.
LayoutMode	`Galley|Paginated`	The layout mode to use for the exported Excel document.
RightToLeft	Boolean	Show sheets right to left to show mirror image of sheets. Note that content is not mirrored. Default is 'False'.

Customizing .xlsx Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <XlsxExportSettings>
                    <OutputFormat Value="OpenXmlStrict" />
                    <UseCompression Value="False" />
                    <OpenXmlStandard Value="Strict" />
                    <EnableToggles Value="True" />
                    <MultiSheet Value="True" />
                    <LayoutMode Value="Galley" />
                    <SheetName Value="Sheet_xlsx" />
                    <UseDefaultPalette Value="True" />
                    <Orientation Value="Landscape" />
                    <PaperSize Value="A5" />
                    <AutoRowHeight Value="True" />
                    <DisplayGridLines Value="False" />
                    <Pagination Value="False" />
                    <RightToLeft Value="True" />
                    <Password Value="123456" />
                    <ProtectedBy Value="USER" />
                    <ReadOnlyRecommended Value="True" />
                    <WritePassword Value="123456" />
                    <FileName Value="ar_xlsx" />
                </XlsxExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

Word 2003 (.doc)

Export to Word 2003 is available for Page and RDLX reports only.

Name	Type	Description
Author	String	The name of the author of the Word document.
BaseHref	String	The Base Url for the relative hyperlinks used in the Word document.
Generator	String	The identity of the generator of the Word document.
PageHeight	Integer	The height of the page of the Word document.
PageWidth	Integer	The width of the page of the Word document
Title	String	The title of the Word document.
FileName	String	The name of the Word document.

Customizing .doc Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <DocExportSettings>
                    <Author Value="USER" Visible="True" />
                    <BaseHref Value="www.com" Visible="True" />
                    <Generator Value="Created by USER" Visible="True" />
                    <PageHeight Value="10" Visible="True" />
                    <PageWidth Value="5" Visible="True" />
                    <Title Value="TITLE_01" Visible="True"></Title>
                    <FileName Value="ar_doc" Visible="True" />
                </DocExportSettings>
```

```
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

Word (.docx)

Export to Word is available for Page and RDLX reports only.

Name	Type	Description
Author	String	The name of the author of the Word document that appears in Word document properties.
CompanyName	String	The name of the company to appear in the Word document properties.
DocumentCompatibilityVersion	`Word2007|Word2010|Word2013` *(default)*	The version in which the Word document should open.
DpiX	Integer	The horizontal resolution of the custom report items. Default is '96'.
DpiY	Integer	The vertical resolution of the custom report items. Default is '96'.
Title	String	The title of the Word document that appears in the Word document properties.
TOCAutoUpdate	Boolean	Indicate whether macros are inserted in the document to automatically update the TOC control on opening the Word document. Default is 'False'.
Orientation	`Default|Portrait|Landscape`	The orientation of the Word document.
PaperSize	`Default|Letter|LetterSmall|Tabloid|Ledger|Legal| Statement|Executive|A3|A4|A4Small|A5|B4|B5`	The size of the Word document.
Password	String	The password required to open the document.
ReadOnlyRecommended	Boolean	Indicate if the Word document was saved as read only recommended. Default is 'False'.
WritePassword	String	The password required to edit the Word document.

FileName	String	The name of the Word document.

Customizing .docx Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <DocxExportSettings>
                    <Author Value="USER" />
                    <CompanyName Value="USER_COMPANY" />
                    <DocumentCompatibilityVersion Value="Word2010" />
                    <DpiX Value="80" />
                    <DpiY Value="80" />
                    <Orientation Value="Landscape" />
                    <PaperSize Value="A5" />
                    <Password Value="123456"  />
                    <ReadOnlyRecommended Value="True" />
                    <WritePassword Value="123456" />
                    <Title Value="TITLE_01" Visible="True"></Title>
                    <TOCAutoUpdate Value="True" />
                    <FileName Value="ar_docx" />
                </DocxExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

PDF (.pdf)

Name	Type	Description

Both rdlx and rpx		
Title	String	The title of the PDF document that appears in the document meta data.
Author	String	The name of the author of the PDF document that appears in the document meta data.
Subject	String	The subject of the PDF document displayed that appears in the document meta data.
Keywords	String	The keywords associated with the document that appear as the document meta data.
Application	String	The string value for the 'Application' field that appears in the document meta data.
EmbedFonts	`Partial` *`(default)`*`|All|None`	Indicate how the fonts used in the report should be embedded in the PDF document.
Version	`PDF-1.2|PDF-1.3|PDF-1.4` *`(default)`*`|PDF-1.5|PDF-1.6|PDF-1.7|PDF-2.0|PDF/A-1a|PDF/A-1b|PDF/A-2a|PDF/A-2b|PDF/A-2u|PDF/A-3a|PDF/A-3b|PDF/A-3u|PDF/UA-1`	The version of the PDF format the exported document is saved in.
UserPassword	String	The password required to open the document.
OwnerPassword	String	The owner password to be entered in the PDF reader that permits full access to the document regardless of the specified user permissions
Encrypt	Boolean	The value indicating whether the document is encrypted or not. Default is 'False'.
FileName	String	The name of the PDF document.
Only rpx		
ConvertMetaToPng	Boolean	Indicate whether Windows metafiles are converted to PNG files in the exported PDF document. Default is 'False'.
ExportBookmarks	Boolean	Indicate whether bookmarks are exported to the PDF document. Default is 'True'.
ImageInterpolation	`None` *`(default)`*`| Auto`	The image interpolation value.
ImageQuality	`Medium` *`(default)`*`|Lowest|Highest`	The quality used for any images that are converted by ActiveReports. Note that if a JPG image is used in the report, it is written directly to PDF without any conversion. Other image formats may incur a conversion, which this value will affect.

Customizing .pdf Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <PdfExportSettings>
                    <Title Value="Document"></Title>
                    <Author Value="USER" />
                    <Subject Value="PDF1" />
                    <Keywords Value="PDF export" />
                    <Application Value="AR" />
                    <EmbedFonts Value="All" />
                    <Version Value="Pdf15" />
                    <UserPassword Value="userpsswd" />
                    <OwnerPassword Value="ownerpsswd" />
                    <Encrypt Value="True" />
                    <ConvertMetaToPng Value="True" />
                    <ExportBookmarks Value="False" />
                    <ImageInterpolation Value="Auto" />
                    <ImageQuality Value="Highest" Visible="False"/>
                    <FileName Value="ar_pdf" Visible="False"/>
                </PdfExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

CSV (.csv)

Export to CSV is available for Page and RDLX reports only.

Name	Type	Description
ColumnsDelimiter	`, (default)|.`	The text to be placed between fields in data row.
Encoding	`Unicode (UTF-8) (default)|US-ASCII|Unicode`	The encoding schema for the output.
NoHeader	Boolean	Indicate whether CSV Header should be omitted. Default is 'False'.
QuotationMode	`Auto quote (default)|Always quote`	The quotations to be added on the exported values, whether to quote only simple values or all exported values.
QuotationSymbol	`" (default)|\`	The quotation symbol or the qualifier character to put around the results.
RowsDelimiter	`\r\n (default)|\\r\\n\\r\\n`	The string to be placed between data rows.
DateTimeFormat	String	Default format for date time values, for example 'MM/dd/yyyy H:mm'.
NumericFormat	String	Default format for numeric values, for example '0.####'.
FileName	String	The name of the CSV document.

Customizing .csv Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <CsvExportSettings>
                    <DateTimeFormat Value="MM/dd/yyyy H:mm" Visible="True" />
                    <NumericFormat Value="money" Visible="True" />
                    <ColumnsDelimiter Value="." Visible="True" />
                    <Encoding Value="us_ascii" Visible="True" />
```

```
                    <NoHeader Value="True" Visible="True" />
                    <QuotationSymbol Value="\'" Visible="True" />
                    <QuotationMode Value="AlwaysQuote" Visible="True" />
                    <RowsDelimiter Value="\\r\\n\\r\\n" Visible="True" />
                    <FileName Value="ar_CSV" Visible="True" />
                </CsvExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

JSON (.json)

Export to JSON is available for Page and RDLX reports only.

Name	Type	Description
Formatted	Boolean	Indicate if the file name should be formatted with tabs and spaces for readability. Default is 'True'.
QuotePropertyNames	Boolean	Indicate if the property names should be enclosed in quotation marks. Default is 'False'.
FileName	String	The name of the JSON document.

Customizing .json Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <JsonExportSettings>
                    <Formatted Value="False" Visible="True" />
```

```
                    <QuotePropertyNames Value="True" Visible="True" />
                    <FileName Value="ar_json" Visible="True" />
                </JsonExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

XML (.xml)

Export to XML is available for Page and RDLX reports only.

Name	Type	Description
Encoding	UTF-8 *(default)*|ASCII|Unicode	The encoding schema for the output.
FileName	String	The name of the XML document.

Customizing .xml Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <XmlExportSettings>
                    <Encoding Value="ASCII" Visible="True" />
                    <FileName Value="ar_xml" Visible="True" />
                </XmlExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
```

```
</body>
</html>
```

Tagged Image (.tiff)

Name	Type	Description
Both rpx and rdlx		
Dither	Boolean	The encoding schema for the output. Default is 'False'.
DpiX	Integer	The horizontal resolution of the rendered images. Default is '200'.
DpiY	Integer	The vertical resolution of the rendered images. Default is '196'.
FileName	String	The name of the TIFF image.
Only rpx		
Pagination	Boolean	Indicate if pagination should be used, that is, whether to render the entire report as a single image or each page of the report as separate images. Default is 'True'.
CompressionScheme	`Ccitt3 (default)| None | Rle | Ccitt4 | Lzw`	The type of compression scheme to use for the reports rendered as TIFF images.
Only rdlx		
Compression	`Ccitt3 (default)| None | Rle | Ccitt4 | Lzw`	The type of compression to use for the reports rendered as TIFF images.

Customizing .tiff Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
```

```
                <DefaultExportSettings>
                <TiffExportSettings>
                    <Compression Value="None" Visible="True" />
                    <Dither Value="True" Visible="True" />
                    <DpiX Value="150" Visible="True" />
                    <DpiY Value="140" Visible="True" />
                    <Pagination Value="False" Visible="True" />
                    <CompressionScheme Value="None" Visible="True" />
                    <FileName Value="ar_tiff" Visible="True" />
                </TiffExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

RTF (.rtf)

Export to RTF is available for Section Report only.

Name	Type	Description
EnableShapes	Boolean	Indicate whether to export the Shapes and Lines to RTF format. You will require Microsoft Word to view them correctly. Default is 'False'.
Pagination	Boolean	Indicate if the property names should be enclosed in quotation marks. Default is 'True'.
FileName	String	The name of the RTF document.

Customizing .rtf Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
```

```
ReportName="Report.rpx">
                <DefaultExportSettings>
                <RtfExportSettings>
                    <EnableShapes Value="True" Visible="True"/>
                    <Pagination Value="False" Visible="True"/>
                    <FileName Value="ar_rtf" Visible="True"/>
                </RtfExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

Web Archive (.mht)

Name	Type	Description
Both rpx and rdlx		
FileName	String	The name of the MHT document.
Only rpx		
BookmarkStyle	`Html` *(default)* | `None`	Specify Html to generate bookmarks if the report has the bookmarks.
CharacterSet	`UnicodeUtf8` *(default)* | `Big5` | `EucJp` | `HzGb2312` | `Ibm850` | `Iso2022Jp` | `Iso8859_1` | `Iso8859_2` | `Iso8859_5` | `Iso8859_6` | `Koi8r` | `ShiftJis` | `UnicodeUtf16`	The character set encoding to use for the outputted HTML pages. This property only takes effect if the IncludeHtmlHeader property is set to 'True'
CreateFramesetPage	Boolean	Indicate whether 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. Default is 'False'.
IncludeHtmlHeader	Boolean	Indicate whether to embed the HTML output in another HTML document. Otherwise, the HTML output includes the usual HTML, HEAD, and BODY elements. Default is 'True'.
IncludePageMargins	Boolean	Indicate whether to include the report margins in the HTML output. Default is 'False'.

MultiPage	Boolean	Indicate whether to generate multiple HTML pages for the document. Default is 'False'.
OutputType	`DynamicHtml` *(default)* `| LegacyHtml`	The type of HTML output.
Pagination	Boolean	Indicate whether to use pagination in the HTML output. Default is 'True'.
RemoveVerticalSpace	Boolean	Indicate whether to remove empty vertical space from the output. Default is 'False'.
Title	String	The tile used in the HEAD of the HTML pages.
Only rdlx		
Fragment	Boolean	Indicate whether to return only the contents inside the body tags so that you can embed it on the Web page. 'False' returns the full HTML text.
OutputTOC	Boolean	Indicate if the table of contents should be included in the output if it is available in the report. Default is 'True'.
LinkTarget	`_blank` *(default)* `| [window_name]: _self | _parent | _top`	The value for a target for hyperlinks contained inside the report. A value of • _blank opens a new window • _self opens the same window • _top opens the new page to load in the full body of the window • _parent similar to '_top' but refers to the immediate parent of a frame in case of nested frames
Mode	`Paginated` *(default)* `| Galley`	The layout mode to use for the exported document. Paginated renders each page as a section inside the HTML document with Page headers and footers. Galley renders one page with a single page header and footer.

Customizing .mht Export Settings

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head runat="server">
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
                <MhtExportSettings>
                    <Fragment Value="True" Visible="True" />
                    <OutputTOC Value="False" Visible="True" />
                    <LinkTarget Value="_top" Visible="True" />
                    <Mode Value="Galley" Visible="True" />
                    <BookmarkStyle Value="None" Visible="True" />
                    <CharacterSet Value="Big5" Visible="True" />
                    <CreateFramesetPage Value="True" Visible="True" />
                    <IncludeHtmlHeader Value="False" Visible="True" />
                    <IncludePageMargins Value="True" Visible="True" />
                    <MultiPage Value="True" Visible="True" />
                    <OutputType Value="LegacyHtml" Visible="True" />
                    <Pagination Value="False" Visible="True" />
                    <RemoveVerticalSpace Value="True" Visible="True" />
                    <Title Value="TITLE_01" Visible="True"></Title>
                    <FileName Value="ar_mht" Visible="True" />
                </MhtExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

Hide Export Settings

To hide an export setting, you need to specify the **Visible** property as 'False' along with the **Value** property.

For example. the following code hides all the .doc export settings.

WebForm1.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<%@ Register Assembly="MESCIUS.ActiveReports.Web" namespace="GrapeCity.ActiveReports.Web"
tagprefix="ActiveReportsWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
```

```
    <title></title>
</head>
<body style="height: 219px">
    <form id="form1" runat="server">
        <div>
            <activeReportsweb:WebViewer ID="WebViewer1" runat="server" Style="z-index: 102;
left: 9px; position: absolute; top: 33px; height: calc(100%)" Width="95%"
ReportName="Report.rdlx">
                <DefaultExportSettings>
<%--hide all doc export settings--%>
                <DocExportSettings>
                    <Author Value="USER" Visible="False" />
                    <BaseHref Value="www.com" Visible="False" />
                    <Generator Value="Created by USER" Visible="False" />
                    <PageHeight Value="10" Visible="False" />
                    <PageWidth Value="5" Visible="False" />
                    <Title Value="TITLE_01" Visible="False"></Title>
                    <FileName Value="ar_doc" Visible="False" />
                </DocExportSettings>
            </DefaultExportSettings>
            </activeReportsweb:WebViewer>
        </div>
    </form>
</body>
</html>
```

Blazor Viewer Application

A Blazor Viewer application uses the same client-server model as the JS Viewer, so all details on the server-side configurations like Custom Font Resolver or Prevent Cross-Site Scripting Attacks are relevant for a Blazor Viewer as well.

This topic describes how to create a web application that embeds Blazor Viewer.

NuGet Packages

The following packages are required to be included in the application:

- MESCIUS.ActiveReports.Aspnetcore.Viewer
- MESCIUS.ActiveReports.Blazor.Viewer

Initialize Blazor Viewer Component

BlazorViewer.razor

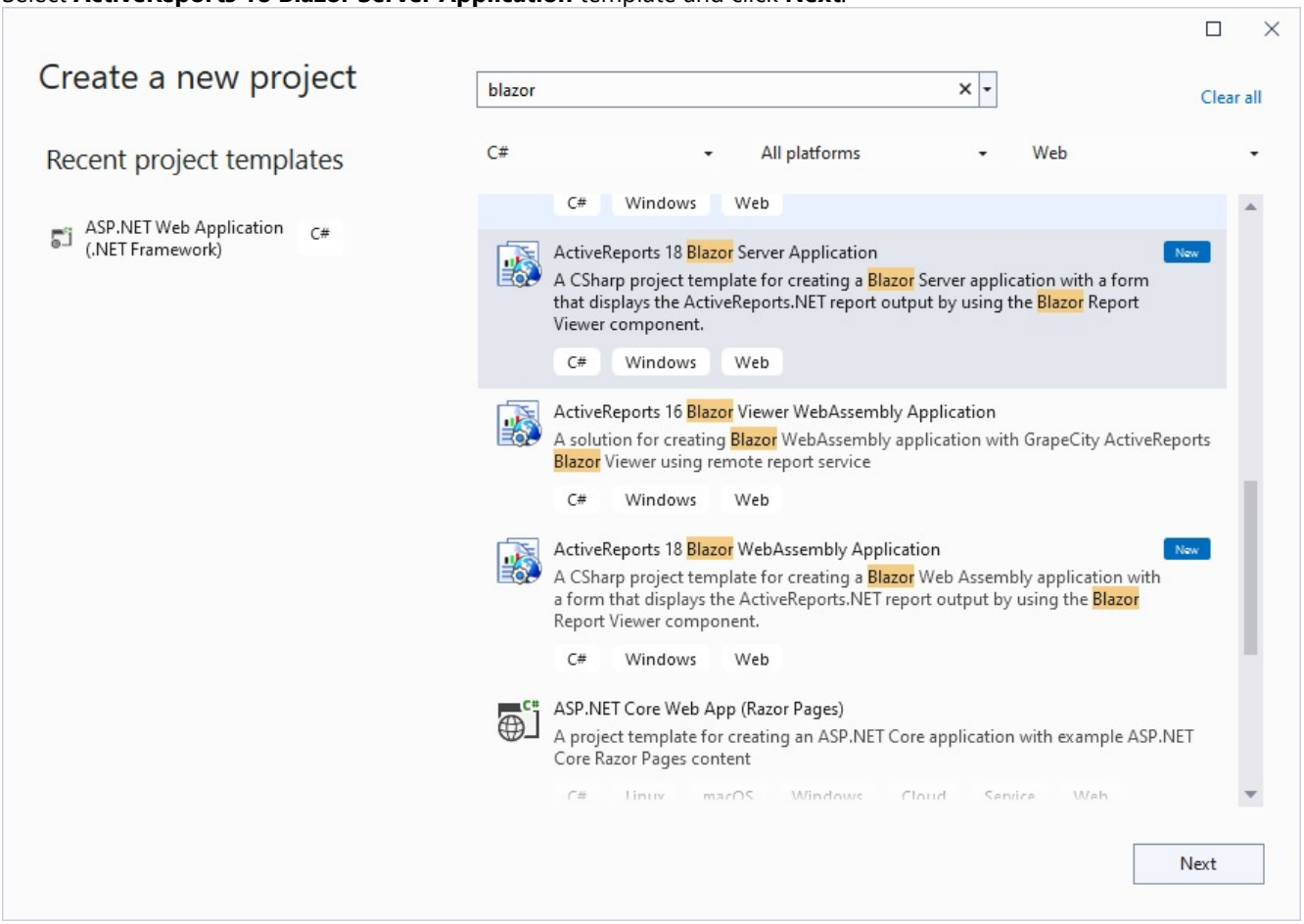
```
<div class="main">
    <div id="viewerContainer">
        <ReportViewer @ref="_viewer" ReportName="@_currentReport"
ViewerInitialized="InitializedViewer" DocumentLoaded="@DocumentLoaded"/>
    </div>
</div>
```

Create ActiveReports Blazor Viewer Server Application

Let us create a Blazor server application with ActiveReports Blazor Viewer using local report service.

Using built-in ActiveReports application template

1. Open **Microsoft Visual Studio 2022**.
2. Select **ActiveReports 18 Blazor Server Application** template and click **Next**.



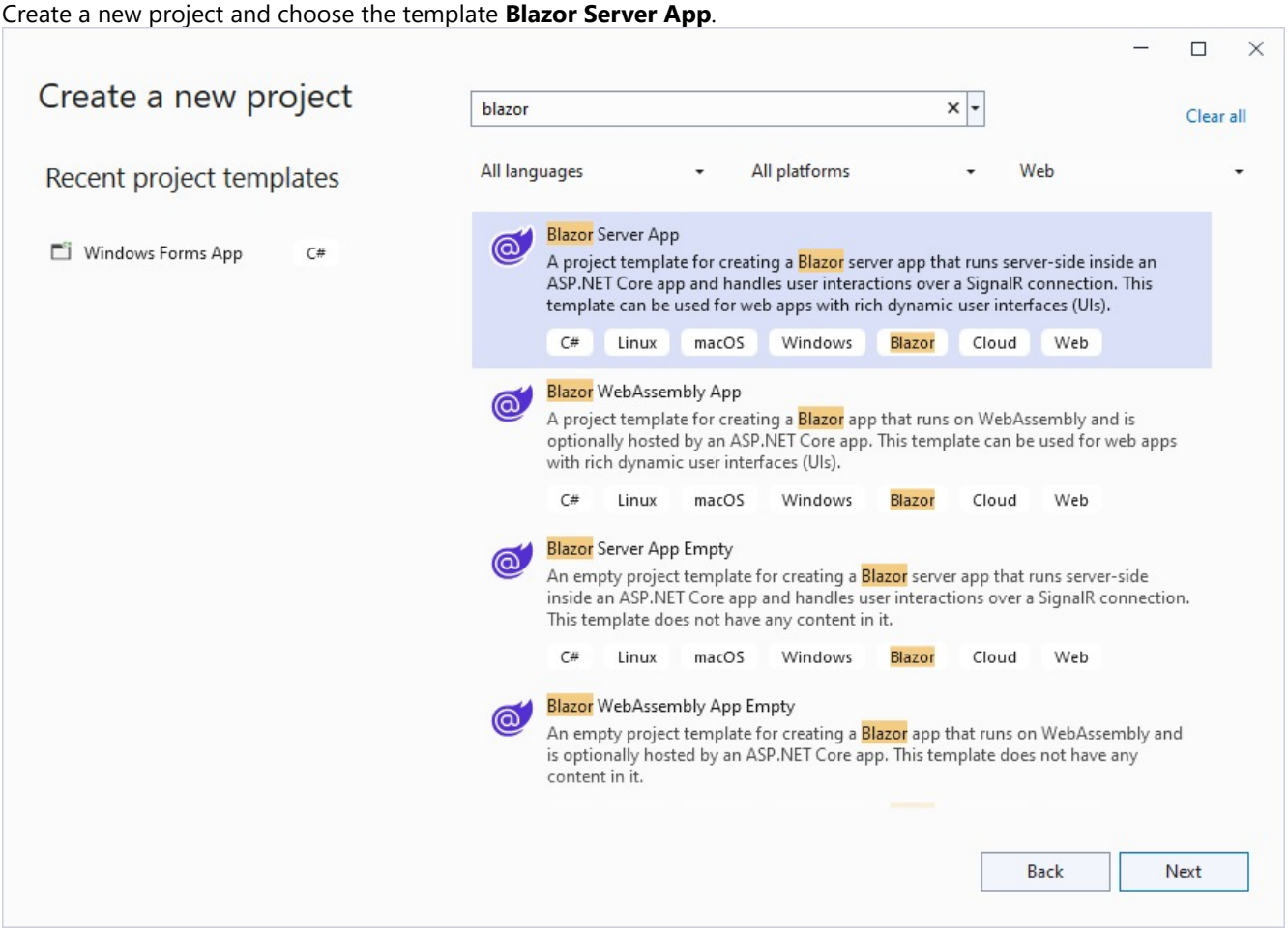
3. Type a name for your project and click **Create.**
4. From the **New Report** dialog, choose a report type, and click **Finish** to skip the data binding at this stage or choose **Next** to bind the data to the report.

   The following required packages are automatically added to project:
   - MESCIUS.ActiveReports.Aspnetcore.Viewer
   - MESCIUS.ActiveReports.Blazor.Viewer
5. Open the report from the 'Reports' folder and design.
6. Make sure to set the **Build Action** property of the report to 'Embedded Resource'.
7. Run the application.

Using Visual Studio Template

1. Open **Microsoft Visual Studio 2022**.

2. Create a new project and choose the template **Blazor Server App**.



3. Type a name for your project (say, BlazorApp1) and click **Next**.
4. Select a target framework and uncheck 'Configure for HTTPS' option, and then click **Create**.
5. Add the following packages:
    - MESCIUS.ActiveReports.Aspnetcore.Viewer
    - MESCIUS.ActiveReports.Blazor.Viewer

6. Add a new folder called 'Reports' in application's root and place the report (say, Report.rdlx) you want to display in the viewer, in this folder.

7. Make sure to set the **Build Action** property of the report to 'Embedded Resource'.

8. Add a Razor component to the folder **Pages** (right-click Pages > Add - Razor component). Set a name for it, for example, BlazorViewer.razor.

9. Write the following code in BlazorViewer.razor page:

BlazorViewer.razor

```
@page "/blazorviewer"

<h3>BlazorViewerTest</h3>
<div style="width:100%; height: 100vh">
    <GrapeCity.ActiveReports.Blazor.Viewer.ReportViewer ReportName="Report.rdlx">
</GrapeCity.ActiveReports.Blazor.Viewer.ReportViewer>
</div>
```

```
@code {

}
```

10. In the **Shared** folder > NavMenu.razor page, add the following code:

NavMenu.razor

```
<li class="nav-item px-3">
            <NavLink class="nav-link" href="blazorviewer">
                <span class="oi oi-list-rich" aria-hidden="true"></span> Blazor Viewer
            </NavLink>
</li>
```

11. In the **Data** folder, add **ReportService.cs** class:

ReportService.cs

```
namespace BlazorApp1.Data
{
    public class ReportsService
    {
        public static string EmbeddedReportsPrefix = "BlazorApp1.Reports";
        public IEnumerable<string> GetReports()
        {
            string[] validExtensions = { ".rdl", ".rdlx", ".rdlx-master", ".rpx" };
            return GetEmbeddedReports(validExtensions);
        }

        private static string[] GetEmbeddedReports(string[] validExtensions) =>
            typeof(ReportsService).Assembly.GetManifestResourceNames()
                .Where(x => x.StartsWith(EmbeddedReportsPrefix))
                .Where(x => validExtensions.Any(x.EndsWith))
                .Select(x => x.Substring(EmbeddedReportsPrefix.Length + 1))
                .ToArray();

    }
}
```

12. Update **Program.cs** file:
    - add following directives:
        - using GrapeCity.ActiveReports.Aspnetcore.Viewer;
        - using System.Reflection;
    - add service to the application
    - provide access to a report output from the browser by adding the **UseReportViewer()** middleware

Program.cs

```
builder.Services.AddSingleton<ReportsService>();

app.UseReportViewer(settings =>
{
    settings.UseEmbeddedTemplates(ReportsService.EmbeddedReportsPrefix,
Assembly.GetAssembly(typeof(ReportsService)));
});
```

> **Note**: Instead of 'UseEmbeddedTemplates', you can use either 'UseFileStore' or 'UseCustomStore' method calls.
>
> - 'UseEmbeddedTemplates' stores reports as resources in dlls.
> - 'UseFileStore' stores reports in the file system.
> - 'UseCustomStore' allows you to store reports in any user-defined location, like a custom database or any other type of location.

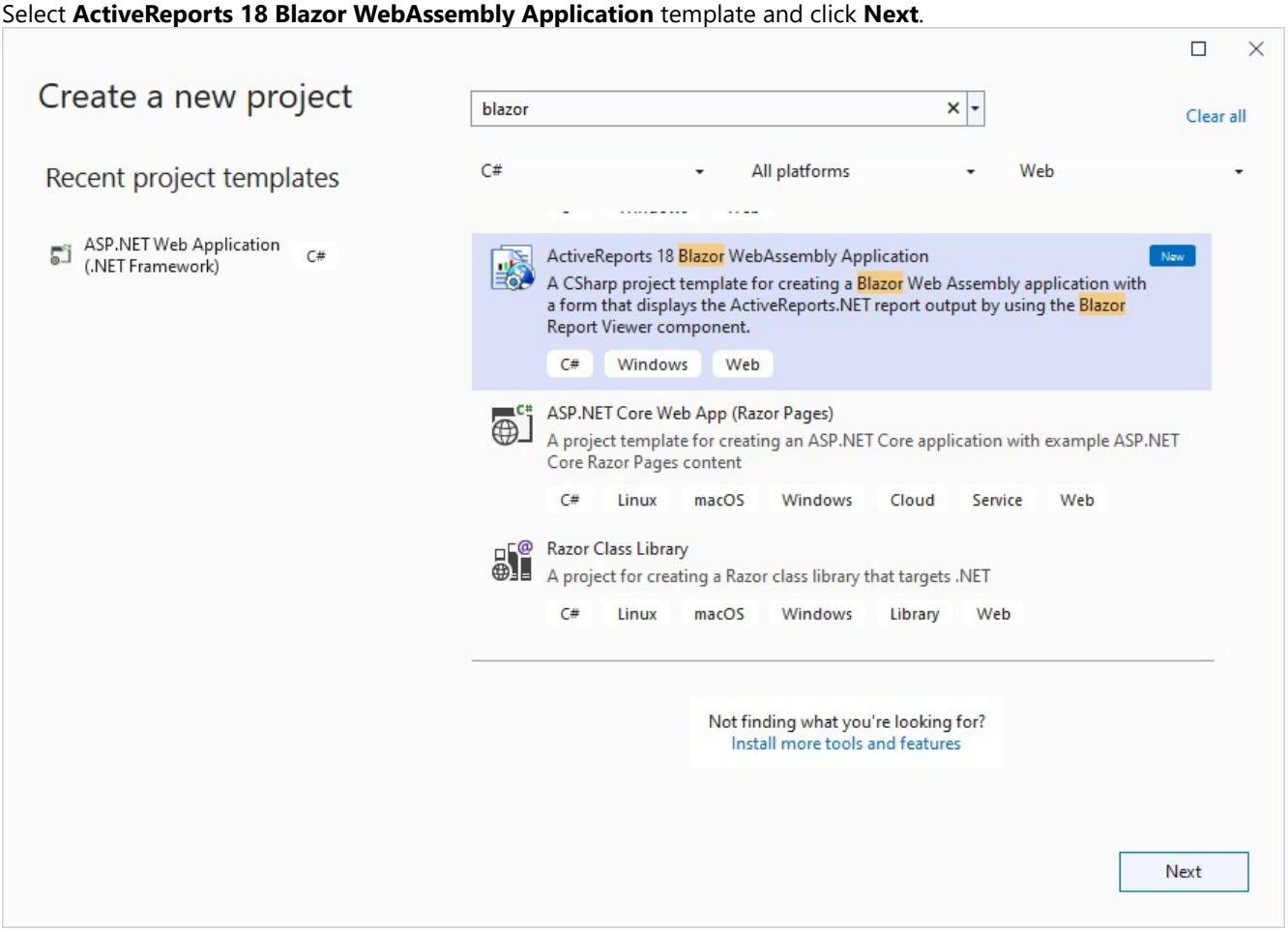
13. Run the application.

Create ActiveReports Blazor Viewer WebAssembly Application

Here we describe creating the application using built-in ActiveReports application template. This application uses remote report service.

Using built-in ActiveReports application template

1. Open **Microsoft Visual Studio 2022**.
2. Select **ActiveReports 18 Blazor WebAssembly Application** template and click **Next**.



3. Type a name for your project and click **Create**.
4. From the **New Report** dialog, choose a report type, and click **Finish** to skip the data binding at this stage or choose **Next** to bind the data to the report.

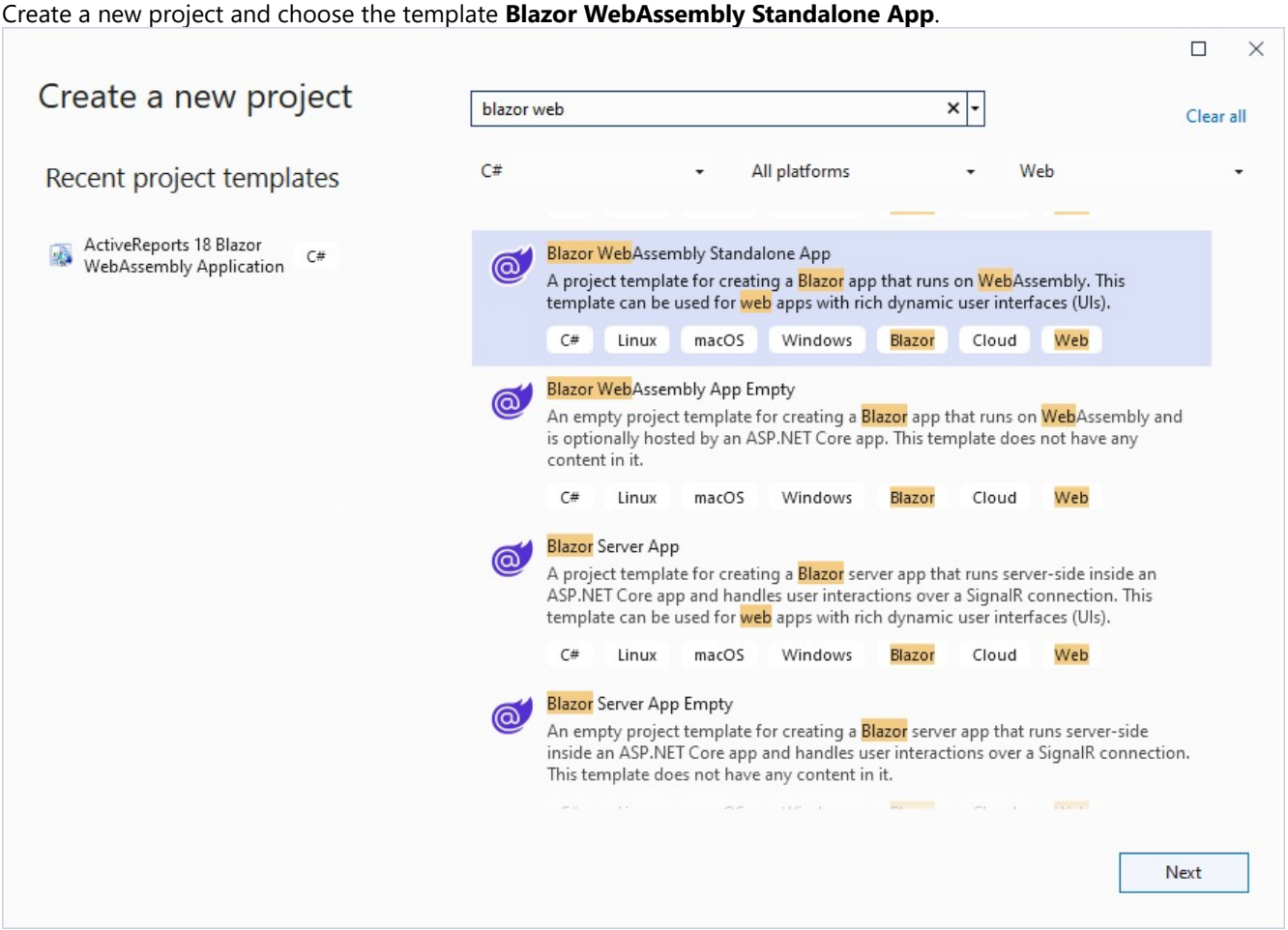
   The following required packages are automatically added to the project:

- MESCIUS.ActiveReports.Blazor.Viewer

5. Open the report from the 'Reports' folder and design.
6. Make sure to set the **Build Action** property of the report to 'Embedded Resource'.
7. Make sure to set multiple startup projects:
    1. In the Solution Explorer, select the solution (the top node).
    2. Choose the solution node's context (right-click) menu and then choose Properties.
    3. In the Solution Property Pages dialog box, expand the Common Properties node, and choose Startup Project.
    4. Choose the **Multiple Startup Projects** option and set the actions of both projects to **Start**.
8. Make sure that the URL is the same in **launchSettings.json** file of the ReportService project and **index.razor** file of BlazorViewerWebAssembly project.
9. Run the application.

Using Visual Studio Template

1. Open **Microsoft Visual Studio 2022**.
2. Create a new project and choose the template **Blazor WebAssembly Standalone App**.



3. Type a name for your project and click **Next**.
4. Select a target framework and uncheck 'Configure for HTTPS' option, and then click **Create**.
5. Add the following package:
    - MESCIUS.ActiveReports.Blazor.Viewer

6. Add a new folder called 'Reports' in application's root and place the report (say, Report.rdlx) you want to display in the viewer, in this folder.
7. Make sure to set the **Build Action** property of the report to 'Embedded Resource'.
8. Update the existing Index.razor page as follows:

BlazorViewer.razor

```
@page "/"
@using MESCIUS.ActiveReports.Blazor.Viewer
<PageTitle>Index</PageTitle><div style="width:100%; height: 100vh">
 <ReportViewer ReportName="Report.rdlx" ReportService="@_reportService"></ReportViewer></div>
@code{
    private ReportServiceSettings _reportService = new ReportServiceSettings()
    {
        Url = "http://localhost:58865/",
    };
}
```

9. Run the application.

Progressive Web Application

To create a Progressive Web Application (PWA), use our Blazor Viewer and following common Microsoft guidelines:

- ASP.NET Core Blazor Progressive Web Application (PWA)
- Building a Progressive Web App with Blazor

UI Customization

The Blazor Viewer API lets developers completely overwrite the toolbar's default user interface and the viewer component's sidebar.

Update layout

Update the toolbar layout to Desktop, Fullscreen, or Mobile as shown.

BlazorViewer.razor

```
@page "/"
@using BlazorViewerServer.Data
@inject ReportsService ReportsService

<div class="main">

    @* Used to render list of Reports on the page *@
    <ReportList ReportsList="@reportsList" CurrentReport="@_currentReport"
OnClickCallback="OnClick"></ReportList>

    <div id="viewerContainer">
```

```
        <ReportViewer @ref="_viewer" ReportName="@_currentReport"
ViewerInitialized="InitializedViewer"/>
    </div>
</div>

@code{

    private ReportViewer _viewer;
    private List<string> reportsList;
    private string _currentReport = null;
    private bool _documentLoaded = false;

    protected override void OnInitialized()
    {

        reportsList = ReportsService.GetReports().ToList();
        _currentReport = reportsList.FirstOrDefault();
        //Sets the first report as the value of _currentReport
    }

    //click event handler to reinitialize the _currentReport value and open the new report in
the viewer
    private async void OnClick(string res)
    {
        _currentReport = res;
        await _viewer.OpenReport(_currentReport);

    }

    private void InitializedViewer()
    {
        _viewer.Toolbar.Desktop.Layout(new string[] { "$zoom", "$split", "$fullscreen",
"$split", "$print" });
        //_viewer.Toolbar.Fullscreen.Layout(new string[] { "$fullscreen", "$print" });
        //_viewer.Toolbar.Mobile.Layout(new string[] { "$navigation"});
    }
}
```

Add custom toolbar item

The following example adds two export buttons - pdf export and excel export.

BlazorViewer.razor

```
@page "/"
@using BlazorViewerServer.Data
@inject ReportsService ReportsService
```

```
<div class="main">

    @* Used to render list of Reports on the page *@
    <ReportList ReportsList="@reportsList" CurrentReport="@_currentReport"
OnClickCallback="OnClick"></ReportList>

    <div id="viewerContainer">
        <ReportViewer @ref="_viewer" ReportName="@_currentReport"
ViewerInitialized="InitializedViewer" DocumentLoaded="@DocumentLoaded" Locale="ja-JP"/>
    </div>
</div>

@code{

    private ReportViewer _viewer;
    private List<string> reportsList;
    private string _currentReport = null;
    private bool _documentLoaded = false;

    protected override void OnInitialized()
    {

        reportsList = ReportsService.GetReports().ToList();
        _currentReport = reportsList.FirstOrDefault();
        //Sets the first report as the value of _currentReport
    }

    //click event handler to reinitialize the _currentReport value and open the new report in
the viewer
    private async void OnClick(string res)
    {
        _currentReport = res;
        _documentLoaded = false;
        await _viewer.OpenReport(_currentReport);

    }

    private async void DocumentLoaded()
    {
        _documentLoaded = true;
    }

    private void InitializedViewer()
    {

        // To add a new item in the ToolBar (it is to be performed before the report is
opened)
```

```
        _viewer.Toolbar.Desktop.AddItem(new ToolbarItem()
        {
            Text = "Export to PDF",
            Key = "$ExportPDF",
            Enabled = true,
            Title = "Export to PDF",
            Action = async () =>
            {
                // you can perform export only after the document is loaded.
                if (_documentLoaded)
                {
                    await _viewer.Export(ExportTypes.Pdf,
                    //Sets the export type
                    (uri) =>
                    {
                        //callback Function that is invoked once the export result is
available (the Url is passed in the callback)
                    },
                    true, // Indicates whether the save as dialog should be shown immediately
once the result is ready
                    new Dictionary<string, string>() { { "Title", "Some Title" } },
                    //Export setting for Rendering Extensions
                    () =>
                    {
                        //checking export cancellation
                        return false;
                    });
                }
            }
        });


        _viewer.Toolbar.Desktop.AddItem(new ToolbarItem()
        {
            Text = "Export to Xlsx",
            Key = "$ExportExcel",
            Enabled = true,
            Title = "Export to Xlsx",
            Action = async () =>
            {
                // you can perform export only after the document is loaded.
                if (_documentLoaded)
                {
                    await _viewer.Export(ExportTypes.Xlsx,
                    //Sets the export type
                    (uri) =>
                    {
                        //   callback Function that is invoked once the export result is
```

```
available (the Url is passed in the callback)
                    },
                    true, // Indicates whether the save as dialog should be shown immediately
once the result is ready
                    new Dictionary<string, string>() { { "Title", "Some Title" } },
//Export setting for Rendering Extensions
                    () =>
                    {
                        //checking export cancellation
                        return false;
                    });
                }
            }
        });

    }
}
```

Show/hide toolbar

In the **InitializedViewer()** method, use the **Toggle** method to set the toolbar visibility.

BlazorViewer.razor

```
private void InitializedViewer()
 {
   _viewer.Toolbar.Toggle(false);
 }
```

Customize sidebar - Show/hide sidebar

In the **InitializedViewer()** method, use the **Toggle** method to set the sidebar visibility.

BlazorViewer.razor

```
private void InitializedViewer()
 {
   _viewer.Sidebar.Toggle(false);
 }
```

Blazor Viewer API

The page describes the Blazor Viewer API that can be used for initialization or at run time while working with the viewer.

RenderMode

Description: The initial render mode - 'Paginated' or 'Galley'. The default value is 'Paginated'.

Type: Enum: RenderMode

Accepted values: 'RenderMode.Paginated', 'RenderMode.Galley'

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" RenderMode="RenderMode.Paginated"/>
```

DefaultExportSettings

Description: The object containing custom default export settings. Use to preset the export settings' default value and its visibility in the export panel.

Type: Dictionary<string, Dictionary<string, ExportSetting>>

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
DefaultExportSettings="@defaultExportSettings"/>
@code{
  Dictionary<string, Dictionary<string, ExportSetting>> defaultExportSettings = new
Dictionary<string, Dictionary<string, ExportSetting>>()
    {
          {
              "xls", new Dictionary<string, ExportSetting>(){
                  {
                      "FileName",  new ExportSetting() {Value = "ar", Visible = true}
                  },
                  {
                      "EnableToggles",  new ExportSetting() {Value = false}
                  }
              }
          }
    };
}
```

AutoBackgroundColor

Description: When set to 'true', the background color of the viewing area is filled with the report's body color. This property is available only for RDLX Dashboard reports.

Type: bool

Accepted values: 'true', 'false'

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" AutoBackgroundColor="true" />
```

AvailableExports

Description: The array of export types available via export functionality of the viewer.

Type: ExportTypes[]

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
AvailableExports="availableExportArr"/>
@code{
    ExportTypes[] availableExportArr = new ExportTypes[] {
            ExportTypes.Pdf, ExportTypes.Xlsx, ExportTypes.Xls, ExportTypes.Json
        };
}
```

Locale

Description: The locale used for displaying the viewer.

Type: String

Accepted values: 'en-US' (for English), 'ja-JP' (for Japanese), and 'zh-CN' (for Chinese)

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" Locale="ja-JP"/>
```

LocaleData

Description: The JSON containing the localization strings.

Type: String

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" LocaleData="@_localeData" >
@code{
    string _localeData = "{\"viewer\": {\"toolbar\": {\"refresh\": \"更新\"} } }";
}
```

LocaleUri

Description: The url of the file containing the localization strings.

Type: String

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" LocaleUri="localization.json"/>
```

PanelsLocation

Description: The location of the panels or panes (search pane, parameters pane, etc.) to the left side ('toolbar') or the right side ('sidebar') of the viewer. The default value is 'toolbar'.

Type: Enum: PanelsLocation

Accepted values: 'PanelsLocation.sidebar', 'PanelsLocation.toolbar'

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
PanelsLocation="PanelsLocation.toolbar" />
```

DisplayMode

Description: Set the single page or continuous page mode for the viewer.

Type: Enum: ViewMode

Accepted values: 'ViewMode.Single', 'ViewMode.Continous'

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" DisplayMode="ViewMode.Single" />
```

Action

Description: The callback that is invoked before the viewer opens the hyperlink, bookmark link, drill-down report, or toggles the report control visibility.

Type: Method (string, object[])

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" Action="actionMethod" />
@code{
    public void actionMethod(string actionType , object[] actionParams)
    {
        System.Diagnostics.Debug.WriteLine(actionType);
        foreach(var obj in actionParams)
        {
            System.Diagnostics.Debug.WriteLine(obj);
        }
                }
}
```

Error

Description: The callback that is invoked when an error occurs in the process of displaying the report.

Type: Method(ErrorInfo obj)

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" Error="errorMethod" />
@code{
    public void errorMethod(ErrorInfo obj)
    {
        System.Diagnostics.Debug.WriteLine("Error Message :" + obj.Message);
                  }
}
```

HideErrors

Description: Specify whether to show errors in the viewer ('false' by default).

Type: bool

Accepted values: 'true', 'false'

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" HideErrors="true" />
```

InitialZoomMode

Description: Set the zoom mode at which the report should open in the viewer.

Type: String

Accepted values: 'FitToPage', 'FitToWidth'

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
InitialZoomMode="@ZoomMode.FitToWidth"/>
```

InitialZoomPercentage

Description: Set the zoom level in percentage at which the report should open in the viewer. If you set this property to, for example, 100, then the **InitialZoomMode** is ignored.

Type: Integer

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" InitialZoomPercentage="50"/>
```

The percentage value can range from 25 to 300.

DocumentLoaded

Description: The callback that is invoked when a document is loaded entirely on the server.

Type: Method()

Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" DocumentLoaded="DocumentLoaded"/>
@code{
    public void DocumentLoaded()
    {
        System.Diagnostics.Debug.WriteLine("Document Loaded");
    }
}
```

ReportService

Description: Set up the settings to connect the Web API.

Type: ReportServiceSettings object

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" ReportService="setting" />
@{
    ReportServiceSettings setting = new ReportServiceSettings()
    {
        Url = "http:example.com/api/reporting",
        SecurityToken = "security_Token",
        OnRequest = (OnRequestInfo obj) =>
        {
            obj.Headers.Add("Authorization", "security_Token");
        }
    };
}
```

ViewerInitialized

Description: The callback that is invoked when the viewer is initialized

Type: Method()

Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
ViewerInitialized="InitializedViewer"/>
@code{
    private void InitializedViewer()
    {
        System.Diagnostics.Debug.WriteLine("Viewer is initailized now");
                }
}
```

ReportLoaded

Description: The callback that is invoked when the viewer obtains the information about the requested report.

Type: Method(ReportInfo obj)

Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" ReportLoaded="ReportLoaded"/>
@code{
    public void ReportLoaded(ReportInfo obj)
    {
```

```
        System.Diagnostics.Debug.WriteLine("The report " + obj.Name + " was successfully
loaded!");
                    }
}
```

ParametersPanelSettings

Description: Set up the parameters panel or pane settings.

Type: ParametersPanelSettings object
Enums:

- **ParameterPanelLocation**: Adjusts the position of the parameters pane.
  Accepted Value: 'Default', 'Top'
- ParameterPanelOpen: Sets the parameters pane to be available by default, irrespective of whether a parameter is set to a default value or requires user input.
- Accepted Value: 'Auto', 'Always'

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
ParametersPanelSettings="parametersPanelSetting"/>
@code{
    ParametersPanelSettings parametersPanelSetting = new ParametersPanelSettings()
    {
        Location = ParameterPanelLocation.Default,
        Open = ParameterPanelOpen.Always
               };
}
```

Sidebar

Description: The viewer's sidebar instance. Use it to toggle the sidebar visibility.

Returns: Sidebar object

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
ViewerInitialized="InitializedViewer"/>
@{
private ReportViewer _viewer;

   private void InitializedViewer()
   {
Sidebar obj = _viewer.Sidebar;
   }
}
```

Parameters

Description: The array of the {name, value} pairs that describe the values of the parameters used to run the report.

Type: Parameter[]

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" Parameters="parameterArray"/>
@code{
    Parameter[] parameterArray = new Parameter[]
    {
        new Parameter
        {
            Name = "Category",
            Values = new string[]{"Business" }
        }
                };
}
```

ReportName

Description: The name of the report to be shown by the viewer.

Type: String

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" />
@code{
private ReportViewer _viewer;
private string _currentReport = null;
    protected override void OnInitialized()
    {
reportsList = ReportsService.GetReports().ToList();
_currentReport = reportsList.FirstOrDefault();
    }
}
```

Width

Description: The width of the viewer, by default 100%.

Type: String

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" Width="50%"/>
```

Height

Description: The width of the viewer, by default 100%.

Type: String

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport" Height="50%"/>
```

Toolbar

Description: The viewer's toolbar instance. Use it to add the custom elements or remove the existing ones.

Returns: Toolbar object

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
ViewerInitialized="InitializedViewer"/>
@{
private ReportViewer _viewer;

   private void InitializedViewer()
   {
Toolbar obj = _viewer.Toolbar;
   }
}
```

BackToParrent

Description: Makes the viewer to display the parent report of the drill-down report.

Type: Method

Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"
ViewerInitialized="InitializedViewer"/>
@code{
private ReportViewer _viewer;
    private void InitializedViewer()
    {
await  _viewer.BackToParrent();
    }
}
```

CurrentPage

Description: The currently displayed page number.

Type: Method

Returns: ValueTask<int>

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"  DocumentLoaded="@DocumentLoaded"/>
@code{
private async void DocumentLoaded()
```

```
    {
        var currentPage = await _viewer.CurrentPage();
                    System.Diagnostics.Debug.WriteLine(currentPage);
                }
}
```

Export

Description: Exports the currently displayed report.
Parameters:

- **exportType**: Specifies the export format.
- **callback**: Function that is invoked once the export result is available (its Url is passed in the callback)
- **saveAsDialog**: Indicates whether the save as dialog should be shown immediately once the export result is ready
- **settings**: The export settings are available for RenderingExtensions
- **isCancelRequested**: The function is periodically called with a check to cancel the export task

Type: Method(ExportTypes, Action<string> callback = null, bool = false, Dictionary<string, string> settings = null, Func<bool> isCancelRequested = null,)

Returns: Void

Sample code

```
<GrapeCity.ActiveReports.Blazor.ReportViewer @ref="_viewer" ReportName="@reportId"
DocumentLoaded="
@DocumentLoaded"/>
@code {
 private ReportViewer _viewer;
 private string reportId = "User defined report columns.rdlx";
 private async void DocumentLoaded()
    {
 await _viewer.Export(ExportTypes.Pdf,
 (uri) =>
    {
    //uri to export result
    },
 false,
 new Dictionary<string, string>() { { "Title", "Some Title" } },
 () =>
    {
   //hecking export cancellation
   return false;
    }
 );
    }
}
```

GetToc

Description: Obtains the report TOC.

Type: Method

Returns: ValueTask<BookmarkInfo>

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"  DocumentLoaded="@DocumentLoaded"/>
@code{
private async void DocumentLoaded()
    {
        var toc = await _viewer.GetToc();
                }
}
```

GoToPage

Description: Makes the viewer display the specific page. Page numeration starts with 1.
Type: int
Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"  DocumentLoaded="@DocumentLoaded"/>
@code{
private async void DocumentLoaded()
    {
                    await _viewer.GoToPage(2);
                }
}
```

OpenReport

Description: Makes the viewer to display the parent report of the drill-down report.
Type: Method(string, Parameter[] = null)
Returns: Void

Sample code

```
_viewer.OpenReport("TestReport.rdlx");
OR
    Parameter[] parameterArray = new Parameter[]
{
        new Parameter
        {
            Name = "Category",
            Values = new string[]{"Business" }
        }
};
_viewer.OpenReport("TestReport.rdlx", parameterArray);
```

PageCount

Description: Gets the page count of the currently displayed report.
Type: Method
Returns: ValueTask<int>

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"  DocumentLoaded="@DocumentLoaded"/>
@code{
private async void DocumentLoaded()
    {
         var countPage = await _viewer.PageCount();
        System.Diagnostics.Debug.WriteLine(countPage);
                }
}
```

Print

Description: Prints the currently displayed report if any.
Type: Method()
Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"  DocumentLoaded="@DocumentLoaded"/>
@code{
private async void DocumentLoaded()
    {
         await _viewer.Print();
                }
}
```

Refresh

Description: Refreshes the report preview.
Type: Method()
Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"  DocumentLoaded="@DocumentLoaded"/>
@code{
private async void DocumentLoaded()
    {
         await _viewer.Refresh();
                }
}
```

Search

Description: Performs the search of a specific term with specific search options (match case, whole word) and invokes the specific callback with the search result passed.
Parameters:

- **searchTerm**: String to find.
- **searchOptions**: The object optionally defines the search options.
- **callback**: The function to call after performing the search.

Type: Method(string, SearchOptions = null, Action<List<SearchResult>> = null)

Returns: Void

Sample code

```
<ReportViewer @ref="_viewer" ReportName="@_currentReport"  DocumentLoaded="@DocumentLoaded"/>
@code{
private async void DocumentLoaded()
    {
        //Searching ALFKI keyword in the report
        await _viewer.Search("ALFKI", new SearchOptions()
        {
            MatchCase = true,
            WholePhrase = false
        },
        (List<SearchResult> results) =>
        {
            if (results != null && results.Count > 0)
            {
                foreach (var res in results)
                {
                    System.Diagnostics.Debug.WriteLine(res.DisplayText);
                }

            }
        });
    }
```

WebDesigner Application

The ActiveReports WebDesigner is a report designer, optimized to be embedded in web applications. It brings all functionality of ActiveReports to the web that includes such features as:

- Intuitive drag-and-drop functionality to add controls, fields, or report items to the design surface. Intuitive and simple, even the most non-technical user can quickly design reports. The design surface is optimized with page layout, margins, snap-to-grid alignment, and built-in preview to help you create the ultimate WYSIWYG reports.
- Support of all classic and modern data sources, including SQL, JSON, XML, CSV, and more. You can customize the designer to prevent users from creating new datasets, or to allow them access to one group of datasets (but not others).
- Use of predefined datasets for new reports by implementing and then registering the IDataSetsService interface.
- Support of all ActiveReports controls.

- Full support of RPX Section Reports. This includes pre-existing reports created in the Desktop Designer or Visual Studio Integrated Designer.

WebDesigner Application is a powerful JS component, which allows creating reports directly in the Web. However, it requires a server-side component with REST API, similar to the Js Viewer and Blazor Viewer.

These topics provide details on how to use WebDesigner with some popular web frameworks:

ASP.NET MVC Core Integration

Angular Application Integration

React Application Integration

VueJS Application Integration

These topics provide information on the WebDesigner configuration:

WebDesigner ASP.NET Middleware

Load Reports

Configure and Use Shared Data Sources

Save Reports

Preview Reports

Update Security Token in Designer Service

WebDesigner API

WebDesigner ASP.NET Middleware

ASP.NET Core middleware is software to handle HTTP requests and responses. Each middleware component serves a specific purpose, one authenticates a user, and the other handles static files like Javascript or CSS files. These middleware components together setup a request processing pipeline.

The default code developed by the **ASP.NET Core Web App** template sets up the request processing pipeline for the application using set of middlewares - **UseDeveloperExceptionPage()** and **UseStaticFiles()** of the IApplicationBuilder interface. The middlewares are executed in the order in which they are added in the pipeline.

To allow designing reports from a browser, you need to configure ActiveReports WebDesigner in ASP.NET Core middleware. It is done by adding the **UseReportDesigner() ('UseReportDesigner Method' in the on-line documentation)** middleware, which configures middleware for the ActiveReports API and handlers. To use the previewing ability of the JSViewer with the designer, add the **UseReportViewer() ('UseReportViewer Method' in the on-line documentation)** middleware along with the UseReportDesigner() middleware.

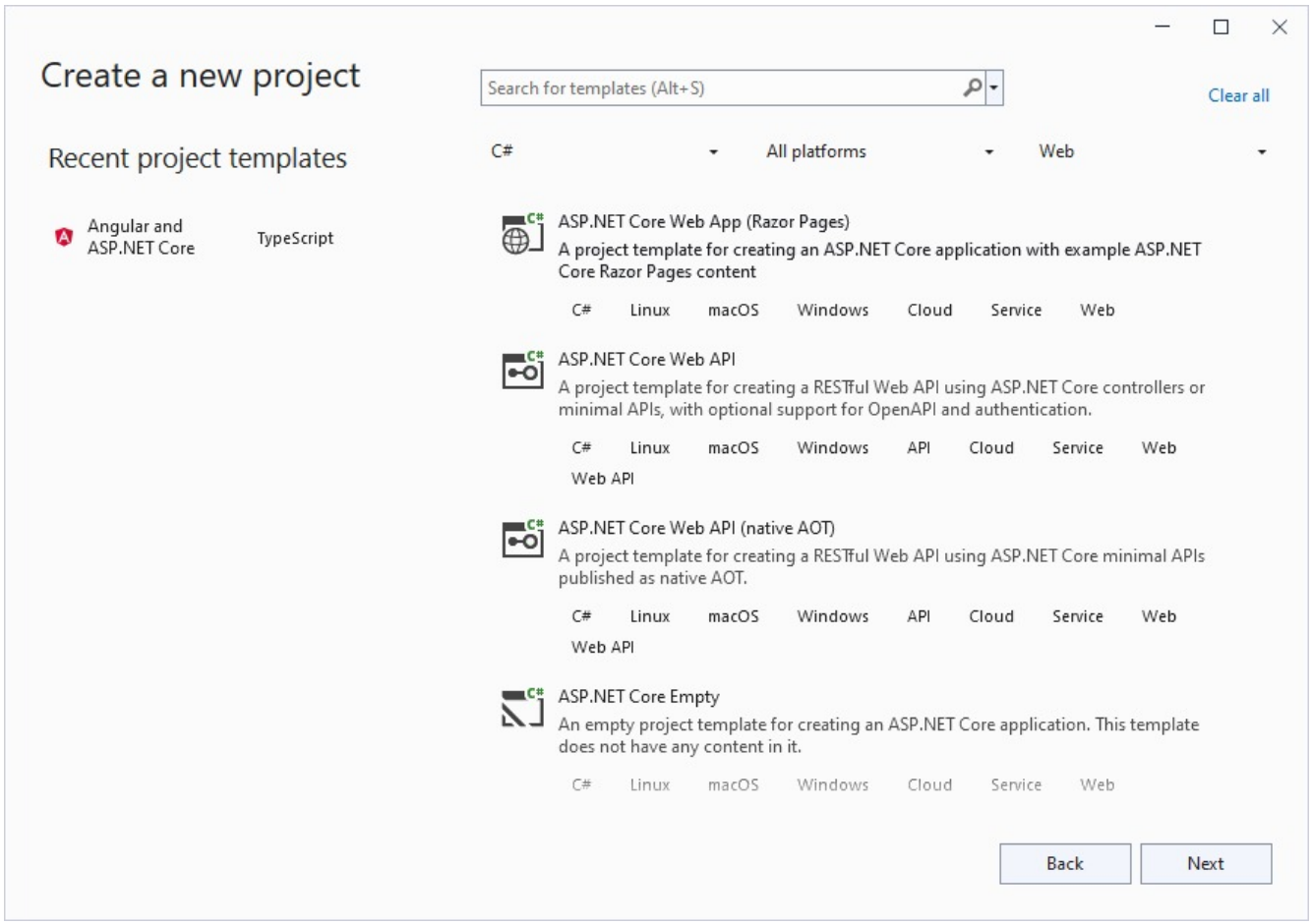
The following are the methods to define middleware settings for WebDesigner available in the **DesignerSettings ('DesignerSettings Class' in the on-line documentation)** class:

- **SetLocateDataSource ('SetLocateDataSource Method' in the on-line documentation)**: Sets data source locator.
- **UseConfig ('UseConfig Method' in the on-line documentation)**: Sets the path to configuration file, ActiveReports.config.
- **UseCustomStore ('UseCustomStore Method' in the on-line documentation)**: Sets the custom resource service to be used.
- **UseDataProviders ('UseDataProviders Method' in the on-line documentation)**: Adds custom data provider.
- **UseFileStore ('UseFileStore Method' in the on-line documentation)**: Use directory as the source of resources.

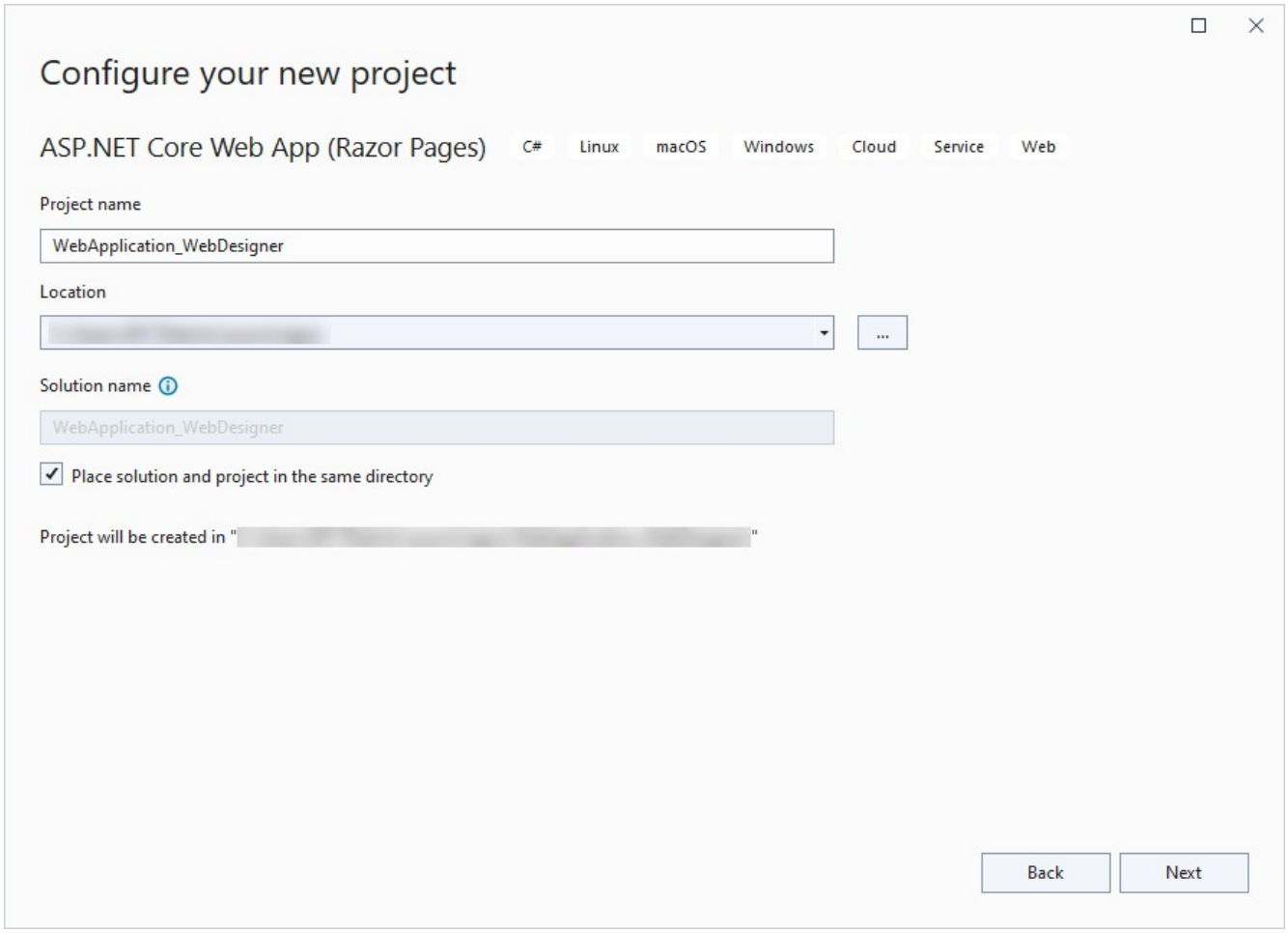
Create an ASP.NET Core Project

The steps to create a WebDesigner sample using ASP.NET MVC Core application are as follows:

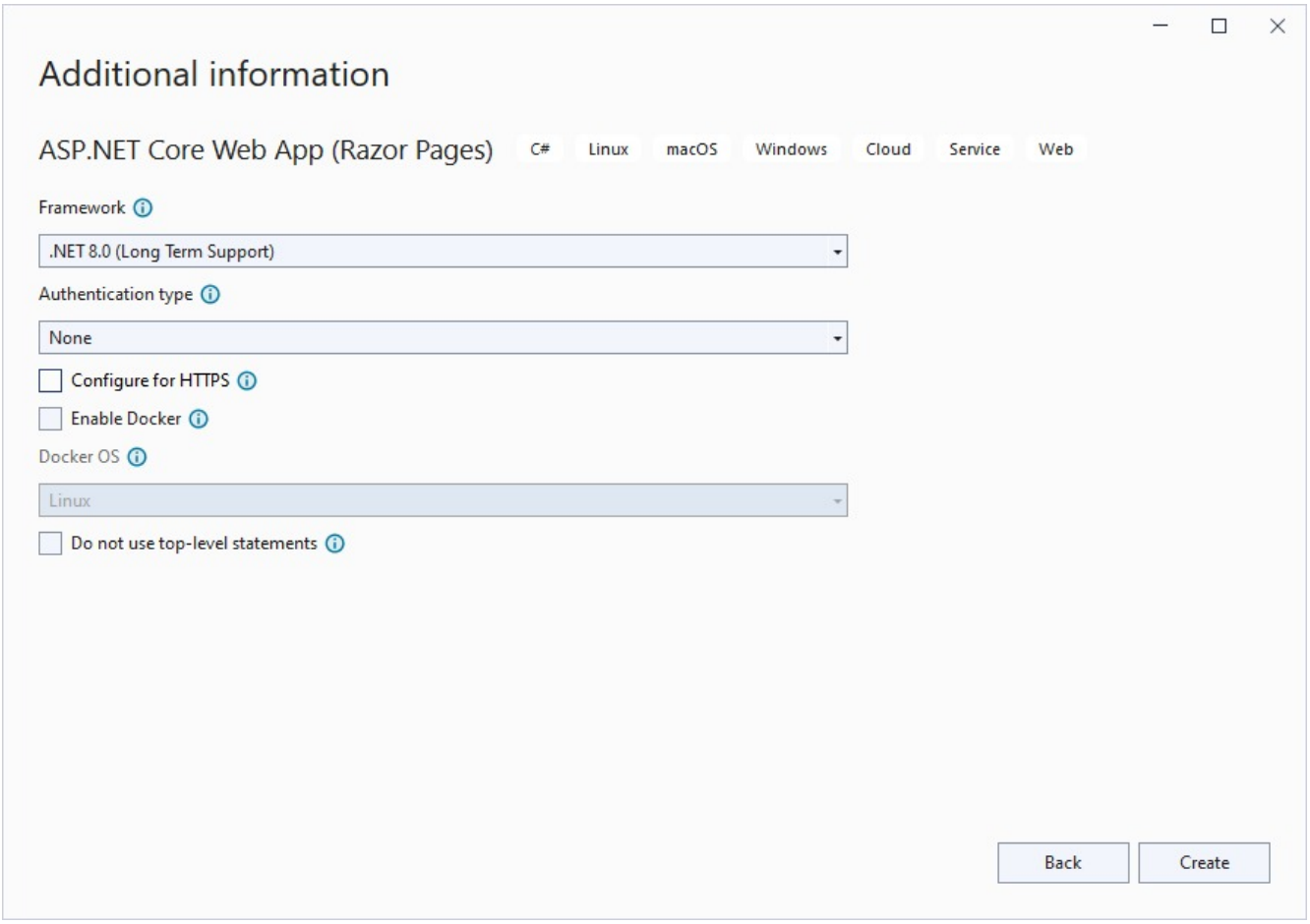
1. Open **Microsoft Visual Studio 2022** and create a new **ASP.NET Core Web App** project which includes example Razor Pages.



2. Type a name for your project and click **Next**.



3. Fill-in the additional info like 'Framework' as .NET 8.0 and click **Create**.



Configure ActiveReports in ASP.NET Core Middleware

4. Right-click the project under the **Solution Explorer** and select **Manage Nuget Packages**.
5. In the window that appears, go to **Browse** and input **Microsoft.AspNetCore.StaticFiles**, select the latest version, and click **Install**.
6. Similarly, browse the following package and install it.

   `MESCIUS.ActiveReports.Aspnetcore.Designer`
7. Create 'resources' folder in your sample project root; you can put your existing reports, themes, and images in this folder.
8. Make sure to set the **Build Action** property of the resources to 'Embedded Resource'.
9. Modify the content for the **Program.cs** file as follows to enable the application to use ActiveReports:

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Designer;

DirectoryInfo ResourcesRootDirectory = new
DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources" +
Path.DirectorySeparatorChar));
var builder = WebApplication.CreateBuilder(args);
```

```
// Add services to the container.
 builder.Services.AddControllersWithViews();
 builder.Services.AddReportDesigner();
var app = builder.Build();

 app.UseHttpsRedirection();
if (!app.Environment.IsDevelopment())
 {
    app.UseDeveloperExceptionPage();
 }
// Configure middleware for ActiveReports API and handlers.
app.UseReportDesigner(config =>
            config.UseFileStore(ResourcesRootDirectory, null,
FileStoreOptions.NestedFoldersLookup));
app.UseDefaultFiles();
app.UseStaticFiles();
app.Run();
```

In case you create the application targeting below .Net 6.0, update the Startup.cs as follows:

Startup.cs

```
using System.IO;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using GrapeCity.ActiveReports.Aspnetcore.Designer;
namespace WebDesignerSample
{
        public class Startup
        {
                // resources (reports, themes, images) location
                private static readonly DirectoryInfo ResourcesRootDirectory = new
DirectoryInfo(".\\resources\\");
                public void ConfigureServices(IServiceCollection services)
                {
                        // web designer services
                        services.AddReportDesigner();
                }
                public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
                {
                        // web designer middleware
                        app.UseReportDesigner(config =>
                                    config.UseFileStore(ResourcesRootDirectory, null,
FileStoreOptions.NestedFoldersLookup));
                        // static files middlewares
                        app.UseDefaultFiles();
                        app.UseStaticFiles();
                }
        }
}
```

ASP.NET MVC Core Integration

Let us create an ASP.NET Core application using ActiveReports and render the reports in the WebDesigner.

1. Follow the steps from step 1 to step 8 in WebDesigner ASP.NET Middleware topic to create an ASP.NET Core Web Application with **ASP.NET Core Web App** template and configure the ActiveReports in ASP.NET Core Middleware.
2. Add the following additional package to the project:

   ```
   MESCIUS.ActiveReports.Aspnetcore.Viewer
   ```
3. To use npm packages, your project must contain a **package.json** file. Open the **Tools** menu > **NuGet Package Manager** > **Package Manager Console** and run the following command in the Package Manager before installing any code dependencies:

   ```
   npm init -y
   ```
4. Add WebDesigner to the application.
   1. Open the **Tools** menu > **NuGet Package Manager** and run the following command in the Package Manager Console to download and install the WebDesigner-related files and folders from NPM

      ```
      npm install @mescius/activereportsnet-designer
      ```

      The designer files/folders will be downloaded in your current directory:
      .\node_modules\@mescius\activereportsnet-designer\dist
   2. Similarly, run the following command in the Package Manager Console, to download and install the Web Viewer-related files and folders from NPM

      ```
      npm install @mescius/activereportsnet-viewer
      ```
5. Copy the following designer and viewer files/folder installed in the **node_modules** to the **wwwroot\js** and **wwwroot\css** folder in the application, respectively.
   - web-designer.css
   - web-designer.js
   - vendor folder
   - jsViewer.min.js
6. In Solution Explorer, right-click 'wwwroot' folder and select **Add** > **New Item**.
7. Select **HTML Page** item type, input **index.html** and click **Add**.
8. In Solution Explorer, find newly-added **index.html** and modify its content as follows:

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>ActiveReports WebDesigner</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <style>
        body, html {
            width: 100%;
            height: 100%;
            margin: 0;
```

```
            padding: 0
        }
        @@keyframes arwd-loader {
            from {
                color: #fff
            }
            to {
                color: #205f78
            }
        }
        .ar-web-designer {
            width: 100%;
            height: 100%
        }
        .ar-web-designer__loader {
            display: flex;
            width: 100%;
            height: 100%;
            background-color: #205f78;
            color: #fff;
            font-size: 18px;
            animation-name: arwd-loader;
            animation-duration: .62s;
            animation-timing-function: ease-in-out;
            animation-iteration-count: infinite;
            animation-direction: alternate;
            justify-content: center;
            align-items: center
        }
    </style>
    <link rel="stylesheet" href="vendor/css/fonts-googleapis.css" type="text/css" />
    <link rel="stylesheet" href="css/jsViewer.min.css" />
    <link rel="stylesheet" href="css/web-designer.css" />
</head>
<body>
    <!-- Required for the ActiveReports Web Viewer -->
    <script src="js/jsViewer.min.js"></script>
    <!-- designer-related js -->
    <script src="js/web-designer.js"></script>
    <!-- Designer root div -->
    <div id="ar-web-designer" class="ar-web-designer">
        <span class="ar-web-designer__loader"><b>AR WebDesigner</b></span>
    </div>
    <script>
       var viewer = null;
       GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
            appBar: {
                openButton: { visible: true },
                saveButton: { visible: true },
                saveAsButton: { visible: true }
            },
```

```
            editor: { showGrid: false },
            data: {
                dataSets: { canModify: true },
                dataSources: { canModify: true }
            },
            preview: {
                openViewer: (options) => {
                    if (viewer) {
                        viewer.openReport(options.documentInfo.id);
                        return;
                    }
                    viewer = GrapeCity.ActiveReports.JSViewer.create({
                        element: '#' + options.element,
                        renderFormat: 'svg',
                        reportService: {
                            url: 'api/reporting',
                        },
                        reportID: options.documentInfo.id,
                        settings: {
                            zoomType: 'FitPage'
                        }
                    });
                }
            }
        });
    </script>
</body>
</html>
```

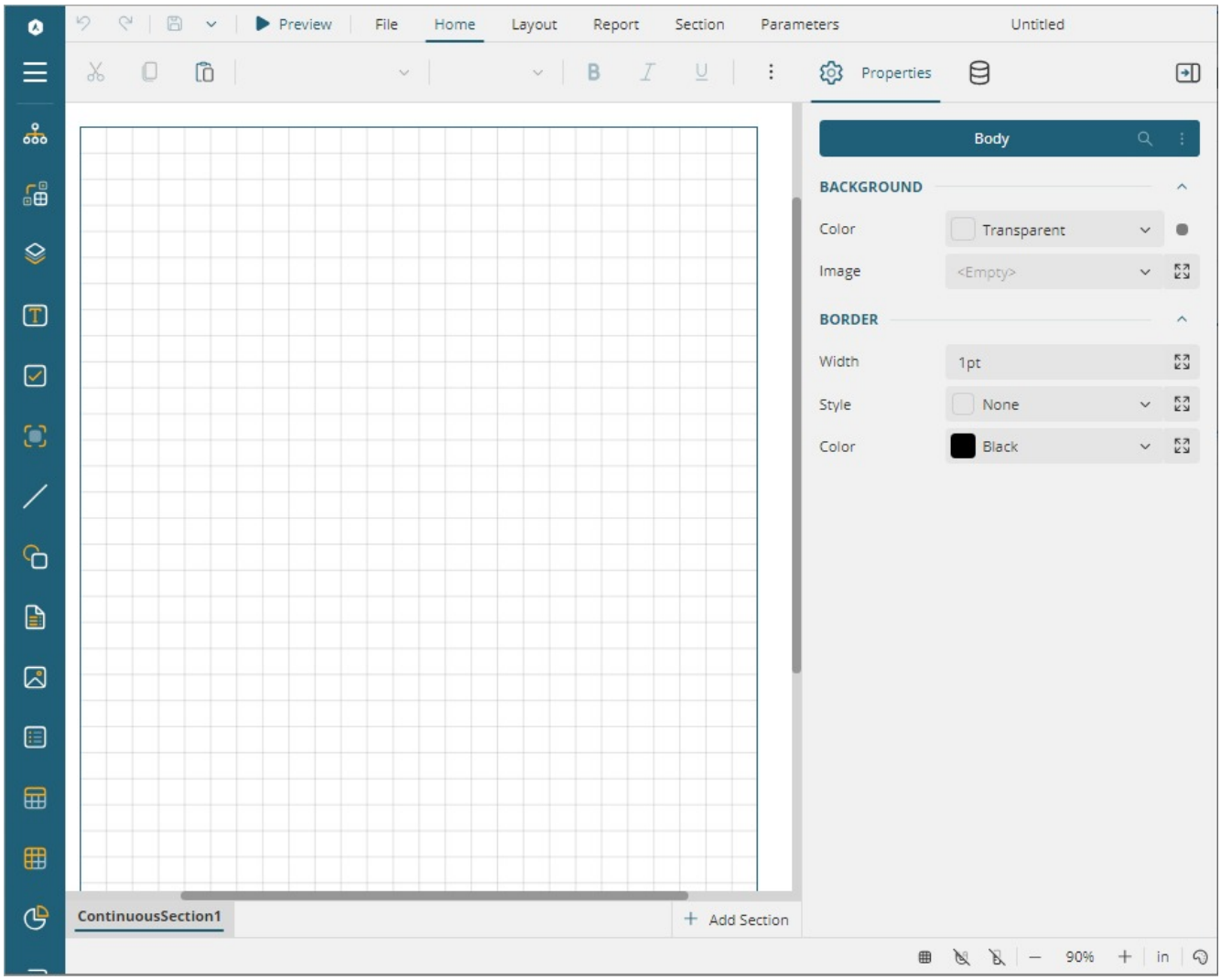
9. Modify the content for the **Program.cs** file as follows to enable the application to use ActiveReports:

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using GrapeCity.ActiveReports.Aspnetcore.Designer;
DirectoryInfo ResourcesRootDirectory = new
DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources" +
Path.DirectorySeparatorChar));
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddReportDesigner();
var app = builder.Build();
app.UseHttpsRedirection();
if (!app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}
// Configure middleware for ActiveReports API and handlers.
app.UseReportDesigner(config => config.UseFileStore(ResourcesRootDirectory));
app.UseReportViewer(config => config.UseFileStore(ResourcesRootDirectory));
app.UseDefaultFiles();
```

```
app.UseStaticFiles();
app.Run();
```

10. Build your solution (Build > Build Solution) and run it. WebDesigner with a blank RDLX report opens in your browser.

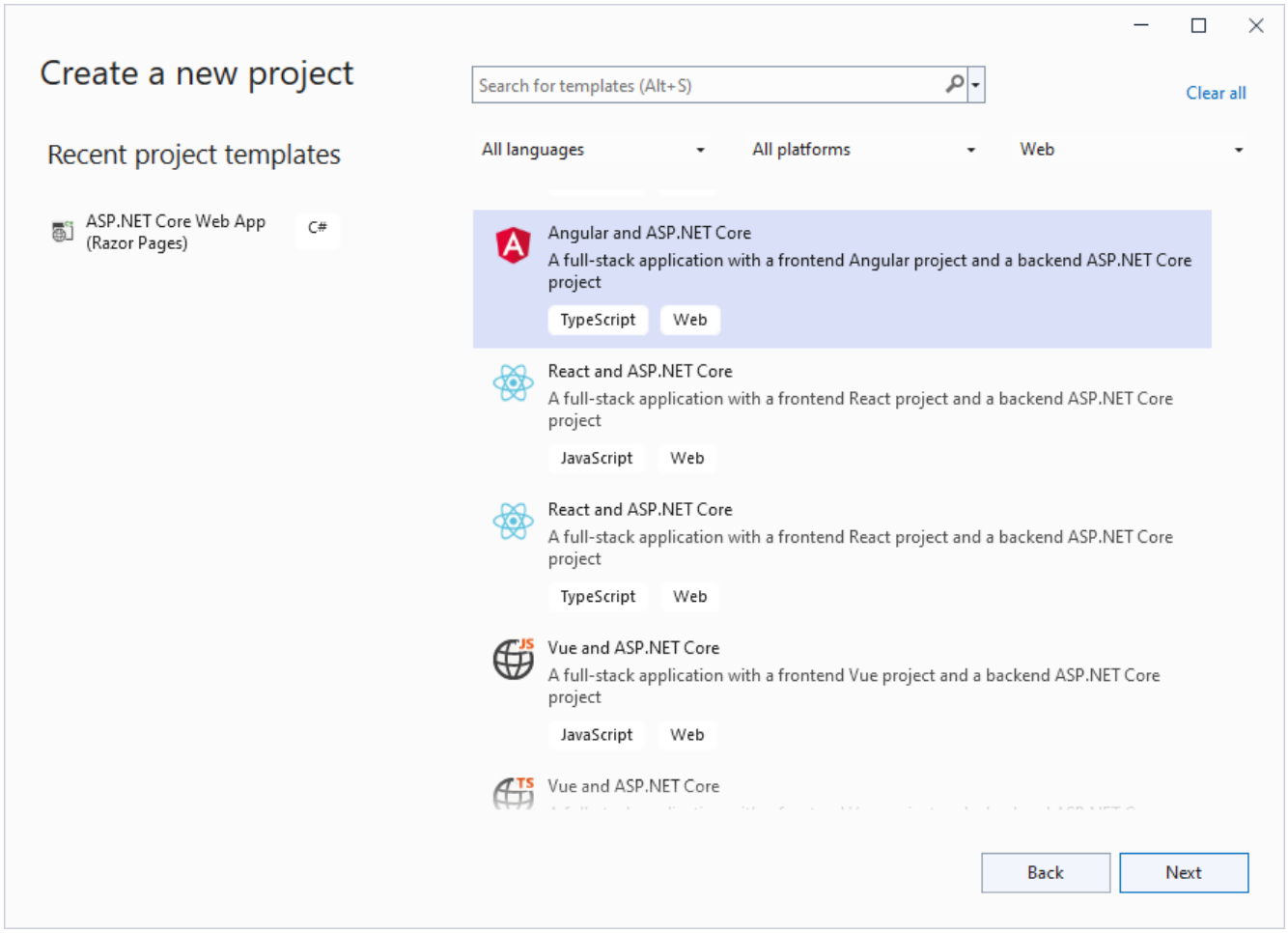


Integration to Angular Application

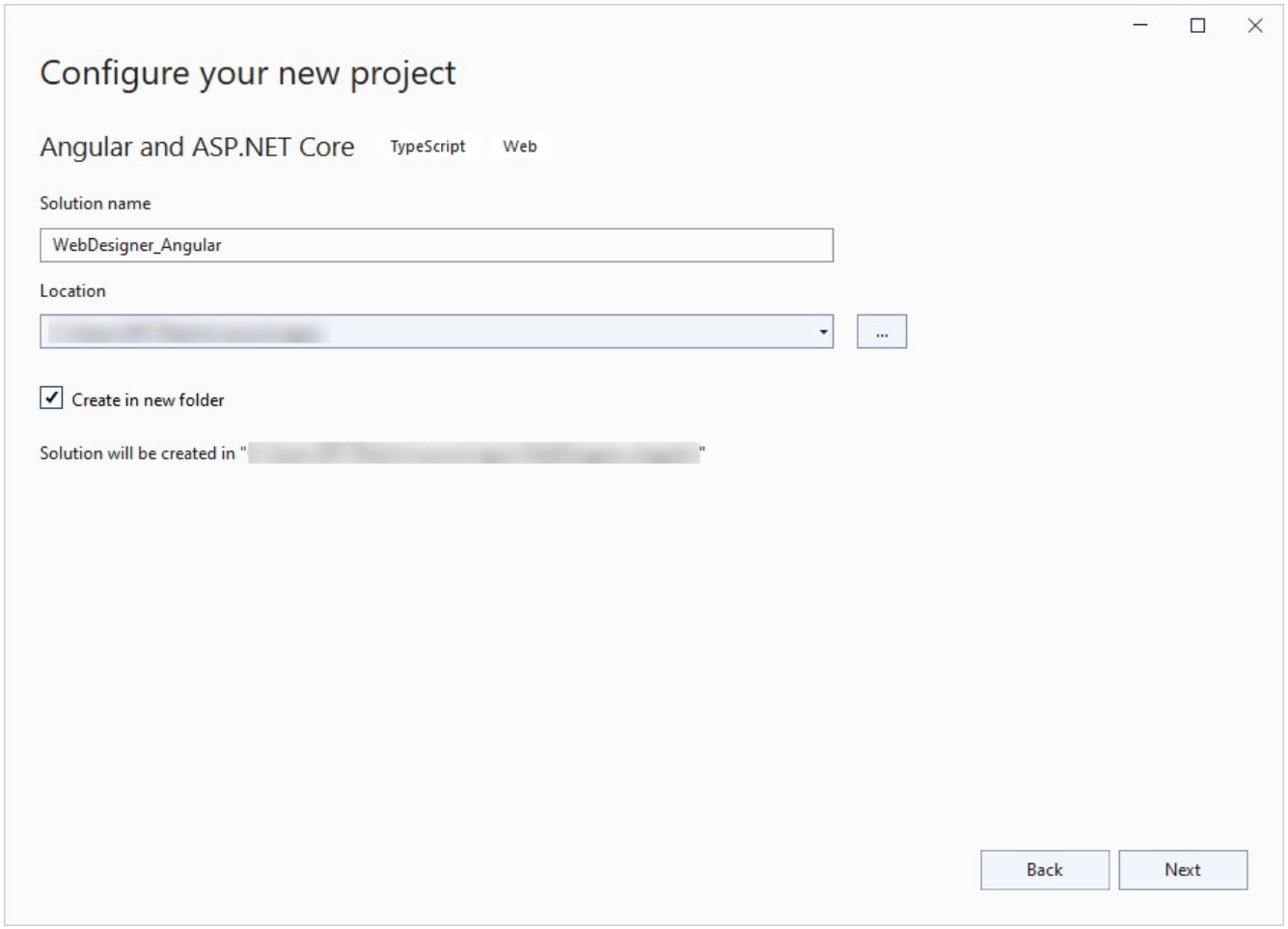
This page explains how you can embed the ActiveReports WebDesigner component in your Angular application. To run the Angular Application Server, you will require the node.js JavaScript runtime and Angular CLI. Use the following command in the terminal or command prompt to install the **Angular CLI**:

```
npm install -g @angular/cli
```

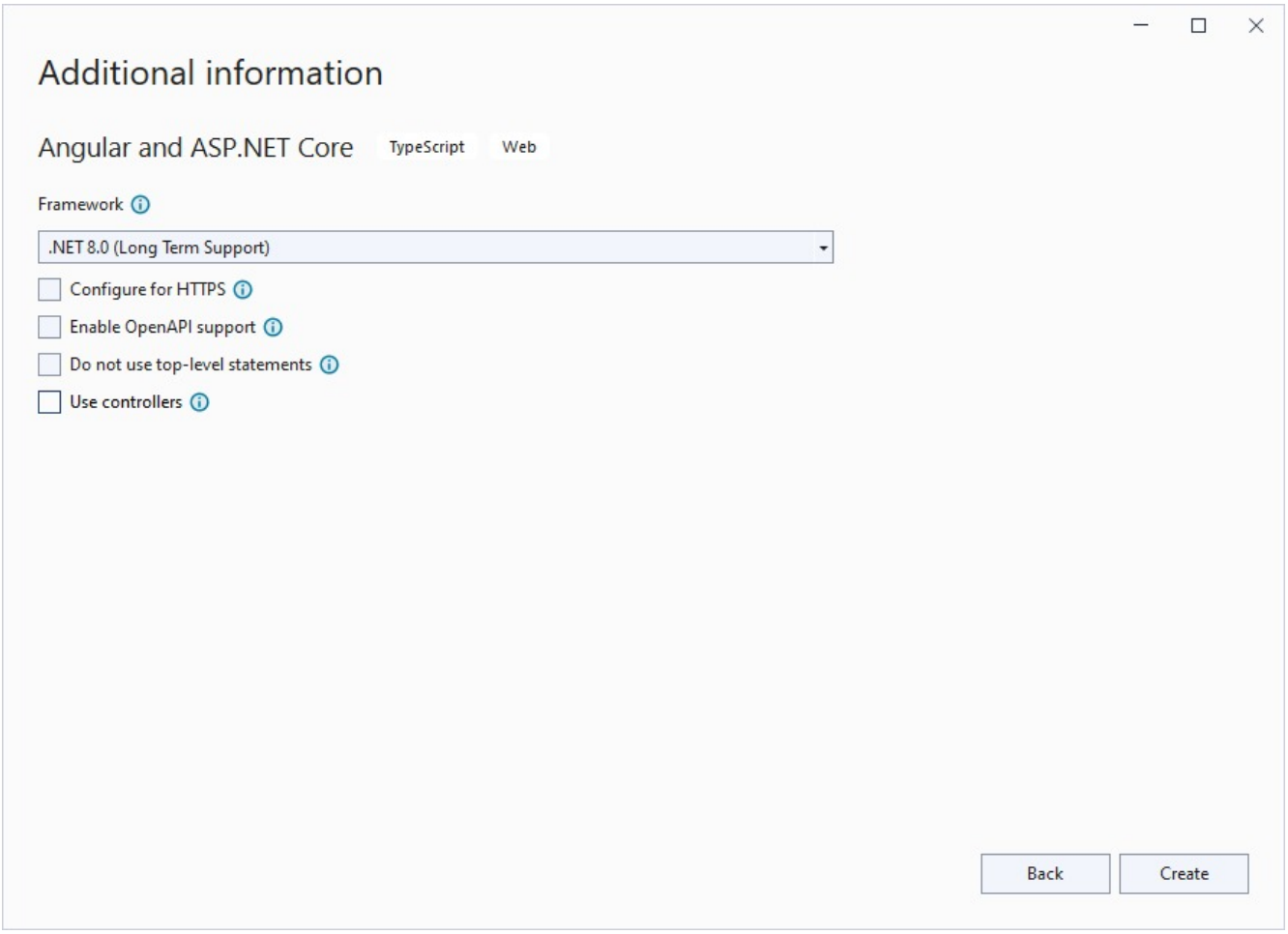
1. Open **Microsoft Visual Studio 2022** and create a new **Angular and ASP.NET Core** project.



2. Type a name for your project and click **Next**.



3. Select the **Framework** to a latest version and uncheck other options.



4. Right-click the '.Server' project in the **Solution Explorer** and select **Manage NuGet Packages**.
5. Add the following packages to the project.

   ```
   MESCIUS.ActiveReports.Aspnetcore.Designer
   MESCIUS.ActiveReports.Aspnetcore.Viewer
   ```

6. Create 'resources' folder in your sample project root; you can put your existing reports, themes, and images in this folder.
7. Make sure to set the **Build** Action property of the resources to 'Embedded Resource'.
8. Open 'Program.cs' file and update the file to include the 'using' statements at the top, and specify the resource folder and add services to container, so that the complete file looks like below.

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Designer;
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using GrapeCity.ActiveReports.Web.Designer;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddReportViewer();
```

```
builder.Services.AddReportDesigner();
var app = builder.Build();
app.UseDefaultFiles();
app.UseStaticFiles();
// Configure the HTTP request pipeline.
app.UseHttpsRedirection();
app.UseAuthorization();
var ResourcesRootDirectory =
    new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"));
app.UseReportDesigner(config => config.UseFileStore(ResourcesRootDirectory, null,
FileStoreOptions.NestedFoldersLookup));

app.MapControllers();
app.MapFallbackToFile("/index.html");
app.Run();
```

9. In the '.client' project, open 'package.json' file and add the following packages under 'dependencies':

   ```
   "@mescius/activereportsnet-designer": "^18.x.x",
   "@mescius/activereportsnet-viewer": "^18.x.x"
   ```

10. Open the '.client' project in the command prompt or terminal window and run the following command to install the npm packages.
    `npm install`

    The designer and viewer files/folders will be downloaded in your current directory: .\node_modules\@mescius\activereportsnet-designer\dist and .\node_modules\@mescius\activereportsnet-viewer\dist.
11. Expand the 'src\app' folder in the '.client' project, open 'app.component.ts' file, and replace the existing code with the following code to initialize the Designer instance.

    app.component.ts

    ```
    import { HttpClient } from '@angular/common/http';
    import { Component, OnInit, ViewEncapsulation } from '@angular/core';
    import { arWebDesigner } from '@mescius/activereportsnet-designer';
    import { JSViewer, createViewer } from '@mescius/activereportsnet-viewer';
    import '@mescius/activereportsnet-designer/dist/web-designer.css';
    import '@mescius/activereportsnet-viewer/dist/jsViewer.min.css';
    @Component({
      selector: 'app-root',
      templateUrl: './app.component.html',
      styleUrl: './app.component.css',
    })
    export class AppComponent implements OnInit {
      forecasts: any;
      private viewer: JSViewer | null = null;
      constructor(private http: HttpClient) { }
      ngOnInit() {
        arWebDesigner.create('#ar-web-designer', {
          rpx: { enabled: true },
          appBar: { openButton: { visible: true } },
          data: { dataSets: { visible: true, canModify: true }, dataSources: { canModify: true }
    ```

```
},
      preview: {
        openViewer: (options: any) => {
          if (this.viewer) {
            this.viewer.openReport(options.documentInfo.id);
            return;
          }
          this.viewer = createViewer({
            element: '#' + options.element,
            reportService: {
              url: 'api/reporting',
            },
            reportID: options.documentInfo.id
          });
        }
      }
    });
  }
  ngOnDestroy() {
    this.viewer?.destroy();
    arWebDesigner.destroy('#ar-web-designer');
  }
  title = 'webdesigner_angular.client';
}
```

12. Open the 'app.component.html' file and replace the existing content with the following markup for hosting the element.

app.component.html

```
<body>
  <div id="ar-web-designer" class="ar-web-designer">
    <span class="ar-web-designer__loader"><b>AR WebDesigner</b></span>
  </div>
</body>
```

13. Open the 'app.component.css' file and modify its content as follows.

app.component.css

```
@keyframes arwd-loader {
  from {
    color: #fff
  }
  to {
    color: #205f78
  }
}
.ar-web-designer {
  width: 100%;
  height: 100%
}
.ar-web-designer__loader {
```

```
  display: flex;
  width: 100%;
  height: 100%;
  background-color: #205f78;
  color: #fff;
  font-size: 18px;
  animation-name: arwd-loader;
  animation-duration: .62s;
  animation-timing-function: ease-in-out;
  animation-iteration-count: infinite;
  animation-direction: alternate;
  justify-content: center;
  align-items: center
}
```

14. Open the '\src\styles.css' file and add the following content.

styles.css

```
body, html {
  width: 100%;
  height: 100%;
  margin: 0;
  padding: 0
}
```

15. Open 'proxy.conf.js' file and update the 'context' section of code as follows.

proxy.conf.js

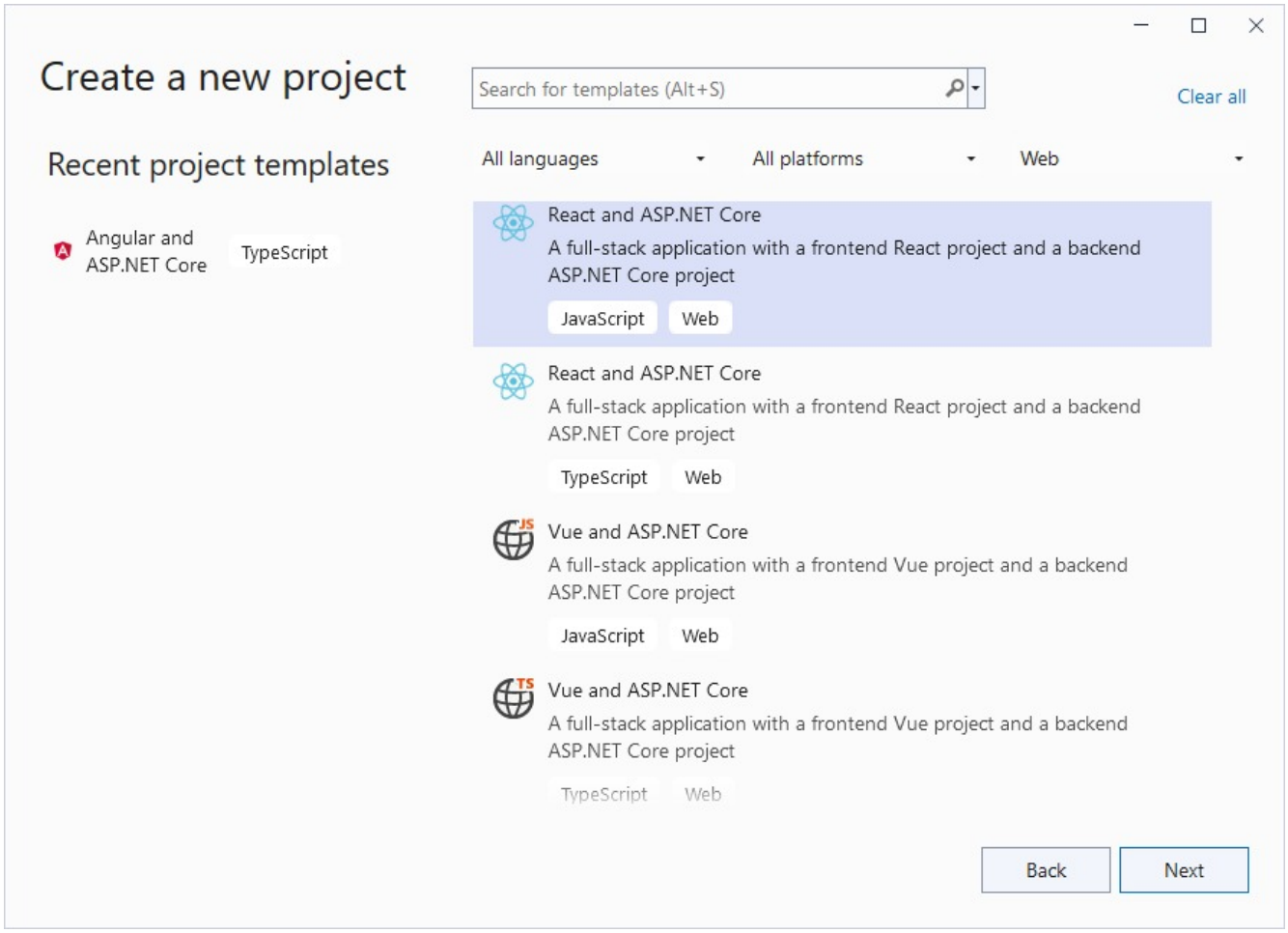
```
context: [
      "/weatherforecast",
      "/api"
   ],
```

16. Press **Ctrl** + **Shift** + **B** to build your application and then press **F5** to run it.
    On running the application, you can find the report placed in the resource folder by navigating to **File** menu > **Open**.

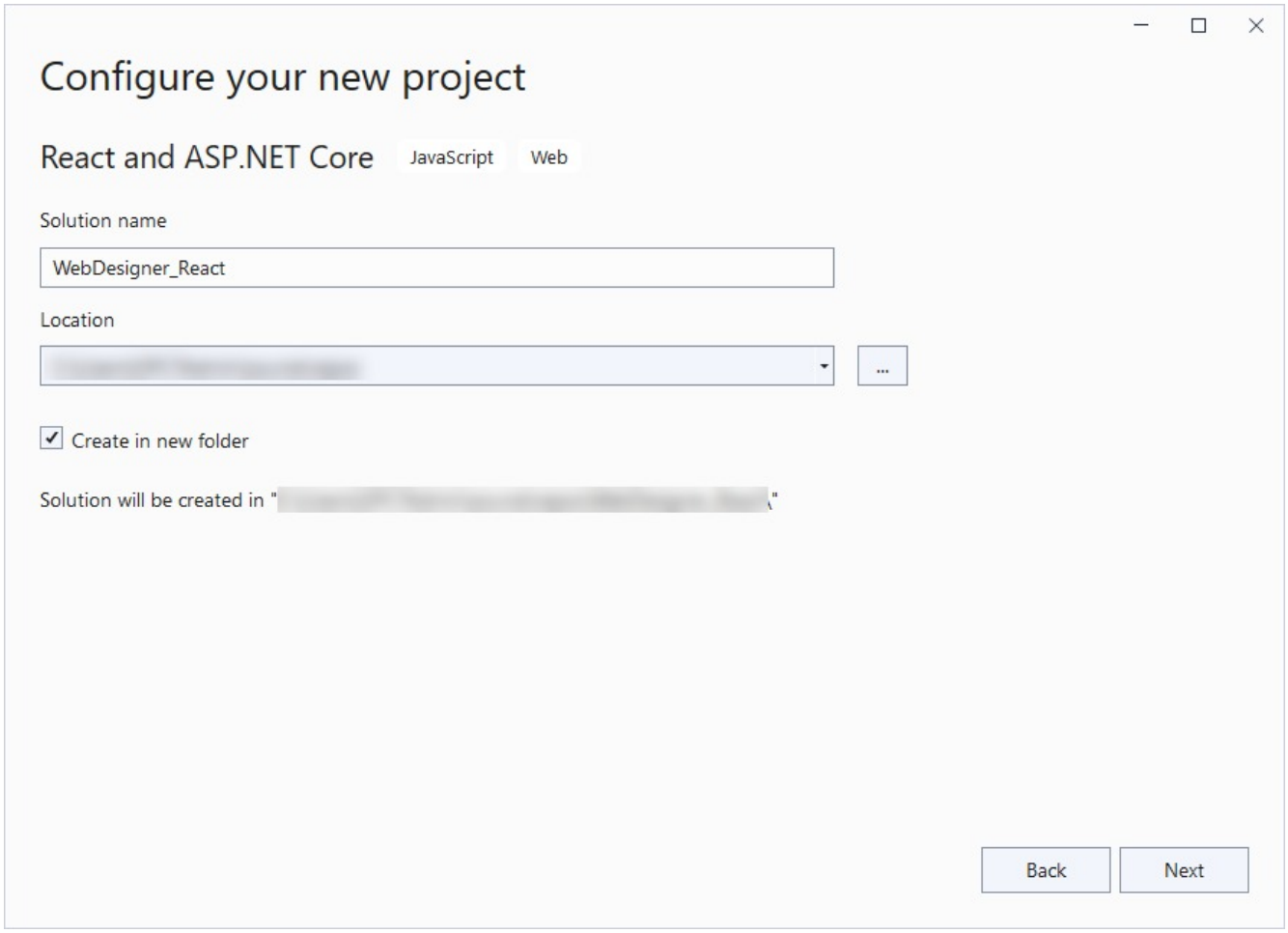
Integration to React Application

This page explains how you can embed the ActiveReports WebDesigner component in your React application (ASP.NET Core). To run the React Application Server, you will require the node.js JavaScript runtime.

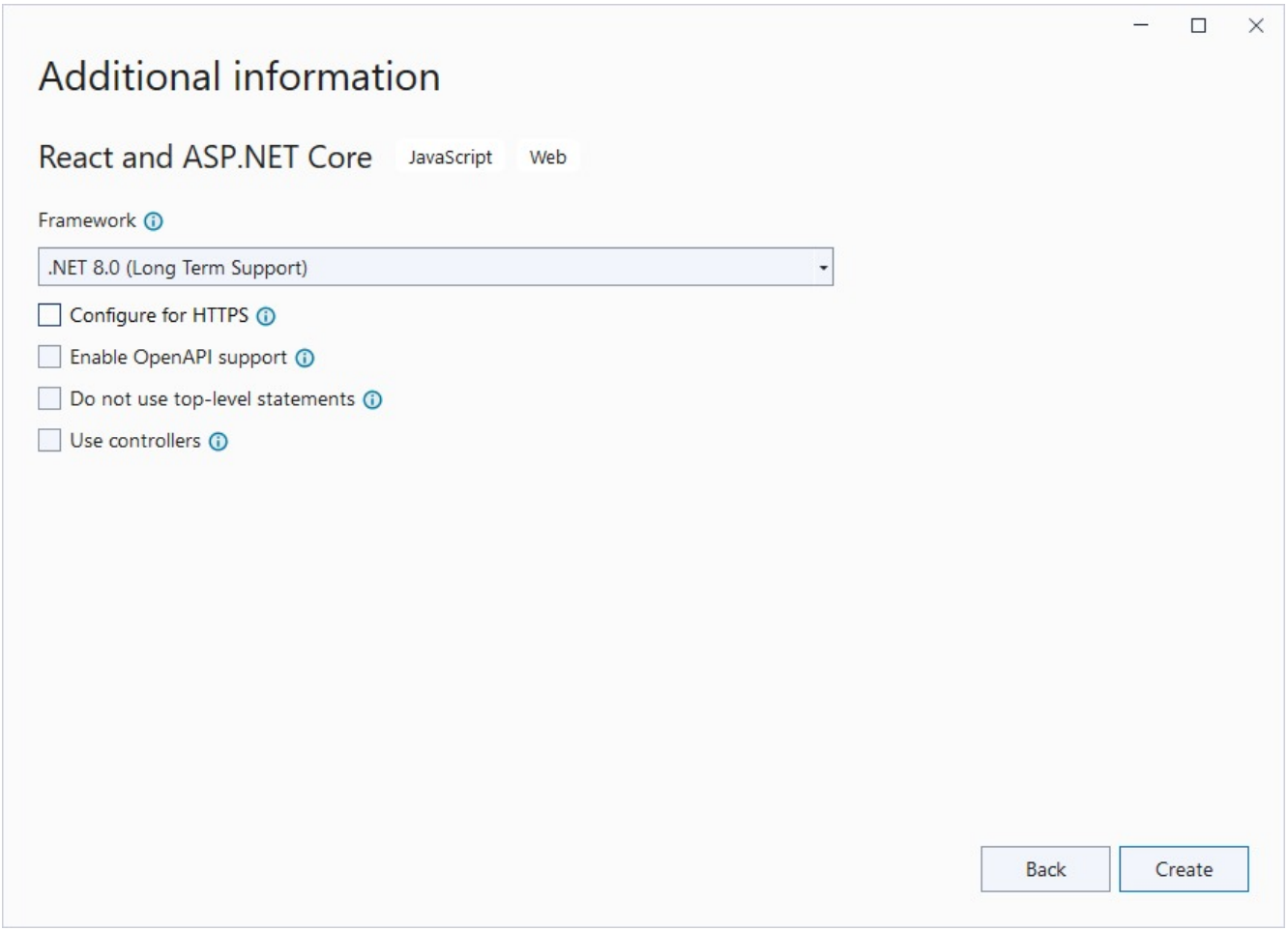
1. Open **Microsoft Visual Studio 2022** and create a new **React and ASP.NET Core** project.



2. Type a name for your project and click **Next**.



3. Select the **Framework** to a latest version and uncheck other options.



4. Right-click the project in the **Solution Explorer** and select **Manage NuGet Packages**.
5. Add the following packages to the project.

   ```
   MESCIUS.ActiveReports.Aspnetcore.Designer
   MESCIUS.ActiveReports.Aspnetcore.Viewer
   ```

6. Create 'resources' folder in your sample project root; you can put your existing reports, themes, and images in this folder.
7. Make sure to set the **Build** Action property of the resources to 'Embedded Resource'.
8. Open 'Program.cs' file and update the file to include the 'using' statements at the top, and specify the resource folder, and add services to container, so that the complete file looks like below.

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Designer;
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using GrapeCity.ActiveReports.Web.Designer;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddReportViewer();
builder.Services.AddReportDesigner();
builder.Services.AddMvc(options => options.EnableEndpointRouting = false);
```

```
var app = builder.Build();
app.UseHttpsRedirection();
if (!app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}
var ResourcesRootDirectory =
    new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"));
app.UseReportViewer(config => config.UseFileStore(ResourcesRootDirectory));
app.UseReportDesigner(config => config.UseFileStore(ResourcesRootDirectory, null,
FileStoreOptions.NestedFoldersLookup));
app.UseDefaultFiles();
app.UseStaticFiles();
app.Run();
```

9. In the '.client' project, open 'package.json' file and add the following packages under 'dependencies':

   ```
   "@mescius/activereportsnet-designer": "^18.x.x",
   "@mescius/activereportsnet-viewer": "^18.x.x"
   ```

10. Open the '.client' project in the command prompt or terminal window and run the following command to install the npm packages.

    ```
    npm install
    ```

    The designer and viewer files/folders will be downloaded in your current directory: .\node_modules\@mescius\activereportsnet-designer\dist and .\node_modules\@mescius\activereportsnet-viewer\dist.

11. Expand the 'src' folder in the '.client' project, and add 'custom.css' to set the designer's host element size to 100%.

    custom.css

    ```
    /* Provide sufficient contrast against white background */
    a {
       color: #0366d6;
     }
    code {
       color: #E01A76;
     }
     .ar-web-designer{
         height:100vh;
         width:100%
     }
     .ar-web-designer__loader {
         display: flex;
         width: 100%;
         height: 100%;
         background-color: #205f78;
         color: #fff;
         font-size: 18px;
         animation-name: arwd-loader;
         animation-duration: .62s;
         animation-timing-function: ease-in-out;
         animation-iteration-count: infinite;
    ```

```
    animation-direction: alternate;
    justify-content: center;
    align-items: center
}
.btn-primary {
    color: #fff;
    background-color: #1b6ec2;
    border-color: #1861ac;
}
```

12. Update the 'root' selector in existing 'App.css' as follows.

App.css

```
#root {
    width: 100%;
}
```

13. Open 'App.jsx' file and replace the existing code with the following code.

App.jsx

```
import { Component } from 'react';
import { arWebDesigner } from '@mescius/activereportsnet-designer';
import { createViewer } from '@mescius/activereportsnet-viewer';
import "@mescius/activereportsnet-designer/dist/web-designer.css";
import "@mescius/activereportsnet-viewer/dist/jsViewer.min.css";
import './custom.css';
import './App.css';
export default class App extends Component {
    constructor() {
        super();
    }
    componentDidMount() {
        console.log("componentDidMount");
            arWebDesigner.create('#ar-web-designer', {
                rpx: { enabled: true },
                appBar: { openButton: { visible: true } },
                data: { dataSets: { visible: true, canModify: true }, dataSources: {
canModify: true } },
                preview: {
                    openViewer: (options) => {
                        if (this.viewer) {
                            this.viewer.openReport(options.documentInfo.id);
                            return;
                        }
                        this.viewer = createViewer({
                            element: '#' + options.element,
                            reportService: {
                                url: 'api/reporting',
                            },
                            reportID: options.documentInfo.id
                        });
                    }
                }
```

```
            })
    }
    componentWillUnmount() {
        console.log("componentWillUnmount");
        this.viewer?.destroy();
        arWebDesigner.destroy('#ar-web-designer');
    }
    render() {
        return (
            <div id="ar-web-designer" className="ar-web-designer"><span className="ar-web-designer__loader"><b>AR WebDesigner</b></span></div>
        );
    }
}
```

14. Open 'vite.config.js' file and update the 'proxy' setting as follows.

vite.config.js

```
proxy: {
    '/api':{
        target: 'http://localhost:5267',
        secure: false
    }
}
```

15. Open 'main.jsx' file and remove `import React from 'react'` statement if using React 17 or higher, and the outer statements `<React.StrictMode>` and `</React.StrictMode>` to disable strict mode. The final main.jsx is as shown.

main.jsx

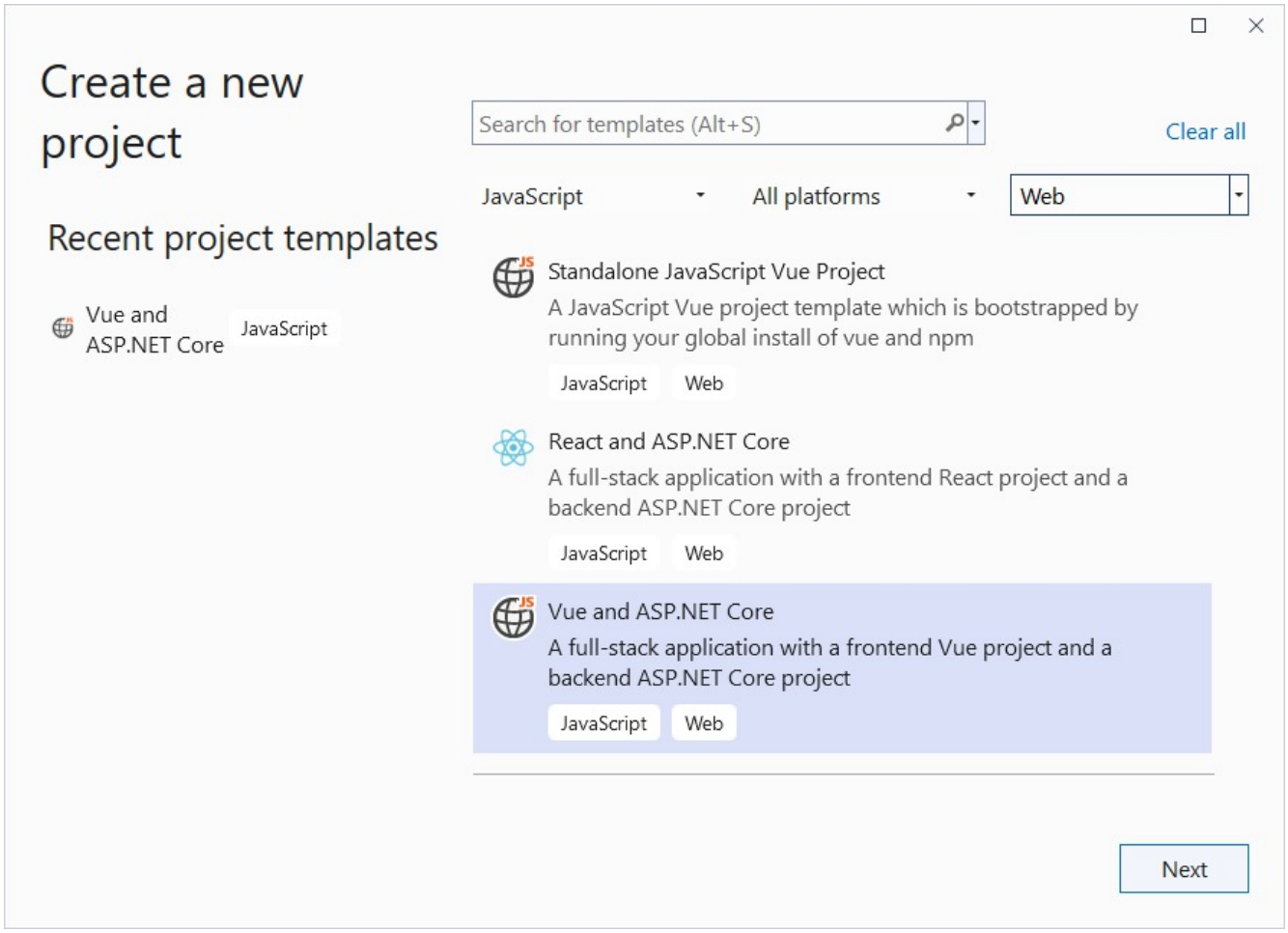
```
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'
ReactDOM.createRoot(document.getElementById('root')).render(
    <App />
)
```

16. Press **Ctrl** + **Shift** + **B** to build your application and then press **F5** to run it.

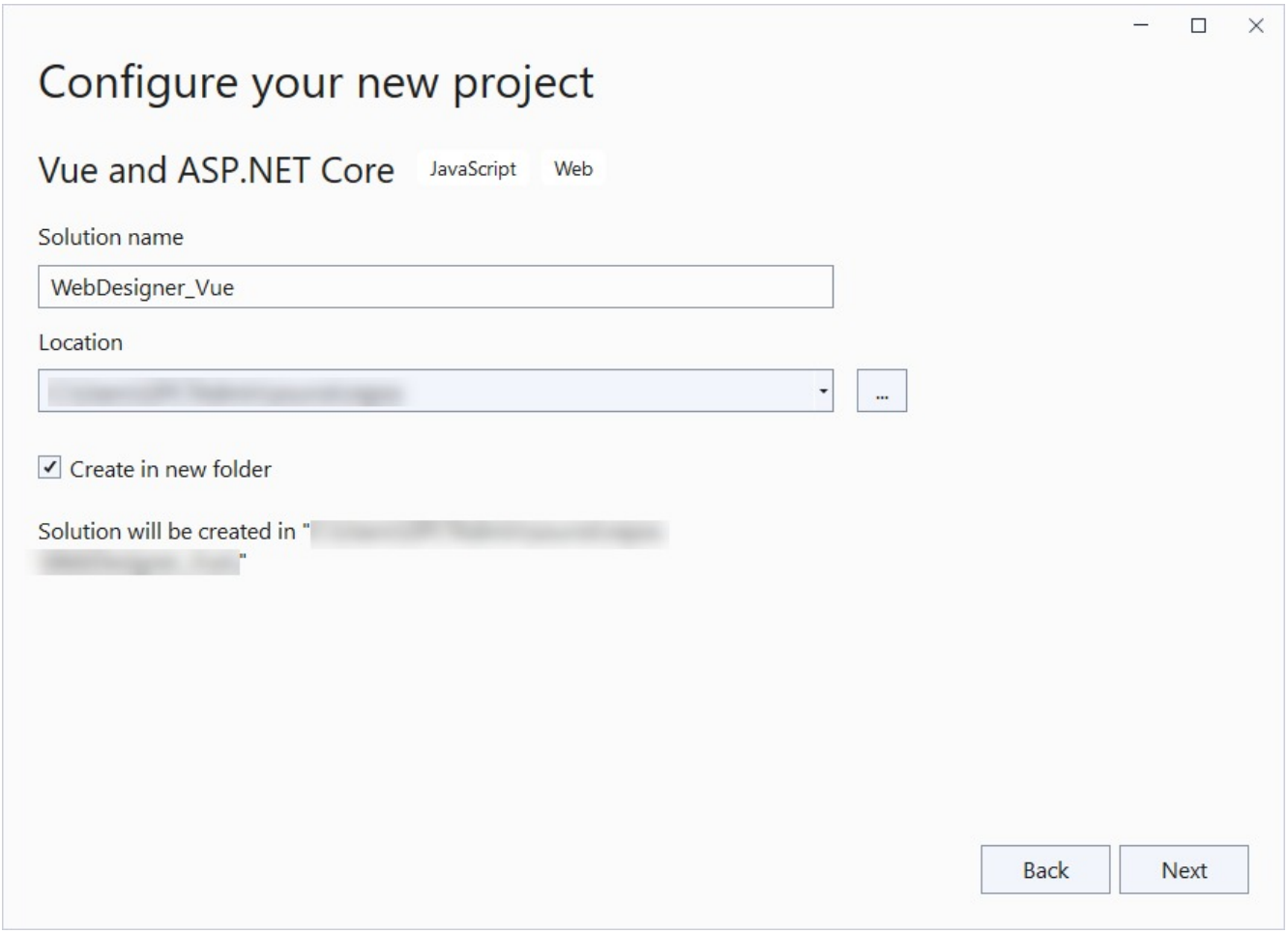
Integration to VueJS Application

This page explains how you can embed the ActiveReports WebDesigner component in your VueJS application (ASP.NET Core). We will use **Vue and ASP.NET Core** with the JavaScript template to create the application.

1. Open **Microsoft Visual Studio 2022** and create a new project, **Vue and ASP.NET Core** with JavaScript.



2. Type a name for your project and click **Create**.



3. In the **Solution Explorer**, right-click the '.Server' project and select **Manage NuGet Packages**.

4. Add the following packages to the project.

   ```
   MESCIUS.ActiveReports.Aspnetcore.Designer
   MESCIUS.ActiveReports.Aspnetcore.Viewer
   ```

5. Add a new folder called 'resources' in the application's root and place the report you want to display in viewer, in this folder. The report you create in WebDesigner, on saving, is saved at this location.

6. Update the 'Program.cs' file as follows:

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Designer;
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using GrapeCity.ActiveReports.Web.Designer;
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddReportViewer();
builder.Services.AddReportDesigner();
builder.Services.AddMvc(options => options.EnableEndpointRouting = false);
var app = builder.Build();
app.UseHttpsRedirection();
```

```
if (!app.Environment.IsDevelopment())
{
    app.UseDeveloperExceptionPage();
}
var ResourcesRootDirectory =
    new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"));
app.UseReportViewer(config => config.UseFileStore(ResourcesRootDirectory));
app.UseReportDesigner(config => config.UseFileStore(ResourcesRootDirectory, null,
FileStoreOptions.NestedFoldersLookup));
app.UseDefaultFiles();
app.UseStaticFiles();
app.Run();
```

7. Open the 'package.json' file and add the following packages for ActiveReports' Viewer and Designer under 'Dependencies':

```
"@mescius/activereportsnet-designer": "^18.x.x",
"@mescius/activereportsnet-viewer": "^18.x.x"
```

8. Add a new 'WebDesigner.vue' file in the **src\components** folder and add the following code.

WebDesigner.vue

```
<template>
    <div id="ar-web-designer"></div>
</template>
<script>
    import { arWebDesigner } from '@mescius/activereportsnet-designer';
    import { createViewer } from '@mescius/activereportsnet-viewer';
    export default {
        mounted() {
            let serverUrl = 'https://localhost:7226';
            arWebDesigner.create('#ar-web-designer', {
                rpx: { enabled: true },
                appBar: { openButton: { visible: true } },
                editor: { showGrid: false },
                data: { dataSets: { visible: true, canModify: true }, dataSources: {
canModify: true } },
                server: {
                    url: serverUrl + '/api'
                },
                preview: {
                    openViewer: (options) => {
                        if (this.viewer) {
                            this.viewer.openReport(options.documentInfo.id);
                            return;
                        }
                        this.viewer = createViewer({
                            element: '#' + options.element,
                            renderFormat: 'svg',
                            reportService: {
                                url: serverUrl + '/api/reporting',
```

```
                        },
                        reportID: options.documentInfo.id
                    });
                }
            }
        });
    }
}
</script>
<style>
    #ar-web-designer {
        height: 100vh;
        float: right;
        width: 100%;
    }
</style>
```

9. Open 'App.vue' inside the **src** folder and replace its default content with the following code.

App.vue

```
<template>
    <div class="main">
        <WebDesigner />
    </div>
</template>
<script>
    import "@mescius/activereportsnet-viewer/dist/jsViewer.min.css";
    import "@mescius/activereportsnet-designer/dist/web-designer.css";
    import WebDesigner from './components/WebDesigner.vue';
    export default {
        name: 'app',
        components: {
            WebDesigner
        },
        methods: {
        }
    }
</script>
<style>
    .main {
        width: 100%;
        overflow-x: hidden
    }
</style>
```

10. To disable browser launch, set the 'launchBrowser' to 'false' in 'launchSettings.json' (in .Server project\Properties).

11. Press **Ctrl** + **Shift** + **B** to build your project and then press **F5** to run it.

Load Reports

By default, the instance of the WebDesigner component displays the blank RDLX report.

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    document: {
        id: 'RPX/Invoice.rpx',
        type: { platform: 'rpx', type: 'report' },
    }
```

To open a report kept on your machine, you need to enable the 'Open' button as shown below

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     appBar: {
         openButton: { visible: true }
}
```

Configure and Use Shared Data Sources

The shared data sources contain connection properties that allow binding multiple reports to same data. The shared data sources can be created and edited only in Standalone Designer or VS Integrated Designer. In WebDesigner, however, you can only reference an existing data source.

Following are the scenarios for using shared data sources in web designer:

- a developer wants to prevent the connection string information from being displayed on the client-side
- a developer wants users to be able to select a pre-defined data sources/data sets

The shared data source solves this problem because the report definition contains only the reference to the data source definition, which is resolved on the server-side when a report is previewed.

> **Note**: Shared data sources are disabled by default.

You must create a shared data source (.rdsx) in Standalone Designer or Visual Studio Integrated Designer. See Work with Local Shared Data Sources for more information.

The steps to use shared data sources are elaborated below. You should first create a WebDesigner-integrated ASP.NET MVC Core application. See the following pages for the information on:

- WebDesigner ASP.NET Middleware
- ASP.NET MVC Core Integration

We will be adding shared data source functionality in an already available sample: WebDesigner_MVC_Core.

1. Open the WebDesigner_MVC_Core application.
2. Place the shared reference, a .rdsx file, in the 'resources' folder of the project.
3. In the script.js where web designer is initialized, use shared property to enable shared data sources. The complete script.js code is as shown.

   script.js

   ```
   import { arWebDesigner } from './web-designer.js';
   import { createViewer } from './jsViewer.min.js';
   ```

```
let viewer = null;
let serverUrl = getServerUrl();
function getServerUrl() {
    let baseUrl = 'api';
    let virtualDirName = document.getElementById('virtualDir');
    if (virtualDirName && virtualDirName.href != window.location.origin + '/') {
        return virtualDirName.href + baseUrl;
    }
    return baseUrl;
}
arWebDesigner.create('#ar-web-designer', {
    server: {
        url: serverUrl
    },
    appBar: { openButton: { visible: true } },
    data: { dataSets: { canModify: true }, dataSources: { canModify: true, shared: { enabled:
true } } } },
    preview: {
        openViewer: (options) => {
            if (viewer) {
                viewer.theme = options.theme;
                viewer.openReport(options.documentInfo.id);
                return;
            }
            viewer = createViewer({
                element: '#' + options.element,
                reportService: {
                    url: 'api/reporting',
                },
                reportID: options.documentInfo.id,
                settings: {
                    zoomType: 'FitPage',
                },
                theme: options.theme
            });
        }
    }
});
```

Implement the IReportStore

4. Create 'Implementation' folder.

5. To the 'Implementation' folder, add ‘ReportStore.cs’ class which will contain implementation for **IReportStore ('IReportStore Interface' in the on-line documentation)**.

ReportStore.cs

```
using System;
using System.Collections.Generic;
using System.IO;
```

```
using System.Linq;
using GrapeCity.ActiveReports.Rendering.Tools;
using GrapeCity.ActiveReports.Web.Designer;
using GrapeCity.ActiveReports.Web.Viewer;
namespace WebDesigner_MVC_Core.Implementation
{
    public class ReportStore : IReportStore
    {
        private static readonly string[] ReportExtensions =
        {
        ".rdl",
        ".rdlx",
        ".rdlx-master",
        ".rpx"
    };
        private readonly Dictionary<string, byte[]> _tempStorage = new Dictionary<string,
byte[]>();
        private readonly DirectoryInfo _rootDirectory;
        public ReportStore(DirectoryInfo rootDirectory)
        {
            _rootDirectory = rootDirectory;
        }
        public ReportDescriptor GetReportDescriptor(string reportId)
        {
            if (_tempStorage.ContainsKey(reportId))
                return new
ReportDescriptor(GetReportTypeByExtension(Path.GetExtension(reportId)));
            var fileInfo = new FileInfo(Path.Combine(_rootDirectory.FullName, reportId));
            return new ReportDescriptor(GetReportTypeByExtension(fileInfo.Extension));
        }
        public Stream LoadReport(string reportId)
        {
            if (_tempStorage.TryGetValue(reportId, out var tempReport))
                return new MemoryStream(tempReport);
            var file = new FileInfo(Path.Combine(_rootDirectory.FullName, reportId));
            return file.OpenRead();
        }
        public string SaveReport(ReportType reportType, string reportId, Stream reportData,
SaveSettings settings = SaveSettings.None)
        {
            if ((settings & SaveSettings.IsTemporary) != 0)
            {
                var tempName = Guid.NewGuid() + GetReportExtension(reportType);
                _tempStorage.Add(tempName, reportData.ToArray());
                return tempName;
            }
            var reportFullPath = Path.Combine(_rootDirectory.FullName, reportId);
            using var fileStream = new FileStream(reportFullPath, FileMode.Create,
FileAccess.Write);
            reportData.CopyTo(fileStream);
            return reportId;
```

```
        }
        public string UpdateReport(ReportType reportType, string reportId, Stream reportData)
        {
            return SaveReport(reportType, reportId, reportData);
        }
        public ReportInfo[] ListReports()
        {
            var reports = _rootDirectory
                .EnumerateFiles("*.*")
                .Where(fileInfo => ReportExtensions.Any(ext =>
                    fileInfo.Extension.EndsWith(ext,
StringComparison.InvariantCultureIgnoreCase)))
                .Select(fileInfo => new ReportInfo()
                {
                    Id = fileInfo.Name,
                    Name = fileInfo.Name,
                    ReportType = GetReportTypeByExtension(fileInfo.Extension),
                }).ToArray();
            return reports;
        }
        private static ReportType GetReportTypeByExtension(string extension)
        {
            switch (extension)
            {
                case ".rdl":
                case ".rdlx":
                    return ReportType.RdlXml;
                case ".rdlx-master":
                    return ReportType.RdlMasterXml;
                case ".rpx":
                    return ReportType.RpxXml;
                default:
                    throw new ArgumentOutOfRangeException(nameof(extension), extension, null);
            }
        }
        private static string GetReportExtension(ReportType type)
        {
            return type switch
            {
                ReportType.RdlXml => ".rdlx",
                ReportType.RdlMasterXml => ".rdlx-master",
                ReportType.RpxXml => ".rpx",
                _ => throw new ArgumentOutOfRangeException(nameof(type), type, null)
            };
        }
        public void DeleteReport(string reportId)
        {
            if (_tempStorage.ContainsKey(reportId))
            {
                _tempStorage.Remove(reportId);
```

```
                return;
            }
            var file = new FileInfo(Path.Combine(_rootDirectory.FullName, reportId));
            if (file.Exists)
                file.Delete(); ;
        }
    }
}
```

Implement the IResourcesService

6. Add 'ResourceService.cs' to the 'Implementation' folder to add implementation for **IResourcesService ('IResourcesService Interface' in the on-line documentation)**.

ResourceService.cs

```
using System.IO;
using System.Linq;
using GrapeCity.ActiveReports;
using GrapeCity.ActiveReports.Rendering.Tools;
using GrapeCity.ActiveReports.Web.Designer;
namespace WebDesigner_MVC_Core.Implementation
{
    public class ResourceProvider : IResourceRepositoryProvider
    {
        private const string SharedDataSourceExtension = ".rdsx";
        private readonly DirectoryInfo _rootDirectory;
        public ResourceProvider(DirectoryInfo rootDirectory)
        {
            _rootDirectory = rootDirectory;
        }
        public Stream GetResource(ResourceInfo resource)
        {
            string absolutePath = Path.Combine(_rootDirectory.FullName, resource.Name);
            var file = new FileInfo(absolutePath);
            if (!file.Exists)
                return null;
            return file.OpenRead();
        }
        public ResourceDescriptor[] ListResources(ResourceType resourceType)
        {
            if (resourceType == ResourceType.SharedDataSource)
            {
                var sharedDataSources = _rootDirectory
                    .EnumerateFiles("*" + SharedDataSourceExtension).Select(fileInfo =>
                    {
                        using var stream = fileInfo.OpenRead();
                        var dataSource = DataSourceTools.LoadSharedDataSource(stream);
                        return new SharedDataSourceResourceDescriptor()
                        {
                            Id = fileInfo.Name,
```

```
                            Name = fileInfo.Name,
                            Type = dataSource.ConnectionProperties.DataProvider
                        };
                    }).ToArray();
                return sharedDataSources;
            }
            return Enumerable.Empty<ResourceDescriptor>().ToArray();
        }
        public ResourceDescriptor[] DescribeResources(ResourceInfo[] resources)
        {
            return Enumerable.Empty<ResourceDescriptor>().ToArray();
        }
    }
}
```

Configure and register services

7. Open Startup.cs and update the file as shown below. The Startup.cs file does the following:

    i. configures the services and middleware used by the application
    ii. registers the '**IReportStore ('IReportStore Interface' in the on-line documentation)**' and '**IResourceRepositoryProvider ('IResourceRepositoryProvider Interface' in the on-line documentation)**' as singleton services
    iii. adds reporting and designer services
    iv. sets the path to the ActiveReports.config file where the SQLite provider is added
    v. configures the reporting and designer middleware
    vi. serves static files

Startup.cs

```
using System.IO;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using GrapeCity.ActiveReports.Aspnetcore.Designer;
using System.Text;
using GrapeCity.ActiveReports.Web.Designer;
using WebDesigner_MVC_Core.Implementation;
using System;
namespace WebDesignerMvcCore
{
    public class Startup
    {
        private static readonly DirectoryInfo ResourcesRootDirectory =
            new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"
+ Path.DirectorySeparatorChar));
        public Startup(IConfiguration configuration)
        {
```

```
            Configuration = configuration;
        }
        public IConfiguration Configuration { get; }
        // This method gets called by the runtime. Use this method to add services to
the container.
        public void ConfigureServices(IServiceCollection services)
        {
            Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);
            services
                .AddReportViewer()
                .AddReportDesigner()
                .AddSingleton<IReportStore>(new ReportStore(ResourcesRootDirectory))
                .AddSingleton<IResourceRepositoryProvider>(new
ResourceProvider(ResourcesRootDirectory))
                .AddMvc(options => options.EnableEndpointRouting = false)
                .AddJsonOptions(options =>
options.JsonSerializerOptions.PropertyNamingPolicy = null);
        }
        // This method gets called by the runtime. Use this method to configure the HTTP
request pipeline.

            public void Configure(IApplicationBuilder app,
            IWebHostEnvironment env,
            IReportStore reportStore,
            IResourceRepositoryProvider resourceProvider
            )
        {
            if (env.IsDevelopment())
            {
                app.UseDeveloperExceptionPage();
            }
            var pathToConfig = Path.Combine(Environment.CurrentDirectory,
"ActiveReports.config");
            app.UseReportDesigner(config =>
            {
                config.UseReportsProvider(reportStore);
                config.UseResourcesProvider(resourceProvider);
                config.UseConfig(pathToConfig);
            });
            app.UseFileServer();
            app.UseMvc();
        }
    }
}
```

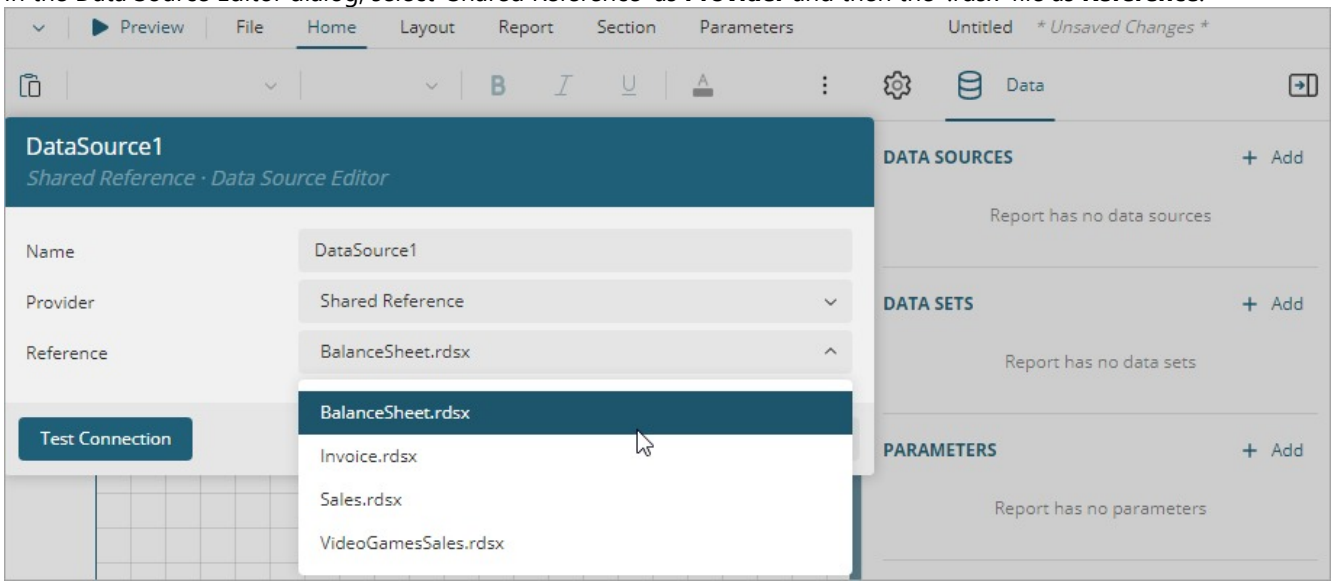
8. Add 'ActiveReports.config' with following content.

ActiveReports.config

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
```

```
    <Extensions>
        <Data>
            <Extension Name="SQLITE" Type="System.Data.SQLite.SQLiteFactory,
System.Data.SQLite"
                       DisplayName="Sqlite Provider" />
        </Data>
    </Extensions>
</Configuration>
```

9. Run the application.
10. Go to the **Data** tab to add the data source.
11. In the Data Source Editor dialog, select 'Shared Reference' as **Provider** and then the '.rdsx' file as **Reference**.



Save Reports

The WebDesigner component contains the **Save** and **Save As** buttons on the toolbar. However, they are hidden by default.

To enable the **Save** and **SaveAs** buttons of Web Designer's toolbar, you should use the code as shown in the example below:

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {

            appBar: {

                saveButton: { visible: true },

                saveAsButton: { visible: true }

            }

        });
```

The process of saving the report is performed by the **UseReportDesigner()** middleware:

Startup.cs

```
app.UseReportDesigner(config => config.UseFileStore(ResourcesRootDirectory, false));
```

The report is saved in the **Resources** folder of the project.

The saving process is performed on the server side and you can customize it with the custom store. See the WebDesigner_CustomStore sample for details.

Preview Reports

You need to configure JS Viewer for preview since preview is part of JS Viewer and not WebDesigner. See JS Viewer Applications for more details.

It is suggested to use our JS Viewer in SVG mode; although this makes previewing slower, you see almost WYSIWYG preview. To configure JS Viewer for preview, use the following code.

```
var viewer = null;
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    preview: {
                openViewer: (options) => {
                    if (viewer) {
                        viewer.openReport(options.documentInfo.id);
                        return;
                    }
                    viewer = GrapeCity.ActiveReports.JSViewer.create({
                        element: '#' + options.element,
                        renderFormat: 'svg',
                        reportService: {
                            url: 'api/reporting',
                        },
                        reportID: options.documentInfo.id,
                        settings: {
                            zoomType: 'FitPage'
                        }
                    });
                }
            }
       });
```

Update Security Token in Designer Service

A site with the WebDesigner application may require to specify tokens. In this case, you should use the code as demonstrated below:

Example Title

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    server: {
        onBeforeRequest: (init) => alert(init.credentials),
    }
```

WebDesigner API

ActiveReports provides a rich API for integrating the web designer components into your web application. Use this API to embed the WebDesigner component in your project. It lets you create, design, and save reports with added capabilities that include - defining the locale for the designer, customizing the default settings of the report items, managing the Data and Properties tab, modifying the application info, and much more. You will find the usage examples for ES, UMD, and TypeScript modules.

export const arWebDesigner

The main object exported by **WebDesigner ES** Module is described below.

create()

Description: Renders the WebDesigner component to the ** ** element with given **selector** using the specified **DesignerSettings** object.

Parameter (Type):

selector: string
settings: **DesignerSettings**

Return Type: Promise<**DesignerAPI**>

Required: Yes

Sample Code

```
import { arWebDesigner } from './web-designer.js';
import { createViewer } from './jsViewer.min.js';
let viewer = null;
arWebDesigner.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } },
    data: { dataSets: { canModify: true }, dataSources: { canModify: true } },
    preview: {
        openViewer: (options) => {
            if (viewer) {
                viewer.openReport(options.documentInfo.id);
                return;
            }
            viewer = createViewer({
                element: '#' + options.element,
                reportService: {
                    url: 'api/reporting',
                },
                reportID: options.documentInfo.id,
                settings: {
                    zoomType: 'FitPage'
                }
            });
        }
    }
});
```

apiof()

Description: Returns the DesignerAPI of the previously created instance of the WebDesigner component.

Parameter (Type):

instanceId: string

Return Type: **DesignerAPI**

Required: Yes

Sample code

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
```

addLanguage()

Description: Adds language resources for all instances of **WebDesigner**.

Parameter (Type):

lng: string

resources: **ResourceBundle[]**

Return Type: void

Required: Yes

Sample code

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.addLanguage('en', [
            {
               "ns": "app",
               "lng": "en",
               "resources": {
                   "about": {
                       "textAppTitleCompact": "",
                   },
               }
           },
]);
```

destroy()

Description: Destroys Designer application.

Parameter (Type):

selector: string (the Designer container selector)
instanceId: string (use only if Designer was created using **DesignerSettings.instanceId**)

Return Type: void

Required: Yes

Sample Code

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#container-1', { ...settings, instanceId: 'instance-1' });
arWebDesigner.destroy('#container-1', 'instance-1');
```

GlobalDesignerAPI

Type of **GrapeCity.ActiveReports.Designer** object exported by the **web-designer.js** module.

create()

Description: Renders the WebDesigner component to the ** ** element with given **selector** using the specified **DesignerSettings** object.

Parameter (Type):

selector: string
settings: **DesignerSettings**

Return Type: Promise<**DesignerAPI**>

Required: Yes

Sample Code

```
<html>
 <head>
     <title>ActiveReports WebDesigner</title>
     <meta charset="utf-8">
     <meta name="viewport" content="width=device-width, initial-scale=1">
     <meta http-equiv="x-ua-compatible" content="ie=edge">
    <link rel="stylesheet" href="vendor/css/bootstrap.css" />
    <link rel="stylesheet" href="vendor/css/fonts-googleapis.css" type="text/css">
    <!-- Optional. Resets the browser's default style, which allows the web designer to occupy the whole page. -->
     <style>
         html, body { width: 100%; height: 100%; margin: 0; padding: 0 }
     </style>
    <!-- Required for the ActiveReports Web Viewer -->
     <link rel="stylesheet" href="jsViewer.min.css" />
     <link rel="stylesheet" href="web-designer.css" />
 </head>
 <body>
     <!-- Required for the ActiveReports Web Viewer -->
     <script src="jsViewer.min.js"></script>
     <script src="web-designer.js"></script>
    <!-- Important! Designer requires a defined size or a container to fill -->
     <div id="ar-web-designer" style="width: 100%; height: 100%;"></div>
    <script>
         /* Required for the ActiveReports Web Viewer */
         var viewer = null;
         GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
             appBar: {
```

```
                saveButton: { visible: false },
                saveAsButton: { visible: false }
            },
            data: {
                dataSets: { canModify: true },
                dataSources: { canModify: true }
            },
            server: {
                url: 'api'
            },
            preview: {
                /* Required for the ActiveReports Web Viewer */
                openViewer: ({ element, documentInfo: { id: documentId } }) => {
                    if (viewer) {
                        viewer.openReport(documentId);
                        return;
                    }
                    viewer = GrapeCity.ActiveReports.JSViewer.create({
                        element: '#' + element,
                        reportID: documentId,
                        renderFormat: options.preferredFormat || 'html',
                        reportService: {
                            url: 'api/reporting'
                        },
                        settings: {
                            zoomType: 'FitPage'
                        }
                    });
                }
            }
        });
    </script>
</body>
</html>
```

apiof()

Description: Returns the DesignerAPI of the previously created instance of the WebDesigner component.

Parameter (Type):

instanceId: string

Return Type: **DesignerAPI** | undefined

Required: Yes

Sample code

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
```

addLanguage()

Description: Adds language resources for all instances of **WebDesigner**.

Parameter (Type):

lng: string
resources: **ResourceBundle[]**

Return Type: void

Required: Yes

Sample code

```
GrapeCity.ActiveReports.Designer.addLanguage('en', [
            {
                "ns": "app",
                "lng": "en",
                "resources": {
                    "about": {
                        "textAppTitleCompact": "",
                    },
                }
            },
]);
```

destroy()

Description: Destroys Designer application.

Parameter (Type):

selector: string (the Designer container selector)

instanceId: string (Optional, use only if Designer was created using **DesignerSettings.instanceId**)

Return Type: void

Required: Yes

Sample Code with instanceId

```
GrapeCity.ActiveReports.Designer.create('#container-1', { settings, instanceId: 'instance-1' });
GrapeCity.ActiveReports.Designer.destroy('#container-1', 'instance-1');
```

Sample Code without instanceId

```
GrapeCity.ActiveReports.Designer.create('#container-2', settings);
GrapeCity.ActiveReports.Designer.destroy('#container-2');
```

ResourceBundle

Localization resource for a specific locale.

lng

Description: Refers to the bundle language.

Return Type: string

Required: Yes

ns

Description: Refers to the bundle namespace.

Return Type: string

Required: Yes

resources

Description: Refers to the localization resources.

Return Type: Record<string, any>

Required: Yes

DesignerAPI

Type of object returned by functions from the **GlobalDesignerAPI**.

app

Description: Contains application-related information.

Required: Yes

about

Description: Contains documentation links and application-related information.

Required: Yes

»help

Description: Contains designer-related documentation links.

Required: Yes

general

Description: Refers to the general documentation.

Return Type: { text: string; url: string }

Required: Yes

transformation

Description: Refers to the report items transformation information.

Return Type: { text (optional): string; url: string }

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     rpx: { enabled: true },
     appBar: { openButton: { visible: true } }
             }).then((designer) => {
     designer.app.about.help.general.text = 'help text';
     designer.app.about.help.general.url = 'http://myurl';
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     rpx: { enabled: true },
     appBar: { openButton: { visible: true } }
             }).then((designer) => {
     designer.app.about.help.general.text = 'help text';
     designer.app.about.help.general.url = 'http://myurl';
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     rpx: { enabled: true },
     appBar: { openButton: { visible: true } }
             }).then((designer: DesignerAPI) => {
     designer.app.about.help.general.text = 'help text';
     designer.app.about.help.general.url = 'http://myurl';
});
```

»applicationTitle

Description: Specifies the application title for the WebDesigner component.

Return Type: string

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.about.applicationTitle = 'Title text';
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.about.applicationTitle = 'Title text';
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer: DesignerAPI) => {
    designer.app.about.applicationTitle = 'Title text';
});
```

»applicationTitleCompact

Description: Specifies the compact version of the application title for the WebDesigner component.

Return Type: string

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.about.applicationTitleCompact = 'Example text';
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.about.applicationTitleCompact = 'Example text';
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer: DesignerAPI) => {
    designer.app.about.applicationTitleCompact = 'Example text';
});
```

»applicationVersion

Description: Specifies the application version for the WebDesigner component.

Return Type: string

Required: Yes

»coreVersion

Description: Refers to the WebDesigner Core version an application is based on.

Return Type: string

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.about.coreVersion = '1.2.3';
    designer.app.about.applicationVersion = '3.4.5';
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.about.coreVersion = '1.2.3';
    designer.app.about.applicationVersion = '3.4.5';
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer: DesignerAPI) => {
    designer.app.about.coreVersion = '1.2.3';
    designer.app.about.applicationVersion = '3.4.5';
});
```

focus()

Description: Returns focus to the WebDesigner component. Focus may be lost when plugged-in or external components are opened or closed. Returning focus is essential to continue using designer hotkeys like Ctrl+Z (undo), Ctrl+Y (redo), etc.

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.focus();
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer) => {
    designer.app.focus();
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: { enabled: true },
    appBar: { openButton: { visible: true } }
}).then((designer: DesignerAPI) => {
    designer.app.focus();
});
```

editor()

Description: Information about the availability of common actions with the report and selected items

Required: Yes

The flags indicate whether the editor is able to perform the corresponding action. The **Return Type** of these flags is 'boolean'. The actions indicate action associated with the flag. The **Return Type** of these actions is 'void'.

Flag	Action
canUndo()	undo()
canRedo()	Redo()
canCut()	Cut()
canPaste()	Paste()
canCopy()	Copy()
canDelete()	Delete()

See the following section for sample usage of setting flags and the corresponding action to take.

»canUndo()/undo()

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
if (designer.app.editor.canUndo) designer.app.editor.undo();
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canUndo) designer.app.editor.undo();
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canUndo) designer.app.editor.undo();
```

»canRedo()/redo()

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
if (designer.app.editor.canRedo) designer.app.editor.redo();
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canRedo) designer.app.editor.redo();
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canRedo) designer.app.editor.redo();
```

»canCut()/cut()

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
if (designer.app.editor.canCut) designer.app.editor.cut();
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canCut) designer.app.editor.cut();
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canCut) designer.app.editor.cut();
```

»canCopy()/copy()

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
if (designer.app.editor.canCopy) designer.app.editor.copy();
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canCopy) designer.app.editor.copy();
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
```

```
if (designer.app.editor.canCopy) designer.app.editor.copy();
```

»canPaste()/paste()

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
if (designer.app.editor.canPaste) designer.app.editor.paste();
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canPaste) designer.app.editor.paste();
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canPaste) designer.app.editor.paste();
```

»canDelete()/delete()

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
if (designer.app.editor.canDelete) designer.app.editor.delete();
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canDelete) designer.app.editor.delete();
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
if (designer.app.editor.canDelete) designer.app.editor.delete();
```

panels

Description: Contains access to the menu and sidebar panels. It contains following objects: menu and sidebar.

menu

Description: Menu API.

»open()

Parameter Type:
id: string

Return Type: void

»pin

Return Type: void

»close

Return Type: void

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
designer.app.panels.menu.open('document-explorer');
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
designer.app.panels.menu.open('document-explorer');
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
designer.app.panels.menu.open('document-explorer');
```

sidebar

Description: Sidebar API.

»open()

Parameter Type:
id: string

Return Type: void

»close

Return Type: void

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
designer.app.panels.sidebar.open('propsTab');
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
designer.app.panels.sidebar.open('propsTab');
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
designer.app.panels.sidebar.open('propsTab');
```

documents

Description: This object includes functions allowing you to create, open, save reports, and more.

Return Type: **DocumentsAPI**

Required: Yes

notifications

Description: Allows to utilize the built-in notifications system in the WebDesigner component.

Return Type: **NotificationsAPI**

Required: Yes

DocumentsAPI

API to create, edit, open, or save report, to fetch information on unsaved reports, and more.

hasUnsavedChanges()

Description: Indicates whether the report has unsaved changes.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
const val = designer.documents.hasUnsavedChanges(); if (val) console.log('Currently edited report has unsaved changes.');
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
const val = designer.documents.hasUnsavedChanges();
if (val) console.log('Currently edited report has unsaved changes.');
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
const val = designer.documents.hasUnsavedChanges();
if (val) console.log('Currently edited report has unsaved changes.');
```

isNew()

Description: Indicates whether the report was saved at least once.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
const val = designer.documents.isNew();
if (val) console.log('New document');
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
const val = designer.documents.isNew();
if (val) console.log('New document');
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
const val = designer.documents.isNew();
if (val) console.log('New document');
```

info()

Description: Returns information about the currently edited report in the WebDesigner component.

Return Type: **CurrentDocumentInfo**

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
var reportInfo = designer.documents.info();
console.log(`Report "${reportInfo.name}" is currently edited.`);
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
var reportInfo = designer.documents.info();
console.log(`Report "${reportInfo.name}" is currently edited.`);
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
var reportInfo = designer.documents.info();
console.log(`Report "${reportInfo.name}" is currently edited.`);
```

create()

Description: Creates a new report to be edited in the WebDesigner component using the specified **CreateReportOptions** object.

Parameter (Type):

options (optional): **CreateDocumentOptions**

Return Type: Promise<**CreateDocumentInfo**>

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
const designer = arWebDesigner.apiOf('ar-web-designer');
var reportInfo = designer.documents.create().then(function() {
console.log('An empty RDLX report is created.'); });
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
var reportInfo = designer.documents.create().then(function() {
console.log('An empty RDLX report is created.');
});
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
var reportInfo = designer.documents.create().then(() => {
console.log('An empty RDLX report is created.');
});
```

open()

Description: Shows the **Open Report** dialog box in the WebDesigner component.

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
var api = arWebDesigner.apiOf('designer-id');
api.documents.open();
```

UMD Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.open();
```

TypeScript Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.open();
```

openById()

Description: Opens an existing report to be edited in the WebDesigner component with a specified id. Optionally, you can pass the report name and content, else it will be loaded from the server.

Parameter (Type):

id: string
type: **SupportedDocumentType**
name (optional): string
content (optional): any

Return Type: Promise<**OpenDocumentInfo**>

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
var api = arWebDesigner.apiOf('designer-id');
api.documents.openById('MyReport.rdlx', { platform: 'rdlx', type: 'report', subType: 'msl'}).then(() => {
console.log('An existing report "MyReport.rdlx" is opened.');
});
var reportContent = { Width: '6.5in', ReportSections: [{ Type: 'Continuous' as any, Name: 'ContinuousSection1', Body: { ReportItems: [ {Type: 'textbox', Name:
'TextBox1', Width: '5in', Height: '1in' } ] }}]};
api.documents.openById('NewReport.rdlx', { platform: 'rdlx', type: 'report', subType: 'msl'}, 'NewReport', reportContent).then(() => {
console.log('New report with one textbox created and opened.');
});
```

UMD Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.openById('MyReport.rdlx', { platform: 'rdlx', type: 'report', subType: 'msl'}).then(() => {
console.log('An existing report "MyReport.rdlx" is opened.');
});
var reportContent = { Width: '6.5in', ReportSections: [{ Type: 'Continuous' as any, Name: 'ContinuousSection1', Body: { ReportItems: [ {Type: 'textbox', Name:
'TextBox1', Width: '5in', Height: '1in' } ] }}]};
api.documents.openById('NewReport.rdlx', { platform: 'rdlx', type: 'report', subType: 'msl'}, 'NewReport', reportContent).then(() => {
console.log('New report with one textbox created and opened.');
});
```

TypeScript Module

```
const api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.openById('MyReport.rdlx', { platform: 'rdlx', type: 'report', subType: 'msl'}).then(() => {
console.log('An existing report "MyReport.rdlx" is opened.');
});
const reportContent = { Width: '6.5in', ReportSections: [{ Type: 'Continuous' as any, Name: 'ContinuousSection1', Body: { ReportItems: [ {Type: 'textbox',
Name: 'TextBox1', Width: '5in', Height: '1in' } ] }}]};
api.documents.openById('NewReport.rdlx', { platform: 'rdlx', type: 'report', subType: 'msl'}, 'NewReport', reportContent).then(() => {
console.log('New report with one textbox created and opened.');
});
```

save()

Description: Saves the currently edited report in the WebDesigner component. If the report is new, then the **Save As** dialog box will be opened.

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
var api = arWebDesigner.apiOf('designer-id');
api.documents.save();
```

UMD Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.save();
```

TypeScript Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.save();
```

saveAs()

Description: Opens the **Save As** dialog box in the WebDesigner component.

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
var api = arWebDesigner.apiOf('designer-id');
api.documents.saveAs();
```

UMD Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.saveAs();
```

TypeScript Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.saveAs();
```

saveById()

Description: Saves the currently edited report in the WebDesigner component using the specified **id**.

Parameter (Type):

id (optional): string
name (optional): string

Return Type: Promise<**SaveDocumentInfo**>

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
var api = arWebDesigner.apiOf('designer-id');
api.documents.saveById('MyReport.rdlx');
```

UMD Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.saveById('MyReport.rdlx');
```

TypeScript Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.saveById('MyReport.rdlx');
```

saveByName()

Description: Saves the report currently edited in the WebDesigner component using the specified **name**.

Parameter (Type):

name: string

Return Type: Promise<**SaveDocumentInfo**>

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
var api = arWebDesigner.apiOf('designer-id');
api.documents.saveByName('MyReport.rdlx');
```

UMD Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.saveByName('MyReport.rdlx');
```

TypeScript Module

```
var api = GrapeCity.ActiveReports.Designer.apiOf('designer-id');
api.documents.saveByName('MyReport.rdlx');
```

RpxReportDocumentType

Section Report (.rpx) document type.

```
type RpxReportDocumentType = { platform: 'rpx'; type: 'report' };
```

RdlxFplReportDocumentType

Page Report (.rdlx) document type.

```
type RdlxFplReportDocumentType = { platform: 'rdlx'; type: 'report'; subType: 'fpl'};
```

RdlxMslReportDocumentType

RDLX Multi-Section (.rdlx) document type.

```
type RdlxMslReportDocumentType = { platform: 'rdlx'; type: 'report'; subType: 'msl'};
```

RdlxMslDashboardDocumentType

RDLX Dashboard (.rdlx) document type.

```
type RdlxMslDashboardDocumentType = { platform: 'rdlx'; type: 'report'; subType: 'msl'};
```

RdlxMasterMultiReportDocumentType

RDLX report (.rdlx) document type.

```
type RdlxMasterMultiReportDocumentType = { platform: 'rdlx'; type: 'master'; subType: 'msl'};
```

RdlxReportDocumentType

RDLX, Page, RDLX Multi-Section, or RDLX Dashboard document type.

Acceptable Values
`RdlxFplReportDocumentType | RdlxMslReportDocumentType | RdlxMslDashboardDocumentType | RdlxMasterMultiReportDocumentType;`

SupportedDocumentType

Type of documents supported by the WebDesigner component.

Acceptable Values
`RpxReportDocumentType | RdlxReportDocumentType;`

NotificationsAPI

API for notifications for a user action, error, warning, and more.

send()

Description: Sends a notification of the specified level, including caption and content.

Parameter (Type):

level: 'info' | 'warning' | 'error'
caption: string
content (optional): string

level refers to notification level. It determines the color and icons used for the notifications.
caption refers to notification caption. It is displayed when the notification pops up by default, then used as a title in Notification Details view.
content refers to notification content. It is only visible when the Notification Details are open.

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.send('info', 'My information');
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.send('info', 'My information');
});
```

TypeScript Module

```
var designer: DesignerAPI = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
 }).then((api: DesignerAPI) => {
     api.notifications.send('info', 'My information');
});
```

info()

Description: Sends a general notification, which can be used to notify when any user-initiated action is complete.

Parameter (Type):

caption: string
text (optional): string

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.info('Notification');
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.info('Notification');
});
```

TypeScript Module

```
var designer: DesignerAPI = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
 }).then((api: DesignerAPI) => {
     api.notifications.info('Notification');
});
```

error()

Description: Sends an error notification.

Parameter (Type):

caption: string
errorText (optional): string

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.error("Application error");
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.error("Application error");
});
```

TypeScript Module

```
var designer: DesignerAPI = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api: DesignerAPI) => {
     api.notifications.error("Application error");
});
```

warning()

Description: Sends a warning notification.

Parameter (Type):

caption: string
warningText (optional): string

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.warning('Warning');
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.warning('Warning');
});
```

TypeScript Module

```
var designer: DesignerAPI = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api: DesignerAPI) => {
     api.notifications.warning('Warning');
});
```

dismissAll()

Description: Dismisses all notifications.

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.dismissAll();
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     storeUnsavedReport: false
}).then((api) => {
     api.notifications.dismissAll();
});
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
designer.notifications.dismissAll();
```

Font

label

Return Type: string

Required: Yes

value

Return Type: string

Required: Yes

FontHeader

header

Return Type: string

Required: Yes

Color

R

Return Type: number

Required: Yes

G

Return Type: number

Required: Yes

B

Return Type: number

Required: Yes

A

Return Type: number

Required: Optional

MenuCssIcon

type

Return Type: css

Required: Yes

class

Return Type: string

Required: Yes

MenuIcon

Menu Icon of the WebDesigner component.

Acceptable Value
`MenuCssIcon`

DesignerSettings

API for designer settings.

instanceId

Description: Indicates the unique identifier for the WebDesigner instance. This is required if there are multiple instances of the WebDesigner on the same page.

Return Type: string

Required: Yes

Sample Code

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
```

language

Description: Specifies the language to use for the WebDesigner component. If **language** is not specified, browser preferences are used. The localization is available in following languages: `'en-US', 'ja-JP', and 'zh-CN'`. The default value for this property is set to 'en-US'.

Sample Code

```
designerSettings.language = 'zh-CN';
```

Return Type: string

Required: Yes

fonts

Description: Specifies the list of fonts displayed in the Font editors all over the WebDesigner component, and supports plain strings, label-value pairs, headers, and splitters. If **fonts** are not specified explicitly here, the default list of fonts is used:
`'Arial', 'Arial Black', 'Comic Sans MS', 'Courier New', 'Geneva', 'Georgia', 'Helvetica', 'Impact', 'Lucida Console', 'Meiryo', 'Meiryo UI', 'MingLiU', 'MingLiU-ExtB', 'MS Gothic', 'MS Mincho', 'MS PGothic', 'MS PMincho', 'MS Song', 'MS UI Gothic', 'NSimSun', 'Osaka', 'PMingLiU', 'PMingLiU-ExtB', 'SimSun', 'SimSun-ExtB', 'Song', 'Tahoma', 'Times New Roman', 'Trebuchet MS', 'Verdana',` and `'Yu Gothic'`.

Return Type: (string | **Font** | **FontHeader**)[]

Required: Yes

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   fonts: [{ header: 'Questionable Choice' }, { label: 'Pretty Font', value: 'Comic Sans MS' }, { header: '' }, 'Arial', 'Courier New', 'Times New Roman']
});
```

themes

default

Description: The default theme to be applied.

Return Type: string

Required: Yes

The default theme can be set from any of these: activeReports, activeReportsDark, defaultDark, darkOled, highContrast, highContrastDark.

ES Module

```
import { arWebDesigner } from './web-designer.js';
 arWebDesigner.create('#ar-web-designer', {
       themes: { default: 'defaultDark'}
 });
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      themes: { default: 'defaultDark'}
});
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      themes: { default: 'defaultDark'}
});
```

list

Description: An array of available themes that can be picked by the user. You can use either built-in theme names, or pass the theme object. A theme object can be created using `GrapeCity.ActiveReports.DesignerThemes.create()`.

Return Type: (string | Record<string, string | Color | boolean>)[];

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     themes: { list: ['default','defaultDark']}
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      themes: { list: ['default','defaultDark']}
});
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      themes: { list: ['default','defaultDark']}
});
```

detectDarkTheme

Description: Indicates the whether designer should automatically detect and switch to a dark theme based on system settings. By default, this setting is set to 'false'.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     themes: { detectDarkTheme: true }
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     themes: { detectDarkTheme: true }
});
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     themes: { detectDarkTheme: true }
});
```

picker

Description: Shows Theme Picker in the UI, which lists all the available themes. By default, this setting is set to 'true'.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     themes: { picker: { enabled: false }
});
```

UMD Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     themes: { picker: { enabled: false }
});
```

TypeScript Module

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     themes: { picker: { enabled: false }
});
```

dateFormats

Description: Specifies the list of supported date formats. See Microsoft Documentation for more information on custom date and time format strings.

Return Type: string[]

Required: Yes

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   dateFormats = ['yyyy/MM/dd HH:mm:ss', 'yyyy/MM/dd', 'HH:mm:ss', 'tt hh:mm:ss']
});
```

imageMimeTypes

Description: Specifies the list of supported image mime-types in the WebDesigner component.

Return Type: string[]

Required: Yes

units

Description: Specifies the default measurement units for the WebDesigner component. The default unit is inches.

Return Type: 'in' | 'cm'

Required: Optional

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  storeUserPreferences: false,
  units: 'cm'
});
```

lockLayout

Description: By default, the **lockLayout** property is set to 'false'. However, when you set this property to 'true', it enables you to modify the properties of existing report items. Operations that can modify the report layout structure are not possible, such as adding a new report item or deleting an existing one, and others.

Return Type: boolean

Required: Yes

Sample Code

```
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
designer.lockLayout = 'true';
```

document

Return Type: { id: string; type: **SupportedDocumentType**; };

Required: Optional

storeUnsavedReport

Description: By default, the **storeUnsavedReport** property is set to 'true'. In this case, the last unsaved report can be restored if the browser tab or browser itself gets accidentally closed. However, if the **storeUnsavedReport** property is set to 'false', the aforementioned functionality is not available.

Return Type: boolean

Required: Yes

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
           storeUnsavedReport: false
});
```

storeUserPreferences

Description: By default, the **storeUserPreferences** property is set to 'true'. In this case, the user preferences will be saved to the browser storage. However, if the **storeUnsavedReport** property is set to

'false', the aforementioned functionality is not available.

Return Type: boolean

Required: Yes

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
            storeUserPreferences: false
});
```

disableFocusTimer

Description: By default, the **disableFocusTimer** property is set to 'true'. In such a case, the focused elements (like buttons) are highlighted only for a short period after the **Tab** key is pressed. However, if the **disableFocusTimer** property is set to 'false', the focus highlighting timer on the buttons is disabled for better accessibility.

Return Type: boolean

Required: Yes

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
            disableFocusTimer: true
});
```

disableSystemClipboard

Description: Disable the usage of the system clipboard. Copy-paste between designer instances will work only in the same browser in the same domain.

Return Type: boolean

Required: Yes

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
            disableSystemClipboard: true
});
```

filterProperties

Description: Return filtered array of descriptors in the order in which descriptors should be rearranged.

Parameter (Type):

descriptors: PropertyDescriptor[]

item: Record<string, any>

platform: 'rdlx' | 'rpx'

Return Type: PropertyDescriptor[]

Required: Optional

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  filterProperties: (descriptors, item, platform) => platform === 'rpx' ? [] : descriptors,
});
```

editor

Description: Settings available for the Editors in the Web designer component.

Required: Yes

rulers

Description: Specifies the ruler settings for the WebDesigner component.

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
   editor: {
       rulers: {
           visible: true
       }
   }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   editor: {
       rulers: {
           visible: true
```

```
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    editor: {
        rulers: {
            visible: true
        }
    }
});
```

»visible

Description: Specifies whether rulers need to be shown in the WebDesigner component, by default.

Return Type: boolean

Required: Yes

»snapStep

Description: Specifies the snapStep value. The default value is { in: 0.25, cm: 0.5 }.

Return Type: { in: number; cm: number; };

Required: Optional

gridSize

Description: Specifies the size of the **Grid Size** editor. By default, 'in' is used.

Return Type: string

Required: Optional

showGrid

Description: Specifies whether the grids must be shown or hidden in the WebDesigner component.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    editor: {
        showGrid: true
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    editor: {
        showGrid: true
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    editor: {
        showGrid: true
    }
});
```

snapToGrid

Description: Specifies whether to display the **Snap To Grid** option in the WebDesigner component. The default value is 'false'.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    editor: {
        snapToGrid: true
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    editor: {
```

```
        snapToGrid: true
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    editor: {
        snapToGrid: true
    }
});
```

snapToGuides

Description: Specifies whether to display the **Snap To Guides** option in the WebDesigner component. The default value is 'false'.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer',
    editor: {
        snapToGuides: true
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    editor: {
        snapToGuides: true
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    editor: {
        snapToGuides: true
    }
});
```

appBar

Description: Settings for the **App** bar in the WebDesigner component.

Required: Yes

visible

Description: Specifies whether the **App** bar needs to be shown in the WebDesigner component. By default, the **App** bar is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        appBar: {
            visible: false
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        appBar: {
            visible: false
        }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        appBar: {
            visible: false
        }
});
```

saveButton

Description: Settings for the **Save** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Save** button needs to be shown in the WebDesigner component. By default, the **Save** button is hidden.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    appBar: {
        saveButton: { visible: true }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        saveButton: { visible: true }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        saveButton: { visible: true }
    }
});
```

saveAsButton

Description: Settings for the **Save As** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **SaveAs** button needs to be shown in the WebDesigner component. By default, the **SaveAs** button is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        appBar: {
           saveAsButton: { visible: false}
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        saveAsButton: { visible: false}
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        saveAsButton: { visible: false}
    }
});
```

openButton

Description: Settings for the **Open** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Open** button needs to be shown in the WebDesigner component. By default, the **Open** button is hidden.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    appBar: {
        openButton: { visible: true }
```

```
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        openButton: { visible: true }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        openButton: { visible: true }
    }
});
```

insertTab

Description: Settings for the **Insert** tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Insert** tab needs to be shown in the Web Designer's application bar. The **ToolBox** and **Insert** tab are interchangeable. By default, this tab is hidden.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    appBar: {
        insertTab: { visible: false}
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        insertTab: { visible: false}
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        insertTab: { visible: false}
    }
});
```

homeTab

Description: Settings for the **Home** tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Home** tab needs to be shown in the Web Designer's application bar. By default, this tab is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    appBar: {
        homeTab: { visible: false}
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        homeTab: { visible: false}
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
```

```
    appBar: {
        homeTab: { visible: false}
    }
});
```

contextActionsTab

Description: Settings for the **Context Actions** tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Context Actions** tab needs to be shown in the Web Designer's application bar. By default, the this tab is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    appBar: {
        contextActionsTab: { visible: false}
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        contextActionsTab: { visible: false}
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        contextActionsTab: { visible: false}
    }
});
```

parametersTab

Description: Settings for the **Parameters** tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Parameters** tab needs to be shown in the Web Designer's application bar. By default, this tab is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    appBar: {
        parametersTab: { visible: false}
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        parametersTab: { visible: false}
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        parametersTab: { visible: false}
    }
});
```

scriptTab

Description: Settings for the **Script** tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Script** tab needs to be shown in the Web Designer's application bar. By default, this tab is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    appBar: {
        scriptTab: { visible: false}
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        scriptTab: { visible: false}
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    appBar: {
        scriptTab: { visible: false}
    }
});
```

toolBar

Description: Settings for the **Tool Bar**.

Required: Yes

visible

Description: Specifies whether the **Tool Bar** needs to be shown in the WebDesigner component. By default, the **Tool Bar** is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    toolBar: { visible: false }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    toolBar: { visible: false }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    toolBar: { visible: false }
});
```

menu

Description: Settings for menus in the WebDesigner component.

Required: Yes

visible

Description: Specifies whether the **Main** menu needs to be shown in the WebDesigner component. By default, the **Main** menu is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    menu: { visible: false }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    menu: { visible: false }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    menu: { visible: false }
});
```

logo

Description: Specifies the settings for the custom **logo** in the menu of WebDesigner.

Required: Optional

»visible

Description: Specifies whether the logo needs to be shown in the menu.

Required: Optional

Return Type: Boolean

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    menu: {
        logo: { visible: false }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   menu: {
       logo: { visible: false }
   }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   menu: {
       logo: { visible: false }
   }
});
```

»custom

Description: Sets a custom logo to be shown in the menu.

Required: Optional

Return Type: MenuIcon

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    menu: {
       logo: { custom: { type: 'css', class: 'my-custom-icon' }; }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   menu: {
      logo: { custom: { type: 'css', class: 'my-custom-icon' }; }
   }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   menu: {
      logo: { custom: { type: 'css', class: 'my-custom-icon' }; }
   }
});
```

toolBox

Description: Settings for the **ToolBox** menu in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **ToolBox** menu needs to be shown in the WebDesigner component. By default, the **ToolBox** menu is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
  menu: {
     toolBox: { visible: false }
  }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  menu: {
     toolBox: { visible: false }
  }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  menu: {
     toolBox: { visible: false }
  }
});
```

documentExplorer

Description: Settings for the **Document Explorer** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Document Explorer** button needs to be shown in the WebDesigner component. By default, the **Document Explorer** button is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
  menu: {
      documentExplorer: { visible: false }
  }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  menu: {
      documentExplorer: { visible: false }
  }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  menu: {
      documentExplorer: { visible: false }
  }
});
```

groupEditor

Description: Settings for the **Group Editor** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Group Editor** button needs to be shown in the WebDesigner component. By default, the **Group Editor** button is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    menu: {
       groupEditor: { visible: false }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   menu: {
```

```
      groupEditor: { visible: false }
  }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    menu: {
      groupEditor: { visible: false }
   }
});
```

layerEditor

Description: Settings for the **Layer Editor** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Layer Editor** button needs to be shown in the WebDesigner component. By default, the **Layer Editor** button is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
  menu: {
      layerEditor: { visible: false }
  }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  menu: {
      layerEditor: { visible: false }
  }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  menu: {
      layerEditor: { visible: false }
  }
});
```

statusBar

Description: Settings for the **Status** bar in the WebDesigner component.

Required: Yes

visible

Description: Specifies whether the **Status Bar** needs to be shown. By default, the Status Bar is visible.

Return Type: boolean

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
  statusBar: { visible: false }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
  statusBar: { visible: false }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   statusBar: { visible: false }
});
```

toggleUnitsButton

Description: Settings for the **Units** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Units** button needs to be shown in the WebDesigner component. By default, the **Units** button is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    statusBar: {
        toggleUnitsButton: {
            visible: true
          }
      }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        toggleUnitsButton: {
            visible: true
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        toggleUnitsButton: {
            visible: true
        }
    }
});
```

toggleGridButton

Description: Settings for the **Show Grid** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Show Grid** button needs to be shown in the WebDesigner component. By default, the **Show Grid** button is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    statusBar: {
        toggleGridButton: {
            visible: true
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        toggleGridButton: {
            visible: true
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        toggleGridButton: {
            visible: true
        }
    }
});
```

gridSizeEditor

Description: Settings for the **Grid Size** editor in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Grid Size** editor needs to be shown in the WebDesigner component. By default, the **Grid Size** editor is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    statusBar: {
        gridSizeEditor: {
            visible: true
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        gridSizeEditor: {
            visible: true
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        gridSizeEditor: {
            visible: true
        }
    }
});
```

rulersButton

Description: Settings for the **Show Rulers** button in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Show Rulers** button needs to be shown in the WebDesigner component. By default, the **Show Rulers** button is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    statusBar: {
        rulersButton: {
            visible: true
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        rulersButton: {
            visible: true
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        rulersButton: {
            visible: true
        }
    }
});
```

propertiesModeButton

Description: Settings for the **Properties Mode** drop-down in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Properties Mode** drop-down needs to be shown in the WebDesigner component. By default, the **Properties Mode** drop-down is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    statusBar: {
        propertiesModeButton: {
            visible: true
           }
       }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        propertiesModeButton: {
            visible: true
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    statusBar: {
        propertiesModeButton: {
            visible: true
        }
    }
});
```

propertyGrid

Description: Settings for the **Property** grid in the WebDesigner component.

Required: Yes

propertiesTab

Description: Settings for the **Properties** tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Properties** tab needs to be shown in the WebDesigner component. By default, the **Properties** tab is visible.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           visible: true
       }
   }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           visible: true
       }
   }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           visible: true
       }
   }
});
```

mode

Description: Specifies the available property modes in the WebDesigner component.

Return Type: 'Basic' | 'Advanced'

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           mode: 'Basic'
       }
   }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           mode: 'Basic'
       }
   }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           mode: 'Basic'
       }
   }
});
```

sort

Description: Specifies the available properties sort modes in the WebDesigner component.

Return Type: 'categorized' | 'alphabetical'

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           sort: 'alphabetical'
       }
   }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           sort: 'alphabetical'
       }
   }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           sort: 'alphabetical'
       }
   }
});
```

collapsibleCategories

Description: Settings for the **collapsibleCategories** in the WebDesigner component.

Required: Optional

»enabled

Description: When set to true, Property grid categories becomes collapsible and app memorizes categories' and editor's expand/collapse states.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
           collapsibleCategories: {
               enabled: false
           }
       }
   }
```

```
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
            collapsibleCategories: {
                enabled: false
            }
       }
   }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       collapsibleCategories: {
           enabled: false
       }
   }
});
```

saveExpandEditorsState

Description: Settings for the **saveExpandEditorsState** in the WebDesigner component.

Required: Optional

»enabled

Description: When set to true, app memorizes editor's expand/collapse states.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
            saveExpandEditorsState: {
                enabled: false
            }
       }
   }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
            saveExpandEditorsState: {
                enabled: false
            }
       }
   }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
   propertyGrid: {
       propertiesTab: {
            saveExpandEditorsState: {
                enabled: false
            }
       }
   }
});
```

documents

Description: Settings for the Document API in the WebDesigner component.

Required: Yes

fileView

Description: Settings for the **File View** tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **File View** tab needs to be shown in the WebDesigner component. By default, the **File View** tab is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        documents: {
            fileView: {
                visible: false,
            }
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        documents: {
            fileView: {
                visible: false,
            }
        }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        documents: {
            fileView: {
                visible: false,
            }
        }
   }
});
```

handlers

Required: Yes

»onBeforeSave()

Description: An async handler, cancels saving process if returned false.

Parameter (Type):

info: **SaveDocumentInfo**

Return Type: Promise<boolean>

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        documents: {
            fileView: {
        handlers: {
            onBeforeSave: (info) => {
                return new Promise((resolve, reject) => {
                    resolve(false);
                });
            }
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
        handlers: {
            onBeforeSave: (info) => {
                return new Promise((resolve, reject) => {
                    resolve(false);
                });
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
        handlers: {
            onBeforeSave: (info: SaveDocumentInfo) => {
                return new Promise((resolve, reject) => {
                    resolve(false);
                });
            }
        }
    }
```

```
});
```

»onAfterSave()

Description: A handler that is called when the save process is completed.

Parameter (Type):

info: **SaveDocumentInfo**

Return Type: void

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    documents: {
        handlers: {
            onAfterSave: (info) => {
                if (!info.success) throw new Error(`Report saving error`);
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
        handlers: {
            onAfterSave: (info) => {
                if (!info.success) throw new Error(`Report saving error`);
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
        handlers: {
            onAfterSave: (info) => {
                if (!info.success) throw new Error(`Report saving error`);
            }
        }
    }
});
```

»onBeforeOpen()

Description: Async handler, cancels opening process if returned false.

Return Type: Promise<boolean>

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    documents: {
       handlers: {
            onBeforeOpen: () => {
                return new Promise((resolve, reject) => {
                    resolve(false);
                });
            }
       }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
            onBeforeOpen: () => {
                return new Promise((resolve, reject) => {
                    resolve(false);
                });
            }
       }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
```

```
            onBeforeOpen: () => {
                return new Promise((resolve, reject) => {
                    resolve(false);
                });
            }
        }
    }
});
```

»onAfterOpen()

Description: A handler that is called when the open process is completed.

Return Type: void

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    documents: {
       handlers: {
           onAfterOpen: () => {
               console.log('New report opened successful.')
           }
       }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
           onAfterOpen: () => {
               console.log('New report opened successful.')
           }
       }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
           onAfterOpen: () => {
               console.log('New report opened successful.')
           }
       }
    }
});
```

»onBeforeCreate()

Description: A async handler, cancels the create process if returned false.

Return Type: Promise<boolean>

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    documents: {
       handlers: {
           onBeforeCreate: () => {return false}
       }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
           onBeforeCreate: () => {return false}
       }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
           onBeforeCreate: () => {return false}
       }
    }
});
```

»onAfterCreate()

Description: A handler that is called when the 'create' process is complete.

Return Type: void

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    documents: {
       handlers: {
          onAfterCreate: () => {
                console.log('New report created successful.')
            }
      }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
          onAfterCreate: () => {
                console.log('New report created successful.')
            }
      }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
       handlers: {
          onAfterCreate: () => {
                console.log('New report created successful.')
            }
      }
    }
});
```

»onInfoUpdate()

Description: A handler that is triggered when report name/id is updated.

Return Type: void

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    documents: {
      handlers: {
          onInfoUpdate: (options) => {
                console.log(`name changed ${options.name}`);
            }
       }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
      handlers: {
          onInfoUpdate: (options) => {
                console.log(`name changed ${options.name}`);
            }
       }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
      handlers: {
          onInfoUpdate: (options:
               {
                   name: string,
                   id?: string,
                   type: SupportedDocumentType['type'],
                   platform: SupportedDocumentType['platform']
               }) =>
                    {console.log(`name changed ${options.name}`);}
       }
    }
```

```
});
```

»onDocumentChanged()

Description: A handler that is triggered when report content is changed.
This function takes an object as an argument that contains two properties:

- **document**: The document property contains the updated version of the document that was changed
- **hasUnsavedChanges**: The hasUnsavedChanges is a boolean value that indicates whether there are any unsaved changes in the document.

Return Type: void

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    documents: {
      handlers: {
            onDocumentChanged: (options) => {
                if (options.hasUnsavedChanges) console.log('Currently edited report has unsaved changes.');
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
      handlers: {
            onDocumentChanged: (options) => {
                if (options.hasUnsavedChanges) console.log('Currently edited report has unsaved changes.');
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    documents: {
      handlers: {
          onInfoUpdate: (options:
              {
                  name: string,
                  id?: string,
                  type: SupportedDocumentType['type'],
                  platform: SupportedDocumentType['platform']
              }) =>
                   {console.log(`name changed ${options.name}`);}
        }
    }
});
```

data

dataTab

Description: Settings related to data in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the **Data** tab needs to be shown in the WebDesigner component. By default, the **Data** tab is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataTab: {
            visible:false
            }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataTab: {
            visible:false
            }
```

```
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataTab: {
            visible:false
        }
    }
});
```

dataSources

Description: Settings for the **Data Sources** section in the **Data** tab.

Required: Yes

»visible

Description: Specifies whether the **Data Sources** section in the **Data** tab needs to be shown in the WebDesigner component. By default, the **Data Sources** section is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSources: {
            visible:false
            }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            visible:false
            }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            visible:false
        }
    }
});
```

»canModify

Description: Specifies whether it is possible to modify (i.e. add, edit, or delete) the data sources in the **Data Sources** section. By default, this feature is disabled.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSources: {
            canModify: true
            }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            canModify: true
            }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
```

```
        dataSources: {
            canModify: true
        }
    }
});
```

»shared

Description: Specifies whether the shared data sources should be enabled in the WebDesigner. The shared data sources are available under the 'Shared Reference' option as a **Provider**.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSources: {
            shared: {
                enabled: true
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            shared: {
                enabled: true
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            shared: {
                enabled: true
            }
        }
    }
});
```

»options

predefinedProviders

Description: Specifies the list of predefined data providers available in the **Data Source** editor. By default, all the predefined providers are available.

Return Type: ('SQL' | 'OLEDB' | 'ODBC' | 'JSON' | 'CSV' | 'XML')[]

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                predefinedProviders: ['SQL', 'JSON']
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                predefinedProviders: ['SQL', 'JSON']
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                predefinedProviders: ['SQL', 'JSON']
```

```
            }
        }
    }
});
```

oleDbProviders

Description: Specifies the list of OLE DB providers available in the **Data Source** editor. By default, 'Microsoft.Jet.OLEDB.4.0', 'SQLOLEDB.1', 'MSDataShape.1', and 'MSDASQL.1' are available.

Return Type: string[]

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                oleDbProviders: ['Microsoft.Jet.OLEDB.4.0', 'SQLOLEDB.1']
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                oleDbProviders: ['Microsoft.Jet.OLEDB.4.0', 'SQLOLEDB.1']
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                oleDbProviders: ['Microsoft.Jet.OLEDB.4.0', 'SQLOLEDB.1']
            }
        }
    }
});
```

customProviders

Description: Specifies the list of custom data providers available in the **Data Source** editor. By default, there are no custom data providers available.

Return Type: {key: string; name: string;}[]

where,
key refers to the data provider identifier. This value is used for 'DataSource.ConnectionProperties.DataProvider' property.
name refers to the data provider label. This value is used as a friendly data provider label in UI.

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                customProviders:[{ key: 'CDP', name: 'Custom Data Provider' }]
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                customProviders:[{ key: 'CDP', name: 'Custom Data Provider' }]
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                customProviders:[{ key: 'CDP', name: 'Custom Data Provider' }]
```

```
            }
        }
    }
});
```

dataSets

Description: Settings for the **Datasets** section in the **Data** tab.

Required: Yes

»visible

Description: Specifies whether the **Datasets** section needs to be shown in the WebDesigner component. By default, the **Datasets** section is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                customProviders:[{ key: 'CDP', name: 'Custom Data Provider' }]
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                customProviders:[{ key: 'CDP', name: 'Custom Data Provider' }]
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSources: {
            options: {
                customProviders:[{ key: 'CDP', name: 'Custom Data Provider' }]
            }
        }
    }
});
```

»canModify

Description: Specifies whether it is possible to modify (including add/edit/remove) datasets in the WebDesigner component. The default value for this property is 'false'.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        dataSets: {
            canModify:true
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSets: {
            canModify:true
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        dataSets: {
            canModify:true
        }
```

```
    }
});
```

parameters

Description: Settings for the **Parameters** section in the **Data** tab.

Required: Yes

»visible

Description: Specifies whether the **Parameters** section needs to be shown in the WebDesigner component. By default, the **Parameters** section is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        parameters: {
            visible: false
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        parameters: {
            visible: false
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        parameters: {
            visible: false
        }
    }
});
```

»canModify

Description: Specifies whether it is possible to modify (i.e. add, edit, or delete) the report parameters in the **Parameters** section. The default value for this property is set to 'true'.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        parameters: {
            canModify: false
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        parameters: {
            canModify: false
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        parameters: {
            canModify: false
        }
    }
});
```

commonValues

Description: Settings for the **Common Values** section in the **Data** tab.

Required: Yes

»visible

Description: Specifies whether the **Common Values** section needs to be shown in the WebDesigner component. By default, the **Common Values** section is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    data: {
        commonValues: {
            visible: false
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        commonValues: {
            visible: false
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    data: {
        commonValues: {
            visible: false
        }
    }
});
```

styles

Description: Style settings for Section Reports in the WebDesigner component.

Required: Yes

stylesTab

Description: Settings for the Styles Tab in the WebDesigner component.

Required: Yes

»visible

Description: Specifies whether the stylesheet for the Section Reports (.rpx) can be modified. By default, the **Styles** tab is visible.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    styles: {
        stylesTab: {
            visible:false
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    styles: {
        stylesTab: {
            visible:false
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    styles: {
        stylesTab: {
            visible:false
        }
    }
});
```

stylesheet

Description: Stylesheet settings in the WebDesigner component.

Required: Yes

»canModify

Description: Specifies whether it is possible to modify the Section Report (.rpx) in the WebDesigner component. The default value for this property is 'true'.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    styles: {
        stylesTab: {
            canModify:false
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    styles: {
        stylesTab: {
            canModify:false
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    styles: {
        stylesTab: {
            canModify:false
        }
    }
});
```

server

Description: Backend-related settings for the WebDesigner component.

Required: Yes

url

Description: Specifies the base URL for the Designer Server API. The default value of the property is 'api'.

Return Type: string

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    server: {
        url: 'api/designer'
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    server: {
        url: 'api/designer'
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    server: {
        url: 'api/designer'
    }
});
```

onBeforeRequest()

Description: A special handler to modify requests in the WebDesigner component.

Parameter (Type):

init: RequestInit

Return Type: RequestInit

Required: Optional

onBeforeRequestAsync()

Description: Async version of onBeforeRequest. Use either this or normal version.

Parameter (Type):

init: RequestInit

Return Type: Promise<RequestInit>

Required: Optional

title

Description: Settings for document title in the WebDesigner component.

Required: Yes

onUpdate()

Description: It is possible to implement custom logic for updating the browser tab's title when the edited document gets updated in the WebDesigner component.

Parameter (Type):

init: **TitleInfo**

Return Type: string

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
      title: {
          onUpdate: (info) =>
                `${info.name}${info.hasUnsavedChanges ? ' - Has Unsaved Changes!' : info.name}`
      }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      title: {
          onUpdate: (info) =>
                `${info.name}${info.hasUnsavedChanges ? ' - Has Unsaved Changes!' : info.name}`
      }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      title: {
          onUpdate: (info: TitleInfo) =>
                `${info.name}${info.hasUnsavedChanges ? ' - Has Unsaved Changes!' : info.name}`
      }
});
```

disabled

Description: Specifies whether the browser tab title can be updated in the WebDesigner component. The default value of the property is 'false'.

Return Type: boolean

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     title: {
           disabled: true
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     title: {
           disabled: true
        }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     title: {
           disabled: true
        }
});
```

preview

Description: Preview settings for the document in the WebDesigner component.

Required: Yes

canPreview

Description: Specifies whether the **Preview** button needs to be shown in the WebDesigner component. The default value of the property is 'false'.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    server: {
        url: 'api/designer'
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    server: {
        url: 'api/designer'
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    server: {
        url: 'api/designer'
    }
});
```

openViewer

Description: You can plug in the **Viewer** component by providing the **openViewer()** method.

Parameter (Type):

settings: **ViewerSettings**

Return Type: void

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
 preview: {
     openViewer: (info) =>
         console.log(info.applicationTitle)
  }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
 preview: {
     openViewer: (info) =>
         console.log(info.applicationTitle)
  }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
 preview: {
     openViewer: (info: ViewerSettings) =>
         console.log(info.applicationTitle)
  }
});
```

rpx

Description: RPX platform-specific settings in the WebDesigner component. Must exist for the Section Reports (.rpx) to work.

Required: Yes

enabled

Description: Set to true to enable RPX support.

Return Type: Boolean

Required: Optional

initTemplates

Return Type: {imperialTemplates (optional): string[]; metricTemplates (optional): string[];};

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rpx: {
            enabled: true,
            metricTemplates: ['{"Name":"Report","Width":"10cm","Layers":[{"Name":"default1"}],"CustomProperties":[{"Name":"DisplayType","Value":"Page"},
{"Name":"SizeType","Value":"Default"},{"Name":"PaperOrientation","Value":"Portrait"}],"Page":
{"PageWidth":"8.5in","PageHeight":"11in","RightMargin":"1in","LeftMargin":"1in","TopMargin":"1in","BottomMargin":"1in","Columns":1,"ColumnSpacing":"0in"},"Body":
{"Height":"0.75cm","ReportItems":[{"Type":"textbox","Name":"TextBox1","CustomProperties":[],"CanGrow":true,"KeepTogether":true,"Style":
{"PaddingLeft":"2pt","PaddingRight":"2pt","PaddingTop":"2pt","PaddingBottom":"2pt"},"Left":"0cm","Top":"0cm","Width":"10cm","Height":"5cm"}]}}'],
        }
    );
     var designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
    designer.documents.create({
    name: 'MyReport.rpx',
    type:
        {
            platform: 'rpx',
            type: 'report'
        }
    });
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rpx: {
            enabled: true,
            metricTemplates: ['{"Name":"Report","Width":"10cm","Layers":[{"Name":"default1"}],"CustomProperties":[{"Name":"DisplayType","Value":"Page"},
{"Name":"SizeType","Value":"Default"},{"Name":"PaperOrientation","Value":"Portrait"}],"Page":
{"PageWidth":"8.5in","PageHeight":"11in","RightMargin":"1in","LeftMargin":"1in","TopMargin":"1in","BottomMargin":"1in","Columns":1,"ColumnSpacing":"0in"},"Body":
{"Height":"0.75cm","ReportItems":[{"Type":"textbox","Name":"TextBox1","CustomProperties":[],"CanGrow":true,"KeepTogether":true,"Style":
{"PaddingLeft":"2pt","PaddingRight":"2pt","PaddingTop":"2pt","PaddingBottom":"2pt"},"Left":"0cm","Top":"0cm","Width":"10cm","Height":"5cm"}]}}'],
        }
    );
     var designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
    designer.documents.create({
    name: 'MyReport.rpx',
    type:
        {
            platform: 'rpx',
            type: 'report'
        }
    });
```

TypeScript Module

```
rpx: {
        enabled: true,
        metricTemplates: ['{"Name":"Report","Width":"10cm","Layers":[{"Name":"default1"}],"CustomProperties":[{"Name":"DisplayType","Value":"Page"},
{"Name":"SizeType","Value":"Default"},{"Name":"PaperOrientation","Value":"Portrait"}],"Page":
{"PageWidth":"8.5in","PageHeight":"11in","RightMargin":"1in","LeftMargin":"1in","TopMargin":"1in","BottomMargin":"1in","Columns":1,"ColumnSpacing":"0in"},"Body":
{"Height":"0.75cm","ReportItems":[{"Type":"textbox","Name":"TextBox1","CustomProperties":[],"CanGrow":true,"KeepTogether":true,"Style":
{"PaddingLeft":"2pt","PaddingRight":"2pt","PaddingTop":"2pt","PaddingBottom":"2pt"},"Left":"0cm","Top":"0cm","Width":"10cm","Height":"5cm"}]}}'],
    }
);
const designer = GrapeCity.ActiveReports.Designer.apiOf('ar-web-designer');
designer.documents.create({
name: 'MyReport.rpx',
type:
        {
            platform: 'rpx',
            type: 'report'
        }
});
```

toolBoxContent

Return Type: RpxToolBoxItem[]

Required: Optional

rdlx

Required: Yes

expressionSyntax

Description: Specifies which Expression syntax will be used in the WebDesigner component: 'i11n' - interpolation syntax or 'rdl' - old rdl expression syntax. By default, the interpolation syntax is used for expressions.

Return Type: 'i11n' | 'rdl'

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
     rdlx: {
         expressionSyntax: 'rdl'
     }
 });
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     rdlx: {
         expressionSyntax: 'rdl'
     }
 });
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     rdlx: {
         expressionSyntax: 'rdl'
     }
 });
```

msl

»enabled

Description: Set to true to enable RDLX report(a multi-section layout (MSL)) support.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        rdlx: {
            msl: {
                enabled: true
            }
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            msl: {
                enabled: true
            }
        }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            msl: {
                enabled: true
            }
        }
});
```

fpl

»enabled

Description: Set to true to enable Fixed Page Layout support.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        rdlx: {
            fpl: {
                enabled: true
            }
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            fpl: {
                enabled: true
            }
```

```
        }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            fpl: {
                enabled: true
            }
        }
});
```

dashboard

»enabled

Description: Set to true to enable Dashboard report support.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        rdlx: {
            dashboard: {
                enabled: true
            }
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            dashboard: {
                enabled: true
            }
        }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            dashboard: {
                enabled: true
            }
        }
});
```

reportParts

Description: Report Parts feature related settings.

Return Type: **ReportPartsSettings**

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rdlx: {
         reportParts: {
             enabled: true,
             libraries: [{
                 name: 'Sales',
                 path: 'Libraries/Lib_Sales.rdlx'
             }]
         }
     }
 });
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
         reportParts: {
             enabled: true,
             libraries: [{
                 name: 'Sales',
                 path: 'Libraries/Lib_Sales.rdlx'
             }]
         }
     }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
         reportParts: {
             enabled: true,
             libraries: [{
                 name: 'Sales',
                 path: 'Libraries/Lib_Sales.rdlx'
             }]
         }
     }
});
```

masterReports

»enabled

Description: Set to true to enable master reports support.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
        rdlx: {
            masterReports: {
                enabled: true
            }
        }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            masterReports: {
                enabled: true
            }
        }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        rdlx: {
            masterReports: {
                enabled: true
            }
        }
});
```

expressionEditorNodesFilter()

Description: Filters the Expression Editor nodes in the WebDesigner component.

Parameter (Type):

key: string

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    designerSettings.rdlx.expressionEditorNodesFilter = (key) => {
        if (key.includes('commonValues')) return false;
        if (key.includes('aggregate.aggregateIfWithScope')) return false;
        return true;
```

UMD Module

```
designerSettings.rdlx.expressionEditorNodesFilter = (key) => {
    if (key.includes('commonValues')) return false;
    if (key.includes('aggregate.aggregateIfWithScope')) return false;
    return true;
}
```

TypeScript Module

```
designerSettings.rdlx.expressionEditorNodesFilter = (key: string) => {
    if (key.includes('commonValues')) return false;
    if (key.includes('aggregate.aggregateIfWithScope')) return false;
    return true;
}
```

toolBoxContent

Description: Specifies the report items and their display order in the toolbox. By default, the available report items for Page Report (fpl) are TextBox, CheckBox, Container, Line, Shape, TableOfContents, Image, List, Table, Tablix, Chart, Bullet, Barcode, FormattedText, RichText, Sparkline, SubReport, BandedList, and InputField.

For RDLX reports, these are the default available report items - TextBox, CheckBox, Container, Line, Shape, TableOfContents, Image, List, Table, Tablix, Chart, Bullet, Barcode, FormattedText, RichText, Sparkline, SubReport, OverflowPlaceholder, BandedList, and InputField.

Return Type: {cpl: **RdlxToolBoxItem[]**; fpl: **RdlxToolBoxItem[]**; };

Required: Optional

Sample Code

```
designerSettings.rdlx.toolBoxContent = {
   cpl: [ 'checkbox', 'container', 'textbox' ],
   fpl: [ 'image', 'list', 'table', 'tablix', 'chart', 'bullet', 'barcode', 'formattedtext' ]
}
```

initTemplates

Description: Use Page reports to set custom templates for specific Page report items, for example, 'OverflowPlaceHolder'. It is preferable to use an RDLX report for setting custom templates for all other report items, as well as for pageHeader and pageFooter. In case of Reports, set ConsumeContainerWhitespace and Page-properties: BottomMargin, LeftMargin, RightMargin, TopMargin, PageHeight, PageWidth, and ColumnSpacing.

For the rest of the report items, all properties are set, including, for example, the number of columns in the table or barcode default symbology. Note that if there is a same report item as in the previous one within a subsequent report in an array. Then, in such a case only the last template for this report item will be set. Furthermore, you can also set multiple init templates for some report items. For this, you should add a report with more than one report item of the same type. Templates with the same names will be replaced. Use the 'TemplateName' custom property of each report item to set a localized name of the template. Use the 'AllowWizard' custom property of a chart report item to allow the use of chart wizard when this report item is created. For example: ... ,`"CustomProperties":[{"Name":"TemplateName","Value":"Doughnut Chart"}, {"Name":"AllowWizard","Value":"true"},...],...`

Return Type: {imperialTemplates?: string[]; metricTemplates?: string[];};

Required: Optional

Sample Code

```
imperialTemplates: ['{"Name":"Report","Width":"5in","Layers":[{"Name":"default"}],"CustomProperties":[{"Name":"DisplayType","Value":"Page"},
{"Name":"SizeType","Value":"Default"},{"Name":"PaperOrientation","Value":"Portrait"}],"Page":
{"PageWidth":"8.5in","PageHeight":"11in","RightMargin":"1in","LeftMargin":"1in","TopMargin":"1in","BottomMargin":"1in","Columns":1,"ColumnSpacing":"0in"},"Body":
{"Height":"0.25in","ReportItems":[{"Type":"textbox","Name":"TextBox1","CustomProperties":[],"CanGrow":true,"KeepTogether":true,"Style":
{"PaddingLeft":"2pt","PaddingRight":"2pt","PaddingTop":"2pt","PaddingBottom":"2pt"},"Width":"5in","Height":"0.25in"}]}}'],
metricTemplates: ['{"Name":"Report","Width":"10cm","Layers":[{"Name":"default"}],"CustomProperties":[{"Name":"DisplayType","Value":"Page"},
{"Name":"SizeType","Value":"Default"},{"Name":"PaperOrientation","Value":"Portrait"}],"Page":
{"PageWidth":"8.5in","PageHeight":"11in","RightMargin":"1in","LeftMargin":"1in","TopMargin":"1in","BottomMargin":"1in","Columns":1,"ColumnSpacing":"0in"},"Body":
{"Height":"0.75cm","ReportItems":[{"Type":"textbox","Name":"TextBox1","CustomProperties":[],"CanGrow":true,"KeepTogether":true,"Style":
{"PaddingLeft":"2pt","PaddingRight":"2pt","PaddingTop":"2pt","PaddingBottom":"2pt"},"Left":"0cm","Top":"0cm","Width":"10cm","Height":"0.75cm"}]}}'],
};
```

reportItemsFeatures

Description: Refers to customizable report item features in the WebDesigner component.

Return Type: **RdlxReportItemsSettings**

Required: Yes

reportStyles

Description: Specifies additional styles to add to report item styles in the WebDesigner component.

Return Type: **ReportStyles[]**

Required: Yes

Sample Code

```
designerSettings.rdlx.reportStyles
    {
        Bullet: [{
                id: 'c8aa4403-83ef-402b-a7da-032063cf625a',
                name: 'additionalBulletStyle1',
                content: {
                    ValueColor: 'red',
                    LabelFontColor: '=Theme.Colors!Dark1',
                    LabelFontFamily: '=Theme.Fonts!MinorFont.Family',
                    LabelFontSize: '=Theme.Fonts!MinorFont.Size',
                    LabelFontStyle: '=Theme.Fonts!MinorFont.Style',
                    TicksLineColor: '=Theme.Colors(1,0)'
                }
            },
        ],
        List: [
            {    id: '3d2c3781-4eea-4ac3-8d50-636edd9328d5',
                name: 'additionalListStyle1',
                content: {},
            },
            {    id: '5b7b4e73-22e5-42ed-99c4-62840bdde79d',
                name: 'additionalListStyle2',
                content: {
                    BackgroundColor: '=Theme.Colors!Accent1',
                    Border: {
```

```
                    Color: '=Theme.Colors(4,4)',
                    Width: '1pt',
                    Style: 'Solid',
                }
            }
        }]
    },
    {    ChartPalette: [{
            id: 'c8aa4403-83ef-402b-a7da-0320444',
            name: 'additionalChartPalette',
            content: ['red', '=Theme.Colors!Accent2', '=Theme.Colors!Accent3', '=Theme.Colors!Accent4', '=Theme.Colors!Accent5', '=Theme.Colors!Accent6',
'=Theme.Colors(5,4)', '=Theme.Colors(5,5)', '=Theme.Colors(5,6)', '=Theme.Colors(5,7)', '=Theme.Colors(5,8)', '=Theme.Colors(5,9)']
        }]
    },
];
```

TitleInfo

Information on browser tab title.

name

Description: Refers to the document name.

Return Type: string

Required: Yes

hasUnsavedChanges

Description: Indicates whether the document has unsaved changes or not in the WebDesigner component.

Return Type: boolean

Required: Yes

RdlxToolBoxItem

Report items available in the toolbox.

Acceptable Values

```
'textbox' | 'checkbox' | 'container' | 'line' | 'shape' | 'tableofcontents' |
'image' | 'list' | 'table' | 'tablix' | 'chart' | 'bullet' | 'barcode' | 'formattedtext' |
'richtext' | 'sparkline' | 'subreport' | 'overflowplaceholder' | 'bandedlist' | 'inputfield'
```

RpxToolBoxItem

Report items available in the toolbox.

Acceptable Values

```
'Label' | 'TextBox' | 'CheckBox' | 'RichTextBox' | 'Shape' | 'Picture' | 'Line' | 'PageBreak' |
'Barcode' | 'SubReport' | 'ReportInfo' | 'CrossSectionLine' | 'CrossSectionBox' | 'InputFieldText' | 'InputFieldCheckBox'
```

DvChartPlotType

Chart types available.

Acceptable Values

```
'Custom'| 'Bar' | 'Line' | 'Area' | 'Scatter' | 'HighLowOpenClose' | 'Candlestick' | 'Column' |'Pie' | 'Pyramid' | 'Funnel' | 'Bubble' | 'Gantt' |
'HighLowClose' | 'PolarColumn' | 'PolarBar'|'RadarArea' | 'RadarBubble' | 'RadarScatter' | 'RadarLine' | 'RangeArea' | 'RangeBar' | 'RangeColumn' | 'Gauge'
```

DvChartEncodingType

The encoding type. Following are the encodings from the adorner panel only.

Acceptable Values

```
'Details' | 'Color' | 'Shape' | 'Size' | 'Text'
```

RdlxReportItemsSettings

Settings for the report items.

barcode

Description: Settings for barcodes in the WebDesigner component.

symbologies

Description: Limits the list of barcode symbologies available for creation. By default, all barcode symbologies supported by ActiveReports are available.

Return Type: **RdlxBarcodeSymbology[]**

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            barcode: {
                symbologies: ['Code_128_A', 'Code_128_B', 'Code_128_C']
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            barcode: {
                symbologies: ['Code_128_A', 'Code_128_B', 'Code_128_C']
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            barcode: {
                symbologies: ['Code_128_A', 'Code_128_B', 'Code_128_C']
            }
        }
    }
});
```

hideUnsupportedBarcodeJSProperties

Description: Hides some unsupported barcode JS properties.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            barcode: {
                hideUnsupportedBarcodeJSProperties: true
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            barcode: {
                hideUnsupportedBarcodeJSProperties: true
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            barcode: {
                hideUnsupportedBarcodeJSProperties: true
            }
        }
    }
});
```

chart

Description: Settings for the chart features.

canUseWizard

Description: Specifies whether the Chart Wizard is available for creating a Chart.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
           chart: {
               canUseWizard: false
           }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     rdlx: {
         reportItemsFeatures: {
            chart: {
                canUseWizard: false
            }
         }
     }
 });
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     rdlx: {
         reportItemsFeatures: {
            chart: {
                canUseWizard: false
            }
         }
     }
 });
```

plotChartTypes

Description: Specifies the plot chart types available for creation.

Return Type: boolean

Required: Optional

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
           chart: {
               plotChartTypes: ['Column', 'Bar', 'Line']
           }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            chart: {
                plotChartTypes: ['Column', 'Bar', 'Line']
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            chart: {
                plotChartTypes: ['Column', 'Bar', 'Line']
            }
        }
    }
});
```

hiddenEncodings

Description: Excludes given encodings from encoding panel in chart adorner.

Return Type: **DvChartEncodingType[]**

Required: Yes

ES Module

```
import { arWebDesigner } from './web-designer.js';
arWebDesigner.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            chart: {
                hiddenEncodings: ['Color', 'Text']
            }
        }
    }
});
```

UMD Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            chart: {
                hiddenEncodings: ['Color', 'Text']
            }
        }
    }
});
```

TypeScript Module

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            chart: {
                hiddenEncodings: ['Color', 'Text']
            }
        }
    }
});
```

table

Description: Settings for tables in the WebDesigner component.

autoFillHeader

Description: Specifies whether the **Table Header** needs to be auto-filled when a field is dropped to the **Table Details** section. For example, if the **ProductName** field is dropped to the Details section, the Product Name value is set to the **Header**. By default, this feature is enabled.

Return Type: boolean

Required: Optional

Sample Code

```
designerSettings.rdlx.reportItemsFeatures.table.autoFillHeader = false;
```

autoFillFooter

Description: Specifies whether **Table Footer** needs to be auto-filled when a field is dropped to the **Table Details** section. For example, if the **ProductName** field is dropped to the Details section, `=Count([ProductName])` value is set to the **Footer**. By default, this feature is disabled.

Return Type: boolean

Required: Optional

Sample Code

```
designerSettings.rdlx.reportItemsFeatures.table.autoFillFooter = true;
```

canMergeCellsVertically

Description: Specifies whether vertical merge of cells is enabled within the **Table Header**, **Details**, and **Footer** sections. By default, this feature is enabled.

Return Type: boolean

Required: Optional

Sample Code

```
designerSettings.rdlx.reportItemsFeatures.table.canMergeCellsVertically = false;
```

hideFrozenRowsColumns

Description: Specifies whether to hide FrozenRows or FrozenColumns properties from the Property grid. By default, hideFrozenRowsColumns is set to 'false'.

Return Type: boolean

Required: Optional

Sample Code

```
designerSettings.rdlx.reportItemsFeatures.table.hideFrozenRowsColumns = true;
```

tablix

Description: Settings for Tablix data region in the WebDesigner component.

crossAggregates

Description: Specifies whether the **Tablix Wizard** should hide the cross-aggregates functionality. By default, this feature is enabled.

Return Type: boolean

Required: Optional

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
        reportItemsFeatures: {
            tablix: {
                 crossAggregates: false
            }
        }
    }
});
```

autoFillCorner

Description: Specifies whether the **Tablix Corner** cell needs to be auto-filled when a field is dropped to the **Tablix Row Group** cell. For example, if the **ProductName** field is dropped to the **Row Group** cell, the **Product Name** value is set to the **Corner** cell. By default, this feature is enabled.

Return Type: boolean

Required: Optional

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
             reportItemsFeatures: {
                tablix: {
                 autoFillCorner: false
                }
             }
    }
});
```

canUseWizard

Description: Specifies whether the **Tablix Wizard** is available for creating or editing the Tablix control in the WebDesigner component. By default, this feature is enabled.

Return Type: boolean

Required: Optional

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
             reportItemsFeatures: {
                tablix: {
                 canUseWizard: false
                }
             }
    }
});
```

hideFrozenRowsColumns

Description: Specifies whether to hide FrozenRows or FrozenColumns properties from the Property grid and Tablix wizard. By default, hideFrozenRowsColumns is set to 'false'.

Return Type: boolean

Required: Optional

Sample Code

```
GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
    rdlx: {
             reportItemsFeatures: {
                tablix: {
          hideFrozenRowsColumns: true
             }
        }
    }
});
```

RdlxBarcodeSymbology

Supported barcode symbologies.

Acceptable Values

```
'Ansi39' | 'Ansi39x' | 'BC412' | 'Codabar' | 'Code_11' | 'Code_128_A' | 'Code_128_B' | 'Code_128_C' | 'Code_128auto' |
'Code_2_of_5' | 'Code_93' | 'Code25intlv' | 'Code39' | 'Code39x' | 'Code49' | 'Code93x' | 'DataMatrix' | 'EAN_13' | 'EAN_8' | 'EAN128FNC1' |
'GS1QRCode' | 'HIBCCode128' | 'HIBCCode39' | 'IATA_2_of_5' | 'IntelligentMail' | 'IntelligentMailPackage' | 'ISBN' | 'ISMN' | 'ISSN' |
'ITF14' | 'JapanesePostal' | 'Matrix_2_of_5' | 'MaxiCode' | 'MicroPDF417' | 'MicroQRCode' | 'MSI' | 'Pdf417' | 'Pharmacode' | 'Plessey' |
'PostNet' | 'PZN' | 'QRCode' | 'RM4SCC' | 'RSS14' | 'RSS14Stacked' | 'RSS14StackedOmnidirectional' | 'RSS14Truncated' | 'RSSExpanded' |
'RSSExpandedStacked' | 'RSSLimited' | 'SSCC_18' | 'Telepen' | 'UCCEAN128' | 'UPC_A' | 'UPC_E0' | 'UPC_E1' |
```

ReportItemStyle<T>

Report item style.

id

Return Type: string
Required: Yes

name

Return Type: string
Required: Yes

content

Return Type: T
Required: Yes

BorderStyle

Border style.

Color

Return Type: string
Required: Optional

Style

Return Type: string
Required: Optional

Width

Return Type: string
Required: Optional

CellStyleContent

Styles for Table cell.

TextBoxStyleContent with the following:

LeftBorder

Return Type: **BorderStyle**
Required: Optional

TopBorder

Return Type: **BorderStyle**
Required: Optional

BottomBorder

Return Type: **BorderStyle**
Required: Optional

RightBorder

Return Type: **BorderStyle**
Required: Optional

TextAlign

Return Type: string
Required: Optional

BottomBorder

Return Type: string
Required: Optional

TextBoxStyleContent

Styles for TextBox control.

Color

Return Type: string
Required: Optional

BackgroundColor

Return Type: string
Required: Optional

FontFamily

Return Type: string
Required: Optional

FontSize

Return Type: string
Required: Optional

FontStyle

Return Type: string
Required: Optional

FontWeight

Return Type: string
Required: Optional

ContainerStyleContent

Styles for Container control.

BackgroundColor

Return Type: string
Required: Optional

Border

Return Type: string
Required: Optional

BulletStyleContent

Styles for Bullet data region.

ValueColor

Return Type: string
Required: Optional

TargetLineColor

Return Type: string
Required: Optional

LabelFontColor

Return Type: string
Required: Optional

LabelFontFamily

Return Type: string
Required: Optional

LabelFontSize

Return Type: string
Required: Optional

LabelFontStyle

Return Type: string
Required: Optional

TicksLineColor

Return Type: string
Required: Optional

SparklineStyleContent

Styles for Sparkline control.

LineColor

Return Type: string
Required: Optional

FillColor

Return Type: string
Required: Optional

GradientEndColor

Return Type: string
Required: Optional

MarkerColor

Return Type: string
Required: Optional

RangeFillColor

Return Type: string
Required: Optional

RangeGradientEndColor

Return Type: string
Required: Optional

TableStyleContent

Styles for Table data region.

Header

Return Type: { Rows: **CellStyleContent[]** };
Required: Yes

Details

Return Type: { Rows: **CellStyleContent[]**; AlternatingExpression: string };
Required: Yes

Footer

Return Type: { Rows: **CellStyleContent[]** };
Required: Yes

TableGroups

Styles

Return Type: { Header: { Rows: **CellStyleContent[]** }; Footer: { Rows: **CellStyleContent[]** };}[];
Required: Yes

Border

Return Type: **BorderStyle**
Required: Optional

TableOfContentsStyleContent

Styles for Table of Contents data region.

BackgroundColor

Return Type: string
Required: Optional

Border

Return Type: **BorderStyle**
Required: Optional

Levels

Required: Yes

Color

Return Type: string
Required: Optional

PaddingLeft

Return Type: string
Required: Optional

FontFamily

Return Type: string
Required: Optional

FontSize

Return Type: string
Required: Optional

FontStyle

Return Type: string
Required: Optional

FontWeight

Return Type: string
Required: Optional

ChartPaletteContent

Chart palette.

Acceptable Values

```
string[]
```

ReportStyles

Styles for toolbox items or report controls.

Bullet

Return Type: ReportItemStyle<**BulletStyleContent**>
Required: Yes

CheckBox

Return Type: ReportItemStyle<**TextBoxStyleContent**>
Required: Yes

FormattedText

Return Type: ReportItemStyle<**ContainerStyleContent**>
Required: Yes

InputField

Return Type: ReportItemStyle<**TextBoxStyleContent**>
Required: Yes

List

Return Type: ReportItemStyle<**ContainerStyleContent**>
Required: Yes

Rectangle

Return Type: ReportItemStyle<**ContainerStyleContent**>
Required: Yes

RichText

Return Type: ReportItemStyle<**TextBoxStyleContent**>
Required: Yes

Shape

Return Type: ReportItemStyle<**ContainerStyleContent**>
Required: Yes

Sparkline

Return Type: ReportItemStyle<**SparklineStyleContent**>
Required: Yes

Table

Return Type: ReportItemStyle<**TableStyleContent**>
Required: Yes

TableOfContents

Tablix

Return Type: ReportItemStyle<**TableStyleContent**>
Required: Yes

TextBox

Return Type: ReportItemStyle<**TextBoxStyleContent**>
Required: Yes

ChartPalette

Return Type: ReportItemStyle<**ChartPaletteContent**>
Required: Yes

Table

Return Type: ReportItemStyle<**TableStyleContent**>
Required: Yes

TableOfContents

Return Type: ReportItemStyle<**TableOfContentsStyleContent**>
Required: Yes

Tablix

TextBox

ChartPalette

ViewerSettings

Viewer settings.

element

Description: Refers to the host element's id where to render the Viewer.
Return Type: string
Required: Yes

applicationTitle

Description: Refers to the application title passed by the WebDesigner component.
Return Type: string
Required: Yes

documentInfo

Description: Information on the document to be previewed.
Required: Yes

id

Description: Refers to the document id.
Return Type: string
Required: Yes

content

Description: Refers to the document content in JSON format that can be useful for viewers with in-browser rendering.
Return Type: unknown
Required: Yes

name

Description: Refers to the document name.
Return Type: string
Required: Yes

temporary

Description: Specifies whether the document to be previewed is an existing report saved on the server-side.
Return Type: boolean
Required: Yes

preferredFormat

Description: Specifies preferred rendering format for the document.
Return Type: 'html' | 'svg'
Required: Yes

CreateDocumentOptions

Define what kind of document must be created.

name

Description: Refers to the name of the document. If this property is not specified, the default 'Untitled' name will be used.
Return Type: string
Required: Optional

type

Description: Refers to the document type to create in the WebDesigner component. If this property is not specified, an RDLX report will be created.
Return Type: **SupportedDocumentType**
Required: Optional

template

Description: Refers to the template to use for the document.

Return Type: **DocumentTemplate**

Required: Optional

dataSetsInfo

Description: Refers to the list of RDLX Data Sets to use with the template in the WebDesigner component. Note that these datasets won't work with Section Reports (.rpx).

Return Type: { id: string;, name: string;}[];

where,
id refers to the dataset id.
name refers to the dataset name.

Required: Optional

CreateDocumentInfo

Information about the created document (if it was successfully created).

type

Description: Refers to the document type in the WebDesigner component.

Return Type: **SupportedDocumentType**

Required: Yes

name

Description: Refers to the document name in the WebDesigner component.

Return Type: string

Required: Yes

template

Description: Refers to the document template in the WebDesigner component.

Return Type: **DocumentTemplate**

Required: Optional

success

Description: Undefined in **onBeforeCreate** handler. Defined in **onAfterCreate** handler.

Return Type: boolean

Required: Optional

CurrentDocumentInfo

Information about the current document.

id

Description: Refers to the document id. If an existing report is edited, **id** is defined. Otherwise, if a new report is edited, **id** is 'null'.

Return Type: string | null

Required: Yes

name

Description: Refers to the document name in the WebDesigner component.

Return Type: string

Required: Yes

type

Description: Refers to the document type in the WebDesigner component.

Return Type: **SupportedDocumentType**

Required: Yes

DocumentTemplate

Template to use for the document.

id

Description: Refers to the document template id.

Return Type: string

Required: Optional

content

Description: Refers to the document template content in JSON format that can be useful in case you would like to create a non-empty new report.

Return Type: unknown

Required: Optional

OpenDocumentOptions

Define options for opening a document.

templateInfo

Description: Refers to the template information. If it is specified for report creation, either **templateInfo.id** or **templateinfo.content** needs to be defined.

Return Type: **DocumentTemplate**

Required: Optional

dataSetsInfo

Description: Refers to the list of RDLX datasets to use with the template. Note that these datasets won't work with Section Reports (.rpx).

Return Type: {id: string; name: string;}[];

where,
id refers to the dataset id.
name refers to the dataset name.

Required: Optional

name

Description: Refers to the new report name. If you do not specify the name, the report name is set to 'Untitled'.

Return Type: string

Required: Optional

type

Description: Refers to the report type. If not specified RDLX report will be created.

Return Type: **SupportedDocumentType**

Required: Optional

SaveDocumentInfo

Information about the saving the document.

type

Description: Refers to the type of document.

Return Type: **SupportedDocumentType**

Required: Yes

id

Description: If an existing document is to be overwritten on saving, the correct id should be specified explicitly.

Return Type: string

Required: Optional

name

Description: The correct name needs to be always specified explicitly.

Return Type: string

Required: Yes

isFirstSave

Description: Indicates that a new document is being saved for the first time.

Return Type: boolean

Required: Yes

success

Description: Undefined in **onBefore** handler. Defined in **onAfter** handler.
Return Type: boolean
Required: Optional

OpenDocumentInfo

Information about an open document.

type

Description: Refers to the type of document.
Return Type: **SupportedDocumentType**
Required: Yes

id

Description: Refers to the document id.
Return Type: string
Required: Yes

name

Description: Refers to the document name.
Return Type: string
Required: Yes

success

Description: Undefined in **onBefore** handler. Defined in **onAfter** handler.
Return Type: boolean
Required: Optional

ReportPartsSettings

API for settings related to Report Parts.

enabled

Description: Specifies whether the report parts should be enabled in the WebDesigner for the end users to be able to use libraries.
Return Type: **SupportedDocumentType**
Required: Optional

libraries

Description: List of report items initially available for user libraries.
Return Type:

```
Array<{
          name: string,
          path: string
        }>
```

Required: Optional

Blazor WebDesigner Application

The ASP.NET Core Blazor designer component, provided by ActiveReports, is quite similar to the WebDesigner component. It also requires both server and client parts.

Blazor allows using the client and server parts differently, please see the topics of this section for details:

Blazor Server Application

Blazor Web Assembly

Configure and Use Shared Data Sources

Configure Preview

Blazor WebDesigner API

Blazor Server Application

This topic describes how to add the Blazor Designer component to your Blazor Server Application.

1. Open **Microsoft Visual Studio 2022**.
2. Create a new project and select the **Blazor Server App Empty** template.
3. Type a name for your project and click **Next**.
4. Select a target framework and uncheck 'Configure for HTTPS' option, and then click **Create**.
5. Add the following NuGet packages:
   MESCIUS.ActiveReports.Aspnetcore.Designer
   MESCIUS.ActiveReports.Blazor.Designer
6. Add a new folder called **Resources** to the project.
7. Add a report to the **Resources** folder.
8. Make sure to set the **Build Action** property of the report to **Embedded Resource**.
9. Update Program.cs with the code as demonstrated in the example below.

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Designer;
var resourcesRootDirectory = new DirectoryInfo(".\\resources\\");
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();
builder.Services.AddServerSideBlazor();
builder.Services
     .AddReportDesigner()
     .AddMvc(options => options.EnableEndpointRouting = false)
     .AddJsonOptions(options => options.JsonSerializerOptions.PropertyNamingPolicy =
null);
var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseReportDesigner(config => config.UseFileStore(resourcesRootDirectory, false));
app.UseStaticFiles();
app.UseRouting();
app.MapBlazorHub();
app.MapFallbackToPage("/_Host");
app.Run();
```

10. Add the WebDesigner Blazor component to 'Pages/Index.razor'.

```
@page "/"
@using GrapeCity.ActiveReports.Blazor.Designer;
@inject IJSRuntime JSRuntime
<div style="height:100vh;width:100%"
```

```
    <ReportDesigner @ref="_designer" Document="@_document" />
</div>
@code {
    private ReportDesigner _designer;
    private Document _document = new Document()
      {
            Id = "report.rdlx",
            Type = SupportedDocumentType.cpl
      };
}
```

11. Build and run the solution.

Blazor Web Assembly

This topic describes how to add the Blazor Designer component to your Blazor Web Assembly Application. This project uses the Report Server.

1. Open **Microsoft Visual Studio 2022**.
2. Create a new project and select the **Blazor WebAssembly App Empty** template.
3. Type a name for your project and click **Next**.
4. Select a target framework and uncheck 'Configure for HTTPS' option, and then click **Create**.
5. Add the **MESCIUS.ActiveReports.Blazor.Designer** NuGet package.
6. Add the Blazor WebDesigner component to 'Pages/Index.razor'.

Pages/Index.razor

```
@page "/"
@using GrapeCity.ActiveReports.Blazor.Designer;
@inject IJSRuntime JSRuntime
<div style="height:100vh;width:100%"
    <ReportDesigner @ref="_designer" Server="@_server" Document="@_document" />
</div>
@code {
    private ReportDesigner _designer;
    private Server _server = new Server()
    {
        Url = "http://localhost:5098"
    };
    private Document _document = new Document()
        {
            Id = "Report.rdlx",
            Type = SupportedDocumentType.cpl
        };
}
```

7. Run the Report Server.
8. Build and run the application.

Configure and Use Shared Data Sources

The shared data sources contain connection properties that allow binding multiple reports to the same data. The shared data sources can be created and edited only in Standalone Designer or VS integrated designer. In WebDesigner, however, you can only reference an existing data source.

Following are the scenarios for using shared data sources in web designer:

- a developer wants to prevent the connection string information from being displayed on the client-side
- a developer wants users to be able to select a pre-defined data sources/data sets

The shared data source solves this problem because the report definition contains only the reference to the data source definition, which is resolved on the server-side when a report is previewed.

> **Note**: Shared data sources are disabled by default.

You must create a shared data source (.rdsx) in Standalone Designer or Visual Studio Integrated Designer. See Work with Local Shared Data Sources for more information.

We will be adding shared data source functionality in an already available Blazor Designer Server Application sample

1. Open the Blazor Designer Server Application.
2. Place the shared reference, a .rdsx file, in the 'resources' folder of the project.
3. In the Index.razor where web designer is initialized, use **Shared ('Shared Property' in the on-line documentation)** property to enable shared data sources. The complete Index.razor code is as shown.

Index.razor

```
@page "/"
@inject IJSRuntime JSRuntime
<PageTitle>Index</PageTitle>
<link href="_content/@(typeof(ReportViewer).Assembly.GetName().Name)/jsViewer.min.css"
rel="stylesheet" />
<div id="designerContainer">
    <ReportDesigner @ref="_designer" RpxSettings="@_rpx" AppBarSettings="@_appBar"
DataSettings="@_data" PreviewSettings="@_preview" />
</div>
@code {
    private ReportDesigner _designer;
    private ReportViewer _viewer;
    private RpxSettings _rpx;
    private AppBarSettings _appBar;
    private DataSettings _data;
    private PreviewSettings _preview;
    public Index()
    {
        _rpx = new RpxSettings
        {
            Enabled = true
        };
        _appBar = new AppBarSettings
        {
            OpenButton = new OpenButton { Visible = true }
        };
        _data = new DataSettings
        {
            DataSets = new DataSets { CanModify = true },
```

```
            DataSources = new DataSources
            {
                CanModify = true,
                Shared = new SharedDataSourceOptions()
                {
                    Enabled = true
                }
            }
        };
        _preview = new PreviewSettings
        {
            OpenViewer = OpenViewer
        };
    }
    private async void OpenViewer(ViewerSettings settings)
    {
        if(_viewer != null)
        {
            await _viewer.SetTheme(settings.Theme);
            await _viewer.OpenReport(settings.DocumentInfo.Id);
            return;
        }
        _viewer = new ReportViewer();
        var initOptions = new InitializationOptions();
        initOptions.ReportID = settings.DocumentInfo.Id;
        initOptions.PanelsLocation = PanelsLocation.toolbar;
        initOptions.Theme = settings.Theme;
        initOptions.ReportLoaded = (reportInfo) =>
        {
        };
        await _viewer.Render(JSRuntime, settings.Element, initOptions);
    }
}
```

Implement the IReportStore

4. Create 'Implementation' folder.
5. To the 'Implementation' folder, add 'ReportStore.cs' class which will contain implementation for **IReportStore ('IReportStore Interface' in the on-line documentation)**.

ReportStore.cs

```
using GrapeCity.ActiveReports.Rendering.Tools;
using GrapeCity.ActiveReports.Web.Designer;
using GrapeCity.ActiveReports.Web.Viewer;
namespace BlazorDesignerServer.Implementation
{
    public class ReportStore : IReportStore
    {
        private static readonly string[] ReportExtensions =
```

```
        {
        ".rdl",
        ".rdlx",
        ".rdlx-master",
        ".rpx"
    };
        private readonly Dictionary<string, byte[]> _tempStorage = new Dictionary<string,
byte[]>();
        private readonly DirectoryInfo _rootDirectory;
        public ReportStore(DirectoryInfo rootDirectory)
        {
            _rootDirectory = rootDirectory;
        }
        public ReportDescriptor GetReportDescriptor(string reportId)
        {
            if (_tempStorage.ContainsKey(reportId))
                return new
ReportDescriptor(GetReportTypeByExtension(Path.GetExtension(reportId)));
            var fileInfo = new FileInfo(Path.Combine(_rootDirectory.FullName, reportId));
            return new ReportDescriptor(GetReportTypeByExtension(fileInfo.Extension));
        }
        public Stream LoadReport(string reportId)
        {
            if (_tempStorage.TryGetValue(reportId, out var tempReport))
                return new MemoryStream(tempReport);
            var file = new FileInfo(Path.Combine(_rootDirectory.FullName, reportId));
            //if (!file.Exists)
            //    throw new ReportNotFoundException();
            return file.OpenRead();
        }
        public string SaveReport(ReportType reportType, string reportId, Stream reportData,
SaveSettings settings = SaveSettings.None)
        {
            if ((settings & SaveSettings.IsTemporary) != 0)
            {
                var tempName = Guid.NewGuid() + GetReportExtension(reportType);
                _tempStorage.Add(tempName, reportData.ToArray());
                return tempName;
            }
            var reportFullPath = Path.Combine(_rootDirectory.FullName, reportId);
            using var fileStream = new FileStream(reportFullPath, FileMode.Create,
FileAccess.Write);
            reportData.CopyTo(fileStream);
            return reportId;
        }
        public string UpdateReport(ReportType reportType, string reportId, Stream reportData)
        {
            return SaveReport(reportType, reportId, reportData);
        }
        public ReportInfo[] ListReports()
        {
```

```
            var reports = _rootDirectory
                .EnumerateFiles("*.*")
                .Where(fileInfo => ReportExtensions.Any(ext =>
                    fileInfo.Extension.EndsWith(ext,
StringComparison.InvariantCultureIgnoreCase)))
                .Select(fileInfo => new ReportInfo()
                {
                    Id = fileInfo.Name,
                    Name = fileInfo.Name,
                    ReportType = GetReportTypeByExtension(fileInfo.Extension),
                }).ToArray();
            return reports;
        }
        private static ReportType GetReportTypeByExtension(string extension)
        {
            switch (extension)
            {
                case ".rdl":
                case ".rdlx":
                    return ReportType.RdlXml;
                case ".rdlx-master":
                    return ReportType.RdlMasterXml;
                case ".rpx":
                    return ReportType.RpxXml;
                default:
                    throw new ArgumentOutOfRangeException(nameof(extension), extension, null);
            }
        }
        private static string GetReportExtension(ReportType type)
        {
            return type switch
            {
                ReportType.RdlXml => ".rdlx",
                ReportType.RdlMasterXml => ".rdlx-master",
                ReportType.RpxXml => ".rpx",
                _ => throw new ArgumentOutOfRangeException(nameof(type), type, null)
            };
        }
        public void DeleteReport(string reportId)
        {
            if (_tempStorage.ContainsKey(reportId))
            {
                _tempStorage.Remove(reportId);
                return;
            }
            var file = new FileInfo(Path.Combine(_rootDirectory.FullName, reportId));
            if (file.Exists)
                file.Delete(); ;
        }
    }
```

```
}
```

Implement the IResourceRepositoryProvider

6. Add 'ResourceProvider.cs' to the 'Implementation' folder to add implementation for **IResourceRepositoryProvider ('IResourceRepositoryProvider Interface' in the on-line documentation)**.

ResourceProvider.cs

```
using GrapeCity.ActiveReports.Rendering.Tools;
using GrapeCity.ActiveReports.Web.Designer;
using GrapeCity.ActiveReports;
namespace BlazorDesignerServer.Implementation
{
    public class ResourceProvider : IResourceRepositoryProvider
    {
        private const string SharedDataSourceExtension = ".rdsx";
        private readonly DirectoryInfo _rootDirectory;
        public ResourceProvider(DirectoryInfo rootDirectory)
        {
            _rootDirectory = rootDirectory;
        }
        public Stream GetResource(ResourceInfo resource)
        {
            string absolutePath = Path.Combine(_rootDirectory.FullName, resource.Name);
            var file = new FileInfo(absolutePath);
            if (!file.Exists)
                return null;
            return file.OpenRead();
        }
        public ResourceDescriptor[] ListResources(ResourceType resourceType)
        {
            if (resourceType == ResourceType.SharedDataSource)
            {
                var sharedDataSources = _rootDirectory
                    .EnumerateFiles("*" + SharedDataSourceExtension).Select(fileInfo =>
                    {
                        using var stream = fileInfo.OpenRead();
                        var dataSource = DataSourceTools.LoadSharedDataSource(stream);
                        return new SharedDataSourceResourceDescriptor()
                        {
                            Id = fileInfo.Name,
                            Name = fileInfo.Name,
                            Type = dataSource.ConnectionProperties.DataProvider
                        };
                    }).ToArray();
                return sharedDataSources;
            }
            return Enumerable.Empty<ResourceDescriptor>().ToArray();
        }
```

```
        public ResourceDescriptor[] DescribeResources(ResourceInfo[] resources)
        {
            return Enumerable.Empty<ResourceDescriptor>().ToArray();
        }
    }
}
```

Configure and register services

7. Open Program.cs and update the file as shown below. The Program.cs file does the following:

    i. configures the services and middleware used by the application
    ii. registers the '**IReportStore ('IReportStore Interface' in the on-line documentation)**' and '**IResourceRepositoryProvider ('IResourceRepositoryProvider Interface' in the on-line documentation)**' as singleton services
    iii. adds reporting and designer services
    iv. sets the path to the ActiveReports.config file where the SQLite provider is added
    v. configures the reporting and designer middleware
    vi. serves static files

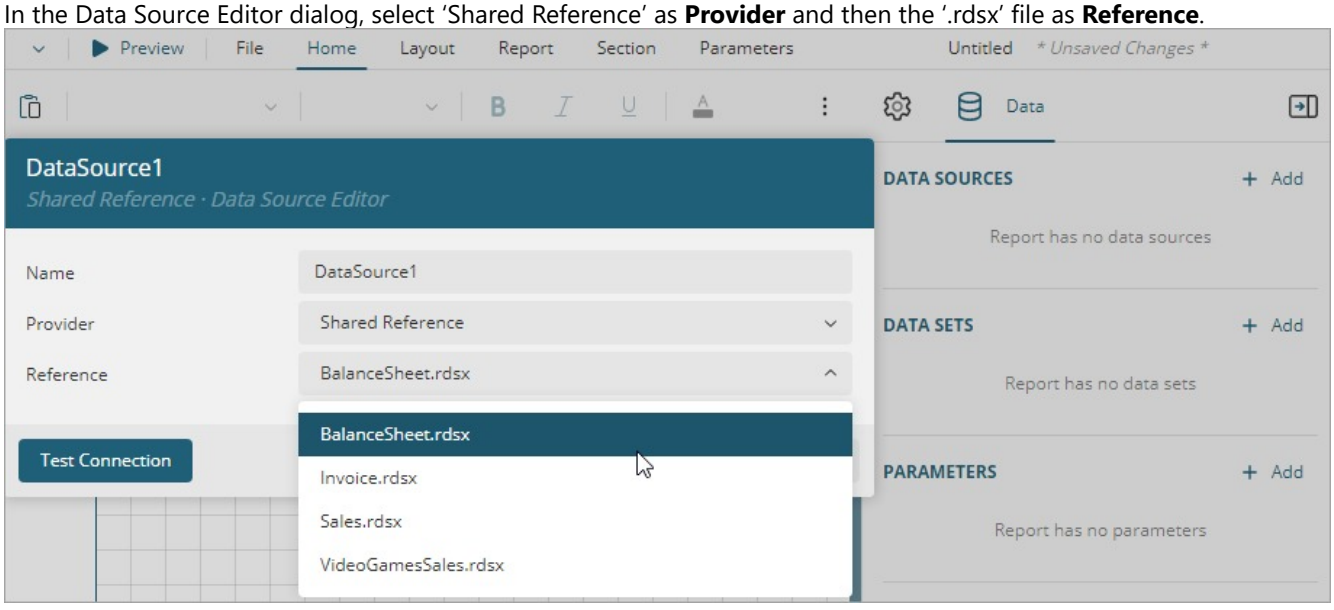
    The complete Program.cs is as shown.

    Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Designer;
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
using Microsoft.AspNetCore.SignalR;
using System.Text;
using GrapeCity.ActiveReports.Web.Designer;
using BlazorDesignerServer.Implementation;
var builder = WebApplication.CreateBuilder(args);
var ResourcesRootDirectory =
    new DirectoryInfo(Path.Combine(Directory.GetCurrentDirectory(), "resources"));
Encoding.RegisterProvider(CodePagesEncodingProvider.Instance);
// Add services to the container.
builder.Services.AddReportViewer();
builder.Services.AddReportDesigner();
builder.Services.AddSingleton<IReportStore>(new ReportStore(ResourcesRootDirectory));
builder.Services.AddSingleton<IResourceRepositoryProvider>(new
ResourceProvider(ResourcesRootDirectory));
builder.Services.AddRazorPages().AddJsonOptions(options =>
options.JsonSerializerOptions.PropertyNamingPolicy = null);
builder.Services.AddServerSideBlazor();
builder.Services.Configure<HubOptions>(options =>
{
    options.MaximumReceiveMessageSize = 524288000; //500MB
});
builder.Services.AddCors();
var app = builder.Build();
// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
```

```
}
// For use as a server for BlazorWebAssembly
app.UseCors(cors => cors.SetIsOriginAllowed(origin => new Uri(origin).Host ==
"localhost")
    .AllowAnyMethod()
    .AllowAnyHeader()
    .AllowCredentials()
    .WithExposedHeaders("Content-Disposition"));
var reportStore = app.Services.GetService<IReportStore>();
var resourceProvider = app.Services.GetService<IResourceRepositoryProvider>();
app.UseReportDesigner(config =>
{
    config.UseReportsProvider(reportStore);
    config.UseResourcesProvider(resourceProvider);
});
app.UseStaticFiles();
app.UseRouting();
app.MapControllers();
app.MapBlazorHub();
app.MapFallbackToPage("/_Host");
app.Run();
```

8. Run the application.
9. Go to the **Data** tab to add the data source.
10. In the Data Source Editor dialog, select 'Shared Reference' as **Provider** and then the '.rdsx' file as **Reference**.



Configure Preview

This topic describes how to use the Blazor Viewer inside the Blazor Designer.

1. In Microsoft Visual Studio 2022, open your Blazor Designer Application project.
2. Add the following NuGet packages for the Blazor Viewer.

MESCIUS.ActiveReports.Aspnetcore.Viewer (for Blazor Server only)
MESCIUS.ActiveReports.Blazor.Viewer

3. [For Blazor Server only] Update Program.cs as demonstrated in the code example below.

Program.cs

```
using GrapeCity.ActiveReports.Aspnetcore.Designer;
using GrapeCity.ActiveReports.Aspnetcore.Viewer;
var resourcesRootDirectory = new DirectoryInfo(".\\resources\\");
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddRazorPages();
builder.Services.AddServerSideBlazor();
builder.Services
     .AddReportViewer()
     .AddReportDesigner()
     .AddMvc(options => options.EnableEndpointRouting = false)
     .AddJsonOptions(options => options.JsonSerializerOptions.PropertyNamingPolicy =
null);
var app = builder.Build();
if (!app.Environment.IsDevelopment())
{
    // The default HSTS value is 30 days. You may want to change this for production
scenarios, see https://aka.ms/aspnetcore-hsts.
    app.UseHsts();
}
app.UseHttpsRedirection();
app.UseReportViewer(config => config.UseFileStore(resourcesRootDirectory));
app.UseReportDesigner(config => config.UseFileStore(resourcesRootDirectory, false));
app.UseStaticFiles();
app.UseRouting();
app.MapBlazorHub();
app.MapFallbackToPage("/_Host");
app.Run();
```

4. Update 'Pages/Index.razor' as demonstrated in the code example below.

Pages/Index.razor

```
@page "/"
@using GrapeCity.ActiveReports.Blazor.Designer;
@using GrapeCity.ActiveReports.Blazor.Viewer
@inject IJSRuntime JSRuntime
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css"
rel="stylesheet" />
<div style="height:100vh;width:100%">
     <ReportDesigner @ref="_designer" Document="@_document" Preview="@_preview" />
</div>
@code {
    private ReportDesigner _designer;
    private ReportViewer _viewer;
    private Document _document = new Document() { Id = "report.rdlx", Type =
```

```
SupportedDocumentType.cpl };
    private Preview _preview;
    public Index()
    {
        _preview = new Preview()
            {
                CanPreview = true,
                OpenViewer = OpenViewer
            };
    }
    private async void OpenViewer(ViewerSettings settings)
    {
        if (_viewer != null)
        {
            await _viewer.OpenReport(settings.DocumentInfo.Id);
            return;
        }
        _viewer = new ReportViewer();
        var initOptions = new InitializationOptions();
        initOptions.ReportID = settings.DocumentInfo.Id;
        initOptions.PanelsLocation = PanelsLocation.toolbar;
        await _viewer.Render(JSRuntime, settings.Element, initOptions);
    }
}
```

5. Build and run the application.

Blazor WebDesigner API

ActiveReports provides a rich API for integrating the Blazor web designer components into your web application. To embed the Blazor WebDesigner component in your project, use the Blazor WebDesigner API. It lets you create, design, and save reports with added capabilities that include defining the locale for the designer, customizing the default settings of the report items, managing the Data and Properties tab, modifying the application info, and much more.

GlobalDesignerAPI

Type of **GrapeCity.ActiveReports.Blazor.Designer.ReportDesigner** object exported by the **GrapeCity.ActiveReports.Blazor.Designer ('GrapeCity.ActiveReports.Blazor.Designer Namespace' in the on-line documentation)** library.

ReportDesigner

Renders the **ReportDesigner** component to the ** ** element. Use the following example for "***.razor**" pages.

Parameter (Type):

PreviewSettings: **PreviewSettings**

Return: Renders the **ReportDesigner** component for the user interface.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

AppBarSettings

Gets or sets the settings for the application bar component.

Parameter (Type):

AppBarSettings: AppBarSettings

- **OpenButton**: OpenButton

  Description: Gets or sets the **OpenButton** button settings.
  Return: Returns the **OpenButton** object representing the Open button settings.

- **AboutButton:** AboutButton

  Description: Gets or sets the **AboutButton** button settings.
  Return: Returns the **AboutButton** object representing the About button settings.

- **ContextActionsTab**: ContextActionsTab

  Description: Gets or sets the Context Actions tab settings.
  Return: Returns the **ContextActionsTab** object representing the Context Actions tab settings.

- **HomeTab**: HomeTab

  Description: Gets or sets the Home tab settings.
  Return: Returns the **HomeTab** object representing the Home tab settings.

- **InsertTab**: InsertTab

  Description: Gets or sets the Insert tab settings.
  Return: Returns the **InsertTab** object representing the Insert tab settings.

- **ParametersTab**: ParametersTab

  Description: Gets or sets the Parameters tab settings.
  Return: Returns the **ParametersTab** object representing the Parameters tab settings.

- **SaveAsButton**: SaveAsButton

  Description: Gets or sets the SaveAs button settings.
  Return: Returns the **SaveAsButton** object representing the SaveAs button settings.

- **SaveButton**: SaveButton

  Description: Gets or sets the Save button settings.
  Return: Returns the **SaveButton** object representing the Save button settings.

- **ScriptTab**: ScriptTab

  Description: Gets or sets the Script tab settings.
  Return: Returns the **ScriptTab** object representing the Script tab settings.

- **Visible**: bool?

  Description: Gets or sets whether to show the Application bar.
  Return: Returns a value representing whether the application bar should be displayed.

Return: Returns an **AppBarSettings** object containing the settings for the Application bar.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner AppBarSettings="@_appBar" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private AppBarSettings _appBar = new AppBarSettings
    {
        OpenButton = new OpenButton { Visible = true },
        AboutButton = new AboutButton { Visible = false },
        ContextActionsTab = new ContextActionsTab { Visible = true },
        HomeTab = new HomeTab { Visible = true },
        InsertTab = new InsertTab { Visible = false },
        ParametersTab = new ParametersTab { Visible = true },
        SaveAsButton = new SaveAsButton { Visible = false },
        SaveButton = new SaveButton { Visible = false },
        ScriptTab = new ScriptTab { Visible = false },
        Visible = false
    };
    public Index()
    {
```

```
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

App

Gets the application API.

Parameter (Type):

App: GrapeCity.ActiveReports.Blazor.Designer.App

- **Panels**: Panels

  Description: Contains access to the Menu and Sidebar panels.
  Return: Returns the **Panels** object representing the menu and sidebar panels.

- **Focus()**: Task

  Description: Returns focus to the Designer. Focus may be lost when plug-in components are opened/closed. Returning focus is essential to continue using Designer hotkeys like Ctrl+Z (undo), Ctrl+Y (redo), etc.
  Return: The **Task** object represents an asynchronous operation.

- **GetAbout()**: ValueTask<About>

  Description: Returns documentation links and application information.
  Return: The **ValueTask<About>** object representing references to documentation and information about the application.

Return: Returns the application **API**.

Sample Code

```
@page "/"
@inject IJSRuntime JSRuntime
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" PreviewSettings="@_preview" />
</div>
@code {
    private ReportDesigner _designer;
    private PreviewSettings _preview;
    private ReportViewer _viewer;
    public Index()
    {
        _preview = new PreviewSettings
```

```
        {
            CanPreview = true,
            OpenViewer = OpenViewer
        };
    }
    private async void OpenViewer(ViewerSettings settings)
    {
        if (_viewer != null)
        {
            await _viewer.OpenReport(settings.DocumentInfo.Id);
            return;
        }
        _viewer = new ReportViewer();
        GrapeCity.ActiveReports.Blazor.Designer.App api = _designer.App;
        var initOptions = new InitializationOptions();
        initOptions.ReportID = settings.DocumentInfo.Id;
        initOptions.PanelsLocation = PanelsLocation.toolbar;
        initOptions.ReportLoaded = (reportInfo) => { };
        await _viewer.Render(JSRuntime, settings.Element, initOptions);
    }
}
```

DataSettings

Sets the data settings for the report designer component.

Parameter (Type):

DataSettings: DataSettings

- **CommonValues**: CommonValues

  Description: Gets or sets the Common Values' section settings.
  Return: Returns the Common Values' section settings.

- **DataSets**: DataSets

  Description: Gets or sets the Data Sets' section settings.
  Return: Returns the Data Sets' section settings.

- **DataSources**: DataSources

  Description: Gets or sets the Data Sources' section settings.
  Return: Returns the Data Sources' section settings.

- **DataTab**: DataTab

  Description: Gets or sets the Data Tab's section settings.
  Return: Returns the Data Tab's section settings.

- **Parameters**: Parameters

Description: Gets or sets the Parameters' section settings.
Return: Returns the Parameters' section settings.

Return: Returns a **DataSettings** object containing the settings for the data.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner DataSettings="@_dataSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private DataSettings _dataSettings = new DataSettings
    {
        DataSets = new DataSets { CanModify = true },
        DataSources = new DataSources { CanModify = true },
        DataSources = new DataSources { CanModify = true,
             Shared = new SharedDataSourceOptions()
             {
                Enabled = true
             }
        },
        DataTab = new DataTab { Visible = true },
        Parameters = new Parameters() { CanModify = true },
        CommonValues = new CommonValues() { Visible = false }
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

DesignerInitialized

Gets or sets the callback that is invoked when a Designer is initialized.

Parameter (Type):

DesignerInitialized: Action

Return: Returns an **Action** object containing the action that is called after the initialization of the designer.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner DesignerInitialized="@DesignerInitializedCallback"
PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private void DesignerInitializedCallback() { }
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

DisableFocusTimer

Disables the focus timer. By default, focused elements (like buttons) are highlighted only for a short period of time after the Tab key is pressed. This setting makes focused elements permanently highlighted.

Parameter (Type):

DisableFocusTimer: Boolean

Return: Returns the current state of an item's focus.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner DisableFocusTimer="true" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

DisableSystemClipboard

Disables the usage of the system clipboard.

Parameter (Type):

DisableSystemClipboard: Boolean

Return: Returns the current state of disabling the use of the system clipboard.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner DisableSystemClipboard="true" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

Document

Opens the specified document.

Parameter (Type):

Document: Document

- **Type**: SupportedDocumentType

  Description: Gets or sets the supported report type. The available values are `SupportedDocumentType.fpl`, `SupportedDocumentType.rpx`, `SupportedDocumentType.dashboard`, `SupportedDocumentType.msl`, and `SupportedDocumentType.cpl`.
  Return: Returns the supported report type.

- **Id**: String

  Description: Gets or sets the report identifier.
  Return: Returns the report identifier.

Return: Returns the specified document.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner Document="@_doc" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private Document _doc = new Document()
    {
        Type = SupportedDocumentType.cpl,
        Id = "Example.rdlx"
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

Documents

Gets the document API.

Parameter (Type):

Documents: Documents

- **Create(CreateDocumentOptions** options**)**: ValueTask<CreateDocumentInfo>

  Description: Creates a new report to be edited in a designer using the specified **CreateReportOptions** object.
  Return: Returns the **ValueTask<CreateDocumentInfo>** object representing the information about the created document.

- **HasUnsavedChanges()**: ValueTask<bool>

  Description: Indicates whether the report has unsaved changes.
  Return: Returns the **ValueTask<bool>** representing the presence of unsaved changes.

- **Info()**: ValueTask<CurrentDocumentInfo>

  Description: Returns information about the currently edited report.
  Return: Returns the **ValueTask<CurrentDocumentInfo>** object representing the document information.

- **IsNew()**: ValueTask<bool>

Description: Indicates whether report was saved before at least once.
Return: Returns the **ValueTask<bool>** object representing whether the report was saved at least once.

- **Open()**: Task

  Description: Shows open report dialog.
  Return: The **Task** object represents an asynchronous operation.

- **OpenById**(**string** id, **SupportedDocumentType** type, **string** name = null, **string** content = null): **ValueTask<OpenDocumentInfo>**

  Description: Opens an existing report to be edited in the Designer with the specified id. Optionally, you can pass the name and the content, else, the report is loaded from the server.
  Return: The **ValueTask<OpenDocumentInfo>**object representing information about the opened document.

- **Save()**: Task

  Description: Saves the report currently edited in Designer. If the report is new, then the "Save As" dialog is opened.
  Return: The **Task** object representing an asynchronous operation.

- **SaveAs()**: Task

  Description: Opens 'Save As' dialog.
  Return: The **Task** object represents an asynchronous operation.

- **SaveById**(**string** id**, string** name = null) : ValueTask<SaveDocumentInfo>

  Description: Saves the report currently edited in Designer using the specified **id**.
  Return: The **ValueTask<SaveDocumentInfo>** object representing a saved document information.

- **SaveByName**(**string** name): ValueTask<SaveDocumentInfo>

  Description: Saves the report currently edited in Designer using the specified **name**.
  Return: The **ValueTask<SaveDocumentInfo>** object representing a saved document information.

Return: Returns the document **API**.

Sample Code

```
@page "/"
@inject IJSRuntime JSRuntime
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" Document="@_doc" PreviewSettings="@_preview" />
</div>
@code {
    private ReportDesigner _designer;
    private PreviewSettings _preview;
    private ReportViewer _viewer;
    private Document _doc = new Document
```

```
    {
        Id = "Example.rdlx",
        Type = SupportedDocumentType.cpl
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = true,
            OpenViewer = OpenViewer
        };
    }
    private async void OpenViewer(ViewerSettings settings)
    {
        if(_viewer != null)
        {
            await _viewer.OpenReport(settings.DocumentInfo.Id);
            return;
        }
        Documents api = _designer.Documents;
        _viewer = new ReportViewer();
        var initOptions = new InitializationOptions();
        initOptions.ReportID = settings.DocumentInfo.Id;
        initOptions.PanelsLocation = PanelsLocation.toolbar;
        initOptions.ReportLoaded = (reportInfo) => { };
        await _viewer.Render(JSRuntime, settings.Element, initOptions);
    }
}
```

DocumentsSettings

Gets or sets the API settings for documents.

Parameter (Type):

DocumentsSettings: DocumentsSettings

- **FileView**: FileView

  Description: Gets or sets the file view settings.
  Return: Returns the current value of the file view settings.

- **Handlers**: Handlers

  Description: Gets or sets the handlers.
  Return: Returns the current value of the handlers.

  - **OnDocumentChanged**: A handler that is triggered when report content is changed.
    This function takes an object as an argument that contains two properties:
    - **document**: The document property contains the updated version of the document that was

changed
- **hasUnsavedChanges**: The hasUnsavedChanges is a boolean value that indicates whether there are any unsaved changes in the document.

Return: Returns the specified settings for documents **API**.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner DocumentsSettings="@_documentsSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private DocumentsSettings _documentsSettings = new DocumentsSettings
    {
        FileView = new FileView { Visible = true },
        Handlers = new Handlers()
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

Sample Code for OnDocumentChanged handler

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner DocumentsSettings="_documentsSettings"/>
</div>
@code {
    private DocumentsSettings _documentsSettings;
    public Index()
    {
        _documentsSettings = new DocumentsSettings()
        {
            Handlers = new Handlers()
            {
                OnDocumentChanged = (options) =>
                {
                    ...
                }
```

```
            }
        };
    }
}
```

EditorSettings

Gets or sets the editor settings.

Parameter (Type):

EditorSettings: EditorSettings

- **ShowGrid**: bool?

  Description: Specifies if the Grid must be shown or hidden by default.
  Return: Returns the current value of the grid display.

- **SnapToGrid**: bool?

  Description: Specifies the default value for the snapToGrid option.
  Return: Returns the current snapToGrid option value.

- **SnapToGuides**: bool?

  Description: Specifies the default value for the snapToGuides option.
  Return: Returns the current snapToGuides option value.

- **GridSize**: String

  Description: Specifies the default Grid Size. If units = 'cm', value = 0.5cm by default. Else, value = 0.25in by default.
  Return: Returns the current Grid Size.

- **Rulers**: Rulers

  Description: Gets or sets the Ruler's settings.
  Return: Returns the current Ruler's settings.

Return: Returns the set editor settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner EditorSettings="@_editorSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private EditorSettings _editorSettings = new EditorSettings
    {
```

```
        ShowGrid = true,
        SnapToGrid = true,
        SnapToGuides = true,
        Rulers = new Rulers(),
        GridSize = "0.5cm"
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

Fonts

Gets or sets the list of allowed fonts for controls.

Parameter (Type):

Fonts: Object[]

Return: Returns the set list of allowed fonts for controls.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner Fonts="@_fonts" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private string[] _fonts = new string[1] { "Arial" };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

Height

Gets or sets the height of the designer.

Parameter (Type):

Height: String

Return: Returns the height of the designer.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner Height="20%" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

ImageMimeTypes

Gets or sets the list of allowed image mime types for controls.

Parameter (Type):

ImageMimeTypes: String[]

Return: Returns the set list of allowed image mime types for controls.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner ImageMimeTypes="@_imageMimeTypes" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private string[] _imageMimeTypes = new string[3] { "image/gif", "image/jpeg", "image/png"
};
    public Index()
    {
        _preview = new PreviewSettings
```

```
        {
            CanPreview = false
        };
    }
}
```

Language

Gets or sets the language of the report designer. Use 'en-US' (for English), 'ja-JP' (for Japanese), and 'zh-CN' (for simplified Chinese).

Parameter (Type):

Language: String

Return: Returns the specified language.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner Language="ja-JP" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

LocalizationResources

Gets or sets a custom resource localization.

Parameter (Type):

LocalizationResources: LocalizationResources[]

- **Language**: String

  Description: Gets or sets the Language.
  Return: Returns the specified language.

- **Resources**: String

Description: Gets or sets the localization resources in json array.
Return: Returns the specified localization resources represented in the json array.

Return: Returns an array of custom localization resources.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner LocalizationResources="@_localizationResources"
PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private LocalizationResources[] _localizationResources =
    {
        new LocalizationResources()
        {
            Language = "en",
            Resources = "[ { \"ns\": \"common\", \"lng\": \"en\", \"resources\": { \"units\":
{ \"cm\": { \"textShortName\": \"CustomName_cm\", \"textFullName\": \"CustomName_Centimeters\"
} } } } ]"
        }
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

LockLayout

Manages the interaction with the Layout. When LockLayout is enabled, it is only possible to modify the properties of existing report items, i.e., operations such as adding a new report item or deleting an existing one, and modifying the report layout structure is not possible.

Parameter (Type):

LockLayout: Boolean

Return: Returns the current state of the lock layout.

Sample Code

```
@page "/"
```

```
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner LockLayout="true" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

MenuSettings

Get or sets the menu settings.

Parameter (Type):

MenuSettings: MenuSettings

- **DocumentExplorer**: DocumentExplorer

  Description: Document/Report Explorer settings.
  Return: Returns an object representing the Document/Report Explorer settings.

- **GroupEditor**: GroupEditor

  Description: Group Editor settings.
  Return: Returns an object representing the Group Editor settings.

- **LayerEditor**: LayerEditor

  Description: Layer Editor settings.
  Return: Returns an object representing the Layer Editor settings.

- **Logo**: Logo

  Description: Logo settings.
  Return: Returns an object representing the Logo settings.

- **ToolBox**: ToolBox

  Description: ToolBox settings.
  Return: Returns an object representing the ToolBox settings.

- **Visible**: bool?

  Description: Specifies whether Main Menu needs to be shown.

Return: Returns a value representing the current visibility state.

Return: Returns the current settings of the menu.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner MenuSettings="@_menuSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private MenuSettings _menuSettings = new MenuSettings
    {
        DocumentExplorer = new DocumentExplorer { Visible = true },
        GroupEditor = new GroupEditor { Visible = true },
        LayerEditor = new LayerEditor { Visible = true },
        Logo = new Logo { Visible = true },
        ToolBox = new ToolBox { Visible = true },
        Visible = true
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

Notifications

Gets the notifications API.

Parameter (Type):

Notifications: Notifications

- **Send()**: Task

  Description: Sends a notification of specified level, caption, and content.
  Return: The **Task** object representing an asynchronous operation.

- **Warning()**: Task

  Description: Sends a warning notification.
  Return: The **Task** object representing an asynchronous operation.

- **Info()**: Task

Description: Sends a general notification. It can be used to notify when any user-initiated action is completed.
Return: The **Task** object representing an asynchronous operation.

- **Error()**: Task

  Description: Sends an error notification.
  Return: The **Task** object representing an asynchronous operation.

- **DismissAll()**: Task

  Description: Dismisses all notifications.
  Return: The **Task** object representing an asynchronous operation.

Return: Returns the notifications **API**.

Sample Code

```
@page "/"
@inject IJSRuntime JSRuntime
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" PreviewSettings="@_preview" />
</div>
@code {
    private ReportDesigner _designer;
    private PreviewSettings _preview;
    private ReportViewer _viewer;
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = true,
            OpenViewer = OpenViewer
        };
    }
    private async void OpenViewer(ViewerSettings settings)
    {
        if (_viewer != null)
        {
            await _viewer.OpenReport(settings.DocumentInfo.Id);
            return;
        }
        Notifications api = _designer.Notifications;
        _viewer = new ReportViewer();
        var initOptions = new InitializationOptions();
        initOptions.ReportID = settings.DocumentInfo.Id;
        initOptions.PanelsLocation = PanelsLocation.toolbar;
```

```
        initOptions.ReportLoaded = (reportInfo) => { };
        await _viewer.Render(JSRuntime, settings.Element, initOptions);
    }
}
```

PreviewSettings

Gets or sets the preview settings.

Parameter (Type):

PreviewSettings: PreviewSettings

- **CanPreview()**: Boolean

  Description: Sets or gets whether to show the **Preview** button.

- **OpenViewer()**: Action<ViewerSettings>

  Description: You can plug-in **Viewer component** by providing OpenViewer function.

Return: Returns the preview settings.

Sample Code

```
@page "/"
@inject IJSRuntime JSRuntime
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" PreviewSettings="@_preview" />
</div>
@code {
    private ReportDesigner _designer;
    private PreviewSettings _preview;
    private ReportViewer _viewer;
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = true,
             OpenViewer = OpenViewer
         };
    }
    private async void OpenViewer(ViewerSettings settings)
    {
        if (_viewer != null)
        {
            await _viewer.OpenReport(settings.DocumentInfo.Id);
            return;
```

```
        }
        _viewer = new ReportViewer();
        var initOptions = new InitializationOptions();
        initOptions.ReportID = settings.DocumentInfo.Id;
        initOptions.PanelsLocation = PanelsLocation.toolbar;
        initOptions.ReportLoaded = (reportInfo) => { };
        await _viewer.Render(JSRuntime, settings.Element, initOptions);
    }
}
```

PropertyGridSettings

Gets or sets the property grid settings.

Parameter (Type):

PropertyGridSettings: PropertyGridSettings

- **PropertiesTab**: PropertiesTab

  Description: Sets or gets properties tab settings.
  Return: The **PropertiesTab** object representing the current settings of properties tab.

- **Mode()**: Mode?

  Description: Specifies default properties mode. The available values are `Mode.Basic` and `Courier New Mode.Advanced`.
  Return: The current mode of the properties tab settings.

- **Sort()**: Sort?

  Description: Specifies default properties sort mode. The available values are `Sort.categorized` and `Sort.alphabetical`.
  Return: The current sort of the properties tab settings

Return: The **PropertyGridSettings** representing the current settings for the property grid.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner PropertyGridSettings="@_propertyGridSettings" PreviewSettings="@_preview"
/>
</div>
@code {
    private PreviewSettings _preview;
    private PropertyGridSettings _propertyGridSettings = new PropertyGridSettings
    {
        Mode = Mode.Advanced,
```

```
        Sort = Sort.categorized,
        PropertiesTab = new PropertiesTab { Visible = false }
    };
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

RdlxSettings

Gets or sets the RDLX platform-specific settings.

Parameter (Type):

RdlxSettings: RdlxSettings

- **ExpressionSyntax**: ExpressionSyntax?

  Description: Gets or sets the expression syntax used in Designer. The available values are `ExpressionSyntax.i11n` and `ExpressionSyntax.rdl`.
  Return: The object representing the current expression syntax settings.

- **ToolBoxContent**: ToolBoxContent

  Description: Gets or sets the report items available and their order.
  Return: The **ToolBoxContent** object representing the available report elements with the current settings and their order.

- **InitTemplates**: InitTemplates

  Description: Gets or sets the reports as rdlx-json strings. Report items from these reports are used as templates for creating new items.
  Return: The **InitTemplates** object representing the current templates settings.

- **ReportItemsFeatures**: RdlxReportItemsSettings

  Description: Gets or sets the customizable report item features.
  Return: The **RdlxReportItemsSettings** object representing the current customizable report item features.

- **ReportStyles**: ReportStyles[]

  Description: Gets or sets the additional styles to add to report item styles.
  Return: The array of **ReportStyles** objects representing the current additional styles to add to the report element styles.

- **Msl**: Msl

  Description: Gets or sets the RDLX Multi-Section report settings.
  Return: The **Msl** object representing the current RDLX Multi-Section report settings.

- **Dashboard**: Dashboard

  Description: Gets or sets the RDLX Dashboard reports settings.
  Return: The **Dashboard** object representing the current RDLX Dashboard report settings.

Return: The **RdlxSettings** object representing the current rdlx report settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner RdlxSettings="@_rdlxSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private RdlxSettings _rdlxSettings = new RdlxSettings
    {
        Dashboard = new Dashboard { Enabled = true },
        Msl = new Msl { Enabled = true },
        ExpressionSyntax = ExpressionSyntax.rdl,
        InitTemplates = new InitTemplates(),
        ReportItemsFeatures = new RdlxReportItemsSettings(),
        ReportStyles = { },
        ToolBoxContent = new ToolBoxContent { Cpl = new[] { RdlxToolBoxItem.formattedtext } }
    };
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

RpxSettings

Gets or sets the RPX platform-specific settings. To set these settings, an RPX report must exist.

Parameter (Type):

RpxSettings: RpxSettings

- **Enabled**: bool?

  Description: Gets or sets the activation value of RPX reports.
  Return: The object represents the current activation value of RPX reports.

- **InitTemplates**: InitTemplates

Description: Gets or sets the reports as json strings. Report items from these reports are used as templates for creating new items.
Return: The **InitTemplates** object representing the current templates settings.

Return: The **RpxSettings** object representing the current rpx reports settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner RpxSettings="@_rpxSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private RpxSettings _rpxSettings = new RpxSettings
    {
        Enabled = true,
        InitTemplates = new InitTemplates()
    };
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

ServerSettings

Gets or sets the back end-related settings.

Parameter (Type):

ServerSettings: ServerSettings

- **Url**: String

  Description: Gets or sets the base URL for Designer Server API.
  Return: The string representing the current base URL for Designer Server API.

- **OnBeforeRequest**: Func<RequestInit, RequestInit>

  Description: Gets or sets the handler to modify requests.
  Return: The **Func<RequestInit, RequestInit>** object representing the current handler to modify requests.

Return: The **ServerSettings** object representing the current back end-related settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner ServerSettings="@_serverSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private ServerSettings _serverSettings = new ServerSettings
    {
        OnBeforeRequest = delegate (RequestInit requestInit)
        {
            return requestInit;
        },
        Url = "http://localhost:5098/"
    };
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

StatusBarSettings

Gets or sets the status bar settings.

Parameter (Type):

StatusBarSettings: StatusBarSettings

- **ToggleUnitsButton**: ToggleUnitsButton

  Description: Gets or sets the toggle units button settings.
  Return: The **ToggleUnitsButton** object representing the current toggle units button settings.

- **GridSizeEditor**: GridSizeEditor

  Description: Gets or sets the grid size editor settings.
  Return: The **GridSizeEditor** object representing the current grid size editor settings.

- **RulersButton**: RulersButton

  Description: Gets or sets the rulers button settings.
  Return: The **RulersButton** object representing the current rulers button settings.

- **Visible**: bool?

Description: Gets or sets the visibility of the status bar.
Return: The value representing the current visibility of the status bar settings.

Return: The **StatusBarSettings** object represented the current status bar settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner StatusBarSettings="@_statusBarSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private StatusBarSettings _statusBarSettings = new StatusBarSettings
    {
        GridSizeEditor = new GridSizeEditor { Visible = true },
        PropertiesModeButton = new PropertiesModeButton { Visible = true },
        RulersButton = new RulersButton { Visible = true },
        ToggleGridButton = new ToggleGridButton { Visible = true },
        ToggleUnitsButton = new ToggleUnitsButton { Visible = true },
        Visible = true
    };
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

StoreUnsavedReport

Manages the report retention. When StoreUnsavedReport is enabled, the last unsaved report can be restored if a browser tab or the browser itself gets accidentally closed.

Parameter (Type):

StoreUnsavedReport: Bool

Return: The value representing the current state of the **StoreUnsavedReport**.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
```

```
<div id="designerContainer">
    <ReportDesigner StoreUnsavedReport="false" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

StoreUserPreferences

Manages the user preferences. When **StoreUserPreferences** is enabled, user preferences are saved to a browser storage.

Parameter (Type):

StoreUserPreferences: Bool

Return: The value representing the current state of the **StoreUserPreferences**.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner StoreUserPreferences="false" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

StylesSettings

Gets or sets the RPX Styles-related settings.

Parameter (Type):

StylesSettings: StylesSettings

- **StylesTab**: StylesTab

  Description: Gets or sets the styles tab settings.
  Return: The **StylesTab** object representing the current styles tab settings.

- **Stylesheet**: Stylesheet

  Description: Gets or sets the Stylesheet settings.
  Return: The **Stylesheet** object representing the current Stylesheet settings.

Return: The StylesSettings object representing the current RPX styles-related settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner StylesSettings="@_styleSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private StylesSettings _styleSettings = new StylesSettings
    {
        Stylesheet = new Stylesheet { CanModify = true },
        StylesTab = new StylesTab { Visible = true }
    };
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

TitleSettings

Gets or sets the document title settings.

Parameter (Type):

TitleSettings: TitleSettings

- **Disabled**: bool?

  Description: Gets or sets whether the document title is disabled.
  Return: The value representing whether the document title is disabled.

Return: The **TitleSettings** object representing the current document title settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner TitleSettings="@_titleSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    private TitleSettings _titleSettings = new TitleSettings
    {
        Disabled = false
    };
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

ToolBarSettings

Gets or sets the ToolBar settings.

Parameter (Type):

ToolBarSettings: ToolBarSettings

- **Visible**: bool?

  Description: Gets or sets the visibility of ToolBar.
  Return: The value represented the visibility of ToolBar.

Return: The **ToolBarSettings** object represented the current ToolBar settings.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner ToolBarSettings="@_toolbarSettings" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
```

```
    private ToolBarSettings _toolbarSettings = new ToolBarSettings
    {
        Visible = true
    };
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

Units

Gets or sets the default measurement units of the report designer. The available values are `Units.In` and `Units.Cm`

Parameter (Type):

Units: Units

Return: The current value of default measurement units of the report designer.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner Units="Units.Cm" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

Width

Gets or sets the width of the designer.

Parameter (Type):

Width: String

Return: Returns the width of the designer.

Sample Code

```
@page "/"
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<div id="designerContainer">
    <ReportDesigner Width="20%" PreviewSettings="@_preview" />
</div>
@code {
    private PreviewSettings _preview;
    public Index()
    {
         _preview = new PreviewSettings
         {
             CanPreview = false
         };
    }
}
```

OpenDocumentExplorer

Manages the panels of the Designer API. To manage panels from the **Blazor Designer API**, use the following example for "***.razor**" pages.

In this example, the explorer panel is called for a report using the Blazor Designer API.

- Add the call button code.

```
<button @onclick="OpenDocumentExplorer">Open document explorer</button>
```

- Add the button click handler code.

```
private async void OpenDocumentExplorer()
{
    await _designer.App.Panels.Menu.Open("document-explorer");
}
```

The complete code is as follows:

Sample Code

```
@page "/"
@inject IJSRuntime JSRuntime
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<button @onclick="OpenDocumentExplorer">Open document explorer</button>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" PreviewSettings="@_preview" />
```

```
</div>
@code {
    private ReportDesigner _designer;
    private PreviewSettings _preview;
    private ReportViewer _viewer;
    private async void OpenDocumentExplorer()
    {
        await _designer.App.Panels.Menu.Open("document-explorer");
    }
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

UndoLastOperation

Manages the clipboard and state selectors of the application API. To manage the clipboard and state selectors of the application API, use the following example for "***.razor**" pages.

This example implements undoing the last operation in the designer using the custom button.

- Add the call button code.

```
<button @onclick="UndoLastOperation">Undo the last operation</button>
```

- Add the button click handler code.

```
private async void UndoLastOperation()
{
    await _designer.App.Editor.Undo();
}
```

The complete code is as follows:

Sample Code

```
@page "/"
@inject IJSRuntime JSRuntime
<link href="_content/GrapeCity.ActiveReports.Blazor.Viewer/jsViewer.min.css" rel="stylesheet"
/>
<button @onclick="UndoLastOperation">Undo the last operation</button>
<div id="designerContainer">
    <ReportDesigner @ref="_designer" PreviewSettings="@_preview" />
</div>
@code {
    private ReportDesigner _designer;
```

```
    private PreviewSettings _preview;
    private ReportViewer _viewer;
    private async void UndoLastOperation()
    {
        await _designer.App.Editor.Undo();
    }
    public Index()
    {
        _preview = new PreviewSettings
        {
            CanPreview = false
        };
    }
}
```

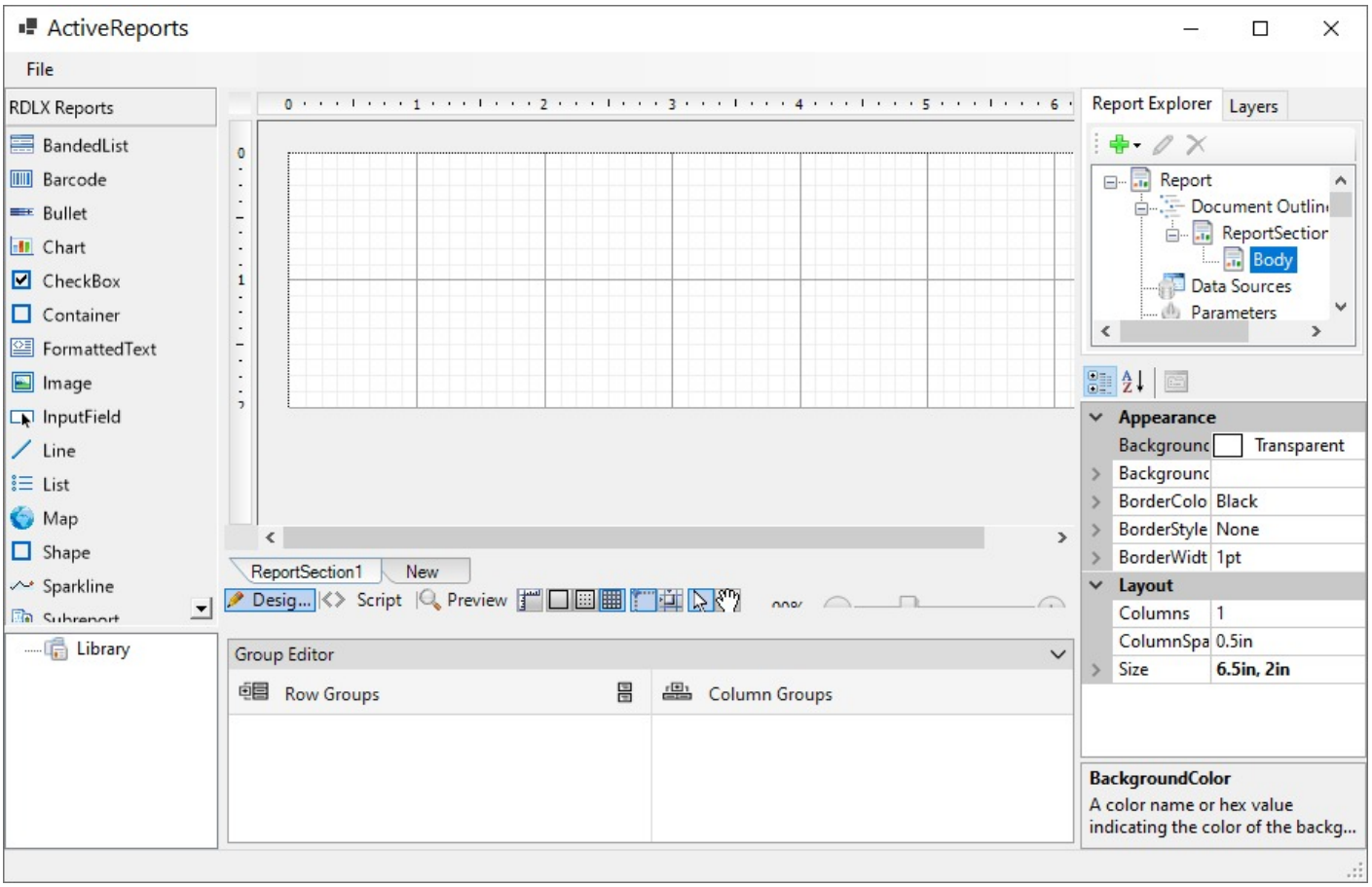
End User Report Designer in WinForms Application

Note: This is a Professional Edition feature. See ActiveReports Editions for details.

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

Create Basic End User Report Designer

This topic demonstrates how to set up a basic End-User Report Designer on a Windows Forms application in the Professional Edition of ActiveReports.



Add the Designer control to the Form

You will add the Designer that could only edit and preview a report file.

1. Create a new **Windows Forms App** project in Visual Studio 2022.
2. In the **Name** field, rename the file to 'CustomEUD' and click **Next**.
3. In the Additional Information screen, select a **Framework** and click **Create**.
4. Install following two packages to make ActiveReports 18 toolbox and the **Designer** control available in Visual Studio:
    - **MESCIUS.ActiveReports**
    - **MESCIUS.ActiveReports.Design.Win**
5. Select a Form and go to the Properties Panel to change the **Name** property to **frmMain** and the **Text** property to **ActiveReports**.
6. Resize the Form so that you can accommodate the controls listed further.
7. From the Visual Studio toolbox, drag the Designer control onto the Form and rename it to '_designer'.
8. From the Visual Studio toolbox, drag the Toolbox control onto the Form and rename it to 'toolbox'.
9. To attach the toolbox control to the designer control, in the Solution Explorer, right-click Form1.cs and select **View Code.**
10. Add the following code (marked in bold) after the InitializeComponent method.

C# code. Paste AFTER the InitializeComponent method
`public frmMain()`

```
{
  InitializeComponent();
  _designer.Toolbox = toolbox;
}
```

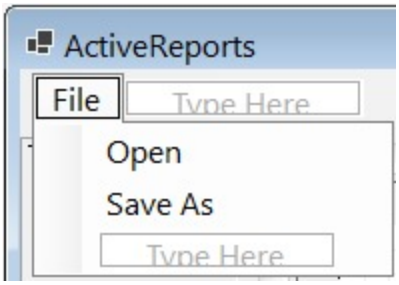
11. At the top of the code view, add a using directive.

C# code. Paste at the top of the Form1 code view

```
using GrapeCity.ActiveReports.Design;
```

Load and/or save the report file

1. From the Visual Studio **Menus & Toolbars** toolbox group, drag the MenuStrip control onto the Form.
2. Create the following structure for the MenuStrip control: **File > Open**, **File > Save As**.



3. On the Form, double-click the **Open** menu item and paste the following code (marked in bold) into the openToolStripMenuItem_Click handler.

C# code. Paste INSIDE the openToolStripMenuItem_Click handler

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            var dialogResult = openFileDialog.ShowDialog();
            if (dialogResult == System.Windows.Forms.DialogResult.OK)
            {
                _designer.LoadReport(new System.IO.FileInfo(openFileDialog.FileName));
            }
        }
```

4. On the Form, double-click the **Save As** menu item and paste the following code (marked in bold) into the saveAsToolStripMenuItem_Click handler.

C# code. Paste INSIDE the openToolStripMenuItem_Click handler

```
private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = GetSaveFilter();
            var dialogResult = saveFileDialog.ShowDialog();
            if (dialogResult == System.Windows.Forms.DialogResult.OK)
            {
               _designer.SaveReport(new System.IO.FileInfo(saveFileDialog.FileName));
            }
```

```
        }
```

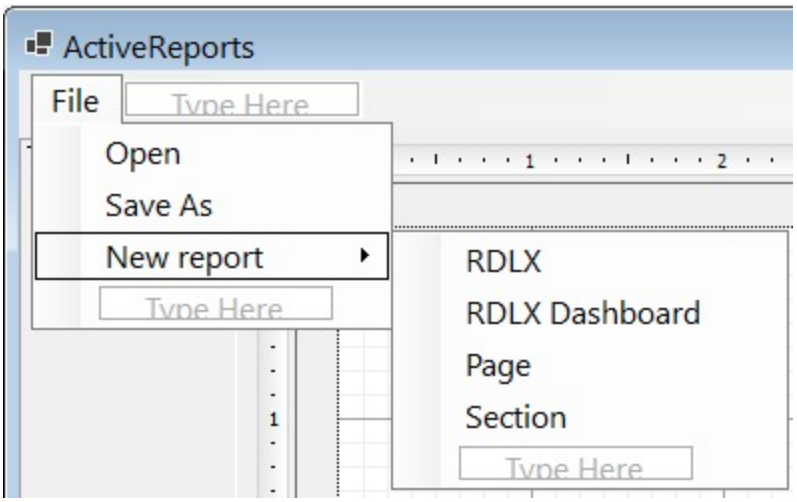
5. After the saveAsToolStripMenuItem_Click handler code, add the code for the GetSaveFilter method as follows.

C# code. Paste AFTER the saveAsToolStripMenuItem_Click handler

```
private string GetSaveFilter()
 {
     switch (_designer.ReportType)
     {
         case DesignerReportType.Section:
             return "Section Report Files (*.rpx)|*.rpx";
         case DesignerReportType.Page:
             return "Page Report Files (*.rdlx)|*.rdlx";
         case DesignerReportType.RdlMultiSection|DesignerReportType.RdlDashboard:
             return "RDLX Report Files (*.rdlx)|*.rdlx";
         default:
             return "RDLX Report Files (*.rdlx)|*.rdlx";
     }
 }
```

Create a new report based on a chosen type

1. Create the following structure for the MenuStrip control: **File > New report > RDLX, RDLX Dashboard, Page, Section**.



2. Double-click the **RDLX** MenuStrip item and add the following code (marked in bold) into the rdlxToolStripMenuItem_Click handler.

C# code. Paste INSIDE the rdlxToolStripMenuItem_Click handler

```
private void rdlxToolStripMenuItem_Click(object sender, EventArgs e)
  {
   _designer.NewReport(DesignerReportType.RdlMultiSection);
  }
```

3. Double-click the **RDLX Dashboard** MenuStrip item and add the following code (marked in bold) into the rdlxdashboardToolStripMenuItem_Click handler.

C# code. Paste INSIDE the rdlxdashboardToolStripMenuItem_Click handler

```
private void rdlxdashboardToolStripMenuItem_Click(object sender, EventArgs e)
  {
   _designer.NewReport(DesignerReportType.RdlDashboard);
  }
```

4. Double-click the **Page** MenuStrip item and add the following code (marked in bold) into the pageToolStripMenuItem_Click handler.

C# code. Paste INSIDE the pageToolStripMenuItem_Click handler

```
private void pageToolStripMenuItem_Click(object sender, EventArgs e)
  {
    _designer.NewReport(DesignerReportType.Page);
  }
```

5. Double-click the **Section** MenuStrip item and add the following code (marked in bold) into the sectionToolStripMenuItem_Click handler.

C# code. Paste INSIDE the sectionToolStripMenuItem_Click handler

```
private void sectionToolStripMenuItem_Click(object sender, EventArgs e)
  {
   _designer.NewReport(DesignerReportType.Section);
  }
```

Add the Export option

1. Install packages from **NuGet** as follows:
   i) Go to **Tools** > **NuGet Package Manager** > **Manage NuGet Packages for Solution**...
   ii) Browse the following package and click Install.

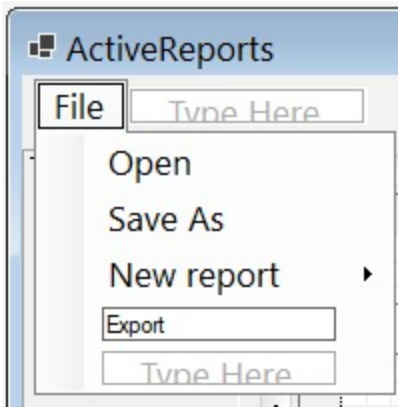
```
MESCIUS.ActiveReports.Export.Pdf
```

2. At the top of the code view, add using statements.

C# code. Paste at the top of the Form1 code view

```
using GrapeCity.ActiveReports.Export.Pdf.Page;
using GrapeCity.ActiveReports.Rendering.IO;
using GrapeCity.ActiveReports;
using System.IO;
```

3. In the MenuStrip control, add the Export menu item to the **File** menu.



4. In the Properties Panel, set the **Enabled** property to **False**. This enables the Export menu item to be displayed in the Preview mode only.
5. To enable the Export menu item to be displayed for all report types, except Section report, add the following code (marked in bold) after the InitializeComponent method.

C# code. Paste AFTER the InitializeComponent method

```
public frmMain()
         {
             InitializeComponent();
             _designer.Toolbox = toolbox;
             _designer.ActiveTabChanged += designer_ActiveTabChanged;
         }
void designer_ActiveTabChanged(object sender, EventArgs e)
        {
             exportToolStripMenuItem.Enabled = _designer.ActiveTab ==
DesignerTab.Preview && _designer.ReportType != DesignerReportType.Section;
        }
```

6. On the Form, double-click the Export item and add the following code (marked in bold) to the exportToolStripMenuItem_Click handler.

C# code. Paste INSIDE the exportToolStripMenuItem_Click handler

```
private void exportToolStripMenuItem_Click(object sender, EventArgs e)
         {
             SaveFileDialog saveFileDialog = new SaveFileDialog();
             saveFileDialog.Filter = "Pdf (*.pdf)|*.pdf";
             var dialogResult = saveFileDialog.ShowDialog();
             if (dialogResult == System.Windows.Forms.DialogResult.OK)
             {
                 var pdfRe = new PdfRenderingExtension();
                 var msp = new MemoryStreamProvider();
                 (_designer.Report as PageReport).Document.Render(pdfRe, msp);
                 using (var stream = msp.GetPrimaryStream().OpenStream())
                using (var fileStream = new FileStream(saveFileDialog.FileName,
FileMode.Create, FileAccess.Write))
                 {
```

```
                    stream.CopyTo(fileStream);
                }
                MessageBox.Show("Export is done");
            }
        }
```

For more information on export filters, rendering extensions and their settings, see Export Reports.

Add other controls to the Form

1. From the Visual Studio toolbox, drag the following controls onto the Form.

Control/Container	Name	Property Value
ReportExplorer	arReportExplorer	ReportDesigner = _designer This binds the ActiveReports Designer to the ReportExplorer control. Resize and move as necessary.
LayerList	arLayerList	ReportDesigner = _designer This binds the ActiveReports Designer to the LayerList control. Resize and move as necessary.
PropertyGrid	arPropertyGrid	Resize and move as necessary.
GroupEditor (under Containers)	arGroupEditor	ReportDesigner = _designer This binds the ActiveReports Designer to the GroupEditor control. Resize and move as necessary.
ReportsLibrary	arReportsLibrary	ReportDesigner = _designer This binds the ActiveReports Designer to the ReportsLibrary control. Resize and move as necessary.

2. On the Form, select the Designer control.
3. In the Properties Panel, set the **PropertyGrid** property of the Designer control to arPropertyGrid. This binds the ActiveReports Designer to the Property Grid control.

View the End User Report Designer

Press F5 to run the project. The **End User Report Designer** opens with an RDLX report.

For information on how you can customize the End User Report Designer and more, refer to the End User Designer sample.

Call Designer

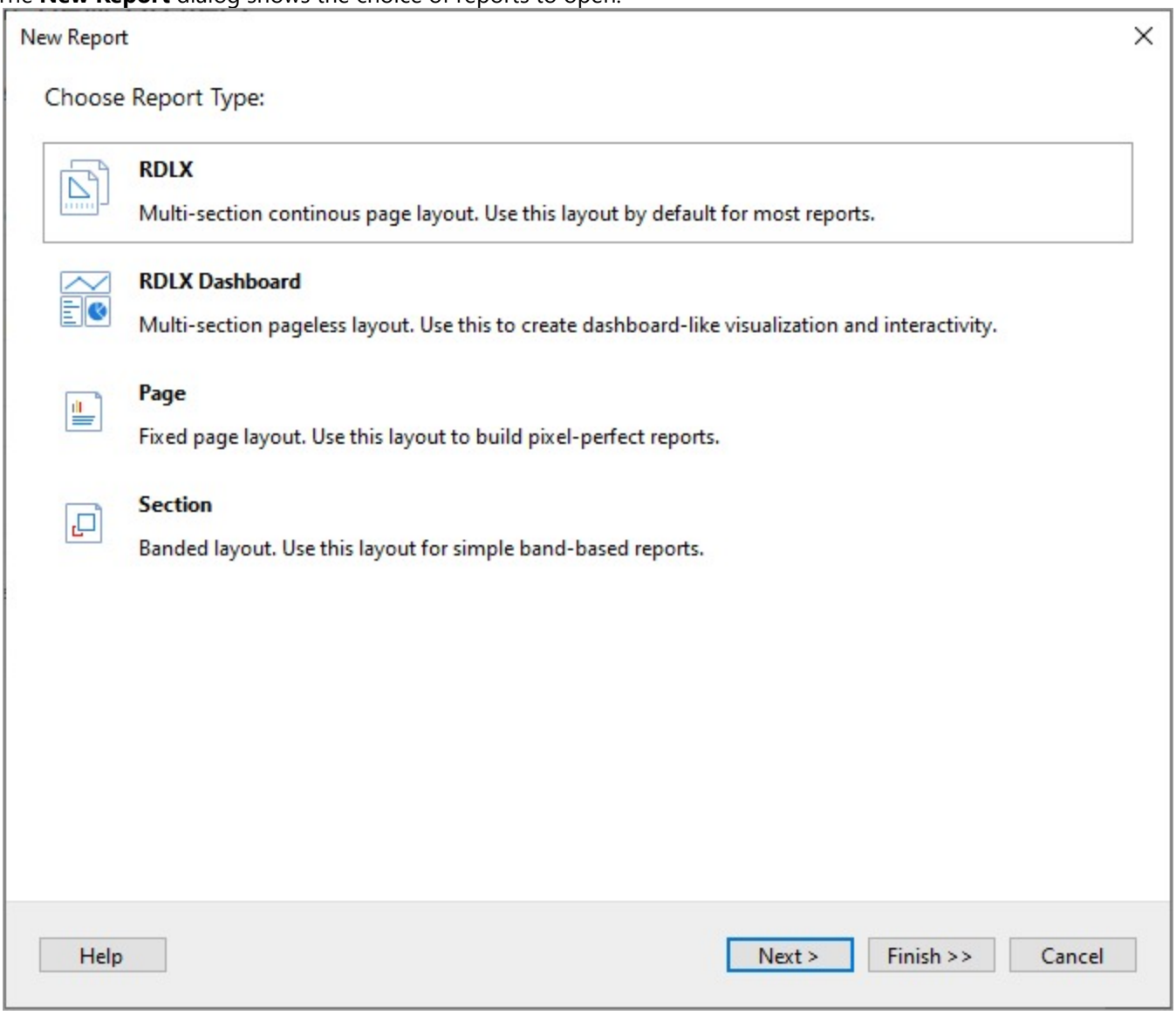
You need to call the **NewReport** designer action in GrapeCity.ActiveReports.Design.**DesignerAction**.

C# code. Paste AFTER the InitializeComponent method

```
public frmMain()
{
  InitializeComponent();
  _designer.ExecuteAction(DesignerAction.NewReport);
```

```
}
```

The **New Report** dialog shows the choice of reports to open.



Set High DPI Support

The End User Report Designer application requires specifying the SetHighDpiMode configuration to support the high DPI displays. To enable high DPI displays in a .NET6 application, use the following code.

C#

```
Application.SetHighDpiMode(HighDpiMode.DpiUnawareGdiScaled);
```

Use End-User Report Designer API

The following set of topics give information on the End-User Report Designer API that allows you to perform many operations with the report itself and the report designer surface.

Work with a Report

The End User Designer API for WinForms allows you to perform common operations with a report using the **Designer ('Designer Class' in the on-line documentation)** class.

Initialize Designer with a new report

Use **Designer.NewReport ('NewReport Method' in the on-line documentation)** method to create a new empty report and initialize the Designer as shown in the code example.

C#. Add using statements on the top of Form.cs

```
using System.Windows.Forms;
using GrapeCity.ActiveReports.Design;
```

C# code. Paste INSIDE the Form Load event

```
var _designer = new Designer() { Dock = DockStyle.Fill };
_designer.NewReport(DesignerReportType.Rdl);
Controls.Add(_designer);
```

Note: You should use the **Report ('Report Property' in the on-line documentation)** property only if you want to get the current state information. Please do not use this property if you need to change the report state or assign a new report.

Initialize Designer with a specified report

Use the **Designer.LoadReport ('LoadReport Method' in the on-line documentation)** method to initialize the Designer by loading a report created in code, as demonstrated in the code example.

```
var report = new PageReport {
      Report = {
        DataSources = {
              new DataSource {
                  Name = "MainDataSource",
                  ConnectionProperties = {
                      ConnectString = "<your_connection_string>",
                      DataProvider = "MSSQL",
                  }
              }
          }
      }
  };
//Serialize and load report into the designer
_designer.LoadReport(XmlReader.Create(new StringReader(report.ToRdlString())),
```

```
DesignerReportType.Rdl);
```

Save a report

Save a report to any storage by passing the XmlWriter object to the **Designer.SaveReport ('SaveReport Method' in the on-line documentation)** method. See the example code for details.

C#. Add using statements on the top of Form.cs

```
using System.IO;
using System.Xml;
using GrapeCity.ActiveReports.Design;
```

C#.

```
private void SaveReport(Stream outStream)
{
  using(var writer = XmlWriter.Create(outStream))
  {
    _designer.SaveReport(writer);
  }
}
```

Define a font resolver

Using the **Designer.FontResolver ('FontResolver Property' in the on-line documentation)** property, you can provide your own fonts to be used for the report design and preview. For that, you must first define your own font resolver and then attach it to the Designer instance. See the code example for details.

C#. Add using statements on the top of Form.cs

```
using GrapeCity.ActiveReports;
using GrapeCity.ActiveReports.Design;
```

C#. Define a custom font resolver

```
public sealed class DirectoryFontResolver : IFontResolver
{
    GrapeCity.Documents.Text.FontCollection _fonts;

    public DirectoryFontResolver(string fontsDirectory)
    {
        _fonts = new GrapeCity.Documents.Text.FontCollection();
        // load standard Windows fonts
        _fonts.RegisterDirectory(Environment.GetFolderPath(Environment.SpecialFolder.Fonts));
        // load fonts from custom directory
        _fonts.RegisterDirectory(fontsDirectory);
        _fonts.DefaultFont = _fonts.FindFamilyName("Arial");
    }
    GrapeCity.Documents.Text.FontCollection IFontResolver.GetFonts(string familyName, bool
```

```
isBold, bool isItalic)
    {
        var fonts = new GrapeCity.Documents.Text.FontCollection();
        fonts.Add(_fonts.FindFamilyName(familyName, isBold, isItalic) ?? _fonts.DefaultFont);
        return fonts;
    }
}
```

C#. Attach a custom font resolver to the Designer instance

```
class MyDesignerForm : Form
{
  public MyDesignerForm()
  {
    InitializeComponent();
    //The designer must be added to the form with the name '_designer'.
    _designer.FontResolver = new DirectoryFontResolver("c:\\Fonts");
  }
}
```

Specify a custom resource locator

Using the **Designer.ResourceLocator ('ResourceLocator Property' in the on-line documentation)** property, you can define a custom resource locator to get the resources (like images, subreports) from the custom storage or at a specific disk location. First, define the custom resource locator and then use it in the Designer. See the code example for details.

C#. Add using statements on the top of Form.cs

```
using System;
using System.IO;
using System.Drawing;
using GrapeCity.ActiveReports;
```

C#. Define a custom resource locator

```
class MyResourceLocator : ResourceLocator
{
    public override Resource GetResource(ResourceInfo resourceInfo)
    {
        if (resourceInfo.Name == "redSquare.png")
        {
            //Here we draw the image with the red square in the center.
            //You can load the image from file system, or from data base, or from assembly
resources.
            var img = new Bitmap(100, 100);
            var graphics = Graphics.FromImage(img);
            var redBrush = new SolidBrush(Color.Red);
            graphics.FillRectangle(redBrush, 10, 10, 80, 80);
            var tmpStream = new MemoryStream();
```

```
            img.Save(tmpStream, System.Drawing.Imaging.ImageFormat.Png);
            tmpStream.Position = 0;
            return new Resource(tmpStream, new Uri("redSquare.png", UriKind.Relative));
        }
        return new Resource();
    }
}
```

C#. Attach a custom resource locator to the Designer instance

```
class MyDesignerForm : Form
{
  public MyDesignerForm()
  {
    InitializeComponent();
    //The designer must be added to the form with the name '_designer'.
    _designer.ResourceLocator = new MyResourceLocator();
  }
```

Execute specific Designer actions

With the **Designer.ExecuteAction(DesignerAction) ('ExecuteAction Method' in the on-line documentation)** method, you can execute some specified designer action. Most Designer APIs related to reports are encapsulated into the **ExecuteAction** method. The code example below demonstrates how you can remove all selected items.

C#

```
private void RemoveSelectedItems()
{
  _designer.ExecuteAction(DesignerAction.EditDelete);
}
```

The code example below demonstrates how you can select all items and remove them.

C#

```
private void Clear()
{
  _designer.ExecuteAction(DesignerAction.SelectAll);
  _designer.ExecuteAction(DesignerAction.EditDelete);
}
```

Supply credentials for a data provider

Use the **Designer.LocateCredentials ('LocateCredentials Event' in the on-line documentation)** event to provide credentials for data providers as demonstrated in the code example.

C#. Add using statements on the top of Form.cs

```
using System.Windows.Forms;
using GrapeCity.ActiveReports;
using GrapeCity.ActiveReports.Design;
```

C#

```
_designer.LocateCredentials += (sender, args) => {
  args.UserName = "sa";
  args.Password = "12345";
}
```

Provide data for a page report

Use the **Designer.LocateDataSource ('LocateDataSource Event' in the on-line documentation)** event to provide data for previewing a page report as demonstrated in the code example.

C#

```
_designer.LocateDataSource += (sender, args) => {
  var dataSourceName = args.DataSet.Query.DataSourceName;
  var dataSource = ((PageReport)designer.Report).Report.DataSources.First(x => x.Name ==
dataSourceName);
  if(dataSource.ConnectionProperties.DataProvider == "OBJECT")
  {
    args.Data = new[] { new { CustomerName = "Gc Inc." } };
  }
};
```

Get the selected items collection

Use the **Designer.Selection ('Selection Property' in the on-line documentation)** property or the **Designer.SelectionChanged ('SelectionChanged Event' in the on-line documentation)** event to react to selection changes. Using the **Designer.Selection ('Selection Property' in the on-line documentation)** property, you can get the selected items collection. The code example below demonstrates changing the 'remove' button state, which must be enabled only when there are selected items.

C#

```
_designer.SelectionChanged += () {
  removeButton.Enabled =
    _designer.Selection.OfType<ReportItem>().Count() > 0
    || designer.Selection.OfType<ARControl>().Count() > 0;
}
```

Get the changed undo history

The **UndoManager.Changed ('Changed Event' in the on-line documentation)** event occurs when the report was changed and therefore the undo history was changed as well. You can use this event in the conjunction with

the **UndoManager.UndoCount ('UndoCount Property' in the on-line documentation)** and **UndoManager.RedoCount ('RedoCount Property' in the on-line documentation)** properties to set the Undo/Redo buttons state as shown in the code example.

C#

```
_designer.UndoManager.Changed += (sender, e) => {
  _undoButton.Enabled = _designer.UndoManager.UndoCount > 0;
  _redoButton.Enabled = _designer.UndoManager.RedoCount > 0;
};
```

Handle actions after changing the report layout

With the **Designer.LayoutChanged ('LayoutChanged Event' in the on-line documentation)** event, you can execute some actions right after the layout has been changed as in the code example below.

C#

```
_designer.LayoutChanged += (sender, args) => {
  //do some work here
};
```

Handle report layout changes

The **Designer.LayoutChanging ('LayoutChanging Event' in the on-line documentation)** event allows to handle the report layout changes. The example below deprecates adding a new control without a dataset.

C#

```
_designer.LayoutChanging += (sender, args) => {
      if (args.Type == LayoutChangeType.ControlAdd
      && designer.Report is PageReport pageReport
      && pageReport.Report.DataSets.Count == 0)
      {
        ((IUIService)designer).ShowError("Add data set first.");
        args.AllowChange = false;
      }
  };
```

Check if a Designer action is enabled

With the **Designer.QueryActionEnabled(DesignerAction) ('QueryActionEnabled Method' in the on-line documentation)** method, you can check if a specific action in the Designer is enabled.

C#

```
var canConvert = designer.QueryActionEnabled(DesignerAction.ConvertToMaster);
//canConvert value will be 'false' for master reports and 'true' for rdl reports.
```

Return a currently opened report

The **Designer.Report ('Report Property' in the on-line documentation)** property returns a currently opened report, which can be a Page report or a Section report.

> To set a report, you should use the **Designer.LoadReport ('LoadReport Method' in the on-line documentation)** method.

C#

```
var extension = switch designer.Report {
  PageReport => ".rdlx",
  SectionReport => ".rpx",
  _ => throw new Exception("Unknown report type.")
};
```

Update some button states after the report changes

The **Designer.ReportChanged ('ReportChanged Event' in the on-line documentation)** event occurs when any change to a report has been applied (layout changes or data layer changes). You can use this event to update some button states as in the example below.

C#

```
_designer.ReportChanged += (sender, args) => {
  saveButton.Enabled = true;
};
```

Show the Data Source creation wizard

The **Designer.RunDataWizard ('RunDataWizard Method' in the on-line documentation)** event opens the data source creation wizard.

C#

```
private void RunDataWizardButtonClick(object sender, EventArgs e)
{
    _designer.RunDataWizard();
}
```

Customize the Design View

Get a currently active tab

By using **Designer.ActiveTabChanged** event, you can add a reaction to switching between the **Script**, **Design**, and **Preview** tabs. By using this event, you can get a currently active tab.

The example code is as follows.

C#

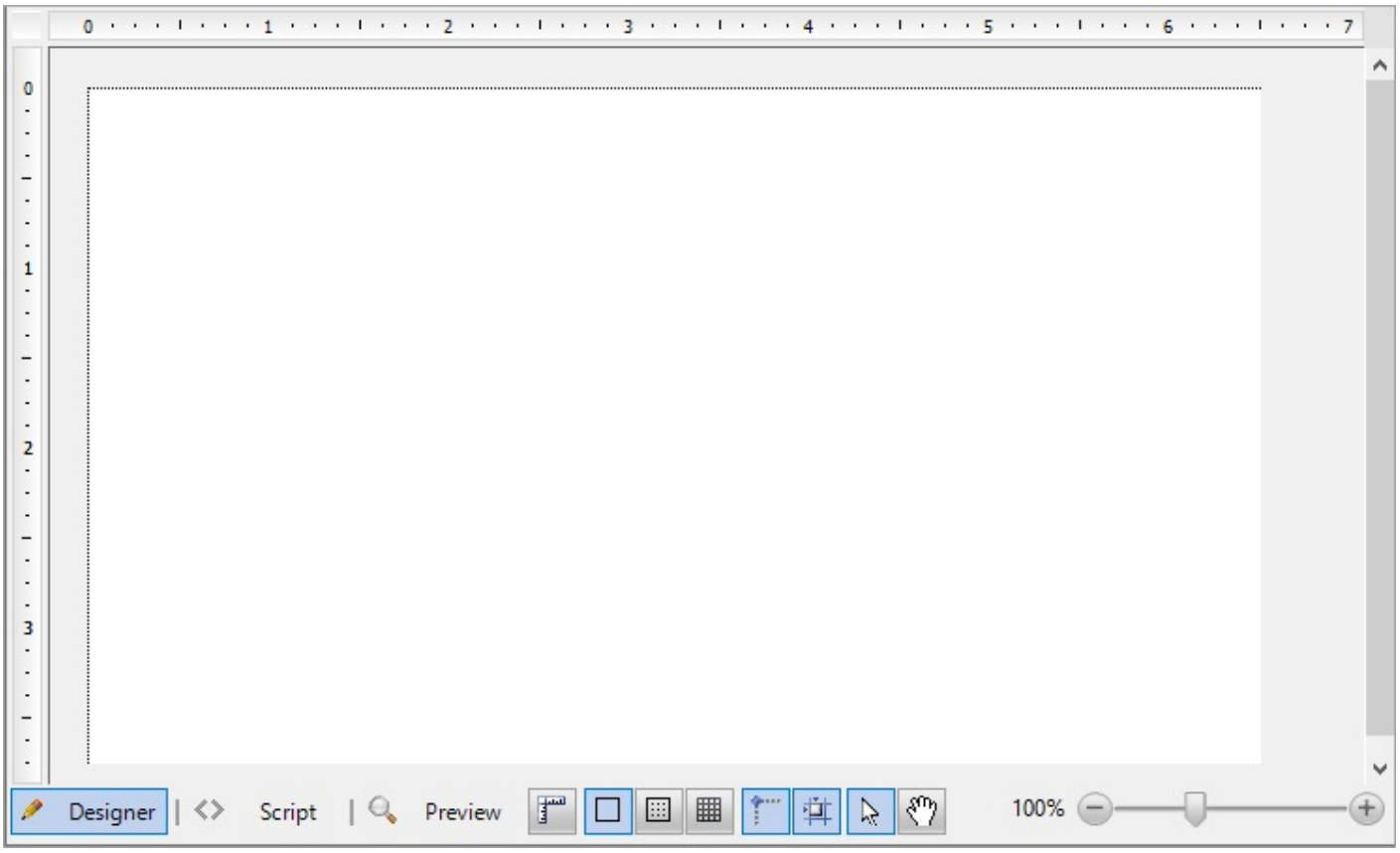
```
public partial class Form1 : Form
{
  public Form1()
  {
    InitializeComponent();
    _designer = new Designer() { Dock = DockStyle.Fill };
    _designer.ActiveTabChanged += (sender, args) => {
        //Get the currently editing report name.
        var reportName = designer.Report switch {
            PageReport pageReport => pageReport.Report.Name,
            SectionReport sectionReport => sectionReport.Name,
            _ => "Report"
        };
        //Set the form title to reflect the current designer state
        this.Text = designer.ActiveTab switch {
            DesignerTab.Script => $"{reportName} - Script",
            DesignerTab.Preview => $"{reportName} - Preview",
            _ => $"{reportName} - Edit"
        };
    };
    Controls.Add(designer);
  }
}
```

Add or remove a grid on the Designer surface

The **Designer.DrawGrid ('DrawGrid Property' in the on-line documentation)** property adds or removes the grid on the design surface. You can setup this property at the designer initialization or any time later.

> **Note:** You can control the grid visibility by using the buttons on the Tool Panel. You can deprecate it by using the **Designer.ToolPanel ('ToolPanel Property' in the on-line documentation)** property.

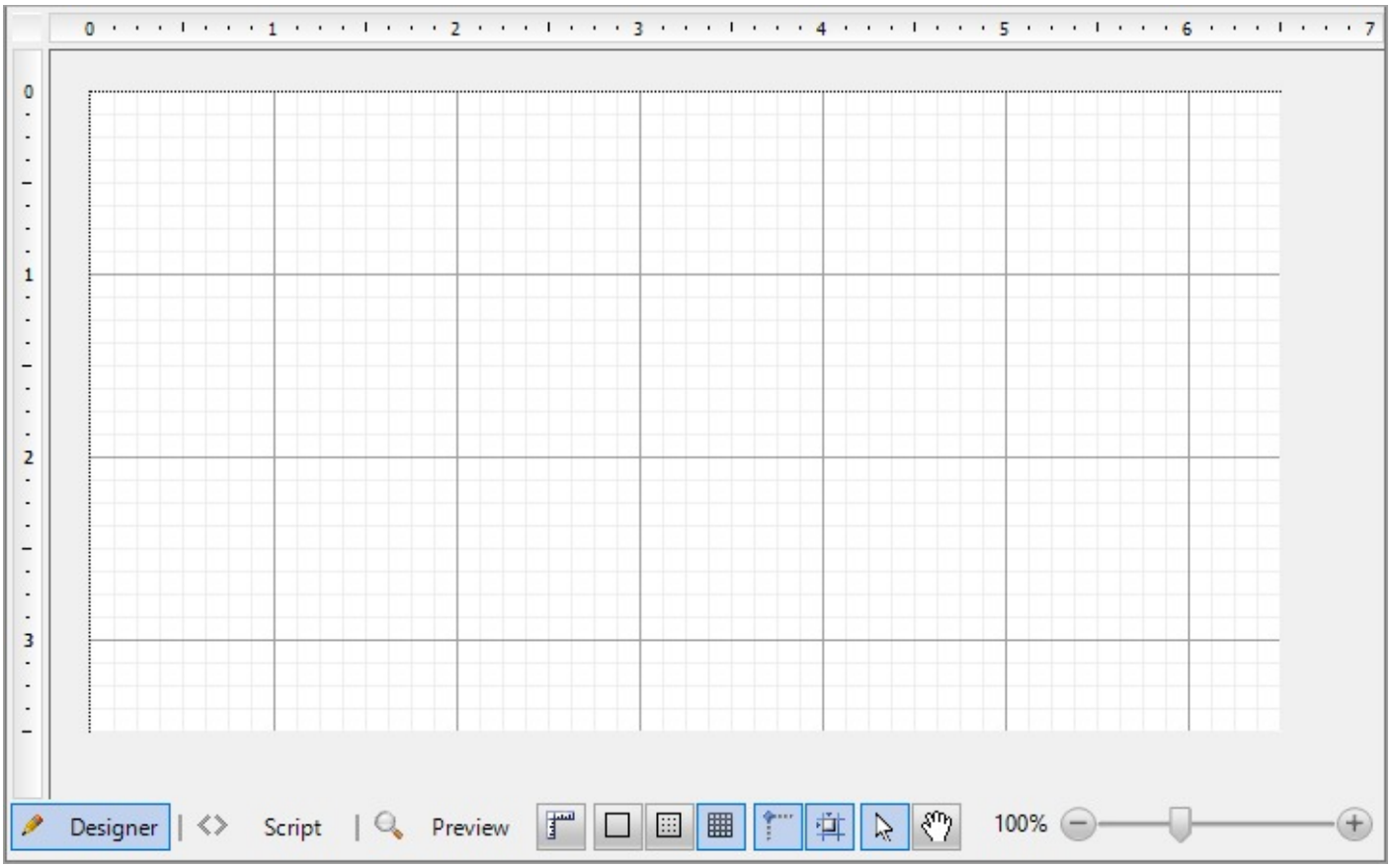
The code example below demonstrates how you can add your own button to switch the grid off.



C#

```
void Button_OnClick(object sender, EventArgs e)
{
    _designer.DrawGrid = false;
}
```

The code example below demonstrates how you can add your own button to switch the grid on.



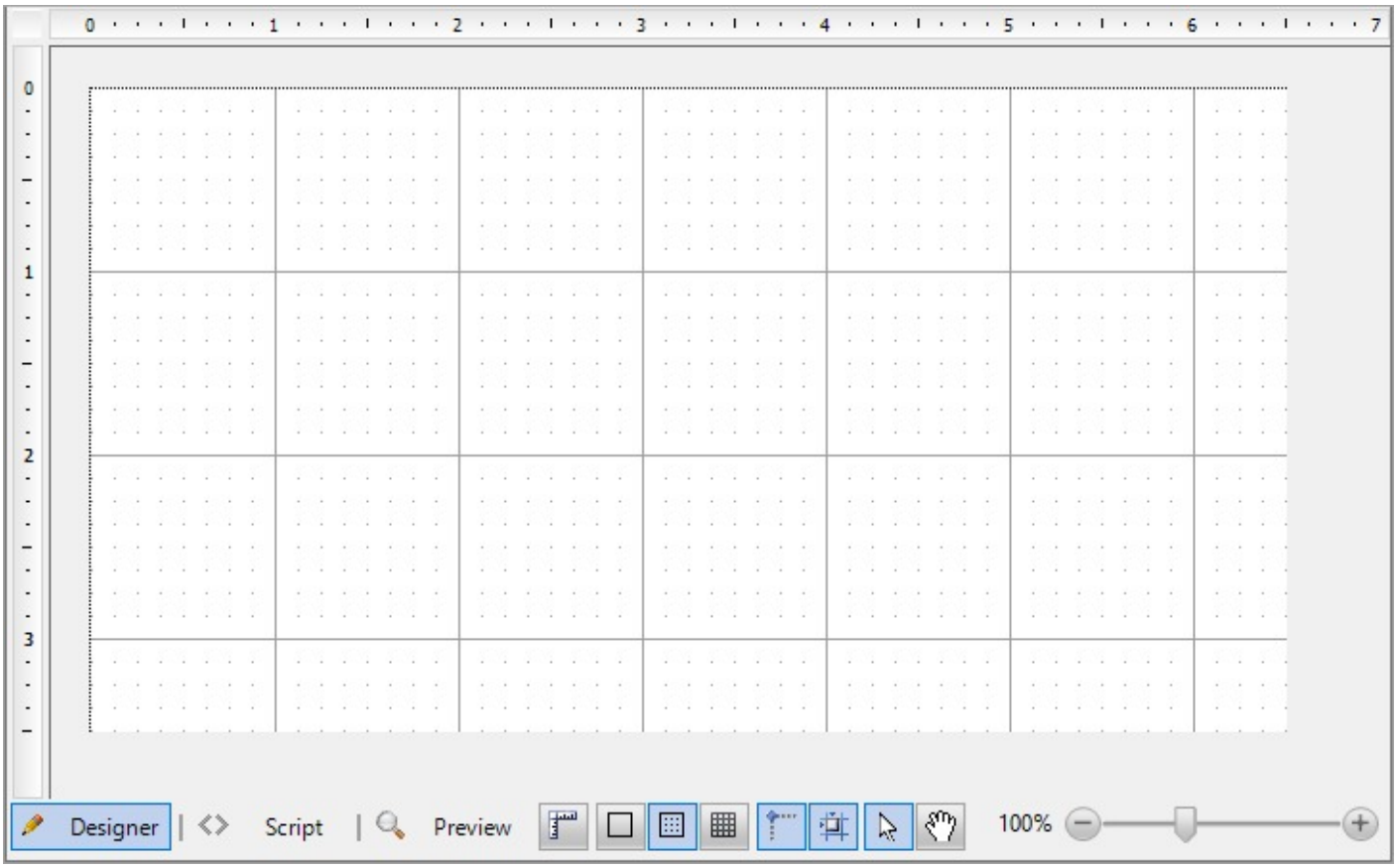
C#

```
void Button_OnClick(object sender, EventArgs e)
 {
     _designer.DrawGrid = true;
 }
```

Switch between the grid types

If the **Designer.DrawGrid ('DrawGrid Property' in the on-line documentation)** property is set to True, there are two possible grid types - Lines or Dots. You can switch between them by using the **Designer.GridMode ('DrawGrid Property' in the on-line documentation)** property.

This code example demonstrates how you can switch to the Dots grid mode.



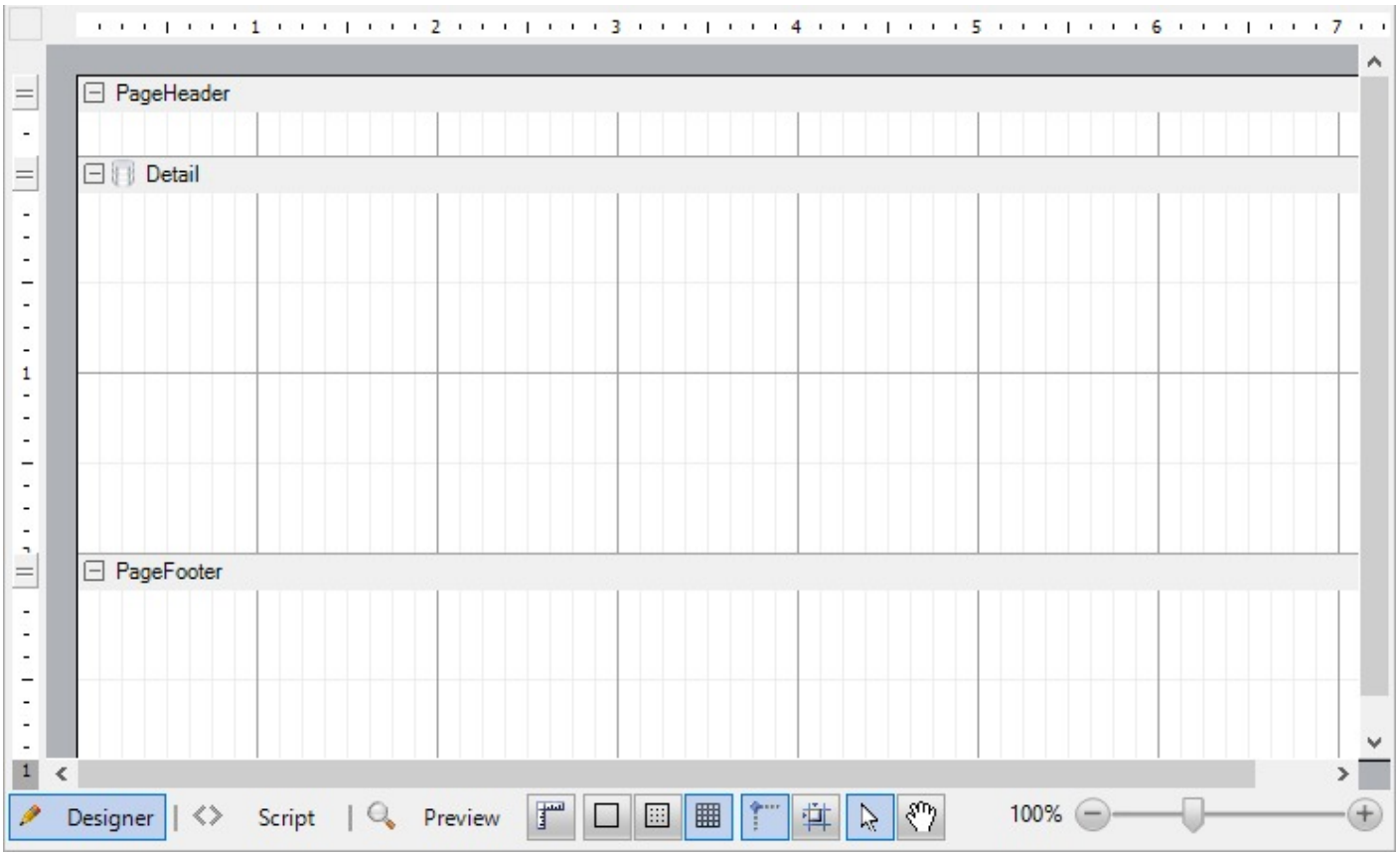
C#

```
_designer.GridMode = GridMode.Dots;
```

Specify a number of grid dots or lines per unit

Note: This is for Section Report designer only.

The **Designer.GridX ('GridX Property' in the on-line documentation)** and **Designer.GridY ('GridY Property' in the on-line documentation)** properties specify how many dots (or lines) of grid per unit need to be drawn on the design surface. If you specify a similar code as in the example below, you will get the following result.

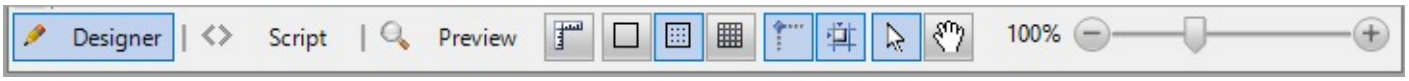


C#

```
_designer.GridX = 8;
_designer.GridY = 2;
```

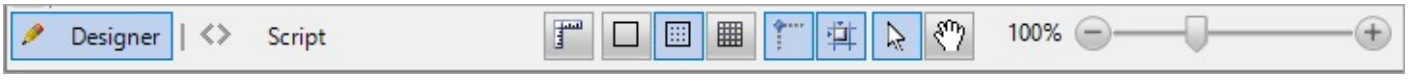
Show or hide the Preview tab

The **Designer.EnablePreview ('EnablePreview Property' in the on-line documentation)** property controls the visibility of the **Preview** tab on the Tool panel of the design surface.



C#

```
_designer.EnablePreview = true;
```

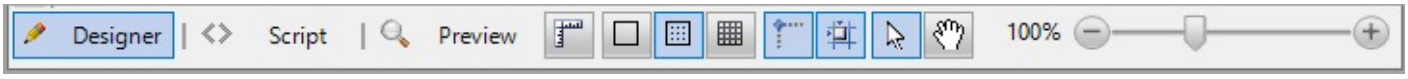


C#

```
_designer.EnablePreview = false;
```

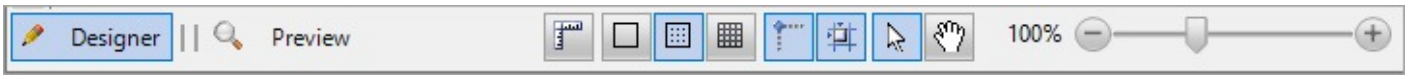
Show or hide the Script tab

The **Designer.EnableScripting ('EnableScripting Property' in the on-line documentation)** property controls the visibility of the **Script** tab on the Tool panel of the design surface.



C#

```
_designer.EnableScripting = true;
```



C#

```
_designer.EnableScripting = false;
```

Change the layout mode of items

The **Designer.LayoutMode ('LayoutMode Property' in the on-line documentation)** property changes the items layout mode and is currently available for Section reports only.

C#

```
private void ButtonSnapGrid_Click(object sender, EventArgs args)
{
  _designer.LayoutMode = designer.LayoutMode ^ LayoutMode.SnapGrid;
}
private void ButtonSnapLines_Click(object sender, EventArgs args)
{
  _designer.LayoutMode = designer.LayoutMode ^ LayoutMode.SnapLines;
}
```

Lock the position or size of report items

The **Designer.LockControls ('LockControls Property' in the on-line documentation)** property makes it impossible to change the position or size of report items.

The code example below opens the Windows Form with the designer with a pre-built report. You can change some properties of the existing controls but it is impossible to change the report layout.

C#

```
class TweakSalesReportForm : Form
{
  public TweakSalesReportForm()
  {
    var _designer = new Designer {Dock = DockStyle.Fill};
    var _propertyGrid = new PropertyGrid(){Dock = DockStyle.Right};
    _designer.LoadReport(new FileInfo("c:\\reports\\Sales.rdlx"));
    _designer.PropertyGrid = propertyGrid;
    _designer.LockControls = false;
    Controls.Add(designer);
    Contorls.Add(propertyGrid);
  }
}
```

Deprecate pages in a Page report

With the **Designer.PageReportDesignerActions ('PageReportDesignerActions Property' in the on-line documentation)** property, you can deprecate some actions under pages in page reports.

C#

```
//Deprecate new page creation.
_designer.PageReportDesignerActions = PageReportDesignerActions.All ^
PageReportDesignerActions.AddPage
```

Show or hide the Designer, Script, and Preview tabs

The **Designer.ReportTabsVisible ('ReportTabsVisible Property' in the on-line documentation)** property controls the visibility of the **Designer**, **Script**, and **Preview** tabs.

C#

```
_designer.ReportTabsVisible = false;
```

Show or hide the Tool panel

The **Designer.ReportTabsPanelVisible ('ReportTabsPanelVisible Property' in the on-line documentation)** property specifies whether the Tool panel is visible.

C#

```
_designer.ReportTabsPanelVisible = false;
```

Specify a scrolling mode for the Viewer

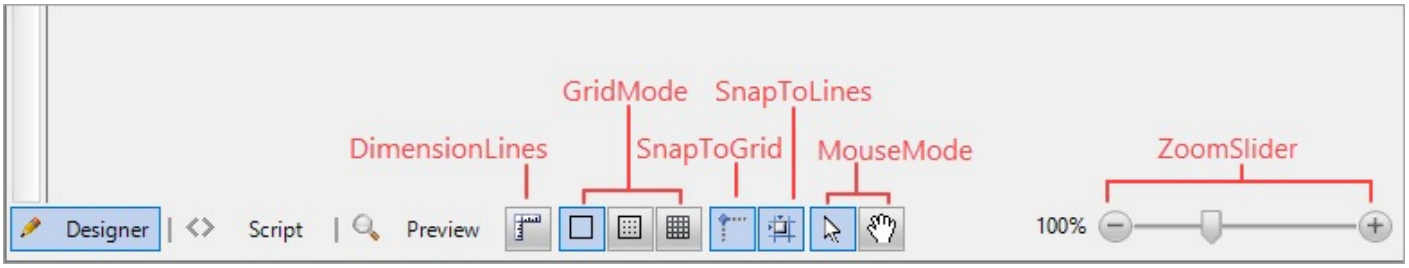
The **Designer.ScrollingMode ('ScrollingMode Property' in the on-line documentation)** property specifies the scrolling mode for the embedded report Viewer.

C#

```
_designer.ScrollingMode = ScrollingMode.Paged;
```

Control the visibility of the Designer tools

The **Designer.ToolPanel ('ToolPanel Property' in the on-line documentation)** property controls the visibility of some items in the Tool panel, which is located next to the Preview tab of the Design surface.



The code example below demonstrates how to remove the zoom slider.

C#

```
_designer.ToolPanel = ToolPanelButton.All ^ ToolPanelButton.ZoomSlider;
```

The Tool panel includes these items that you can remove:

- DimensionLines
- GridMode
- SnapGrid
- SnapLines
- MouseMode
- ZoomSlider

Show or hide the data source icon in the Detail section of Section report

The **Designer.ShowDataSourceIcon ('ShowDataSourceIcon Property' in the on-line documentation)** property is used for Section reports and allows to control the visibility of the data source icon in the Detail section.

C#

```
public partial class Form1 : Form
{
  Designer designer;
  public Form1()
  {
    InitializeComponent();
```

```
    _designer = new Designer() { Dock = DockStyle.Fill };
    _designer.NewReport(DesignerReportType.Section);
    _designer.ShowDataSourceIcon = false;
  }
}
```

Undo the latest operation

You can call the **Designer.Undo() ('Undo Method' in the on-line documentation)** method of the object returned by UndoManager property to undo the latest operation. Just add the button you want and assign the Click event handler like in this sample code.

C#

```
private void UndoButton_Click(object sender, EventArgs args)
{
  _designer.UndoManager.Undo();
}
```

Redo the undone operation

To redo the undone operation, you can call the **Designer.Redo() ('Redo Method' in the on-line documentation)** method of the object returned by UndoManager property. Just add the button you want and assign the Click event handler like in this sample code.

C#

```
private void RedoButton_Click(object sender, EventArgs args)
{
  _designer.UndoManager.Redo();
}
```

Specify zoom as scale factor

The **Designer.Zoom ('Zoom Property' in the on-line documentation)** property specifies zoom as scale factor.

C#

```
private void ResetZoom_Click(object sender, EventArgs e)
{
  _designer.Zoom = 1;
}
```

Add the PropertyGrid to the Designer

You can add the PropertyGrid component that allows you to set the properties of a selected report item.

Use WinForms Designer

1. From the **ActiveReports 18 Toolbox**, drag and drop the **Designer** component on the form.
2. Drag and drop the **PropertyGrid** component from the **All Windows Forms** Toolbox group onto the surface of your form or the Designer component.
3. Select the **Designer** component and from properties pane, set the **PropertyGrid** property to the name of the PropertyGrid component.

Use code

The code example below demonstrates adding the PropertyGrid component using the **Designer.PropertyGrid ('PropertyGrid Property' in the on-line documentation)** property.

C#

```
using GrapeCity.ActiveReports.Design;
using GrapeCity.ActiveReports.Design.ReportExplorer;
class MyForm : Form
{
  MyForm()
  {
    var _designer = new Designer() { Dock = DockStyle.Fill };
    var _propertyGrid = new PropertyGrid { Dock = DockStyle.Right };
    _designer.PropertyGrid = _propertyGrid;
    Controls.Add(_designer);
    Controls.Add(_propertyGrid);
  }
}
```

Manage the Report Explorer

Report Explorer is a component that displays the structure of the current report opened in the designer as a tree. It also allows you to change the report, add or remove entities from it, affects the selection.

Attach Report Explorer to the existing Designer instance

Use WinForms Designer

1. From the **ActiveReports 18 Toolbox**, drag and drop the **Designer** component on the form.
2. Drag and drop the **ReportExplorer** component onto the surface of your form or the Designer component.
3. Select the ReportExplorer component and from properties pane, set the **ReportDesigner** property to the name of the designer component.

Use code

The code example below demonstrates how to attach Report Explorer to the Designer programmatically, using the **ReportExplorerInternal.ReportDesigner** property in **GrapeCity.ActiveReports.Design.ReportExplorer ('GrapeCity.ActiveReports.Design.ReportExplorer Namespace' in the on-line documentation)** namespace.

C#. Paste the code in Form.cs

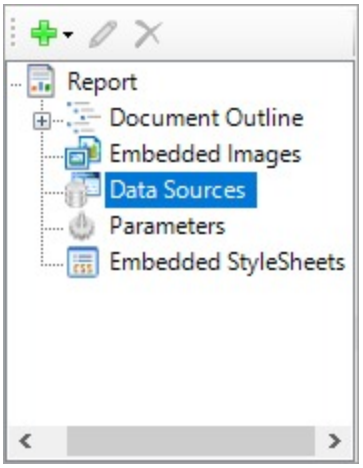
```
using GrapeCity.ActiveReports.Design;
using GrapeCity.ActiveReports.Design.ReportExplorer;

class MyForm : Form
{
  MyForm()
  {
    var _designer = new Designer() { Dock = DockStyle.Fill };
    var reportExplorer = new ReportExplorer  {
        Dock = DockStyle.Left,
        //now the reportExplorer instance will reflect the state of the report opened
        //in the designer control represented by '_designer' variable.
        ReportDesigner = _designer
    };
    Controls.Add(_designer);
    Controls.Add(reportExplorer);
  }
}
```

Control the visibility of nodes in Report Explorer

All nodes in the Report Explorer are visible by default. However there are specific nodes that you can hide from the Report Explorer. These specific nodes are:

- Data Sources
- Parameters
- Settings
- CommonValues

The code example below demonstrates how to make the Report Explorer display the two nodes: Data Sources (including DataSet node) and Parameters.



C#

```
reportExplorer.VisibleNodes = ReportExplorerNodes.DataSourceAndFields |
ReportExplorerNodes.Parameters;
```

Control the visibility of the Report Explorer content

All report changes are reflected in the Report Explorer, which may look unattractive in case of multiple operations.

With the **ReportExplorerInternal.ContentsVisible** property in **GrapeCity.ActiveReports.Design.ReportExplorer ('GrapeCity.ActiveReports.Design.ReportExplorer Namespace' in the on-line documentation)** namespace, you can hide the Report Explorer content for some time and then show it again as demonstrated in the code example below.

C#

```
reportExplorer.ContentsVisible = false;
BulkUpdate();//performs some bulk update operation
reportExplorer.ContentsVisible = true;
```

Set the Report Explorer enabled nodes

You can set enabled nodes to control actions to be performed to a report. For example, you can deprecate modifying the Parameters list from the ReportExplorer using the **ReportExplorerInternal.EnabledNodes** property in **GrapeCity.ActiveReports.Design.ReportExplorer ('GrapeCity.ActiveReports.Design.ReportExplorer Namespace' in the on-line documentation)** namespace. In this case, existing report parameters will be visible in the Report Explorer but a user will not be able to edit them.

C#

```
reportExplorer.EnabledNodes = ReportExplorerEnabledNodes.All ^
ReportExplorerEnabledNodes.Parameters;
```

Customize the Toolbox

You can change the Designer toolbox items to show specific controls or remove a control from the toolbox.

Add Toolbox panel

The code example below demonstrates adding a Toolbox panel to the Designer using the **Designer.Toolbox ('Toolbox Property' in the on-line documentation)** property.

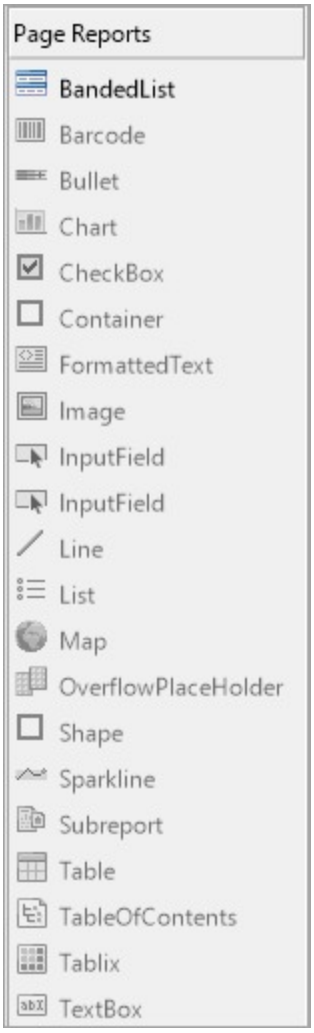
C#. Paste the code in Form.cs

```
using GrapeCity.ActiveReports.Design;
using GrapeCity.ActiveReports.Design.ReportExplorer;
class MyForm : Form
{
  MyForm()
  {
```

```
    var _designer = new Designer() { Dock = DockStyle.Fill };
    var toolbox = new Toolbox { Dock = DockStyle.Right };
    _designer.Toolbox = toolbox;
    Controls.Add(_designer);
    Controls.Add(toolbox);
  }
}
```

Control the Toolbox items

With the **Toolbox.ConfigureToolboxItems ('ConfigureToolboxItems Method' in the on-line documentation)** method, you can control items that are available in the Toolbox.



You first need to define a state provider as demonstrated in the sample below.

C#

```
class ToolboxStateProvider : IToolboxUser
{
```

```
   public bool GetToolSupported(ToolboxItem tool)
    {
        if (tool.TypeName ==
"GrapeCity.ActiveReports.Design.DdrDesigner.Designers.BandedList.BandedListDesigner")
            return true;
        return false;
    }
    public void ToolPicked(ToolboxItem tool)  { }
}
```

After the state provider is configured, you need to configure the Toolbox with it as demonstrated in the sample below.

C#

```
toolbox.ConfigureToolboxItems(new ToolboxStateProvider());
```

Remove an item from the Toolbox

You can use the **Toolbox.RemoveToolboxItem ('RemoveToolboxItem Method' in the on-line documentation)** method together with the **Toolbox.GetToolboxItems ('GetToolboxItems Method' in the on-line documentation)** method to remove an item from the Toolbox.

C#

```
private static void RemoveBandedListFromToolBox(Toolbox toolbox)
{
  //Find the item to be removed.
  var bandedList = toolbox.GetToolboxItems()
      .OfType<ToolboxItem>()
      .Single(items => items.TypeName.EndsWith("BandedListDesigner"));
  //Remove the banded list item from the toolbox.
  toolbox.RemoveToolboxItem(items);
}
```

Work with Report Parts

Report parts are groups of items in a report that you can reuse in other reports. The report parts can be added and managed in the Reports Library.

Specify a directory for the reports library

With the **Designer.ReportPartsDirectory ('ReportPartsDirectory Property' in the on-line documentation)** property, you can specify the directory for loading the report items in the report library.

C#

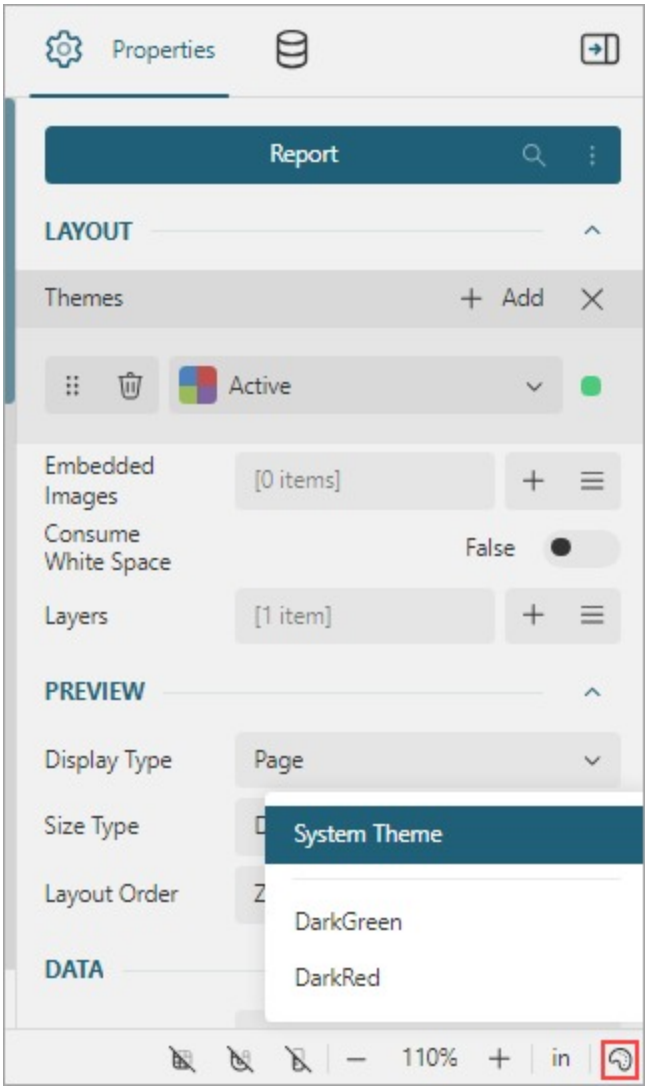
```
_designer.ReportPartsDirectory = "c:\\Data\Reports Library\\";
```

Apply Themes to WebDesigner and Js Viewer Components

Apply Themes to WebDesigner

Use the Applications UI

Use **Theme Picker** to select a System Theme or from the pre-defined themes in the application.



Use API

Enable/Disable Theme Picker in the UI

You can choose to show or hide Theme Picker in the UI, which lists all the available themes. Use 'picker' API as shown to hide the theme picker:

```
<script>
```

```
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     themes: { picker: { enabled: false }
});
< /script>
```

Apply default theme

Use 'default' API as shown to set a default theme from built-in themes.

```
<script>
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      themes: { default: 'defaultDark'}
});
< /script>
```

The built-in themes to choose from are: activeReports, activeReportsDark, defaultDark, darkOled, highContrast, highContrastDark.

Provide options to select from an array of themes

You can use either built-in themes or pass the theme object as option to choose in a Theme Picker. Use 'list' API as shown:

```
<script>
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
      themes: { list: ['default','defaultDark']}
});
< /script>
```

A theme object can be created as shown in section '**Add Custom Themes**'.

Apply dark theme if detected on system

You can choose whether the WebDesigner application should automatically detect and switch to a dark theme based on system settings. Use 'detectDarkTheme' API as shown to enable dark theme detection and set it:

```
<script>
const designer = GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
     themes: { detectDarkTheme: true }
});
< /script>
```

Apply Themes to Js Viewer using API

Apply a default theme

Use 'theme' API as shown to set a default theme from built-in themes.

```
<script>
 var viewer = GrapeCity.ActiveReports.JSViewer.create({
            theme: GrapeCity.ActiveReports.JSViewer.Themes.darkOled
        });
 viewer.openReport("Report.rdlx");
 }
</script>
```

The built-in themes to choose from are: activeReports, activeReportsDark, defaultDark, darkOled, highContrast, highContrastDark.

Apply same theme as designer

Use 'theme' API as shown to set the designer's theme to the viewer on preview.

```
<script>
     var viewer = null;
     GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
         themes: {
             default: 'DarkGreen',
         },
         appBar: {
             openButton: { visible: true },
             saveButton: { visible: true },
             saveAsButton: { visible: true }
         },

         preview: {
             openViewer: (options) => {
                 if (viewer) {
                     viewer.openReport(options.documentInfo.id);
                     return;
                 }
                 viewer = GrapeCity.ActiveReports.JSViewer.create({
                     element: '#' + options.element,
                     theme: options.theme,
                     reportService: {
                         url: 'api/reporting',
                     },
                     reportID: options.documentInfo.id
                 });
             }
```

```
        }
    });
</script>
```

Add Custom Themes in WebDesigner and Js Viewer using API

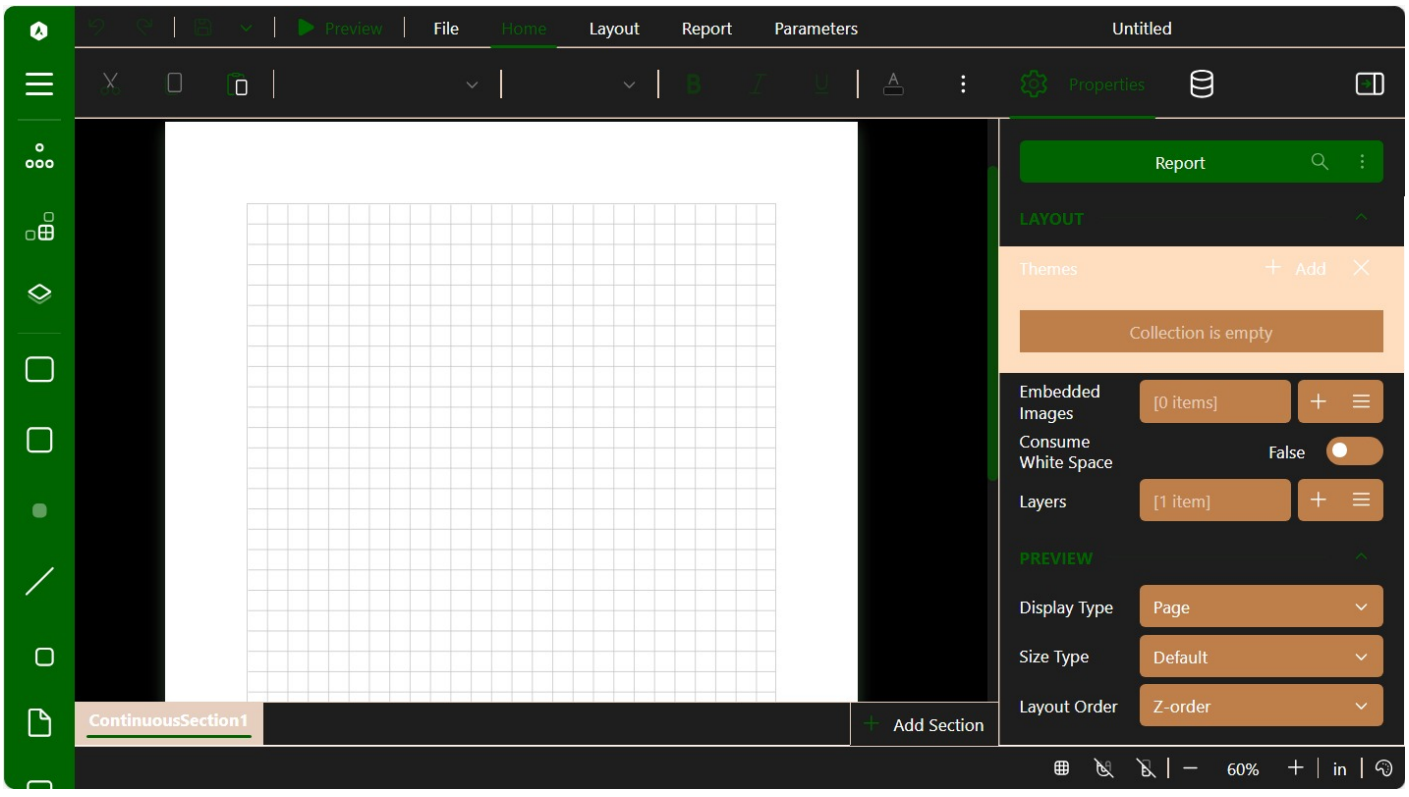
You can setup your own theme in WebDesigner and Js Viewer.

In WebDesigner, you can create a color theme object using: `GrapeCity.ActiveReports.DesignerThemes.color.rgba()` or `GrapeCity.ActiveReports.DesignerThemes.color.hex()`. And then pass the theme to the WebDesigner using the Themes API.

In Js Viewer, you can create a color theme object using: `GrapeCity.ActiveReports.ViewerThemes.color.rgba()` and `GrapeCity.ActiveReports.ViewerThemes.color.hex()`. And then pass the theme to the Js Viewer using the theme API.

> **Note**: If you create a theme using the WebDesigner API, you can pass the same theme to the Js Viewer. You can't pass the theme created using the Js Viewer API to the WebDesigner.

The following image and example snippet shows setting theme colors, drop shadow, and border radius, and apply it to the WebDesigner and its preview.

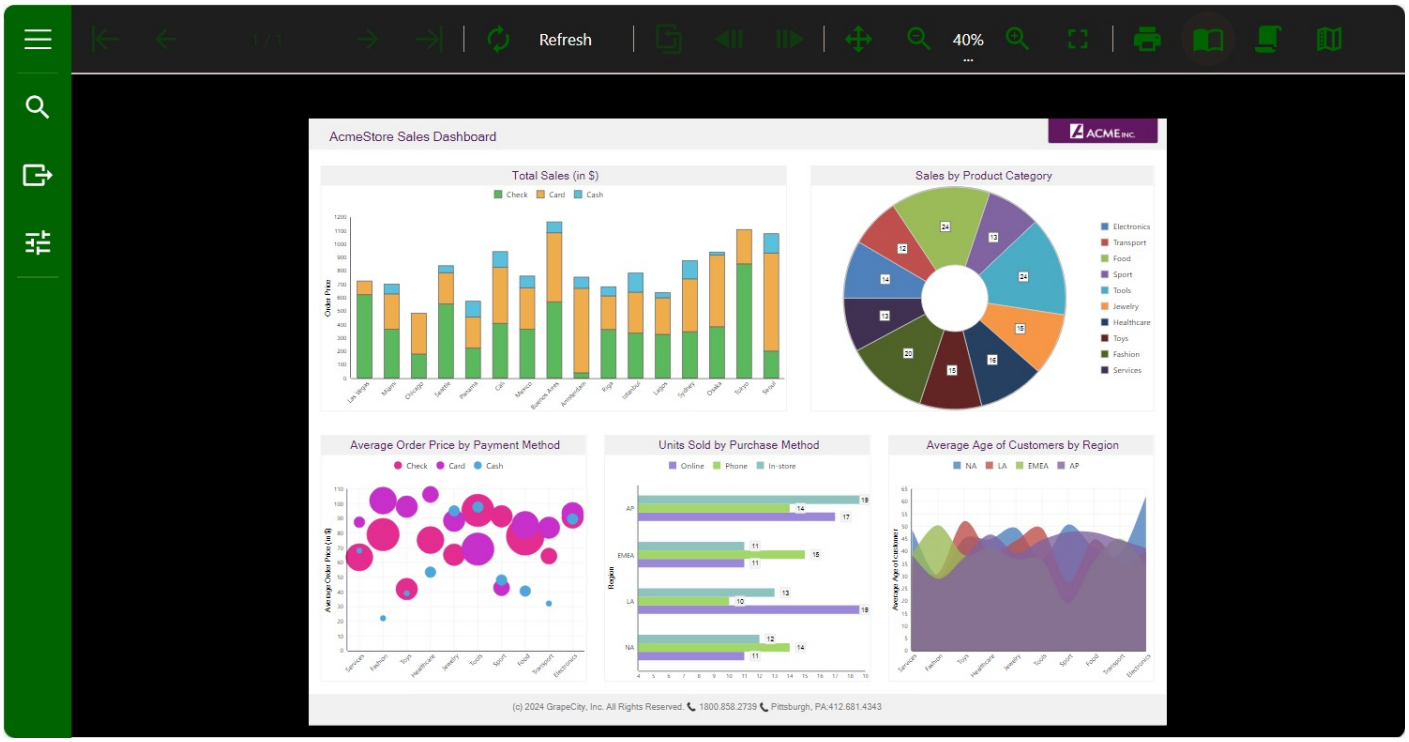



```
<script>
    var viewer = null;
    var DarkGreenTheme = {
```

```
        name: "DarkGreen",
        accent: GrapeCity.ActiveReports.DesignerThemes.color.rgba(0, 100, 0, 255),
        accentText: GrapeCity.ActiveReports.DesignerThemes.color.rgba(0, 100, 0, 255),
        accentSecondary: GrapeCity.ActiveReports.DesignerThemes.color.rgba(0, 100, 0, 255),
        accentWarning: GrapeCity.ActiveReports.DesignerThemes.color.rgba(0, 100, 0, 255),
        accentError: GrapeCity.ActiveReports.DesignerThemes.color.rgba(0, 100, 0, 255),
        colorContrast: GrapeCity.ActiveReports.DesignerThemes.color.rgba(255, 255, 255, 255),
        colorContrastText: GrapeCity.ActiveReports.DesignerThemes.color.rgba(255, 255, 255,
255),
        backgroundBody: GrapeCity.ActiveReports.DesignerThemes.color.rgba(0, 0, 0, 255),
        backgroundPanels: GrapeCity.ActiveReports.DesignerThemes.color.rgba(30, 30, 30, 255),
        shadow: "0 0 10px 1px rgba(0, 0, 0, 0.2)",
        shadowBorder: "0 0 10px 1px rgba(0, 0, 0, 0.1)",
        overlay: GrapeCity.ActiveReports.DesignerThemes.color.rgba(0, 0, 0, 38),
        textColor: GrapeCity.ActiveReports.DesignerThemes.color.rgba(255, 255, 255, 255),
        borderRadius: 4,
        elemBackground: GrapeCity.ActiveReports.DesignerThemes.color.rgba(191, 127, 74, 13),
        elemBackgroundHover: GrapeCity.ActiveReports.DesignerThemes.color.rgba(191, 127, 74,
39),
        btnGroupHeader: GrapeCity.ActiveReports.DesignerThemes.color.rgba(246, 229, 215,
255),
        btnGroupHeaderHover: GrapeCity.ActiveReports.DesignerThemes.color.rgba(242, 218, 198,
255),
        dropdownBackground: GrapeCity.ActiveReports.DesignerThemes.color.rgba(255, 255, 255,
255),
        dropdownBorder: GrapeCity.ActiveReports.DesignerThemes.color.rgba(230, 207, 190,
191),
        bindingModified: GrapeCity.ActiveReports.DesignerThemes.color.rgba(77, 202, 125,
255),
        bindingBound: GrapeCity.ActiveReports.DesignerThemes.color.rgba(254, 207, 0, 255),
        backgroundPanelsSection: GrapeCity.ActiveReports.DesignerThemes.color.rgba(255, 222,
191, 64),
        backgroundPanelsBorder: GrapeCity.ActiveReports.DesignerThemes.color.rgba(230, 207,
190, 191),
    };
    GrapeCity.ActiveReports.Designer.create('#ar-web-designer', {
        themes: {
            default: 'DarkGreen',
            list: [DarkGreenTheme]
        },
        appBar: {
            openButton: { visible: true },
            saveButton: { visible: true },
            saveAsButton: { visible: true }
        },

        preview: {
            openViewer: (options) => {
```

```
                if (viewer) {
                    viewer.openReport(options.documentInfo.id);
                    return;
                }
                viewer = GrapeCity.ActiveReports.JSViewer.create({
                    element: '#' + options.element,
                    theme: DarkGreenTheme,
                    reportService: {
                        url: 'api/reporting',
                    },
                    reportID: options.documentInfo.id
                });
            }
        }
    });
</script>
```

The following image and example snippet shows setting theme colors, drop shadow, and border radius, and pass it to the Js Viewer using the 'theme' API.




```
<script type="text/javascript">
    let viewer;
    function loadViewer() {
        var DarkGreenTheme = {
            name: "DarkGreen",
            accent: GrapeCity.ActiveReports.ViewerThemes.color.rgba(0, 100, 0, 255),
```

```
            accentText: GrapeCity.ActiveReports.ViewerThemes.color.rgba(0, 100, 0, 255),
            accentSecondary: GrapeCity.ActiveReports.ViewerThemes.color.rgba(0, 100, 0, 255),
            accentWarning: GrapeCity.ActiveReports.ViewerThemes.color.rgba(0, 100, 0, 255),
            accentError: GrapeCity.ActiveReports.ViewerThemes.color.rgba(0, 100, 0, 255),
            colorContrast: GrapeCity.ActiveReports.ViewerThemes.color.rgba(255, 255, 255,
255),
            colorContrastText: GrapeCity.ActiveReports.ViewerThemes.color.rgba(255, 255, 255,
255),
            backgroundBody: GrapeCity.ActiveReports.ViewerThemes.color.rgba(0, 0, 0, 255),
            backgroundPanels: GrapeCity.ActiveReports.ViewerThemes.color.rgba(30, 30, 30,
255),
            shadow: "0 0 10px 1px rgba(0, 0, 0, 0.2)",
            shadowBorder: "0 0 10px 1px rgba(0, 0, 0, 0.1)",
            overlay: GrapeCity.ActiveReports.ViewerThemes.color.rgba(0, 0, 0, 38),
            textColor: GrapeCity.ActiveReports.ViewerThemes.color.rgba(255, 255, 255, 255),
            borderRadius: 4,
            elemBackground: GrapeCity.ActiveReports.ViewerThemes.color.rgba(191, 127, 74, 13),
            elemBackgroundHover: GrapeCity.ActiveReports.ViewerThemes.color.rgba(191, 127, 74,
39),
            btnGroupHeader: GrapeCity.ActiveReports.ViewerThemes.color.rgba(246, 229, 215,
255),
            btnGroupHeaderHover: GrapeCity.ActiveReports.ViewerThemes.color.rgba(242, 218,
198, 255),
            dropdownBackground: GrapeCity.ActiveReports.ViewerThemes.color.rgba(255, 255, 255,
255),
            dropdownBorder: GrapeCity.ActiveReports.ViewerThemes.color.rgba(230, 207, 190,
191),
            bindingModified: GrapeCity.ActiveReports.ViewerThemes.color.rgba(77, 202, 125,
255),
            bindingBound: GrapeCity.ActiveReports.ViewerThemes.color.rgba(254, 207, 0, 255),
            backgroundPanelsSection: GrapeCity.ActiveReports.ViewerThemes.color.rgba(255, 222,
191, 64),
            backgroundPanelsBorder: GrapeCity.ActiveReports.ViewerThemes.color.rgba(230, 207,
190, 191),
        };
        viewer = GrapeCity.ActiveReports.JSViewer.create({
            element: '#viewerContainer',
            theme: DarkGreenTheme
        });
        viewer.openReport("Report.rdlx");
    }
</script>
```

Extensibility in ActiveReports

This section covers some common developers tasks related to using our API to extend ActiveReports functions at at various levels:

- Customization of report behavior
    - Fonts customization (see Custom Font Resolver)
    - Resources customization (see Custom Resource Locator)
    - Report item customization (see Create a Custom Report Item)
    - Map tile provider customization (see Custom Tile Provider)
- Create a new data provider (see Custom Data Provider)
- Customize Viewers and Designers (see Other Customization Options)
- Create a custom export (see Custom PDF Export sample)

If you still have questions or doubts, please contact our Support Team.

Custom Font Resolver

You can run reports on different platforms, and it is important to ensure that a report uses the same fonts. With the ActiveReports' custom font resolver, you can configure fonts for Page, RDLX, and Section reports (in the **CrossPlatform** compatibility mode) on different platforms.

> **Note**: Custom font settings have higher priority over fonts from the config file or the system font settings.

To learn about how to configure custom fonts in the ActiveReports.config file, see ActiveReports Configuration File.

Use the Custom Font Resolver in Preview

In some situations, you can use the custom fonts resolver through code by specifying the **PageReport.FontResolver ('FontResolver Property' in the on-line documentation)** property, in the **Windows Forms Viewer** and the **JavaScript Viewer** as in the following code examples.

C# Code for Windows Forms Viewer

```
var report = new PageReport(…);
report.FontResolver = …;
viewer.LoadDocument(report.Document);
```

C# Code for JavaScript Viewer

```
app.UseReportViewer(config => {
  config.FontResolver = ...
  config.UseFileStore(ResourcesRootDirectory)
});
```

Configure Fonts for Preview and Export

You can configure fonts for preview and export on all platforms without installation as in the following examples.

Example 1

```
public sealed class WindowsFontResolver : GrapeCity.ActiveReports.IFontResolver
{
   static readonly GrapeCity.Documents.Text.FontCollection _fonts = new
GrapeCity.Documents.Text.FontCollection();
```

```
   static WindowsFontResolver()
   {
        GrapeCity.Documents.Text.Windows.FontLinkHelper.UpdateFontLinks(_fonts, true);
        _fonts.DefaultFont = _fonts.FindFamilyName("Arial");
   }
   public static GrapeCity.ActiveReports.IFontResolver Instance = new WindowsFontResolver();
   private WindowsFontResolver() { }
   GrapeCity.Documents.Text.FontCollection GrapeCity.ActiveReports.IFontResolver.GetFonts(string
familyName, bool isBold, bool isItalic)
   {
        var fonts = new GrapeCity.Documents.Text.FontCollection();
        fonts.Add(_fonts.FindFamilyName(familyName, isBold, isItalic) ?? _fonts.DefaultFont);
        GrapeCity.Documents.Text.Windows.FontLinkHelper.UpdateEudcLinks(fonts);
        return fonts;
   }
}
```

Example 2

```
public sealed class DirectoryFontResolver : GrapeCity.ActiveReports.IFontResolver
{
   static readonly GrapeCity.Documents.Text.FontCollection _fonts = new
GrapeCity.Documents.Text.FontCollection();
   static DirectoryFontResolver()
   {
      // see https://developers.redhat.com/blog/2018/11/07/dotnet-special-folder-api-linux/

_fonts.RegisterDirectory(System.Environment.GetFolderPath(System.Environment.SpecialFolder.Fonts));
      _fonts.DefaultFont = _fonts.FindFamilyName("Arial");
   }
   public static GrapeCity.ActiveReports.IFontResolver Instance = new DirectoryFontResolver();
   private DirectoryFontResolver() { }
   GrapeCity.Documents.Text.FontCollection GrapeCity.ActiveReports.IFontResolver.GetFonts(string
familyName, bool isBold, bool isItalic)
   {
      var fonts = new GrapeCity.Documents.Text.FontCollection();
      var font = _fonts.FindFamilyName(familyName, isBold, isItalic);
      if (font != null) fonts.Add(font);
      fonts.Add(_fonts.DefaultFont);
      return fonts;
   }
}
```

See the FontResolver sample for more details.

Link one Font to Another

If you need to link just one font to another (for example, link an EUDC font to an installed font), you can do it with DsPdf and DsImaging APIs, using the following code.

C# Code. PASTE to the beginning of the Main function

```
GrapeCity.Documents.Text.Windows.FontLinkHelper.UpdateFontLinks(null, true);
var fonts = new System.Collections.Generic.List();
GrapeCity.Documents.Text.FontCollection.SystemFonts.SelectByFamilyName("MS UI Gothic", fonts);
if (fonts.Count > 0)
 {

    var eudcFonts = new GrapeCity.Documents.Text.FontCollection();
    using (var stream = new System.IO.FileStream(@"C:\EudcFonts\DFHSG3J.tte",
System.IO.FileMode.Open, System.IO.FileAccess.Read, System.IO.FileShare.Read))
    eudcFonts.LoadFonts(stream);
    foreach (var font in fonts)
      {

        font.ClearEudcFontLinks();
        foreach (var eudcFont in eudcFonts) font.AddEudcFont(eudcFont);
      }
  }
```

Custom Resource Locator

Reports can depend on external resources. ActiveReports provides the possibilities to resolve reporting tasks, related to using any resources that your report may require. With the ActiveReports **custom resource locator**, you can place any resources - themes, image files, subreports, CSS, etc, to any location in your file system.

In ActiveReports, you can place resources such as themes, image files, subreports, CSS, to any location with some exceptions:

- Section report requires using scripts to bind subreports.
- Some data source types require using connection strings with absolute paths.

Page and RDLX reports can resolve resources from your file system using file paths, but sometimes resources are preserved in very specific sources, such as a database. With RDLX report, you can create a custom resource locator to read any resources that might be required by your reports from any type of location. You can use it for resources such as images and theme files, or for reports to use in drillthrough links, subreports, or master reports.

The Custom Resource Locator sample demonstrates a custom resource locator that looks for files in the current user's **My Pictures** folder by looking for a special MyPictures protocol. The custom resource locator in the sample is implemented by deriving from GrapeCity.ActiveReports.ResourceLocator class and overriding the **GetResource ('GetResource Method' in the on-line documentation)** method. The **GetResource** method returns **ParentUri ('ParentUri Property' in the on-line documentation)** and **Value ('Value Property' in the on-line documentation)** properties. The **Value** property contains the located resource as a memory stream. The **ParentUri** property contains the string URI of the parent of the resource within the resource hierarchy.

You can customize the resource locator for all report types in the following ActiveReports components:

- Windows Forms Designer and Windows Forms Viewer
- WPF Viewer
- all ASP.NET viewer backends
- all exports

See the **ResourceLocator ('ResourceLocator Property' in the on-line documentation)** property for details.

> **Note**: The WebDesigner uses a completely different mechanism to access resources. See the WebDesigner_CustomStore sample for details.

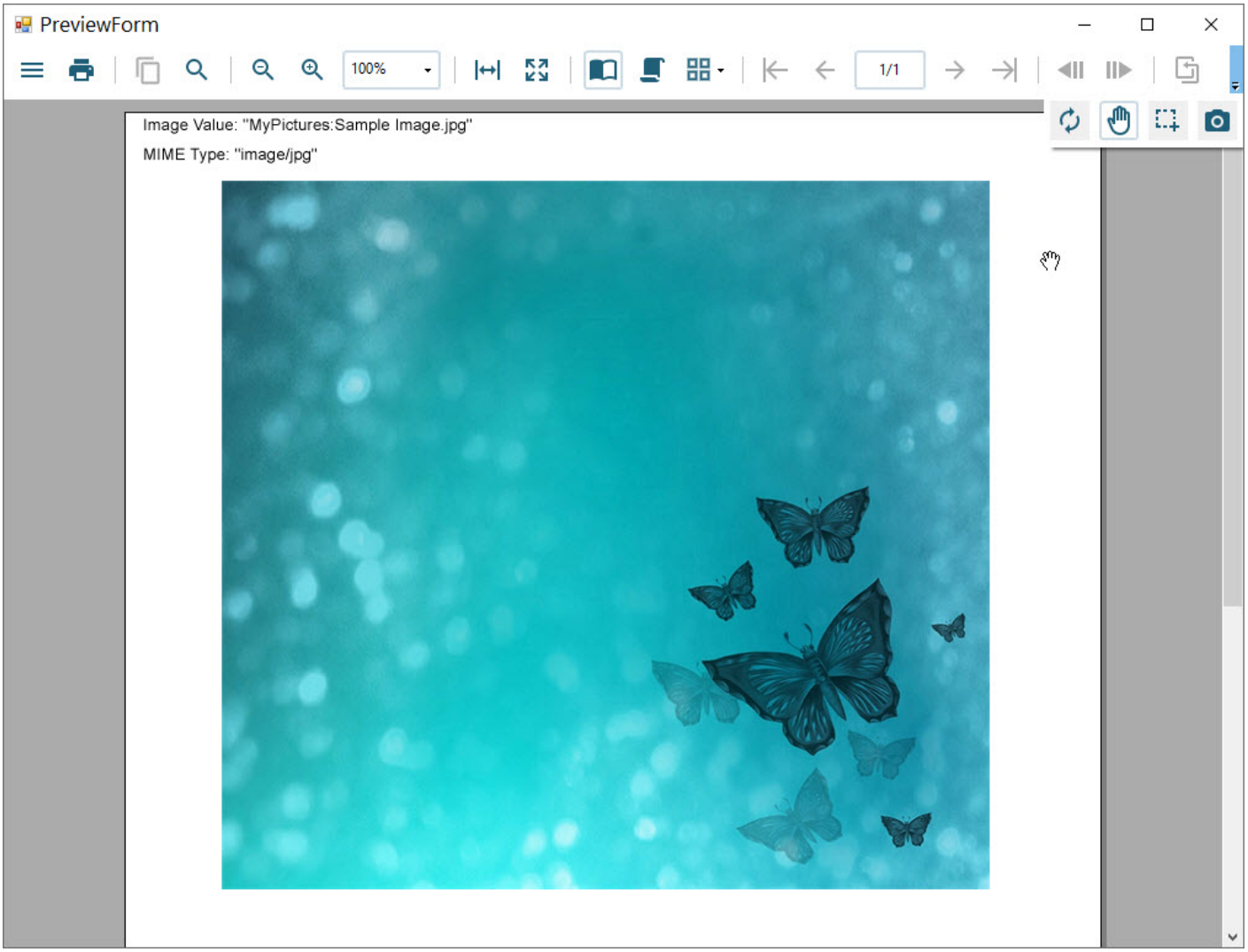
Load Pictures from My Pictures directory

Page reports can get resources from your file system using file paths, but sometimes resources are preserved in very specific sources, such as a database. With Page reports, you can create a custom resource locator to read any resources that might be required by your reports from any location.

The following steps illustrate how to load pictures from the user's **My Pictures** or **Pictures** directory based on the **Custom Resource Locator** sample.

> **Note:** Although these steps use Page reports, you can also implement this using RDLX reports.

When you complete these steps, you get a layout that looks similar to the following at run time.



Add an ActiveReports to the Visual Studio project

1. Create a new Visual Studio Windows Forms Application project.
2. From the **Project** menu, select **Add New Item**.
3. In the Add New Item dialog that appears, select **ActiveReports 18 Page report** and in the Name field, rename the file as DemoReport.rdlx.
4. Click the **Add** button to open a new Page Report.

Create a layout for the report

1. From the toolbox, drag an Image control onto the design surface and in the Properties panel, set the following properties.

Property Name	Property Value
Name	Image1
Location	0.1in, 0.1in
Size	2.8in, 2.8in
Value	MyPictures:Penguins.jpg

2. From the toolbox, drag another Image control onto the design surface and in the Properties panel, set the following properties.

Property Name	Property Value
Name	Image2
Location	3.1in, 0.1in
Size	2.8in, 2.8in
Value	MyPictures:Desert.jpg

3. In the Solution Explorer, select DemoReport.rdlx and in the Properties panel, set **Build Action** to **Embedded Resource**.

Add the new MyPicturesLocator class

1. In the Solution Explorer window, right-click on your project name and select **Add** and then **New Item**.
2. In the **Add New Item** dialog that appears, select **Class**.
3. Change the name of the class to **MyPicturesLocator** and click the **Add** button.
4. Replace the existing code with the following code to the new class.

To write the code in Visual Basic.NET

VB code. Paste on TOP

```
Imports System
Imports System.Drawing
Imports GrapeCity.ActiveReports.Extensibility
Imports System.Globalization
Imports System.IO
```

```
Imports System.Runtime.InteropServices
```

VB code. Paste INSIDE the class

```
 Inherits ResourceLocator

    Private Const UriSchemeMyImages As String = "MyPictures:"

    ' Obtain and return the resource.
    Public Overrides Function GetResource(resourceInfo As ResourceInfo) As Resource
        Dim name As String = resourceInfo.Name
        If name Is Nothing OrElse name.Length = 0 Then
            Throw New ArgumentException("The name of resource to be obtained should be
non-empty string.", "name")
        End If
        Dim uri As New Uri(name)
        Dim stream As Stream = GetPictureFromSpecialFolder(name)
        If stream Is Nothing Then
            stream = New MemoryStream()
        End If
        Return New Resource(stream, uri)
    End Function

    ' Returns the specified image from Public Pictures folder.
    Private Shared Function GetPictureFromSpecialFolder(path As String) As Stream
        Dim startPathPos As Integer = UriSchemeMyImages.Length
        If startPathPos >= path.Length Then
            Return Nothing
        End If
        Dim pictureName As String = path.Substring(startPathPos)
        Dim myPicturesPath As String = Environment.GetEnvironmentVariable("public") &
"\Pictures"
        If Not myPicturesPath.EndsWith("\") Then
            myPicturesPath += "\"
        End If
        Dim picturePath As String = System.IO.Path.Combine(myPicturesPath, pictureName)
        If Not File.Exists(picturePath) Then
            Return Nothing
        End If
        Dim stream As New MemoryStream()
        Try
            Dim picture As Image = Image.FromFile(picturePath)
            picture.Save(stream, picture.RawFormat)
            stream.Position = 0
        Catch generatedExceptionName As OutOfMemoryException
            ' The file is not valid image, or GDI+ doesn't support such images.
            Return Nothing
        Catch generatedExceptionName As ExternalException
            Return Nothing
```

```
    End Try
    Return stream
End Function
```

To write the code in C#

C# code. Paste on TOP

```
using System;
using System.Drawing;
using System.Globalization;
using System.IO;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using GrapeCity.ActiveReports.Extensibility;
using your_project_name.Properties;
```

C# code. Paste BELOW the Using statements

```
namespace your_project_name
{
    // Look for the resources in My Pictures folder.
    internal sealed class MyPicturesLocator : ResourceLocator
    {
        private const string UriSchemeMyImages = "MyPictures:";
        // Obtain and return the resource.
        public override Resource GetResource(ResourceInfo resourceInfo)
        {
            string name = resourceInfo.Name;
            if (name == null || name.Length == 0)
            {
                throw new ArgumentException("The name of resource to be obtained should
be non-empty string.", "name");
            }
            Uri uri = new Uri(name);
            Stream stream = GetPictureFromSpecialFolder(name);
            if (stream == null)
            {
               stream = new MemoryStream();
            }
            return new Resource(stream, uri);
        }
        // Returns the specified image from Public Pictures folder.
        private static Stream GetPictureFromSpecialFolder(string path)
        {
            int startPathPos = UriSchemeMyImages.Length;
            if (startPathPos >= path.ToString().Length)
            {
                return null;
```

```
            }
            string pictureName = path.ToString().Substring(startPathPos);
            string myPicturesPath = Environment.GetEnvironmentVariable("public") +
\\Pictures;
            if (!myPicturesPath.EndsWith("\\")) myPicturesPath += "\\";
            string picturePath = Path.Combine(myPicturesPath, pictureName);
                        if (!File.Exists(picturePath)) return null;
            MemoryStream stream = new MemoryStream();
            try
            {
                Image picture = Image.FromFile(picturePath);
                picture.Save(stream, picture.RawFormat);
                stream.Position = 0;
            }
            catch (OutOfMemoryException) // The file is not valid image, or GDI+ doesn't
support such images.
            {
                return null;
            }
            catch (ExternalException)
            {
                return null;
            }
            return stream;
        }
    }
}
```

Create the PreviewForm

1. In the Solution Explorer, select the Form1 in the Design view and in the Properties panel, set the properties as follows.

Property Name	Property Value
Name	PreviewForm
Text	Preview Form
Size	1015, 770

2. From the Visual Studio toolbox, drag the Viewer control onto the PreviewForm and in the Properties panel, set the following properties.

Property Name	Property Value
Name	reportPreview1
Dock	Fill

3. Double-click the PreviewForm to create an instance for the Load event and add the following code.

   To write the code in Visual Basic.NET

VB code. Paste BELOW the Import statements

```
Imports GrapeCity.ActiveReports.Document
Imports System.IO
Imports GrapeCity.ActiveReports
```

VB code. Paste INSIDE the Load event

```
Dim reportData As Stream = [GetType]
().Assembly.GetManifestResourceStream("your_project_name.DemoReport.rdlx")
reportData.Position = 0
Dim reader As New StreamReader(reportData)
Dim def As New PageReport(reader)
def.ResourceLocator = New MyPicturesLocator()
Dim runtime As New PageDocument(def)
reportPreview1.ReportViewer.LoadDocument(runtime)
```

To write the code in C#

C# code. Paste BELOW the Using statements

```
using GrapeCity.ActiveReports.Document;
using System.IO;
using GrapeCity.ActiveReports;
```

C# code. Paste INSIDE the Load event

```
string myPicturesPath = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
Stream reportData =
GetType().Assembly.GetManifestResourceStream("your_project_name.DemoReport.rdlx");
reportData.Position = 0;
StreamReader reader = new StreamReader(reportData);
PageReport def = new PageReport(reader);
def.ResourceLocator = new MyPicturesLocator();
PageDocument runtime = new PageDocument(def);
reportPreview1.ReportViewer.LoadDocument(runtime);
```

4. Press **F5** to run the project.

Custom Tile Provider

You can add and configure a Custom Tile Provider in the Map control using the **IMapTileProvider ('IMapTileProvider Interface' in the on-line documentation)** and **IMapTile ('IMapTile Interface' in the on-line documentation)** interfaces.

The **IMapTileProvider** interface contains detailed settings that are required to communicate with the tile server, whereas the **IMapTile** interface represents a single tile of a Map's tile layer that fetches the tile image based on the configurations in the **IMapTileProvider** interface.

Set a Custom Tile Provider

Adding a custom tile provider also requires making some modifications in the ActiveReports.config file. Follow these steps to learn how to set a custom tile provider:

1. Create a Class Library Project, for example **MyClassLib**, in Visual Studio.
2. Add a new **Class** to the project and name the class, for example, **MyTileProvider**. You may add functions and features to this class for getting the Tile images based on your tile server settings and details. This class serves as the interface between your Map control and your custom tile server. Replace the existing code with the following in the **MyTileProvider** class to implement the **IMapTileProvider** interface.

To write the code in Visual Basic.NET

VB code. Paste on TOP

```
Imports System
Imports System.Collections.Specialized
Imports GrapeCity.ActiveReports.Extensibility.Rendering.Components.Map
```

VB code. Paste BELOW the Imports statements

```
Namespace MyClassLib
   Public Class MyTileProvider Implements IMapTileProvider
        ' Tile provider settings, like ApiKey, Language, Style and etc.
        Public Property Settings() As NameValueCollection
        ' Add your code here.
        End Property
        ' Get instance of tile by specifying tile coordinates and details.
        Public Sub GetTile(key As MapTileKey, success As Action(Of IMapTile),
 [error] As Action(Of Exception))
        ' Add your code here.
        End Sub
    End Class

End Namespace
```

To write the code in C#

C# code. Paste on TOP

```
using System;
using System.Collections.Specialized;
using GrapeCity.ActiveReports.Extensibility.Rendering.Components.Map;
```

C# code. Paste BELOW the Using statements

```
namespace MyClassLib
{
    public Class MyTileProvider :IMapTileProvider
    {// Tile provider settings, like ApiKey, Language, Style and etc.
    public NameValueCollection Settings { get; private set;}
    // Get instance of tile by specifying tile coordinates and details.
```

```
    public void GetTile(MapTileKey key, Action<IMapTile> success, Action<Exception>
error);
    // Add your code here.
    }
}
```

3. Add a new **Class** to the project and name the class, for example, **MyMapTile**. Replace the existing code with the following in the **MyMapTile** class to implement the **IMapTile** interface.

To write the code in Visual Basic.NET

VB code. Paste on TOP

```
Imports System.IO
Imports GrapeCity.ActiveReports.Extensibility.Rendering.Components.Map
```

VB code. Paste BELOW the Imports statements

```
Namespace MyClassLib
   Public Class MyMapTile Implements IMapTile
       ' Gets the tile identifier
       Public Property Id() As MapTileKey
       ' Add your code here
       End Property
       ' Gets the tile image stream.
       Public Property Image() As Stream
       ' Add your code here.
       End Property
   End Class
End Namespace
```

To write the code in C#

C# code. Paste on TOP

```
using System.IO;
using GrapeCity.ActiveReports.Extensibility.Rendering.Components.Map;
```

C# code. Paste BELOW the Using statements

```
namespace MyClassLib
   public class MyMapTile : IMapTile
   {   // Gets the tile identifier.
       public MapTileKey Id { get; private set; }
       // Gets the tile image stream.
       public Stream Image { get; private set; }
       // Add your code here.
    }
}
```

4. Add another **Class** to the project and name the class, for example, **WebRequestHelper**. Replace the existing code with the following in the **WebRequestHelper** class to implement the loading of raw website data into the

System.IO.MemoryStream class.

To write the code in Visual Basic.NET

VB code. Paste on TOP

```
Imports System.IO
Imports System.Net
```

VB code. Paste BELOW the Imports statements

```
Namespace MyClassLib
    Module StringExtensions
        Public Sub CopyTo(ByVal input As Stream, ByVal output As Stream)
        'Add your code here
        End Sub
        Private Function InlineAssignHelper(Of T)(ByRef target As T, value As T) As T
        'Add your code here
        End Function
    End Module
    Friend NotInheritable Class WebRequestHelper
        Private Sub New()
        End Sub
        ' Load raw data into MemoryStream from specified Url.
        Public Shared Function DownloadData(url As String, timeoutMilliseconds As
Integer) As Stream
        'Add your code here
        End Function
        'Load raw data into MemoryStream from specified Url.
        Public Shared Sub DownloadDataAsync(url As String, timeoutMilliseconds As
Integer,
 success As Action(Of MemoryStream), [error] As Action(Of Exception))
        'Add your code here
        End Sub
        Private Shared Function InlineAssignHelper(Of T)(ByRef target As T, value As T)
As T
        'Add your code here
        End Function
    End Class
End Namespace
```

To write the code in C#

C# code. Paste on TOP

```
using System.IO;
using System.Net;
```

C# code. Paste BELOW the Using statements

```
namespace MyClassLib
```

```
{
    internal static class WebRequestHelper
    {   // Load raw data into MemoryStream from specified Url.
        public static Stream DownloadData(string url, int timeoutMilliseconds)
        {//Add your code here  }
        public static void DownloadDataAsync(string url, int timeoutMilliseconds,
 Action<MemoryStream> success, Action<Exception> error)
        {//Add your code here  }
        public static void CopyTo(this Stream input, Stream output)
        {//Add your code here  }
    }
}
```

5. Save and build your class library project and locate the new .dll file in its **Bin>Debug** folder. This file has the same name as your class library project, with a .dll extension.
6. Create a Basic End User Designer in a new solution following the steps in Creating a Basic End User Designer.
7. Run your Basic End User Designer project to create a EndUserDesigner.exe in your projects **Bin>Debug** folder.
8. Copy the ActiveReports.config file from the *C:\Program Files (x86)\MESCIUS\ActiveReports 18* location and paste it into your End User Designer project's **Bin>Debug** folder.

   > ⚠ **Caution**: ActiveReports.config file should always be placed inside the same folder as the EndUserDesigner.exe file for displaying a tile layer on a Map.

9. Right-click on the ActiveReports.config file and select **Include in this Project** to make changes in the config file.
10. Double-click to open the ActiveReports.config file and paste the following code between the **<Configuration>** and **</Configuration>** tags:

    Paste between the <Configuration></Configuration> tags.

    ```
    <!-- Register and configure custom tile provider. -->
    <MapTileProvider Name="Custom" DisplayName="Custom Provider" type="YourTileProvider,
    AssemblyName,
     Version = x.x.x.x">
        <Settings>
          <add key="ApiKey" value="API Key" />
        </Settings>
    </MapTileProvider>
    ```

    > **Note:** Replace ***YourTileProvider*** with fully qualified class name and ***AssemblyName*** with the name of the assembly created after implementing IMapTileProvider and IMapTile interfaces.

11. Add the Class Library project created in step 5 to your Basic End User Designer project.
12. Copy the *YourProjectName.dll* created in step 5 and paste it to the current project's **Bin > Debug** folder together with the EndUserDesigner.exe.
13. Save and Run the project.
14. Create a Report containing a Map control in the **Basic End User Designer**. See Map Data Region in Reports for more information.
15. Add a Tile layer to the Map control. Right click the Tile layer and select **Edit** to view the custom tile provider added in the Provider drop-down. See Tile Layer for more information.

Custom Data Provider

Data sources customization in ActiveReports is one of the most popular demands. Our Oracle Data Provider sample demonstrates the custom data providers implementation details.

> **Note**: Our built-in Visual Query Designer supports only a limited number of data providers or the SQL dialect. Thus, in most cases you need to use a plain data provider without any additional schema and other implementations.

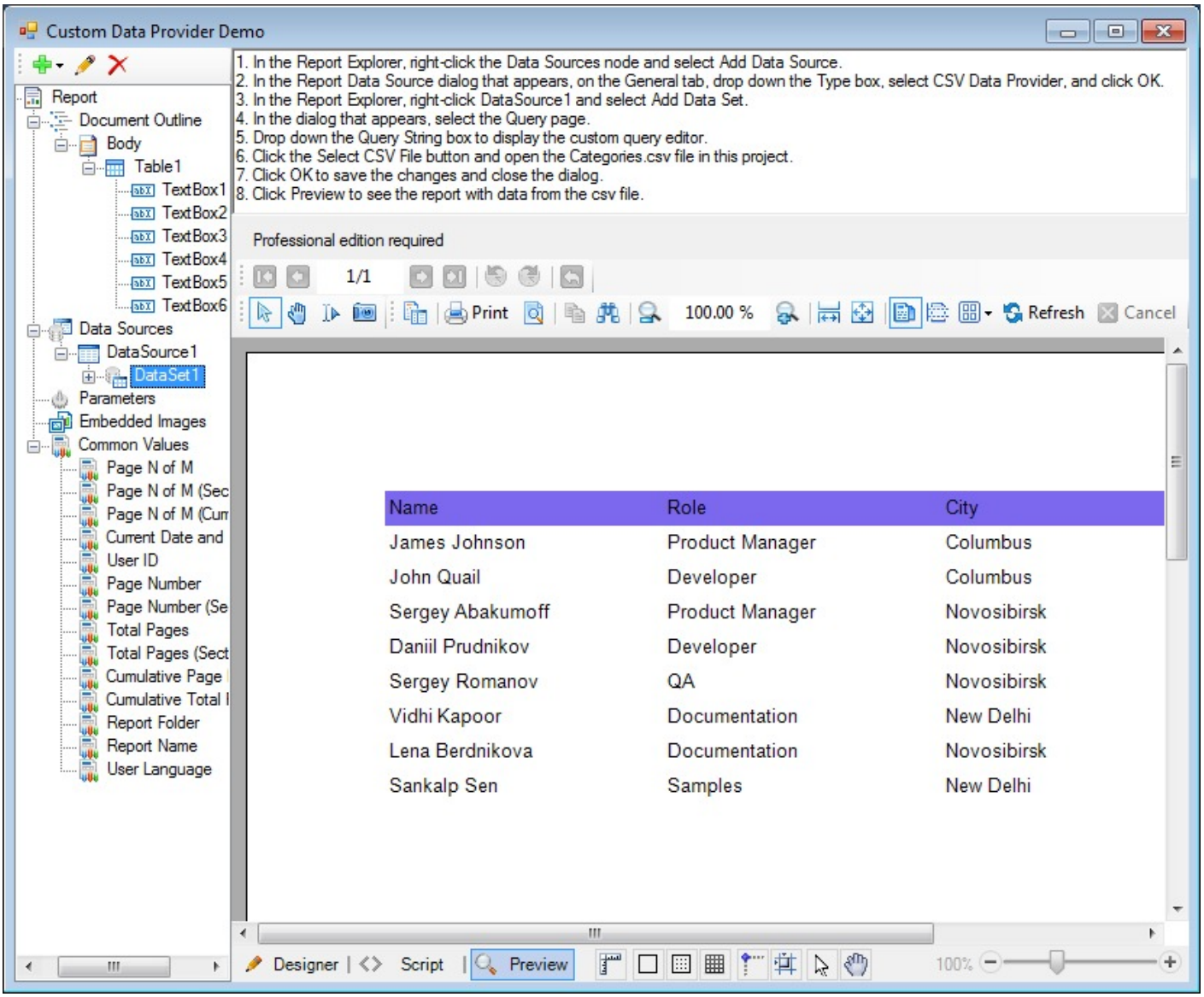
To support a custom data provider in Web, you need to perform some additional steps:

- Specify configuration with the **UseConfig ('UseConfig Method' in the on-line documentation)** method.
- Specify new data providers on both server and client sides for the WebDesigner. See our Custom Data Providers sample for details.

Create a Custom Data Provider

A custom data provider allows you to use non-traditional data sources in your Page reports, both at run time and at design time. Let us learn how to create a solution with projects that create a custom data provider and demonstrate how it pulls data from a comma separated values (CSV) file.

When you complete these steps, you will have a designer pre-loaded with a report that pulls data from a CSV file and looks like the following.



Create a Designer project to demonstrate the custom data provider

1. In Visual Studio, create a Windows Forms project and name it **CustomDataProviderDemo**.
2. From the Visual Studio toolbox ActiveReports 18 tab, drag a ReportExplorer and drop it onto the default Windows form, resizing the form to a comfortable working area.
3. In the Properties window, set the **Dock** property of the ReportExplorer control to **Left**.
4. From the toolbox Common Controls tab, drag a RichTextBox control onto the form and set the **Dock** property to **Top**.
5. Add the following text to the **Text** property. (Drop down the box to ensure that all of the lines of text are added.)

Text. Paste in the Text property of the RichTextBox.

```
1. In the Report Explorer, right-click the Data Sources node and select Add Data Source.
2. In the Report Data Source dialog that appears, on the General tab, drop down the Type box, select CSV Data Provider, and click OK.
```

```
3. In the Report Explorer, right-click DataSource1 and select Add Data Set.
4. In the dialog that appears, select the Query page.
5. Drop down the Query String box to display the custom query editor.
6. Click the Select CSV File button and open the Categories.csv file in this project.
7. Click OK to save the changes and close the dialog.
8. Click Preview to see the report with data from the csv file.
```

6. From the toolbox ActiveReports 18 tab, drag a Designer control and drop it on the empty part of the form.
7. Set the **Dock** property to **Fill**, then right-click the Designer control on the form and select **Bring to front**.
8. Select the ReportExplorer control and in the Properties window, drop down the **ReportDesigner** property and select **Designer1**.
9. Double-click on the form's title bar to create a form Load event, and add code like the following above the class.

Visual Basic.NET code. Paste above the class.

```
Imports System.Xml
Imports System.IO
Imports GrapeCity.ActiveReports.Design
```

C# code. Paste above the class.

```
using System.Xml;
using System.IO;
using GrapeCity.ActiveReports.Design;
```

10. Add the following code to the form Load event.

Visual Basic.NET code. Paste inside the form Load event.

```
Using reportStream = File.OpenRead("DemoReport.rdlx")
        Using reader = XmlReader.Create(reportStream)
                Designer1.LoadReport(reader, DesignerReportType.Page)
        End Using
End Using
```

C# code. Paste inside the form Load event.

```
using (var reportStream = File.OpenRead("DemoReport.rdlx"))
 {
     using (var reader = XmlReader.Create(reportStream))
      {
         designer1.LoadReport(reader, DesignerReportType.Page);
      }
 }
```

Configure the project to use a custom data provider

1. In the Solution Explorer, right-click the project and select **Add**, then **New Item**.
2. In the dialog that appears, select **Text File**, name it **ActiveReports.config**, and click **Add**.
3. In the Solution Explorer, select the new file and in the Properties window, set its **Copy to Output Directory** property to **Copy always**.
4. Paste the following text into the file and save it. (You can safely ignore the warning that the 'Configuration' element is not declared.)

Paste into the config file.

```
<?xml version="1.0" encoding="utf-8" ?>
<Configuration>
  <Extensions>
    <Data>
      <Extension Name="CSV" DisplayName="CSV Data Provider"
        Type="CustomDataProvider.CsvDataProvider.CsvDataProviderFactory,
          CustomDataProvider"
          CommandTextEditorType="CustomDataProvider.CSVDataProvider.QueryEditor,
        CustomDataProvider"/>
    </Data>
     </Extensions>
</Configuration>
```

5. In the Solution Explorer, right-click the project and select **Add**, then **New Item**.
6. In the dialog that appears, select **Text File**, name it **Categories.csv**, and click **Add**.
7. In the Solution Explorer, click to select the file, and in the Properties window, change the **Copy to Output Directory** property to **Copy always**.
8. Paste the following text into the file and save it.

Paste into the text file.

```
EmployeeID(int32),LastName,FirstName,Role,City
1,James,Yolanda,Owner,Columbus
7,Reed,Marvin,Manager,Newton
9,Figg,Murray,Cashier,Columbus
12,Snead,Lance,Store Keeper,Columbus
15,Halm,Jeffry,Store Keeper,Columbus
17,Hames,Alma,Store Keeper,Oak Bay
18,Nicki,Aubrey,Store Keeper,Columbus
24,Cliett,Vikki,Store Keeper,Newton
```

Add a report to show the data from the custom data provider

1. In the Solution Explorer, right-click the project and select **Add**, then **New Item**.
2. In the dialog that appears, select **ActiveReports 18 RDLX report**, name it **DemoReport**, and click **Add**.
3. In the Solution Explorer, click to select the report, and in the Properties window, change the **Copy to Output Directory** property to **Copy always**.
4. From the ActiveReports 18 RDLX report Toolbox, drag a Table report control onto the report.

   > **Note**: In case you are still working on Page Report layout, set the FixedSize property of the Table control to display all data on one page.

5. Click inside the table to reveal the table adorners, then right-click the table adorner to the left of the footer row and select **Delete Rows**. The footer row is removed from the table.
6. In the Report Explorer, select each of the textboxes in turn and set the properties as in the following table. (If you do not see the Report Explorer, from the View menu, select Other Windows, then Report Explorer.)

TextBox Name	Value Property	BackgroundColor Property
TextBox1	Name	MediumSlateBlue

TextBox2	Role	MediumSlateBlue
TextBox3	City	MediumSlateBlue
TextBox4	=Fields!FirstName.Value & " " & Fields!LastName.Value	
TextBox5	=Fields!Role.Value	
TextBox6	=Fields!City.Value	

7. In the Report Explorer, select the Table1 node and in the Properties window, set the **Location** property to 0in, 1in and the **Size** property to **6in, 0.5in** to make the table wide enough to see all of the data.
8. With Table1 still selected in the Properties window, in the **DataSetName** property, enter the text **DataSet1**.

Add a class library project to the solution to contain the custom data provider

1. From the File menu, select **Add**, then **New Project**.
2. In the Add New Project dialog, select **Class Library**, and name the project **CustomDataProvider**.
3. In the Solution Explorer, right-click the default class and select **Delete**. (We will add our classes to a folder below.)
4. Install packages from **nuget** as follows:
   i) Go to **Tools** > **Nuget Package Manager** > **Manage Nuget Packages for Solution**...
   ii) Browse the following packages one by one and click Install.
   MESCIUS.ActiveReports
   MESCIUS.ActiveReports.Extensibility
5. Right-click the CustomDataProvider project and select **Add**, then **New Folder**, and name the folder **CSVDataProvider**.
6. Right-click the folder and select **Add**, then **Class**, then name the class **CsvColumn** and add code like the following to replace the default stub in the class.

Visual Basic.NET code. Paste it to replace the default stub in the class.

```
Namespace CSVDataProvider
        ' Represents information about fields in the data source.
        Friend Structure CsvColumn
                Private ReadOnly _fieldName As String
                Private ReadOnly _dataType As Type
                ' Creates a new instance of the CsvColumn class.
                ' The fieldName parameter is the name of the field represented by this
instance of the CsvColumn.
                ' The dataType parameter is the Type of the field represented by this
instance of the CsvColumn.
                Public Sub New(fieldName As String, dataType As Type)
                        If fieldName Is Nothing Then
                                Throw New ArgumentNullException("fieldName")
                        End If
                        If dataType Is Nothing Then
                                Throw New ArgumentNullException("dataType")
                        End If
```

```
                        _fieldName = fieldName
                        _dataType = dataType
                End Sub

                ' Gets the name of the field represented by this instance of the
CsvColumn.
                Public ReadOnly Property FieldName() As String
                        Get
                                Return _fieldName
                        End Get
                End Property

                ' Gets the the Type of the field represented by this instance of the
CsvColumn.
                Public ReadOnly Property DataType() As Type
                        Get
                                Return _dataType
                        End Get
                End Property

                ' Returns a String that represents this instance of the CsvColumn.
                Public Overrides Function ToString() As String
                        Return [String].Concat(New String() {FieldName, "(",
DataType.ToString(), ")"})
                End Function

                ' Determines whether two CsvColumn instances are equal.
                ' The obj represents the CsvColumn to compare with the current
CsvColumn.
                ' Returns True if the specified CsvColumn is equal to the current
CsvColumn; otherwise, False.
                Public Overrides Function Equals(obj As Object) As Boolean
                        Dim flag As Boolean
                        If TypeOf obj Is CsvColumn Then
                                flag = Equals(CType(obj, CsvColumn))
                        Else
                                flag = False
                        End If
                        Return flag
                End Function
                Private Overloads Function Equals(column As CsvColumn) As Boolean
                        Return column.FieldName = FieldName
                End Function

                ' Serves as a hash function for a CsvColumn, suitable for use in hashing
algorithms and data structures like a hash table.
                ' Returns a hash code for the current CsvColumn instance.
                Public Overrides Function GetHashCode() As Integer
```

```
                        Return (FieldName.GetHashCode() + DataType.GetHashCode())
                End Function
        End Structure
End Namespace
```

C# code. Paste it to replace the default stub in the class.

```
using System;
namespace CustomDataProvider.CSVDataProvider
{
        // Represents information about fields in the data source.
        internal struct CsvColumn
        {
                private readonly string _fieldName;
                private readonly Type _dataType;
                // Creates a new instance of the CsvColumn class.
                // The fieldName parameter is the name of the field represented by this
instance of the CsvColumn.
                // The dataType parameter is the Type of the field represented by this
instance of the CsvColumn.
                public CsvColumn(string fieldName, Type dataType)
                {
                        if (fieldName == null)
                                throw new ArgumentNullException("fieldName");
                        if (dataType == null)
                                throw new ArgumentNullException("dataType");
                        _fieldName = fieldName;
                        _dataType = dataType;
                }

                // Gets the name of the field represented by this instance of the
CsvColumn.
                public string FieldName
                {
                        get { return _fieldName; }
                }

                // Gets the the Type of the field represented by this instance of the
CsvColumn.
                public Type DataType
                {
                        get { return _dataType; }
                }

               // Returns a String that represents this instance of the CsvColumn.
                public override string ToString()
                {
                        return String.Concat(new string[] {FieldName, "(",
DataType.ToString(), ")"});
```

```
                }

                // Determines whether two CsvColumn instances are equal.
                // The obj represents the CsvColumn to compare with the current
CsvColumn.
                // Returns True if the specified CsvColumn is equal to the current
CsvColumn; otherwise, False.
                public override bool Equals(object obj)
                {
                        bool flag;
                        if (obj is CsvColumn)
                        {
                                flag = Equals((CsvColumn) obj);
                        }
                        else
                        {
                                flag = false;
                        }
                        return flag;
                }
                private bool Equals(CsvColumn column)
                {
                        return column.FieldName == FieldName;
                }

                // Serves as a hash function for a CsvColumn, suitable for use in
hashing algorithms and data structures like a hash table.
               // Returns a hash code for the current CsvColumn instance.
                public override int GetHashCode()
                {
                        return (FieldName.GetHashCode() + DataType.GetHashCode());
                }
        }
}
```

7. Right-click the CSVDataProvider folder and select **Add**, then **Class**, then name the class **CsvDataReader** and add code like the following to replace the default stub in the class.

Visual Basic.NET code. Paste it to replace the default stub in the class.

```
Imports System
Imports System.Collections
Imports System.Globalization
Imports System.IO
Imports System.Text.RegularExpressions
Imports GrapeCity.ActiveReports.Extensibility.Data
Namespace CSVDataProvider
    ' Provides an implementation of IDataReader for the .NET Framework CSV Data
Provider.
    Friend Class CsvDataReader
```

```
        Implements IDataReader
        'NOTE: HashcodeProvider and Comparer need to be case-insensitive since TypeNames
are capitalized differently in places.
        'Otherwise data types end up as strings when using Int32 vs int32.
        Private _typeLookup As New
Hashtable(StringComparer.Create(CultureInfo.InvariantCulture, False))
        Private _columnLookup As New Hashtable()
        Private _columns As Object()
        Private _textReader As TextReader
        Private _currentRow As Object()

        'The regular expressions are set to be pre-compiled to make it faster. Since we
were concerned about
        'multi-threading, we made the properties read-only so no one can change any
properties on these objects.
        Private Shared ReadOnly _rxDataRow As New Regex(",(?=(?:[^""]*""[^""]*"")*(?!
[^""]*""))", RegexOptions.Compiled)
        'Used to parse the data rows.
        Private Shared ReadOnly _rxHeaderRow As New Regex("(?<fieldName>(\w*\s*)*)\((?
<fieldType>\w*)\)", RegexOptions.Compiled)
        'Used to parse the header rows.

        ' Creates a new instance of the CsvDataReader class.
        ' The textReader parameter represents the TextReader to use to read the data.
        Public Sub New(textReader As TextReader)
            _textReader = textReader
            ParseCommandText()
        End Sub

        ' Parses the passed-in command text.
        Private Sub ParseCommandText()
            If _textReader.Peek() = -1 Then
                Return
            End If
            'Command text is empty or at the end already.
            FillTypeLookup()
            Dim header As String = _textReader.ReadLine()
            header = AddDefaultTypeToHeader(header)
            If Not ParseHeader(header) Then
                Throw New InvalidOperationException( _
                "Field names and types are not defined. " & _
                "The first line in the CommandText must contain the field names and data
types. e.g FirstName(string)")
            End If
        End Sub
        'A hashtable is used to return a type for the string value used in the header
text.
        Private Sub FillTypeLookup()
```

```
            _typeLookup.Add("string", GetType([String]))
            _typeLookup.Add("byte", GetType([Byte]))
            _typeLookup.Add("boolean", GetType([Boolean]))
            _typeLookup.Add("datetime", GetType(DateTime))
            _typeLookup.Add("decimal", GetType([Decimal]))
            _typeLookup.Add("double", GetType([Double]))
            _typeLookup.Add("int16", GetType(Int16))
            _typeLookup.Add("int32", GetType(Int32))
            _typeLookup.Add("int", GetType(Int32))
            _typeLookup.Add("integer", GetType(Int32))
            _typeLookup.Add("int64", GetType(Int64))
            _typeLookup.Add("sbyte", GetType([SByte]))
            _typeLookup.Add("single", GetType([Single]))
            _typeLookup.Add("time", GetType(DateTime))
            _typeLookup.Add("date", GetType(DateTime))
            _typeLookup.Add("uint16", GetType(UInt16))
            _typeLookup.Add("uint32", GetType(UInt32))
            _typeLookup.Add("uint64", GetType(UInt64))
        End Sub

        ' Returns a type based on the string value passed in from the header text
string. If no match is found,
        ' a string type is returned.
        ' The fieldType parameter represents the String value from the header command
text string.
        Private Function GetFieldTypeFromString(fieldType As String) As Type
            If _typeLookup.Contains(fieldType) Then
                Return TryCast(_typeLookup(fieldType), Type)
            End If
            Return GetType([String])
        End Function

        ' Parses the first line in the passed-in command text string to create the field
names and field data types.
        ' The field information is stored in a CsvColumn struct, and these column info
items are stored
        ' in an ArrayList. The column name is also added to a hashtable for easy lookup
later.
        ' The header parameter represents the header string that contains all the
fields.
        ' Returns True if it can parse the header string; otherwise False.
        Private Function ParseHeader(header As String) As Boolean
            Dim fieldName As String
            Dim index As Integer = 0
            If header.IndexOf("(") = -1 Then
                Return False
            End If
            Dim matches As MatchCollection = _rxHeaderRow.Matches(header)
```

```
            _columns = New Object(matches.Count - 1) {}
            For Each match As Match In matches
                fieldName = match.Groups("fieldName").Value
                Dim fieldType As Type =
GetFieldTypeFromString(match.Groups("fieldType").Value)
                _columns.SetValue(New CsvColumn(fieldName, fieldType), index)
                _columnLookup.Add(fieldName, index)
                index += 1
            Next
            Return True
        End Function

        ' Ensures that the header contains columns in the form of name(type)
        ' The line parameter represents the raw header line from the file to fix up.
        ' Returns a modified header with default types appended to column names.
        Private Shared Function AddDefaultTypeToHeader(line As String) As String
            Const ColumnWithDataTypeRegex As String = "[""]?\w+[\""]?\(.+\)"
            Dim columns As String() = line.Split(New String() {","},
StringSplitOptions.None)
            Dim ret As String = Nothing
            For Each column As String In columns
                If Not String.IsNullOrEmpty(ret) Then
                    ret += ","
                End If
                If Not Regex.Match(column, ColumnWithDataTypeRegex).Success Then
                    ret += column + "(string)"
                Else
                    ret += column
                End If
            Next
            Return ret
        End Function

        ' Parses a row of data using a regular expression and stores the information
inside an object
        ' array that is the current row of data.
        ' If the row does not have the correct number of fields, an exception is raised.
        ' The dataRow parameter represents the String value representing a comma
delimited data row.
        ' Returns True if it can parse the data string; otherwise False.
        Private Function ParseDataRow(dataRow As String) As Boolean
            Dim index As Integer = 0
            Dim tempData As String() = _rxDataRow.Split(dataRow)
            _currentRow = New Object(tempData.Length - 1) {}
            If tempData.Length <> _columns.Length Then
                Dim [error] As String = String.Format(CultureInfo.InvariantCulture, _
                                "Invalid row ""{0}"". The row does not contain the same
number of data columns as the table header definition.", dataRow)
```

```
                Throw New InvalidOperationException([error])
            End If
            For i As Integer = 0 To tempData.Length - 1
                Dim value As String = tempData(i)
                If value.Length > 1 Then
                    If value.IndexOf("""c, 0) = 0 AndAlso value.IndexOf("""c, 1) =
value.Length - 1 Then
                        value = value.Substring(1, value.Length - 2)
                    End If
                End If
                _currentRow.SetValue(ConvertValue(GetFieldType(index), value), index)
                index += 1
            Next
            Return True
        End Function

        ' Coverts the string value coming from the command text to the appropriate data
type, based on the field's type.
        ' This also checks a few string value rules to decide if a String.Empty of
System.Data.DBNull needs to be returned.
        ' The type parameter represents the Type of the current column the data belongs
to.
        ' The originalValue parameter represents the String value coming from the
command text.
        ' Returns the object resulting from the converted string, based on the type.
        Private Function ConvertValue(type As Type, originalValue As String) As Object
            Dim fieldType As Type = type
            Dim invariantCulture As CultureInfo = CultureInfo.InvariantCulture
            Try
                If originalValue = """""" OrElse originalValue = " " Then
                    Return String.Empty
                End If
                If originalValue = "" Then
                    Return DBNull.Value
                End If
                If originalValue = "DBNull" Then
                    Return DBNull.Value
                End If
                If fieldType.Equals(GetType([String])) Then
                    Return originalValue.Trim()
                End If
                If fieldType.Equals(GetType(Int32)) Then
                    Return Convert.ToInt32(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType([Boolean])) Then
                    Return Convert.ToBoolean(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType(DateTime)) Then
```

```
                    Return Convert.ToDateTime(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType([Decimal])) Then
                    Return Convert.ToDecimal(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType([Double])) Then
                    Return Convert.ToDouble(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType(Int16)) Then
                    Return Convert.ToInt16(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType(Int64)) Then
                    Return Convert.ToInt64(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType([Single])) Then
                    Return Convert.ToSingle(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType([Byte])) Then
                    Return Convert.ToByte(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType([SByte])) Then
                    Return Convert.ToSByte(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType(UInt16)) Then
                    Return Convert.ToUInt16(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType(UInt32)) Then
                    Return Convert.ToUInt32(originalValue, invariantCulture)
                End If
                If fieldType.Equals(GetType(UInt64)) Then
                    Return Convert.ToUInt64(originalValue, invariantCulture)
                End If
            Catch e As Exception
                Throw New InvalidOperationException(String.Format("Input value '{0}'
could not be converted to the type '{1}'.", originalValue, type), e)
            End Try
            'If no match is found return DBNull instead.
            Return DBNull.Value
        End Function
#Region "IDataReader Members"

       ' Advances the CsvDataReader to the next record.
        ' Returns True if there are more rows; otherwise, False.
        Public Function Read() As Boolean Implements IDataReader.Read
            If _textReader.Peek() > -1 Then
                ParseDataRow(_textReader.ReadLine())
            Else
                Return False
```

```
            End If
            Return True
        End Function
#End Region
#Region "IDisposable Members"

        ' Releases the resources used by the CsvDataReader.
        Public Sub Dispose() Implements IDisposable.Dispose
            Dispose(True)
            GC.SuppressFinalize(Me)
        End Sub
        Private Sub Dispose(disposing As Boolean)
            If disposing Then
                If _textReader IsNot Nothing Then
                    _textReader.Close()
                End If
            End If
            _typeLookup = Nothing
            _columnLookup = Nothing
            _columns = Nothing
            _currentRow = Nothing
        End Sub

        ' Allows an Object to attempt to free resources and perform
        ' other cleanup operations before the Object is reclaimed by garbage collection.
        Protected Overrides Sub Finalize()
            Try
                Dispose(False)
            Finally
                MyBase.Finalize()
            End Try
        End Sub
#End Region
#Region "IDataRecord Members"

        ' Gets the number of columns in the current row.
        Public ReadOnly Property FieldCount() As Integer Implements
IDataRecord.FieldCount
            Get
                Return _columns.Length
            End Get
        End Property

        ' The i parameter represents the index of the field to find.
        ' Returns the Type information corresponding to the type of Object that would be
returned from GetValue.
        Public Function GetFieldType(i As Integer) As Type Implements
IDataReader.GetFieldType
```

```
            If i > _columns.Length - 1 Then
                Return Nothing
            End If
            Return DirectCast(_columns.GetValue(i), CsvColumn).DataType
        End Function

        ' Gets the name for the field to find.
        ' The i parameter represents the index of the field to find.
        ' Returns the name of the field or an empty string (""), if there is no value to
return.
        Public Function GetName(i As Integer) As String Implements IDataRecord.GetName
            If i > _columns.Length - 1 Then
                Return String.Empty
            End If

            Return DirectCast(_columns.GetValue(i), CsvColumn).FieldName
        End Function

        ' The name parameter represents the name of the field to find.
        ' Returns the index of the named field.
        Public Function GetOrdinal(name As String) As Integer Implements
IDataRecord.GetOrdinal
            Dim value As Object = _columnLookup(name)
            If value Is Nothing Then
                Throw New IndexOutOfRangeException("name")
            End If
            Return CInt(value)
        End Function

        ' The i parameter represents the index of the field to find.
        ' Returns the Object which contains the value of the specified field.
        Public Function GetValue(i As Integer) As Object Implements IDataRecord.GetValue
            If i > _columns.Length - 1 Then
                Return Nothing
            End If
            Return _currentRow.GetValue(i)
        End Function
        Public Overridable Function GetData(fieldIndex As Integer) As IDataReader
Implements IDataReader.GetData
            Throw New NotSupportedException()
        End Function
#End Region
    End Class
End Namespace
```

C# code. Paste it to replace the default stub in the class.

```
using System;
using System.Collections;
```

```
using System.Globalization;
using System.IO;
using System.Text.RegularExpressions;
using GrapeCity.ActiveReports.Extensibility.Data;
namespace CustomDataProvider.CSVDataProvider
{
        // Provides an implementation of IDataReader for the .NET Framework CSV Data
Provider.
        internal class CsvDataReader : IDataReader
        {
                //NOTE: HashcodeProvider and Comparer need to be case-insensitive since
TypeNames are capitalized differently in places.
                                //Otherwise data types end up as strings when using
Int32 vs int32.
                private Hashtable _typeLookup =
                        new
Hashtable(StringComparer.Create(CultureInfo.InvariantCulture, false));
                private Hashtable _columnLookup = new Hashtable();
                private object[] _columns;
                private TextReader _textReader;
                private object[] _currentRow;

                //The regular expressions are set to be pre-compiled to make it faster.
Since we were concerned about
                //multi-threading, we made the properties read-only so no one can change
any properties on these objects.
                private static readonly Regex _rxDataRow = new Regex(@",(?=(?:[^""]*""
[^""]*"")*(?![^""]*""))", RegexOptions.Compiled);
                        //Used to parse the data rows.
                private static readonly Regex _rxHeaderRow =
                        new Regex(@"(?<fieldName>(\w*\s*)*)\((?<fieldType>\w*)\)",
RegexOptions.Compiled);
                        //Used to parse the header rows.

                // Creates a new instance of the CsvDataReader class.
                // The textReader parameter represents the TextReader to use to read the
data.
                public CsvDataReader(TextReader textReader)
                {
                        _textReader = textReader;
                        ParseCommandText();
                }

                // Parses the passed-in command text.
                private void ParseCommandText()
                {
                        if (_textReader.Peek() == -1)
                                return; //Command text is empty or at the end already.
```

```
                        FillTypeLookup();
                        string header = _textReader.ReadLine();
                        header = AddDefaultTypeToHeader(header);
                        if (!ParseHeader(header))
                                throw new InvalidOperationException(
                                        "Field names and types are not defined. The
first line in the CommandText must contain the field names and data types. e.g
FirstName(string)");
                }
                //A hashtable is used to return a type for the string value used in the
header text.
                private void FillTypeLookup()
                {
                        _typeLookup.Add("string", typeof (String));
                        _typeLookup.Add("byte", typeof (Byte));
                        _typeLookup.Add("boolean", typeof (Boolean));
                        _typeLookup.Add("datetime", typeof (DateTime));
                        _typeLookup.Add("decimal", typeof (Decimal));
                        _typeLookup.Add("double", typeof (Double));
                        _typeLookup.Add("int16", typeof (Int16));
                        _typeLookup.Add("int32", typeof (Int32));
                        _typeLookup.Add("int", typeof (Int32));
                        _typeLookup.Add("integer", typeof (Int32));
                        _typeLookup.Add("int64", typeof (Int64));
                        _typeLookup.Add("sbyte", typeof (SByte));
                        _typeLookup.Add("single", typeof (Single));
                        _typeLookup.Add("time", typeof (DateTime));
                        _typeLookup.Add("date", typeof (DateTime));
                        _typeLookup.Add("uint16", typeof (UInt16));
                        _typeLookup.Add("uint32", typeof (UInt32));
                        _typeLookup.Add("uint64", typeof (UInt64));
                }

                // Returns a type based on the string value passed in from the header
text string. If no match is found, a string type is returned.
                // The fieldType parameter represents the String value from the header
command text string.
                private Type GetFieldTypeFromString(string fieldType)
                {
                        if (_typeLookup.Contains(fieldType))
                                return _typeLookup[fieldType] as Type;
                        return typeof (String);
                }

                // Parses the first line in the passed-in command text string to create
the field names and field data types. The field information
                // is stored in a CsvColumn struct, and these column info items are
stored in an ArrayList. The column name is also added
```

```
                // to a hashtable for easy lookup later.

                // The header parameter represents the header string that contains all
the fields.
                // Returns True if it can parse the header string; otherwise False.
                private bool ParseHeader(string header)
                {
                        string fieldName;
                        int index = 0;
                        if (header.IndexOf("(") == -1)
                                return false;
                        MatchCollection matches = _rxHeaderRow.Matches(header);
                        _columns = new object[matches.Count];
                        foreach (Match match in matches)
                        {
                                fieldName = match.Groups["fieldName"].Value;
                                Type fieldType =
GetFieldTypeFromString(match.Groups["fieldType"].Value);
                                _columns.SetValue(new CsvColumn(fieldName, fieldType),
index);
                                _columnLookup.Add(fieldName, index);
                                index++;
                        }
                        return true;
                }

                // Ensures that the header contains columns in the form of name(type)
                // The line parameter represents the raw header line from the file to
fix up.
                // Returns a modified header with default types appended to column
names.
                private static string AddDefaultTypeToHeader(string line)
                {
                        const string ColumnWithDataTypeRegex = @"[""]?\w+[\""]?\(.+\)";
                        string[] columns = line.Split(new string[] { "," },
StringSplitOptions.None);
                        string ret = null;
                        foreach (string column in columns)
                        {
                                if (!string.IsNullOrEmpty(ret))
                                        ret += ",";
                                if (!Regex.Match(column,
ColumnWithDataTypeRegex).Success)
                                {
                                        ret += column + "(string)";
                                }
                                else
                                {
```

```
                                        ret += column;
                                }
                        }
                        return ret;
                }

                // Parses a row of data using a regular expression and stores the
information inside an object array that is the current row of data.
                // If the row does not have the correct number of fields, an exception
is raised.
                // The dataRow parameter represents the String value representing a
comma delimited data row.
                // Returns True if it can parse the data string; otherwise False.
                private bool ParseDataRow(string dataRow)
                {
                        int index = 0;
                        string[] tempData = _rxDataRow.Split(dataRow);
                        _currentRow = new object[tempData.Length];
                        if (tempData.Length != _columns.Length)
                        {
                                string error =
                                        string.Format(CultureInfo.InvariantCulture,
                                                "Invalid row \"{0}\". The row does
not contain the same number of data columns as the table header definition.",
                                                dataRow);
                                throw new InvalidOperationException(error);
                        }
                        for (int i = 0; i < tempData.Length; i++)
                        {
                                string value = tempData[i];
                                if (value.Length > 1)
                                {
                                        if (value.IndexOf('"', 0) == 0 &&
value.IndexOf('"', 1) == value.Length - 1)
                                                value = value.Substring(1, value.Length
- 2);
                                }
                                _currentRow.SetValue(ConvertValue(GetFieldType(index),
value), index);
                                index++;
                        }
                        return true;
                }

                // Coverts the string value coming from the command text to the
appropriate data type, based on the field's type.
                // This also checks a few string value rules to decide if a String.Empty
of System.Data.DBNull needs to be returned.
```

```
                // The type parameter represents the Type of the current column the data
belongs to.
                // The originalValue parameter represents the String value coming from
the command text.
                // Returns the object resulting from the converted string, based on the
type.
                private object ConvertValue(Type type, string originalValue)
                {
                        Type fieldType = type;
                        CultureInfo invariantCulture = CultureInfo.InvariantCulture;
                        try
                        {
                                if (originalValue == "\"\"" || originalValue == " ")
                                        return string.Empty;
                                if (originalValue == "")
                                        return DBNull.Value;
                                if (originalValue == "DBNull")
                                        return DBNull.Value;
                                if (fieldType.Equals(typeof (String)))
                                        return originalValue.Trim();
                                if (fieldType.Equals(typeof (Int32)))
                                        return Convert.ToInt32(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (Boolean)))
                                        return Convert.ToBoolean(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (DateTime)))
                                        return Convert.ToDateTime(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (Decimal)))
                                        return Convert.ToDecimal(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (Double)))
                                        return Convert.ToDouble(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (Int16)))
                                        return Convert.ToInt16(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (Int64)))
                                        return Convert.ToInt64(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (Single)))
                                        return Convert.ToSingle(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (Byte)))
                                        return Convert.ToByte(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (SByte)))
```

```
                                        return Convert.ToSByte(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (UInt16)))
                                        return Convert.ToUInt16(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (UInt32)))
                                        return Convert.ToUInt32(originalValue,
invariantCulture);
                                if (fieldType.Equals(typeof (UInt64)))
                                        return Convert.ToUInt64(originalValue,
invariantCulture);
                        }
                        catch (Exception e)
                        {
                                throw new InvalidOperationException(
                                        string.Format("Input value '{0}' could not be
converted to the type '{1}'.", originalValue, type), e);
                        }
                        //If no match is found return DBNull instead.
                        return DBNull.Value;
                }
                #region IDataReader Members

                // Advances the CsvDataReader to the next record.
                // Returns True if there are more rows; otherwise, False.
                public bool Read()
                {
                        if (_textReader.Peek() > -1)
                                ParseDataRow(_textReader.ReadLine());
                        else
                                return false;
                        return true;
                }
                #endregion
                #region IDisposable Members

                // Releases the resources used by the CsvDataReader.
                public void Dispose()
                {
                        Dispose(true);
                        GC.SuppressFinalize(this);
                }
                private void Dispose(bool disposing)
                {
                        if (disposing)
                        {
                                if (_textReader != null)
                                        _textReader.Close();
```

```
                        }
                        _typeLookup = null;
                        _columnLookup = null;
                        _columns = null;
                        _currentRow = null;
                }

                // Allows an Object to attempt to free resources and perform other
cleanup operations before the Object is reclaimed by garbage collection.
                ~CsvDataReader()
                {
                        Dispose(false);
                }
                #endregion
                #region IDataRecord Members

                // Gets the number of columns in the current row.
                public int FieldCount
                {
                        get { return _columns.Length; }
                }

                // The i parameter represents the index of the field to find.
                // Returns the Type information corresponding to the type of Object that
would be returned from GetValue.
                public Type GetFieldType(int i)
                {
                        if (i > _columns.Length - 1)
                                return null;
                        return ((CsvColumn) _columns.GetValue(i)).DataType;
                }

                // Gets the name for the field to find.
                // The i parameter represents the index of the field to find.
                // Returns the name of the field or an empty string (""), if there is no
value to return.
                public string GetName(int i)
                {
                        if (i > _columns.Length - 1)
                                return string.Empty;

                        return ((CsvColumn) _columns.GetValue(i)).FieldName;
                }

                // The name parameter represents the name of the field to find.
                // Returns the index of the named field.
                public int GetOrdinal(string name)
                {
```

```
                        object value = _columnLookup[name];
                        if (value == null)
                                throw new IndexOutOfRangeException("name");
                        return (int) value;
                }

                // The i parameter represents the index of the field to find.
                // Returns the Object which contains the value of the specified field.
                public object GetValue(int i)
                {
                        if (i > _columns.Length - 1)
                                return null;
                        return _currentRow.GetValue(i);
                }
                public virtual IDataReader GetData(int fieldIndex)
                {
                        throw new NotSupportedException();
                }
                #endregion
        }
}
```

8. Right-click the CSVDataProvider folder and select **Add**, then **Class**, then name the class **CsvCommand** and add code like the following to replace the default stub in the class.

Visual Basic.NET code. Paste it to replace the default stub in the class.

```
Imports System
Imports System.IO
Imports GrapeCity.ActiveReports.Extensibility.Data
Namespace CSVDataProvider
    ' Provides the IDbCommand implementation for the .NET Framework CSV Data Provider.
    Public NotInheritable Class CsvCommand
        Implements IDbCommand
        Private _commandText As String
        Private _connection As IDbConnection
        Private _commandTimeout As Integer
        Private _commandType As CommandType

        ' Creates a new instance of the CsvCommand class.
        Public Sub New()
            Me.New(String.Empty)
        End Sub

        ' Creates a new instance of the CsvCommand class with command text.
        ' The commandText parameter represents the command text.
        Public Sub New(commandText As String)
            Me.New(commandText, Nothing)
        End Sub
```

```
        ' Creates a new instance of the CsvCommand class with command text and a
CsvConnection.
        ' The commandText parameter represents the command text.
        ' The connection parameter represents a CsvConnection to a data source.
        Public Sub New(commandText As String, connection As CsvConnection)
            _commandText = commandText
            _connection = connection
        End Sub

        ' Gets or sets the command to execute at the data source.
        Public Property CommandText() As String Implements IDbCommand.CommandText
            Get
                Return _commandText
            End Get
            Set(value As String)
                _commandText = value
            End Set
        End Property

        ' Gets or sets the wait time before terminating an attempt to execute the
command and generating an error.
        Public Property CommandTimeout() As Integer Implements IDbCommand.CommandTimeout
            Get
                Return _commandTimeout
            End Get
            Set(value As Integer)
                _commandTimeout = value
            End Set
        End Property

        ' Gets or sets a value indicating how the CommandText property is interpreted.
        ' Remarks: We don't use this one for the Csv Data Provider.
        Public Property CommandType() As CommandType Implements IDbCommand.CommandType
            Get
                Return _commandType
            End Get
            Set(value As CommandType)
                _commandType = value
            End Set
        End Property

        ' Gets or sets the CsvConnection used by this instance of the CsvCommand.
        Public Property Connection() As IDbConnection
            Get
                Return _connection
            End Get
            Set(value As IDbConnection)
                _connection = value
```

```
            End Set
        End Property

       ' Sends the CommandText to the CsvConnection, and builds a CsvDataReader using
one of the CommandBehavior values.
        ' The behavior parameter represents a CommandBehavior value.
        ' Returns a CsvDataReader object.
        Public Function ExecuteReader(behavior As CommandBehavior) As IDataReader
Implements IDbCommand.ExecuteReader
            Return New CsvDataReader(New StringReader(_commandText))
        End Function

        ' Returns a string that represents the command text with the parameters expanded
into constants.
        Public Function GenerateRewrittenCommandText() As String Implements
IDbCommand.GenerateRewrittenCommandText
            Return _commandText
        End Function

        ' Sends the CommandText to the CsvConnection and builds a CsvDataReader.
        ' Returns a CsvDataReader object.
        Public Function ExecuteReader() As IDataReader Implements
IDbCommand.ExecuteReader
            Return ExecuteReader(CommandBehavior.SchemaOnly)
        End Function
#Region "Non implemented IDbCommand Members"
        Public ReadOnly Property Parameters() As IDataParameterCollection Implements
IDbCommand.Parameters
            Get
                Throw New NotImplementedException()
            End Get
        End Property
        Public Property Transaction() As IDbTransaction Implements
IDbCommand.Transaction
            Get
                Throw New NotImplementedException()
            End Get
            Set(value As IDbTransaction)
                Throw New NotImplementedException()
            End Set
        End Property
        Public Sub Cancel() Implements IDbCommand.Cancel
        End Sub
        Public Function CreateParameter() As IDataParameter Implements
IDbCommand.CreateParameter
            Throw New NotImplementedException()
        End Function
#End Region
```

```
#Region "IDisposable Members"

        ' Releases the resources used by the CsvCommand.
        Public Sub Dispose() Implements IDisposable.Dispose
            Dispose(True)
            GC.SuppressFinalize(Me)
        End Sub

        Private Sub Dispose(disposing As Boolean)
            If disposing Then
                If _connection IsNot Nothing Then
                    _connection.Dispose()
                    _connection = Nothing
                End If
            End If
        End Sub
#End Region
    End Class
End Namespace
```

C# code. Paste it to replace the default stub in the class

```
using System;
using System.IO;
using GrapeCity.ActiveReports.Extensibility.Data;
namespace CustomDataProvider.CSVDataProvider
{
       // Provides the IDbCommand implementation for the .NET Framework CSV Data
Provider.
        public sealed class CsvCommand : IDbCommand
        {
                private string _commandText;
                private IDbConnection _connection;
                private int _commandTimeout;
                private CommandType _commandType;

                /// Creates a new instance of the CsvCommand class.
                public CsvCommand()
                        : this(string.Empty)
                {
                }

                // Creates a new instance of the CsvCommand class with command text.
                // The commandText parameter represents the command text.
                public CsvCommand(string commandText)
                        : this(commandText, null)
                {
                }
```

```
		// Creates a new instance of the CsvCommand class with command text and
a CsvConnection.
		// The commandText parameter represents the command text.
		// The connection parameter represents a CsvConnection to a data
source.?
		public CsvCommand(string commandText, CsvConnection connection)
		{
			_commandText = commandText;
			_connection = connection;
		}

		// Gets or sets the command to execute at the data source.
		public string CommandText
		{
			get { return _commandText; }
			set { _commandText = value; }
		}

		// Gets or sets the wait time before terminating an attempt to execute
the command and generating an error.
		public int CommandTimeout
		{
			get { return _commandTimeout; }
			set { _commandTimeout = value; }
		}

		// Gets or sets a value indicating how the CommandText property is
interpreted.
		// Remarks: We don't use this one for the Csv Data Provider.
		public CommandType CommandType
		{
			get { return _commandType; }
			set { _commandType = value; }
		}

		// Gets or sets the CsvConnection used by this instance of the
CsvCommand.
		public IDbConnection Connection
		{
			get { return _connection; }
			set { _connection = value; }
		}

		// Sends the CommandText to the CsvConnection, and builds a
CsvDataReader using one of the CommandBehavior values.
		// The behavior parameter represents a CommandBehavior value.
		// Returns a CsvDataReader object.
		public IDataReader ExecuteReader(CommandBehavior behavior)
```

```
                {
                        return new CsvDataReader(new StringReader(_commandText));
                }

               // Returns a string that represents the command text with the parameters
expanded into constants.
                public string GenerateRewrittenCommandText()
                {
                        return _commandText;
                }

                // Sends the CommandText to the CsvConnection and builds a
CsvDataReader.
                // Returns a CsvDataReader object.
                public IDataReader ExecuteReader()
                {
                        return ExecuteReader(CommandBehavior.SchemaOnly);
                }
                #region Non implemented IDbCommand Members
                public IDataParameterCollection Parameters
                {
                        get { throw new NotImplementedException(); }
                }
                public IDbTransaction Transaction
                {
                        get { throw new NotImplementedException(); }
                        set { throw new NotImplementedException(); }
                }
                public void Cancel()
                {

                }
                public IDataParameter CreateParameter()
                {
                        throw new NotImplementedException();
                }
                #endregion
                #region IDisposable Members

                // Releases the resources used by the CsvCommand.
                public void Dispose()
                {
                        Dispose(true);
                        GC.SuppressFinalize(this);
                }

                private void Dispose(bool disposing)
                {
```

```
                        if (disposing)
                        {
                                if (_connection != null)
                                {
                                        _connection.Dispose();
                                        _connection = null;
                                }
                        }
                }
                #endregion
        }
}
```

9. Right-click the CSVDataProvider folder and select **Add**, then **Class**, then name the class **CsvConnection** and add code like the following to replace the default stub in the class. (You can safely ignore the errors, as they will go away when you add the CsvConnection class.)

Visual Basic.NET code. Paste it to replace the default stub in the class.

```
Imports System
Imports System.Collections.Specialized
Imports GrapeCity.ActiveReports.Extensibility.Data
Namespace CSVDataProvider
    ' Provides an implementation of IDbConnection for the .NET Framework CSV Data
Provider.
    Public NotInheritable Class CsvConnection
        Implements IDbConnection
        Private _localizedName As String
        ' Creates a new instance of the CsvConnection class.
        Public Sub New()
            _localizedName = "Csv"
        End Sub
       ' Creates a new instance of the CsvConnection class.
       ' The localizedName parameter represents the localized name for the CsvConnection
instance.
        Public Sub New(localizeName As String)
            _localizedName = localizeName
        End Sub
#Region "IDbConnection Members"
        ' Gets or sets the string used to open the connection to the data source.
        ' Remarks: We don't use this one for the Csv Data Provider.
        Public Property ConnectionString() As String Implements
IDbConnection.ConnectionString
            Get
                Return String.Empty
            End Get
            Set(value As String)

            End Set
        End Property
```

```
        ' Gets the amount of time to wait while trying to establish a connection before
terminating
        ' the attempt and generating an error.
        ' Remarks: We don't use this one for the Csv Data Provider.
        Public ReadOnly Property ConnectionTimeout() As Integer Implements
IDbConnection.ConnectionTimeout
            Get
                Throw New NotImplementedException()
            End Get
        End Property
        ' Begins a data source transaction.
        ' Returns an object representing the new transaction.
        ' Remarks: We don't use this one for the Csv Data Provider.
        Public Function BeginTransaction() As IDbTransaction Implements
IDbConnection.BeginTransaction
            Return Nothing
        End Function
        ' Opens a data source connection.
        ' Remarks: We don't use this one for the Csv Data Provider.
        Public Sub Open() Implements IDbConnection.Open

        End Sub
        ' Closes the connection to the data source. This is the preferred method of
closing any open connection.
        Public Sub Close() Implements IDbConnection.Close
            Dispose()
        End Sub
        ' Creates and returns a CsvCommand object associated with the CsvConnection.
        Public Function CreateCommand() As IDbCommand Implements
IDbConnection.CreateCommand
            Return New CsvCommand(String.Empty)
        End Function
        Public Property DataProviderService() As IDataProviderService Implements
IDbConnection.DataProviderService
            Get
                Return Nothing
            End Get
            Set(value As IDataProviderService)
            End Set
        End Property
#End Region
#Region "IDisposable Members"
        ' Releases the resources used by the CsvConnection.
        Public Sub Dispose() Implements IDisposable.Dispose
            Dispose(True)
            GC.SuppressFinalize(Me)
        End Sub
        Private Sub Dispose(disposing As Boolean)
```

```
        End Sub
       ' Allows an Object to attempt to free resources and perform other cleanup
operations
        ' before the Object is reclaimed by garbage collection.
        Protected Overrides Sub Finalize()
            Try
                Dispose(False)
            Finally
                MyBase.Finalize()
            End Try
        End Sub
#End Region
#Region "IExtension Members"
       ' Gets the localized name of the CsvConnection.
        Public ReadOnly Property LocalizedName() As String Implements
IDbConnection.LocalizedName
            Get
                Return _localizedName
            End Get
        End Property
       ' Specifies any configuration information for this extension.
       ' The configurationSettings parameter represents a NameValueCollection of the
settings.
        Public Sub SetConfiguration(configurationSettings As NameValueCollection)
Implements IDbConnection.SetConfiguration
        End Sub
#End Region
    End Class
End Namespace
```

C# code. Paste it to replace the default stub in the class.

```
using System;
using System.Collections.Specialized;
using GrapeCity.ActiveReports.Extensibility.Data;
namespace CustomDataProvider.CSVDataProvider
{
        // Provides an implementation of IDbConnection for the .NET Framework CSV Data
Provider.
        public sealed class CsvConnection : IDbConnection
        {
                private string _localizedName;

                // Creates a new instance of the CsvConnection class.
                public CsvConnection()
                {
                        _localizedName = "Csv";
                }
```

```
                // Creates a new instance of the CsvConnection class.
                // The localizedName parameter represents the localized name for the
CsvConnection instance.
                public CsvConnection(string localizeName)
                {
                        _localizedName = localizeName;
                }
                #region IDbConnection Members

               // Gets or sets the string used to open the connection to the data
source.
               // Remarks: We don't use this one for the Csv Data Provider.
                public string ConnectionString
                {
                        get { return string.Empty; }
                        set { ; }
                }

                // Gets the amount of time to wait while trying to establish a
connection before terminating the attempt and generating an error.
                // Remarks: We don't use this one for the Csv Data Provider.
                public int ConnectionTimeout
                {
                        get { throw new NotImplementedException(); }
                }

               // Begins a data source transaction.
               // Returns an object representing the new transaction.
               // Remarks: We don't use this one for the Csv Data Provider.
                public IDbTransaction BeginTransaction()
                {
                        return null;
                }

               // Opens a data source connection.
               // Remarks: We don't use this one for the Csv Data Provider.
                public void Open()
                {
                        ;
                }

                // Closes the connection to the data source. This is the preferred
method of closing any open connection.
                public void Close()
                {
                        Dispose();
                }
```

```
                // Creates and returns a CsvCommand object associated with the
CsvConnection.
                public IDbCommand CreateCommand()
                {
                        return new CsvCommand(string.Empty);
                }
                public IDataProviderService DataProviderService
                {
                        get { return null; }
                        set { }
                }
                #endregion
                #region IDisposable Members

                // Releases the resources used by the CsvConnection.
                public void Dispose()
                {
                        Dispose(true);
                        GC.SuppressFinalize(this);
                }
                private void Dispose(bool disposing)
                {
                }

                // Allows an Object to attempt to free resources and perform other
cleanup operations before the Object is reclaimed by garbage collection.
                ~CsvConnection()
                {
                        Dispose(false);
                }
                #endregion
                #region IExtension Members

                // Gets the localized name of the CsvConnection.
                public string LocalizedName
                {
                        get { return _localizedName; }
                }

                // Specifies any configuration information for this extension.
                // The configurationSettings parameter represents a NameValueCollection
of the settings.
                public void SetConfiguration(NameValueCollection configurationSettings)
                {
                }
                #endregion
        }
}
```

10. Right-click the CSVDataProvider folder and select **Add**, then **Class**, then name the class **CsvDataProviderFactory** and add code like the following to replace the default stub in the class.

Visual Basic.NET code. Paste it to replace the default stub in the class.

```
Imports GrapeCity.ActiveReports.Extensibility.Data
Imports GrapeCity.BI.Data.DataProviders
Namespace CSVDataProvider
	' Implements the DataProviderFactory for .NET Framework CSV Data Provider.
	 Public Class CsvDataProviderFactory
		Inherits DataProviderFactory
		' Creates new instance of the CsvDataProviderFactory class.
		Public Sub New()
		End Sub
		' Returns a new instance of the the CsvCommand.
		Public Overrides Function CreateCommand() As IDbCommand
			Return New CsvCommand()
		End Function
		' Returns a new instance of the the CsvConnection.
		Public Overrides Function CreateConnection() As IDbConnection
			Return New CsvConnection()
		End Function
	 End Class
End Namespace
```

C# code. Paste it to replace the default stub in the class.

```
using GrapeCity.ActiveReports.Extensibility.Data;
using GrapeCity.BI.Data.DataProviders;
namespace CustomDataProvider.CSVDataProvider
{
	// Implements the DataProviderFactory for .NET Framework CSV Data Provider.
	 public class CsvDataProviderFactory : DataProviderFactory
	 {

		// Creates new instance of the CsvDataProviderFactory class.
		public CsvDataProviderFactory()
		{
		}

		// Returns a new instance of the the CsvCommand.
		public override IDbCommand CreateCommand()
		{
			return new CsvCommand();
		}

		// Returns a new instance of the the CsvConnection.
		public override IDbConnection CreateConnection()
		{
			return new CsvConnection();
```

```
                }
        }
}
```

Add a button to the query editor

1. In the Solution Explorer, right-click the CSVDataProvider folder and select **Add**, then **Class**, then name the class **QueryEditor** and add code like the following to replace the default stub in the class.

Visual Basic.NET code. Paste it to replace the default stub in the class.

```
Imports System.Collections.Generic
 Imports System.Drawing.Design
 Imports System.IO
 Imports System.Linq
 Imports System.Text
 Imports System.Text.RegularExpressions
 Imports System.Windows.Forms
 Imports System.Windows.Forms.Design

 Namespace CustomDataProvider.CSVDataProvider
     Public NotInheritable Class QueryEditor
         Inherits UITypeEditor
         Public Overrides Function GetEditStyle(context As
System.ComponentModel.ITypeDescriptorContext) As UITypeEditorEditStyle
             Return UITypeEditorEditStyle.DropDown
         End Function
         Public Overrides Function EditValue(context As
System.ComponentModel.ITypeDescriptorContext, provider As System.IServiceProvider, value
As Object) As Object
             Dim edSvc As IWindowsFormsEditorService =
DirectCast(provider.GetService(GetType(IWindowsFormsEditorService)),
IWindowsFormsEditorService)
             Dim path = ""
             Dim btn = New Button()
             btn.Text = "Select CSV File..."
             Dim pdg = btn.Padding
             pdg.Bottom += 2
             btn.Padding = pdg
             btn.Click += Sub() Using openDlg = New OpenFileDialog()
                 openDlg.Filter = "CSV Files (*.csv)|*.csv|All Files (*.*)|*.*"
                 If openDlg.ShowDialog() <> DialogResult.OK Then
                     path = ""
                 Else
                     path = openDlg.FileName
                 End If
             End Using
             edSvc.DropDownControl(btn)
             If String.IsNullOrEmpty(path) Then
```

```
                Return String.Empty
            End If
            If Not File.Exists(path) Then
                Return String.Empty
            End If
            Return GetCSVQuery(path)
        End Function
        Private Function GetCSVQuery(path As String) As Object
            Dim sr As StreamReader = Nothing
            Try
                sr = New StreamReader(path)
                Dim ret As String = String.Empty
                Dim currentLine As String
                Dim line As Integer = 0
                While (InlineAssignHelper(currentLine, sr.ReadLine())) IsNot Nothing
                    If line = 0 Then
                        ret += ProcessColumnsDefinition(currentLine) &
Convert.ToString(vbCr & vbLf)
                    Else
                        ret += currentLine & Convert.ToString(vbCr & vbLf)
                    End If
                    line += 1
                End While
                Return ret
            Catch generatedExceptionName As IOException
                Return String.Empty
            Finally
                If sr IsNot Nothing Then
                    sr.Close()
                End If
            End Try
        End Function
        Private Function ProcessColumnsDefinition(currentLine As String) As String
            Const  ColumnWithDataTypeRegex As String = "[""]?\w+[\""]?\(.+\)"
            Dim columns As String() = currentLine.Split(New String() {","},
StringSplitOptions.None)
            Dim ret As String = Nothing
            For Each column As String In columns
                If Not String.IsNullOrEmpty(ret) Then
                    ret += ","
                End If
                If Not Regex.Match(column, ColumnWithDataTypeRegex).Success Then
                    ret += column & Convert.ToString("(string)")
                Else
                    ret += column
                End If
            Next
            Return ret
```

```
        End Function
        Private Shared Function InlineAssignHelper(Of T)(ByRef target As T, value As T)
As T
            target = value
            Return value
        End Function
    End Class
 End Namespace
```

C# code. Paste it to replace the default stub in the class.

```
using System;
 using System.Collections.Generic;
 using System.Drawing.Design;
 using System.IO;
 using System.Linq;
 using System.Text;
 using System.Text.RegularExpressions;
 using System.Windows.Forms;
 using System.Windows.Forms.Design;
 namespace CustomDataProvider.CSVDataProvider
 {
     public sealed class QueryEditor : UITypeEditor
     {
         public override UITypeEditorEditStyle
GetEditStyle(System.ComponentModel.ITypeDescriptorContext context)
         {
             return UITypeEditorEditStyle.DropDown;
         }
        public override object EditValue(System.ComponentModel.ITypeDescriptorContext
context, System.IServiceProvider provider, object value)
         {
             IWindowsFormsEditorService edSvc =
(IWindowsFormsEditorService)provider.GetService(typeof(IWindowsFormsEditorService));
             var path = "";
             var btn = new Button();
             btn.Text = "Select CSV File...";
             var pdg = btn.Padding;
             pdg.Bottom += 2;
             btn.Padding = pdg;
             btn.Click += delegate
             {
                 using (var openDlg = new OpenFileDialog())
                 {
                     openDlg.Filter = "CSV Files (*.csv)|*.csv|All Files (*.*)|*.*";
                     if (openDlg.ShowDialog() != DialogResult.OK)
                         path = "";
                     else
                         path = openDlg.FileName;
```

```
                }
            };
            edSvc.DropDownControl(btn);
            if (string.IsNullOrEmpty(path)) return string.Empty;
            if (!File.Exists(path)) return string.Empty;
            return GetCSVQuery(path);
        }
        private object GetCSVQuery(string path)
        {
            StreamReader sr = null;
            try
            {
                sr = new StreamReader(path);
                string ret = string.Empty;
                string currentLine;
                int line = 0;
                while ((currentLine = sr.ReadLine()) != null)
                {
                    if (line == 0)
                        ret += ProcessColumnsDefinition(currentLine) + "\r\n";
                    else
                        ret += currentLine + "\r\n";
                    line++;
                }
                return ret;
            }
            catch (IOException)
            {
                return string.Empty;
            }
            finally
            {
                if (sr != null)
                    sr.Close();
            }
        }
        private string ProcessColumnsDefinition(string currentLine)
       {
            const string ColumnWithDataTypeRegex = @"[""]?\w+[\""]?\(.+\)";
            string[] columns = currentLine.Split(new string[] { "," },
StringSplitOptions.None);
            string ret = null;
            foreach (string column in columns)
            {
                if (!string.IsNullOrEmpty(ret))
                    ret += ",";
                if (!Regex.Match(column, ColumnWithDataTypeRegex).Success)
                {
```

```
                    ret += column + "(string)";
                }
                else
                {
                    ret += column;
                }
            }
            return ret;
        }
    }
}
```

2. In the Solution Explorer, right-click the CustomDataProviderDemo project and select **Add Reference**. In the **Reference Manager** dialog that appears, on the Projects tab, select **CustomDataProvider** and click **OK**.
3. Run the project, and follow the instructions in the RichTextBox to see the custom data provider in action.

External Customization

This article discusses the possibilities to use ActiveReports.NET with other technologies.

ActiveReports gives you opportunity to customize reports, viewers, and exports if you are looking for customization at different levels of using the product.

Reports

ActiveReports provides the following extension possibilities for reports:

- Scripts or external assemblies
- Custom report items
- Custom data providers
- Custom map tile providers

ActiveReports provides many reports-related features. If you are looking for more possibilities, you can try the following extensions:

- **Customize with scripts or external assemblies**

  To have custom assemblies load in restricted domains, you should add a configuration file with the name of *yourApp*.exe.config file as follows.

  yourApp.exe.config file

  ```
  <configuration>
      <runtime>
         <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
            <probing privatePath="PrivateBin"/>
         </assemblyBinding>
      </runtime>
   </configuration>
  ```

  Limitation: Private assemblies are deployed in the same directory structure as the application. If the directories, specified in the **privatePath** setting are not under ApplicationBase, they are ignored.

- **Customize with custom report items**

  ActiveReports delivers a number of samples demonstrating how to use custom report items. See the Samples page for details.

- **Customize with custom data providers**

  To learn about possible options, see Configure ActiveReports.

- **Customize with custom map tile providers**

  To learn about possible options, see Configure ActiveReports.

Viewers

ActiveReports provides various ways to create your own preview control, including different platforms for custom viewers based on the JS Viewer component and the Blazor Viewer control.

The **Custom Preview** sample demonstrates how to create your own preview control and show report output in a custom preview form. You can find the **Custom Preview** sample here. You can modify both viewer's mouse mode and touch mode toolbars and set custom commands. You can also customize the Viewer control to make it a perfect fit for your Windows application by adding and removing toolbar buttons, adding and removing menu items, creating custom dialogs, and calling them from custom click events. For more information, see Customize the WinForms Viewer Control.

With the JS Viewer component and the Blazor Viewer control, you can implement your own preview control on many different platforms like:

- PWAs
- .NET MAUI (WebView)
- .NET MAUI Blazor (BlazorWebView)

Exports

You can create custom exports in your ActiveReports applications. ActiveReports provides a lot of various powerful exports that cover most situations.

However, you can write your own exports as demonstrated in the CustomPdfExport sample. It shows how to write custom exports for non-embedded fonts. Such tasks have some limitations because they are dependent on the internal API.

However, the simplest way is to write custom export to graphics-like formats: PDF, SVG, DrawingML, XPS, etc. More complex formats may require a lot of work.

Environment

ActiveReports delivers implementation solutions that can work in different environments. We recommend that you consider this logic when making a decision on what option to choose:

- Use Page/RDLX reports, Section report requires some knowledge on writing scripts and is intended mainly for developers.
- Use custom font resolving (and distribute fonts together with your application).
- Use the new Blazor viewer for the cross-platform UI. See following articles from Visual Studio
  - https://visualstudiomagazine.com/articles/2021/02/17/net-6-preview-1.aspx

- https://visualstudiomagazine.com/articles/2021/04/09/blazorwebview.aspx

Export Reports

In this section, learn about the independent ways to export a report to different formats, and some export implementations.

- **Export Page/RDLX Reports using Rendering Extensions**

  Use the Render method in Rendering Extensions of the PageDocument class to render a Page report and RDLX report to Image, Html, Pdf, Xml, Word, and Excel formats.

- **Export Page/RDLX/Section Reports using Export Filters**

  Use the Export method of the corresponding ExportFilter class to export a Section report, Page report, and RDLX Report. Exporting in Section Report is only possible through Export Filters.

- **Set PDF Print Presets**

  Use the PrintPresets class to export a Section report, Page report, and RDLX Report. The print preset properties are only available with the Professional Edition license.

.

> **Note**: In **ASP.NET Core** applications, supported export formats are - Excel (.xlsx), Word (.docx), PDF, CSV, JSON, and TIFF.

For more information on the supported export formats, see Exports.

Export Page/RDLX Reports

To export Page/RDLX reports, use the **Render** method in the rendering extensions of the **PageDocument** class. Each export format provides its own set of properties to control how the report is rendered. Explore the following export abilities available in Page/RDLX reports.

> To export Page/RDLX reports to RTF format, use **MESCIUS.ActiveReports.Export.Rdf ('MESCIUS.ActiveReports.Export.Rdf Assembly' in the on-line documentation)** package. See Section report's RTF Export topic.

- HTML Export
- PDF Export
- Image Export
- XML Export
- Excel Export
- Word Export
- CSV Export
- JSON Export
- Excel Data Export
- Text Print Export

HTML Export

HTML, or hypertext markup language, is a format that opens in a Web browser. You can export your reports to HTML or MHT formats. It is a good format for delivering content because virtually all users have an HTML browser. The **HTMLRenderingExtension ('HtmlRenderingExtension Class' in the on-line documentation)** renders your report in this format with improved table border rendering and high-quality SVG output for charts. If you do not want to use SVG in charts, set the RenderingEngine property to **Html**.

HTML Rendering Properties

ActiveReports offers several options to control how reports render to HTML.

Property	Description
Fragment ('Fragment Property' in the on-line documentation)	Determine whether or not the full HTML text will be returned or just the contents contained inside the body tag will be returned. True indicates only the content inside the body tag will be returned. The default is false.
MhtOutput ('MhtOutput Property' in the on-line documentation)	Gets or sets whether or not the output should be in Mht format. True indicates the output should be in Mht format; otherwise, false. The default is false.
RenderingEngine ('RenderingEngine Property' in the on-line documentation)	The RenderingEngine property is set to **Mixed** by default for improved quality output. The choices are Html or Mixed, where Mixed uses SVG to render charts.
StyleStream ('StyleStream Property' in the on-line documentation)	Set the StyleStream to True to create an external .css file containing style information from your report controls' style properties. If you prefer to have style information embedded in the HTML file, set the StyleStream property to False.
LinkTarget ('LinkTarget Property' in the on-line documentation)	Specify a link target to control whether drill-down reports and other links open in a new window or reuse the current window. By default, no value is set and links open in the same window. A value of _blank opens the link in a new window, or you can specify a window using window_name. By default, this value is not set.
Mode ('Mode Property' in the on-line documentation)	Galley mode renders the report in one HTML stream. Select Paginated mode to render each page as a section inside the HTML document.
OutputTOC ('OutputTOC Property' in the on-line documentation)	Indicates whether the report's existing TOC should be added to the output.

Interactivity

Reports rendered in HTML support a number of interactive features. Hyperlinks, Bookmarks, and Drill through links can be rendered to HTML. However, Document Maps are not available in this format. For a drill-down report, make sure

that the data you want to display is expanded before rendering, otherwise, it renders in the hidden state.

Limitations

- HTML is not the best format for printing. Use the PDF rendering extension instead.
- Diagonal lines, Page margins, Border/Line styles (like DashedDotDot, DashDot, WindowInset) are not supported.
- CheckBox color does not affect the color of the square.
- TextIndent and FillCharacter properties of the TableOfContents control's Level setting are not supported.

Export Report using HTML Rendering Extension

The following steps provide an example of rendering a report in HTML format.

1. Create a new or open an existing Visual Studio project.
2. If you are creating a new project, select **ActiveReports 18 Page report Application** in **Create a New Project** dialog, specify a name for the project, and click OK.
   If you are using an existing project, in the Solution Explorer, right-click the project and select **Add** > **New Item**. Select **ActiveReports 18 Page report** and click **Add**. By default, a Page report is added to the project.
3. Add a reference to **MESCIUS.ActiveReports.Export.Html** package in the project.
4. On the Form.cs or Form.vb that opens, double-click the high-quality title bar to create the Form_Load event.
5. Add the following code inside the Form_Load event.

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

```
' Provide the Page report you want to render.
Dim rptPath As New IO.FileInfo("..\..\PageReport1.rdlx")

Dim pageReport As New GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyHTML")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim htmlSetting As New GrapeCity.ActiveReports.Export.Html.Page.Settings()
Dim setting As GrapeCity.ActiveReports.Extensibility.Rendering.ISettings = htmlSetting

' Set the rendering extension and render the report.
Dim htmlRenderingExtension As New
GrapeCity.ActiveReports.Export.Html.Page.HtmlRenderingExtension()
Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(htmlRenderingExtension, outputProvider, htmlSetting)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.

System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");
GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyHTML");
outputDirectory.Create();

// Provide settings for your rendering output.
GrapeCity.ActiveReports.Export.Html.Page.Settings htmlSetting = new
GrapeCity.ActiveReports.Export.Html.Page.Settings();
GrapeCity.ActiveReports.Extensibility.Rendering.ISettings setting = htmlSetting;

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Html.Page.HtmlRenderingExtension htmlRenderingExtension = new
GrapeCity.ActiveReports.Export.Html.Page.HtmlRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;

pageReport.Document.Render(htmlRenderingExtension, outputProvider, htmlSetting);
```

PDF Export

Portable Document Format (PDF), is a format recommended for printing and for preserving formatting. You can use the **PDFRenderingExtension ('PdfRenderingExtension Class' in the on-line documentation)** to render your report in this format. With the PDF rendering extension, you can use features such as font linking, digital signatures, and end user-defined characters (EUDC). These features are only available in the Professional Edition of ActiveReports.

PDF Rendering Properties

ActiveReports offers a number of options to control how reports render to PDF.

Property	Description
Application ('Application Property' in the on-line documentation)	Set the value that appears for application in the Document Properties dialog of the PDF viewer application.
Author ('Author Property' in the on-line documentation)	Enter the name of the author to appear in the Document Properties dialog of the PDF viewer application.
CenterWindow ('CenterWindow Property' in the on-line documentation)	Set to True to position the document's window in the center of the screen.
DisplayMode ('DisplayMode Property' in the on-line documentation)	Specifies how the document is displayed when opened. FullScreen mode displays the document with no menu bar, window controls, or any other window visible.
DisplayTitle	Set to True to display text you enter in the Title property. When set to False it displays the name of the PDF file.

('DisplayTitle Property' in the on-line documentation)	
DpiX ('DpiX Property' in the on-line documentation)	Set the horizontal resolution of the rendered PDF file.
DpiY ('DpiY Property' in the on-line documentation)	Set the vertical resolution of the rendered PDF file.
EmbedFonts ('EmbedFonts Property' in the on-line documentation)	Select how the fonts used in the report should be embedded in the PDF document. **Note:** By default, all fonts get embedded in the exported PDF document.
Encrypt ('Encrypt Property' in the on-line documentation)	Determines whether the document is encrypted or not. **Note**: If Encrypt is set to False, permissions and passwords have no effect.
EndPage ('EndPage Property' in the on-line documentation)	The last page of the report to render. The default value is the value for StartPage, that is, 0.
FallbackFonts ('FallbackFonts Property' in the on-line documentation)	Gets or sets a comma-delimited string of font families to locate missing glyphs from the original font.
FitWindow ('FitWindow Property' in the on-line documentation)	True to resize the document's window to fit the size of the first displayed page. Default value: false.
HideMenubar ('HideMenubar Property' in the on-line documentation)	True to hide the viewer application's menu bar when the document is active. Default value: false.
HideToolbar ('HideToolbar Property' in the on-line documentation)	True to hide the viewer application's toolbars when the document is active. Default value: false.
HideWindowUI ('HideWindowUI Property' in the on-line documentation)	True to hide user interface elements in the document's window (such as scroll bars and navigation controls), leaving only the document's contents displayed. Default value: false.
ImageInterpolation ('ImageInterpolation Property' in the on-line documentation)	Interpolation value of images. Allows enabling and disabling image interpolation, when exporting the file to PDF.
Keywords ('Keywords Property' in the on-line documentation)	Keywords associated with the document.
OwnerPassword ('OwnerPassword Property' in the on-line documentation)	The owner password that can be entered in the reader, which permits full access to the document regardless of the specified user permissions.
PageHeight ('PageHeight Property' in the on-line documentation)	The page height value, in inches, to set for the report. You must include an integer or decimal value followed by "in" (for example, 1in). This value overrides the report's original settings.
PageWidth ('PageWidth Property' in the on-line documentation)	The page width value, in inches, to set for the report. You must include an integer or decimal value followed by "in" (for example, 1in). This value overrides the report's original settings.
Permissions ('Permissions Property' in the on-	Specifies the user permissions for the document. Permissions can be combined using a comma between values.

line documentation)	
PrintLayoutMode ('PrintLayoutMode Property' in the on-line documentation)	Specifies layout mode to be used for PDF documents.
PrintOnOpen ('PrintOnOpen Property' in the on-line documentation)	Gets or sets the value indicating whether the document should be printed after it is open.
PrintPresets ('PrintPresets Property' in the on-line documentation)	Gets or sets the PDF print preset dialog.
SizeToFit ('SizeToFit Property' in the on-line documentation)	Determines whether PDF pages are fit to the selected paper size or not.
StartPage ('StartPage Property' in the on-line documentation)	The first page of the report to render. A value of 0 indicates that all pages are rendered.
Subject ('Subject Property' in the on-line documentation)	The subject of the document.
Title ('Title Property' in the on-line documentation)	The title of the document.
UserPassword ('UserPassword Property' in the on-line documentation)	The user password that can be entered in the reader. If this value is left empty, the user will not be prompted for a password, however, the user will be restricted by the specified permissions.
Version ('Version Property' in the on-line documentation)	Set the output PDF version. The supported versions are: PDF-1.2 | PDF-1.3 | PDF-1.4(default) | PDF-1.5 | PDF-1.6 | PDF-1.7 | PDF-2.0 | PDF/A-1a | PDF/A-1b | PDF/A-2a | PDF/A-2b | PDF/A-2u | PDF/A-3a | PDF/A-3b | PDF/A-3u | PDF/UA-1
WatermarkAngle ('WatermarkAngle Property' in the on-line documentation)	Specify the degree of angle for the watermark text on the PDF document. Valid values range from 0 to 359, where 0 is horizontal, left to right.
WatermarkColor ('WatermarkColor Property' in the on-line documentation)	Select a color for the watermark text on the PDF document. The default value for the watermark color is gray, but you can select any Web, System, or Custom color.
WatermarkFontName ('WatermarkFontName Property' in the on-line documentation)	Set the font to use for the watermark on the PDF document.
WatermarkFontSize ('WatermarkFontSize Property' in the on-line documentation)	Set the font size to use for the watermark on the PDF document.
WatermarkFontStyle ('WatermarkFontStyle Property' in the on-line documentation)	Set the font style to use for the watermark on the PDF document.
WatermarkPrintOnly ('WatermarkPrintOnly Property' in the on-line documentation)	Specify whether to print a report with a watermark on it via a printer or a Microsoft print to PDF. The default value is False.
WatermarkTitle ('WatermarkTitle Property' in the on-line documentation)	Enter text (i.e. CONFIDENTIAL) to use as the watermark on the PDF document.

PDF Print Presets Properties

ActiveReports allows you to preset the printing properties for PDF report exports using the **PrintPresets ('PrintPresets Class' in the on-line documentation)** class. This prepopulates the print settings in the Print dialog box. Please see Use PDF Printing Presets for more information.

> **Note:** The print preset properties are only available with the Professional Edition license. An evaluation message is displayed when used with the Standard Edition license.

Property	Description
PageScaling ('PageScaling Property' in the on-line documentation)	Specify scaling for the printable area. You can select Default to shrink to the printable area, or you can select None for the actual size.
DuplexMode ('DuplexMode Property' in the on-line documentation)	Specify the duplex mode of the printer. For the best results with the duplex option, the selected printer should support duplex printing. You can choose from the following values, • Simplex: Prints on one side of the paper. This is the default value. • Duplex (Flip on long edge): Prints on both sides of the paper with paper flip on the long edge. • Duplex (Flip on short edge): Prints on both sides of the paper with paper flip on the short edge.
PaperSourceByPageSize ('PaperSourceByPageSize Property' in the on-line documentation)	Determines the output tray based on PDF page size, rather than page setting options. This option is useful when printing PDFs with multiple page sizes, where different sized output trays are available. By default, this option is set to False.
PrintPageRange ('PrintPageRange Property' in the on-line documentation)	Specify the range of page numbers 1-3 or 1, 2, 3.
NumberOfCopies ('NumberOfCopies Property' in the on-line documentation)	Specify the number of copies to print. You can select any number of copies from 2 to 5, or select Default to specify a single copy.

> **Note:** These properties are available in PDF version 1.7 or higher. The **PageScaling** property is supported in PDF version 1.6.

PDF Features

Interactivity

PDF is considered the best format for printing and it also supports interactive features like Document Map, Bookmarks, and Hyperlinks. However, in case you have any data hidden (like in a drill-down report) at the time of rendering, it does not show up in the output. Therefore, it is recommended to expand all toggle items prior to rendering.

Editable PDF: InputField Control

In Page and RDLX reports, you can use the InputField report control. This control provides support for editable fields in an exported PDF report file where the InputField's value can be modified.

There are two types of InputField - **Text** and **Checkbox**, which you can set in the **InputType** property. Each type has its own set of properties: the **Text** type of the InputField control gets the set of properties of the TextBox control. If the **Checkbox** type is selected, then the new control inherits the set of properties of the CheckBox control.

Tagged PDF

You can generate PDFs with tagging on the report content by setting the **AccessibleDescription** property of the report controls: Bullet, Barcode, Chart, Image, Line, Map, Shape, Sparkline, and FormattedText controls; and all custom report items. The PDF versions that support tagged PDFs conforming with the PDF/UA standard are PDF/A-1a, PDF/A-2a, PDF/A-2u, PDF/A-3a, PDF/A-3u, and PDF/UA-1.

PDF/A Support Limitations

- The **NeverEmbedFonts** property is ignored, so all fonts of a report are embedded into the PDF document.
- The **Security.Encrypt** property is ignored and the PDF export behaves as if this property is always set to False.
- The **OnlyForPrint** property is ignored and the PDF export behaves as if this property is always set to False.
- The **DocumentToAddBeforeReport** and **DocumentToAddAfterReport** properties of the PDF Rendering Extension settings are ignored.
- **Transparent images** lose their transparency when exported to PDF/A-1.
- **External hyperlinks** are exported as plain text.

Export Report using PDF Rendering Extension

The following steps provide an example of rendering a report in PDF format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default, a Page report is added to the project.
4. Add a reference to **MESCIUS.ActiveReports.Export.Pdf** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

VB code. Paste INSIDE the Form Load event.

```
' Provide the Page report you want to render.
Dim rptPath As New IO.FileInfo("..\..\PageReport1.rdlx")
Dim pageReport As New GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyPDF")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim pdfSetting As New GrapeCity.ActiveReports.Export.Pdf.Page.Settings()
' Set the rendering extension and render the report.
Dim pdfRenderingExtension As New GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension()
Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(pdfRenderingExtension, outputProvider, pdfSetting)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");
GrapeCity.ActiveReports.PageReport pageReport = new GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyPDF");
outputDirectory.Create();

// Provide settings for your rendering output.
GrapeCity.ActiveReports.Export.Pdf.Page.Settings pdfSetting = new GrapeCity.ActiveReports.Export.Pdf.Page.Settings();

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension pdfRenderingExtension = new
        GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension();

GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
        System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists
outputProvider.OverwriteOutputFile = true;

pageReport.Document.Render(pdfRenderingExtension, outputProvider, pdfSetting);
```

ZUGFeRD and Factur-X electronic invoices

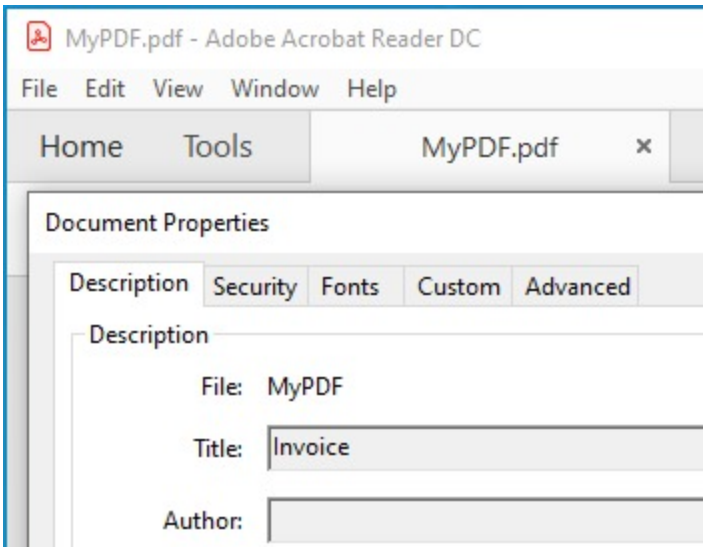
Metadata in PDFs

ActiveReports provides a special API for performing additional manual steps, required for creating ZUGFeRD reports. For details about this extension, see https://www.ferd-net.de/standards/zugferd-version-archive/zugferd-version-archive.html.

You should note the information below when working with a ZUGFeRD report:

- It should have the PDF/A-3 format.
- It should have the ZUGFeRD-invoice.xml attachment. We suggest to use the **Data Excel** export or **CSV**/**XML**/**JSON** exports to obtain data. We also suggest to use this library to generate ZUGFeRD-invoice.xml (for ZUGFeRD 1.0) or zugferd-invoice.xml (for ZUGFeRD 2.0) or factur-x.xml (for ZUGFeRD 2.1 and Factur-X).
- It should have additional XMP metadata.

Adding Metadata



Metadata such as keywords, descriptions are used by the search engines to narrow down the searches. You can add a number of predefined accessors, such as title, contributors, creators, copyright, description, etc using **AdditionalMetadata ('AdditionalMetadata Property' in the on-line documentation)** property. The allowed namespaces are:

- Dublin Core Properties
- XMP Core Properties
- PDF Properties

VB code. Paste INSIDE the Form Load event.

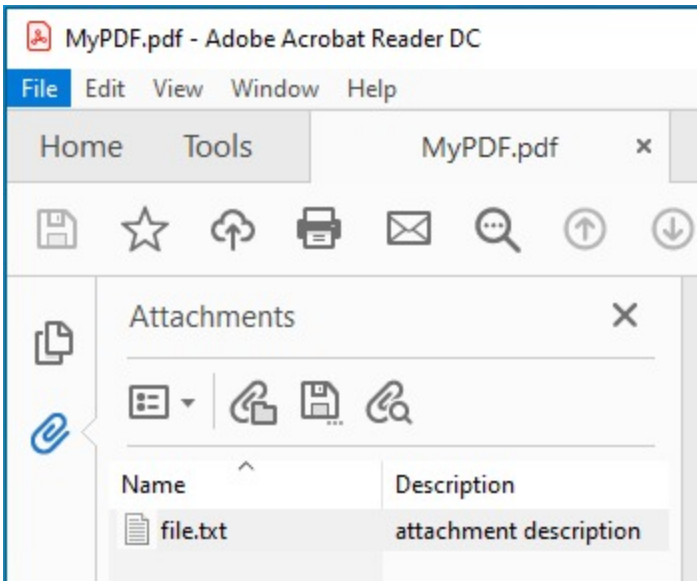
```
Dim rptPath As System.IO.FileInfo = New System.IO.FileInfo("..\..\PageReport1.rdlx")
Dim pageReport As GrapeCity.ActiveReports.PageReport = New
GrapeCity.ActiveReports.PageReport(rptPath)
Dim outputDirectory As System.IO.DirectoryInfo = New System.IO.DirectoryInfo("C:\MyPDF")
outputDirectory.Create()
Dim pdfSetting = New GrapeCity.ActiveReports.Export.Pdf.Page.Settings()
'Imports GrapeCity.ActiveReports.Export.Pdf
pdfSetting.AdditionalMetadata.Add(New AdditionalMetadataInfo With {
        .[Namespace] = AdditionalMetadataNamespace.PurlOrg,
        .Key = "title",
        .Value = "Invoice"
    })

Dim pdfRenderingExtension As GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension =
```

```
New GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension()
Dim outputProvider As GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider = New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(pdfRenderingExtension, outputProvider, pdfSetting)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");
GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyPDF");
outputDirectory.Create();

// Add meta data.
var pdfSetting = new GrapeCity.ActiveReports.Export.Pdf.Page.Settings();

// using GrapeCity.ActiveReports.Export.Pdf;
pdfSetting.AdditionalMetadata.Add(new AdditionalMetadataInfo
{
    Namespace = AdditionalMetadataNamespace.PurlOrg, // Dublin Core Properties
    Key = "title",
    Value = "Invoice"
});
GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension pdfRenderingExtension = new
GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));
outputProvider.OverwriteOutputFile = true;
pageReport.Document.Render(pdfRenderingExtension, outputProvider, pdfSetting);
```

Adding Attachment



You can include an attachment as metadata (such as invoices) to exported PDFs using **Attachments ('Attachments Property' in the on-line documentation)** property. This property allows attaching files such as a .xml or a .txt file in PDF. Below is the code example to export RDLX and Page reports to PDF and attach a file to the exported PDF.

VB code. Paste INSIDE the Form Load event.

```
' Provide the Page report you want to render.
Dim rptPath As System.IO.FileInfo = New System.IO.FileInfo("..\..\PageReport1.rdlx")
Dim pageReport As GrapeCity.ActiveReports.PageReport = New
GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As System.IO.DirectoryInfo = New System.IO.DirectoryInfo("C:\MyPDF")
outputDirectory.Create()

' Add attachment.
Dim pdfSetting = New GrapeCity.ActiveReports.Export.Pdf.Page.Settings()
pdfSetting.Attachments.Add(New AttachmentInfo With {
   .Name = "file.txt",
   .Content = System.IO.File.ReadAllBytes("D:\Reports\file.txt"),
   .Description = "attachment description" ' optional
})
Dim pdfRenderingExtension As GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension =
New GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension()
Dim outputProvider As GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider = New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))
outputProvider.OverwriteOutputFile = True
pageReport.Document.Render(pdfRenderingExtension, outputProvider, pdfSetting)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
```

```
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");
GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyPDF");
outputDirectory.Create();

// Add attachment.
var pdfSetting = new GrapeCity.ActiveReports.Export.Pdf.Page.Settings();
// using GrapeCity.ActiveReports.Export.Pdf;
pdfSetting.Attachments.Add(new AttachmentInfo
{
    Name = "file.txt",
    Content = System.IO.File.ReadAllBytes(@"D:\Reports\file.txt"),
    Description = "attachment description" // optional
});
// or
//{
//   Name = "file.xml",
//   Content = File.ReadAllBytes(Application.StartupPath + "\\file.xml")
//};
GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension pdfRenderingExtension = new
GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));
outputProvider.OverwriteOutputFile = true;
pageReport.Document.Render(pdfRenderingExtension, outputProvider, pdfSetting);
```

Open the exported PDF and you should see the attachment. Check the left sidebar in Adobe Acrobat Reader DC.

> **Note:** Metadata in PDFs is part of the Professional Edition. It is supported with the PDF version PDF/A-3b (or higher).

Image Export

Image is the format that converts your report to an image file. You can use the **ImageRenderingExtension ('ImageRenderingExtension Class' in the on-line documentation)** to render your report in this format. Make sure that you select an **ImageType ('ImageType Property' in the on-line documentation)** to any of the six different image formats available: BMP, GIF, JPEG, TIFF, and PNG.

> **Note:** By default, the image rendering extension creates a separate file for each page in a report and adds an index to each corresponding file name (for example, image001.PNG, image002.PNG, etc).
>
> To render the entire report as a single image, set the **Pagination ('Pagination Property' in the on-line documentation)** setting to **False**.

Image Rendering Properties

ActiveReports offers several options to control how reports render to Image.

Property	Description
Compression ('Compression Property' in the on-line documentation)	Sets or returns a value which specifies the compression to be used when exporting.
Dither ('Dither Property' in the on-line documentation)	Specifies whether the image should be dithered when saving to a black and white output format, like CCITT3 or Rle. This property has no effect if the CompressionScheme property is set to Lzw or None(represents color output).
DpiX ('DpiX Property' in the on-line documentation)	Adjust the horizontal resolution of rendered images. The default value is 96.
DpiY ('DpiY Property' in the on-line documentation)	Adjust the vertical resolution of rendered images.
EndPage ('EndPage Property' in the on-line documentation)	The default value of **0** in this property renders all of the report pages. Otherwise, set this value to the number of the last page to render. Please note that if the **StartPage** property is set to **0**, all of the pages of the report render. In order to use the **EndPage** property, you must set the **StartPage** property to a valid non-zero number.
ImageType ('ImageType Property' in the on-line documentation)	Select the type of image to which you want to render the report. Supported types are BMP, GIF, JPEG, TIFF, and PNG.
PageHeight ('PageHeight Property' in the on-line documentation)	Set the value in inches to use for the height of the image. The format is an integer or decimal with "in" as the suffix, for example, "11in" for 11 inches. The value set in this property overrides the report's settings.
PageWidth ('PageWidth Property' in the on-line documentation)	Set the value in inches to use for the height of the image. The format is an integer or decimal with "in" as the suffix, for example, "11in" for 11 inches. The value set in this property overrides the report's settings.
Pagination ('Pagination Property' in the on-line documentation)	By default, each page of a report is rendered as a separate image. Set this value to **False** to render the entire report as a single image.
PrintLayoutMode ('PrintLayoutMode Property' in the on-line documentation)	Select how to lay out the pages of the report in the image. • **OneLogicalPageOnSinglePhysicalPage** • **TwoLogicalPagesOnSinglePhysicalPage** • **FourLogicalPagesOnSinglePhysicalPage**

	• **EightLogicalPagesOnSinglePhysicalPage** • **BookletMode** (lays the pages out for booklet printing)
Quality ('Quality Property' in the online documentation)	Gets or sets the quality of the report to be rendered as an image.
SizeToFit ('SizeToFit Property' in the online documentation)	By default, rendered report pages are not resized to fit within the selected image size. Set this value to **True** to resize the report pages.
Start Page ('StartPage Property' in the online documentation)	The default value of zero in this and the **EndPage** properties render all of the report pages to images. Otherwise, set this value to the number of the first page to render.
WatermarkAngle ('WatermarkAngle Property' in the online documentation)	Specify the degree of angle for the watermark text on the image. Valid values range from 0 to 359, where 0 is horizontal, left to right.
WatermarkColor ('WatermarkColor Property' in the online documentation)	Select a color for the watermark text on the image. The default value for the watermark color is gray, but you can select any Web, System, or Custom color.
WatermarkFontName ('WatermarkFontName Property' in the online documentation)	Set the font to use for the watermark on the PDF document.
WatermarkFontSize ('WatermarkFontSize Property' in the online documentation)	Set the font size to use for the watermark on the PDF document.
WatermarkFontStyle ('WatermarkFontStyle Property' in the online documentation)	Set the font style to use for the watermark on the PDF document.
WatermarkTitle ('WatermarkTitle Property' in the online documentation)	Sets text (i.e. CONFIDENTIAL) to use as the watermark on the image.

Export Report using Image Rendering Extension

Reports rendered as images do not support any of the interactive features of Active Reports. Any data hidden at the time of export is hidden in the image. To display all data in a drill-down report, expand all toggle items prior to exporting.

The following steps provide an example of rendering a report in Image format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default, a Page report is added to the project.
4. Add a reference to **MESCIUS.ActiveReports.Export.Image** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

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

```
' Provide the Page report you want to render.

Dim rptPath As New IO.FileInfo("..\..\PageReport1.rdlx")

Dim pageReport As New GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyImage")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim imageSetting As New GrapeCity.ActiveReports.Export.Image.Page.Settings()
Dim setting As GrapeCity.ActiveReports.Extensibility.Rendering.ISettings = imageSetting

' Set the rendering extension and render the report.
Dim imageRenderingExtension As New
GrapeCity.ActiveReports.Export.Image.Page.ImageRenderingExtension()
Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(imageRenderingExtension, outputProvider, imageSetting)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyImage");
outputDirectory.Create();

// Provide settings for your rendering output.
GrapeCity.ActiveReports.Export.Image.Page.Settings imageSetting = new
```

```
GrapeCity.ActiveReports.Export.Image.Page.Settings();
GrapeCity.ActiveReports.Extensibility.Rendering.ISettings setting = imageSetting;

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Image.Page.ImageRenderingExtension imageRenderingExtension =
new GrapeCity.ActiveReports.Export.Image.Page.ImageRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;

pageReport.Document.Render(imageRenderingExtension, outputProvider, imageSetting);
```

XML Export

XML is a useful format for delivering data to other applications as the resulting XML file opens in an internet browser. You can use the **XmlRenderingExtension ('XmlRenderingExtension Class' in the on-line documentation)** to render your report in this format. XML is a good format for delivering data to other applications. The resulting XML file opens in an internet browser.

Xml Rendering Properties

ActiveReports offers several options to control how reports render to XML.

Property	Description
Encoding ('Encoding Property' in the on-line documentation)	Select the encoding schema to use in the XML transformation.
XslStylesheet ('XslStylesheet Property' in the on-line documentation)	Select the existing XSL Stylesheet file to use to transform the resulting XML file. **Note**: When using the XslStylesheet option, be sure to save the file in the correct file format, such as HTML.

Controlling XML Output

You can also control XML output through properties on the individual report controls. These properties are:

- **DataElementName** Indicates the name to use for the data element or attribute.
- **DataElementOutput** Indicates whether the report controls render in the XML output.
- **DataElementStyle** Indicates whether a text box renders as an element or an attribute.
- **DetailDataCollectionName** Indicates the name to use for the collection of all instances of the detail group in the XML output.

- **DetailDataElementName** Indicates the name to use for instances of the detail group in the XML output.
- **DetailDataElementOutput** Indicates whether the details appear in the XML output.
- **DataInstanceElementOutput** Indicates whether a list appears in the XML output. (This property is ignored if there is a grouping in the list.)
- **DataInstanceName** Indicates the name to use for instances of the list in the XML output.

Interactivity

XML format does not support interactive features except that when rendering a report to XML, complete drill-down data is shown regardless of whether the data is rendered in an expanded state or not.

Export Report using XML Rendering Extension

The following steps provide an example of rendering a report in XML format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default, a Page report is added to the project.
4. Add a reference to **MESCIUS.ActiveReports.Export.Xml** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

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

```
' Provide the Page report you want to render.

Dim rptPath As New IO.FileInfo("..\..\PageReport1.rdlx")

Dim pageReport As New GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyXml")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim xmlSetting As New GrapeCity.ActiveReports.Export.Xml.Page.Settings()
Dim setting As GrapeCity.ActiveReports.Extensibility.Rendering.ISettings = xmlSetting

' Set the rendering extension and render the report.
Dim xmlRenderingExtension As New
GrapeCity.ActiveReports.Export.Xml.Page.XmlRenderingExtension()

Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
```

```
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(xmlRenderingExtension, outputProvider, xmlSetting)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyXml");
outputDirectory.Create();

// Provide settings for your rendering output.
GrapeCity.ActiveReports.Export.Xml.Page.Settings xmlSetting = new
GrapeCity.ActiveReports.Export.Xml.Page.Settings();
GrapeCity.ActiveReports.Extensibility.Rendering.ISettings setting = xmlSetting;

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Xml.Page.XmlRenderingExtension xmlRenderingExtension = new
GrapeCity.ActiveReports.Export.Xml.Page.XmlRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;

pageReport.Document.Render(xmlRenderingExtension, outputProvider, xmlSetting);
```

Excel Export

Microsoft Excel is one of the formats to which you can render your report using **ExcelRenderingExtension ('ExcelRenderingExtension Class' in the on-line documentation)**. You can export Excel files in two formats, i.e. Xls and Xlsx.

Excel Rendering Properties

ActiveReports offers several options to control how reports render to Microsoft Excel.

Property	Description
PageSettings ('PageSettings Property' in the on-line documentation)	Returns an **ExcelRenderingExtensionPageSettings ('ExcelRenderingExtensionPageSettings Class' in the on-line documentation)** object for initializing Excel file print setting.

Pagination ('Pagination Property' in the on-line documentation)	Forces pagination or galley report layout mode.
RightToLeft ('RightToLeft Property' in the on-line documentation)	Shows direction of sheets from right to left.
Security ('Security Property' in the on-line documentation)	Returns an **ExcelRenderingExtensionSecurity ('ExcelRenderingExtensionSecurity Class' in the on-line documentation)** object for initializing document security.
UseDefaultPalette ('UseDefaultPalette Property' in the on-line documentation)	Indicates whether to export the document with the default Excel palette.
FileFormat ('FileFormat Property' in the on-line documentation)	Specifies the output format of the Excel document, i.e. Xls or Xlsx.
OpenXmlStandard ('OpenXmlStandard Property' in the on-line documentation)	Specifies the level of Open XML document conformance on exporting in Xlsx file format. You can choose from the following values: • **Transitional:** The default value. • **Strict:** The Excel file generated using Strict cannot be viewed on iOS devices.
MultiSheet ('MultiSheet Property' in the on-line documentation)	Indicates whether to generate a single-sheet or multi-sheet Excel document.
EnableToggles ('EnableToggles Property' in the on-line documentation)	Allows to export collapsible rows in the detail and row groups of the **Table** control of an RDLX report. This property gets displayed in the Export menu when the **Pagination** property is set to **False**.

Interactivity

Reports rendered in Excel support a number of interactive features like Bookmarks and Hyperlinks. However, in case

you have any data hidden at the time of rendering (like in a drill-down report), it does not show up in the output. It is recommended that you expand all toggle items prior to rendering.

Limitations

- BackgroundImage and rounded corners (for Shape and Container) are not exported.
- Overlapping controls are not supported and an incorrect result will be obtained in the case of export.
- LineSpacing is not retained after export.
- Exported FormattedText control does not preserve styles and formatting. It exports FormattedText as TextBox with plain text without any tags.
- Barcodes are exported as an image object so scanning of barcode images may fail in some cases. It depends on printer settings and the scanner quality. Barcodes may be blurred on export and the caption may get truncated in case of physical printing.
- Exported Boolean values are displayed in uppercase in both Xls and Xlsx files.
- TextIndent and FillCharacter properties of the TableOfContents control's Level setting are not supported.
- Text decoration (Underline and LineThrough) gets applied to the indented area when left padding is applied to the TableOfControl's levels.
- When a report is exported to Excel, CharWrap mode is ignored.
- The following properties of ActiveReports are not exported to Excel, since Excel does not have such settings:
  - CharacterSpacing
  - BorderWidth
  - MinCondenseRate

Export Report using Excel Rendering Extension

The following steps provide an example of rendering a report in Microsoft Excel format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default, a Page report is added to the project.
4. Add a reference to **MESCIUS.ActiveReports.Export.Excel** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

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

```
' Provide the Page report you want to render.

Dim rptPath As New IO.FileInfo("..\..\PageReport1.rdlx")

Dim pageReport As New GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyExcel")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim excelSetting As New
GrapeCity.ActiveReports.Export.Excel.Page.ExcelRenderingExtensionSettings()
```

```
excelSetting.FileFormat = GrapeCity.ActiveReports.Export.Excel.Page.FileFormat.Xls
Dim setting As GrapeCity.ActiveReports.Extensibility.Rendering.ISettings = excelSetting

' Set the rendering extension and render the report.
Dim excelRenderingExtension As New
GrapeCity.ActiveReports.Export.Excel.Page.ExcelRenderingExtension()
Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(excelRenderingExtension, outputProvider,
setting.GetSettings())
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyExcel");
outputDirectory.Create();

// Provide settings for your rendering output.
GrapeCity.ActiveReports.Export.Excel.Page.ExcelRenderingExtensionSettings excelSetting =
new GrapeCity.ActiveReports.Export.Excel.Page.ExcelRenderingExtensionSettings();
excelSetting.FileFormat = GrapeCity.ActiveReports.Export.Excel.Page.FileFormat.Xls;
GrapeCity.ActiveReports.Extensibility.Rendering.ISettings setting = excelSetting;

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Excel.Page.ExcelRenderingExtension
excelRenderingExtension = new
GrapeCity.ActiveReports.Export.Excel.Page.ExcelRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;

pageReport.Document.Render(excelRenderingExtension, outputProvider,
setting.GetSettings());
```

Word Export

The **WordRenderingExtension ('WordRenderingExtension Class' in the on-line documentation)** class renders your reports to the native Microsoft Word file formats. You can export Page reports and RDLX reports to **Microsoft Office Open XML (OOXML)** format (.Docx) or Word HTML format (.Doc) using the **FileFormat ('FileFormat Property' in the on-line documentation)** property.

The Word HTML format (.Doc) provides greater layout accuracy for Page and RDLX Reports in Microsoft Word, on the other hand, OOXML format (.Docx) provides excellent editing experience for the exported reports.

The OOXML format (.Docx) is recommended in the following scenarios:

- **Open exported reports in a wide range of applications:** Users can open and modify the exported Word document in any of the following applications.
  - Microsoft Office 2013+
  - Microsoft Office for Mac 2016+
  - iWork and Pages for OS X (all supported versions)
  - LibreOffice
  - Google Quickoffice for Android
  - Documents Free (Mobile Office Suite) by SavySoda for iOS

  > **Note:** Besides the applications listed above, .Docx files exported through ActiveReports WordRenderingExtension class might work in other applications that support the .Docx format.
- **Customize reports after exporting:** Positioning and arrangement of report elements in the exported document is implemented using the OOXML format (.Docx) which provides a natural document flow for editing the exported documents.
- **Use Word automation features:** With support for automation features in the OOXML format (.Docx), tasks that previously required manual adjustments in the exported Word document are now handled automatically. Report elements such as page header and footer, expressions, heading levels, and table of contents are automatically transformed to the OOXML format (.Docx).
- **Set compatibility mode:** You can render a report as a Word document that is compatible with Microsoft Word 2007, 2010, or 2013 using the **DocumentCompatibleVersion** property from the export settings.

Word Rendering Extension Properties

ActiveReports offers several options to control how reports render to Microsoft Word.

Common properties (HTML and OOXML)

Property	Description
Author ('Author Property' in the on-line documentation)	Sets the name of the author that appears in the Author field of the Properties dialog in the rendered Word document.
FileFormat ('FileFormat Property' in the on-line documentation)	Sets the output file format to HTML (.Doc) or OOXML (.Docx). By default, the file format is set to HTML format.

Title ('Title Property' in the on-line documentation)	Sets the title for a document that appears in the Title field of properties dialog in the rendered Word document.

HTML format

Property	Description
BaseUrl ('BaseUrl Property' in the on-line documentation)	Sets the base URL for any relative hyperlinks that appear in the Hyperlink base field of the Properties dialog in the rendered Word document.
Generator ('Generator Property' in the on-line documentation)	Sets the identity of the document generator in the rendered Word document.
PageHeight ('PageHeight Property' in the on-line documentation)	Sets the height of the report pages in inches for the rendered Word document. The value in this property overrides the original settings in the report.
PageWidth ('PageWidth Property' in the on-line documentation)	Sets the width of the report pages in inches for the rendered Word document. The value in this property overrides the original settings in the report.
UseMhtOutput ('UseMhtOutput Property' in the on-line documentation)	Indicates whether Mht output is to be used for the resultant Word document or not.

OOXML format

Property	Description
CompanyName ('CompanyName Property' in the on-line documentation)	Sets the name of the organization or company that appears in the Company field of Properties dialog in the rendered Word document.
DocumentCompatibilityVersion ('DocumentCompatibilityVersion Property' in the on-line documentation)	Sets the compatibility mode of the document to previous versions (Microsoft Word 2007 - 2013) of Word. By default, the compatibility version is set to Word2013.

DpiX ('DpiX Property' in the on-line documentation)	Sets the horizontal resolution of the images in the rendered Word document. By default, DpiX is set to 96.
DpiY ('DpiY Property' in the on-line documentation)	Sets the vertical resolution of the images in the rendered Word document. By default, DpiY is set to 96.
PageOrientation ('PageSettings Property' in the on-line documentation)	Sets a value that specifies whether the document pages should be printed in portrait or landscape in the rendered Word document.
PaperSize ('PageSettings Property' in the on-line documentation)	Sets the paper size for the page.
Password ('SecuritySettings Property' in the on-line documentation)	Sets a password that must be provided to open the rendered Word document.
ReadOnlyRecommended ('SecuritySettings Property' in the on-line documentation)	Sets a value that indicates whether Microsoft Office Word displays a message whenever a user opens the document, suggesting that the document is read-only.
WritePassword ('SecuritySettings Property' in the on-line documentation)	Sets the write password that is required for saving changes in the rendered Word document.
TOCAutoUpdate ('TOCAutoUpdate Property' in the on-line documentation)	Automatically updates the TableOfContents control while opening the Word document. By default, TOCAutoUpdate is set to False.

Interactivity

HTML format

Reports rendered in a Word format supports both Bookmarks and Hyperlinks. However, if visibility toggling (like in a drill-down report) is essential to your report, it is recommended to use the HTML rendering extension. If a Document map is essential to your report, it is recommended to use the PDF rendering extension.

OOXML format

- **Hyperlinks -** Hyperlinks on TextBox and Image controls are rendered as hyperlinks in Microsoft Word.
- **Bookmarks** - Bookmarks in the report are rendered as Microsoft Word bookmarks. Bookmark links are rendered as hyperlinks that link to the bookmark labels within the document.
- **TOC AutoUpdate** - TableofContents control in the report is rendered as a Microsoft Word table of contents.

Limitations

HTML format

- Although background colors for controls export to Word documents, background colors for sections such as Body and Page Header or Footer do not.
- The **BackgroundImage** is not supported when used in TextBox and CheckBox controls embedded in data

regions such as Table and Tablix.
- The **BackgroundImage** is not supported for reports, and for **List**, **Container**, **Shape**, **FormattedText**, **Table**, and **Tablix** report controls.
- **KeepTogether** property of Table/Tablix is not supported.
- Some FormattedText tags, for example bi and <s>es</s>, are not exported to Word.
- Image alignments other than the defaults (HorizontalAlignment: Left and VerticalAlignment: Top) are not supported.
- Checkbox color does not affect the color of the square.

OOXML format

Report properties

- The **LineSpacing** property of a report's style sheet, **StartPageNumber** property of the report, **PrintOnLastPage** property of the PageHeader and PageFooter are not supported.
- For Page reports, some of the **NumberingStyle** property (**DocumentMap** settings) options are not supported. The supported **NumberingStyle** options are **Decimal**, **DecimalZero**, **LowerLetter**, **UpperLetter**, **LowerRoman**, **UpperRoman**.
- Background image is not supported for report item except Shape.
- The **BackgroundRepeat** property of the BackgroundImage is not supported in Page (Page reports) and Body (RDLX reports).
- For RDLX reports, **Background** and **Border** properties of Page Header or Page Footer are not supported.
- Microsoft Word calculates the width of the columns by the document width. an RDLX report calculates the width of the columns based on the body width, therefore, the width of columns in an exported RDLX report may differ from an original RDLX report.
- In Microsoft Word, the maximum supported page size is 22 inches (55.87 cm) wide and 22 inches (55.87 cm) high. If an exported report exceeds the maximum size, some data may be lost during export.
- In Microsoft Word, a table can have a maximum of 63 columns. If an exported report table has more than 63 columns, then the table is split and therefore an exported document may differ from an original report.
- A repeated Table Footer (the **RepeatOnNewPage** property of Table Footer) or multiple repeated headers on a single page are not supported.
- The **OverflowPlaceHolder** control is not supported.
- If the **PrintOnFirstPage** property of PageHeader or PageFooter is set to **False**, then both PageHeader and PageFooter will not be available on the first page of the exported document.
- The report data gets rendered only in the first theme if a Page report containing multiple themes is exported to Docx format.

Report controls

- The **Inset**, **Outset**, and **Windowsinset** border styles (in **BorderStyle** property) are not supported.
- The **Map**, **Chart**, **Image**, **Barcode**, **SparkLine**, **Bullet**, and **CustomControl** report controls are exported as an image. If a report control uses the BorderColor, BorderStyle or BorderWidth properties, a report is exported as a table.
- The **BorderWidth** property of report controls is not exported as-is and may differ from the original BorderWidth value.
- **PageBreaks** are not fully supported. The report contents exported to the Word's table or cell items do not support the page breaks.
- For **Shape** report control, if the **BorderStyle** property is set to **Double**, it is exported as Solid.
- For **Line** control, if **LineStyle** property is set to Double/Transparent, it is exported as Solid.
- For **BandedList** data region, only the BandedList Header is repeated on each page. The BandedList Footer,

GroupHeaders and GroupFooters are not supported.
- **Tablix** data region is exported as a single table without horizontal split.
- For **Image** control, **Border** properties and **Padding** properties are not supported if the **Sizing** property is set to **Clip**.
- For **Container** report control, rounding corners (the **RoundingRadius** property) are not supported.
- **RepeatToFill** property for Table and Tablix is not supported.
- Images embedded in a Table data region are not properly supported.
- Pagination is not supported due to difference in the layouts of ActiveReports and Word.
- Page Number in Section (Page N of M(Section)) is not supported.
- **KeepTogether** property of Table/Tablix is not supported.
- Horizontal aligned images may overlap in iWord.
- The properties set to the 'Body' region of a Subreport are not exported.
- If a report contains overlapped report controls, these controls appear side by side in an exported Word document and not overlapped as in the Designer's preview.

FormattedText

- FormattedText is exported as it is. It does not support all HTML and CSS features.
- The <a> tag without a href attribute, <abbr> and <q> tags are exported as simple text.
- For Border Styles - Inset and Outset, the <ul> tags are not exported.
- The **BackgroundColor**, **BackgroundImage**, **BorderColor**, **BorderStyle** and **BorderWidth** properties are not supported.
- Anchors with an href attribute are exported as hyperlinks.
- Headers like h1, h2, etc. are exported as corresponding Microsoft Word built-in header styles.

TableOfContents

- The **TextAlign**, **DisplayPageNumber**, **TextIndent**, and **Overline TextDecoration** properties are not supported.
- The **Source** property of the Document Map settings is not fully supported. Only the **Headings Only** option of the **Source** property is supported.
- If a report uses more than one TableOfContents controls, the properties of the first TableOfContents are applied to the other TableOfContents controls in the exported document.
- The **FillCharacter** property is exported as dots.
- The background of TOC control appears black on opening the exported file in LibreOffice.

TextBox/CheckBox

- If the **Format** property is set to **Numeric** or **Date,** the exported TextBox has the right alignment. Other Format values are exported with the left alignment.
- The **Transparent color** for text is exported as white.
- The **Underline** for numbered lists, Right-To-Left (RTL) option of the **Direction, Angle**, **ShrinkToFit,** and **Overline TextDecoration** properties are not supported.
- The action **Jump to report** is not supported.
- The **tb-rl** (vertical text) option of the **WritingMode** property is not supported. TextBoxes with the **WritingMode** property set to tb-rl are exported as lr-tb.
- The **NoWrap** option of the **WrapMode** property is exported as WordWrap.
- The **LineSpacing** property of an exported document will differ from the original report. This is because, in Microsoft Word, the line spacing is calculated by the font size value of a report control plus the line spacing value of a report control.
- For **CheckBox** control, the **CheckAlignment** property is exported as MiddleRight for TopRight, MiddleRight, and BottomRight options. Other CheckAlignment options are exported as MiddleLeft.

- **Paddings** exceeding 31 inches is exported as border spaces.
- **Right-To-Left** text direction does not work in the LibreOffice.
- On exporting to Word 2013, when the background (shading) and padding are applied to a paragraph, the padding is also applied to the background, so a gap between the border and the background appears on the left side of the paragraph.
- **CharWrap** property is not supported.
- The fields in a TextBox control are evaluated as follows:
  - PageNumber and TotalPages expressions are exported as special fields, evaluated by a text editor (Word or other).
  - The fields placed in Header or Footer are automatically evaluated.
  - The fields placed in the Body should be re-evaluated manually by clicking 'Update field' from the context menu.

Export Report using Word Rendering Extension

The following steps provide an example of rendering a report in Word format (.doc or .docx).

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default, a Page report is added to the project.
4. Add a reference to **MESCIUS.ActiveReports.Export.Word** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event to render your report in .OOXML or .HTML file format.

> **Note:** To export your report in Word HTML format (.Doc), change the **FileFormat** property option from OOXML to HTML format as shown.
> `wordSetting.FileFormat = GrapeCity.ActiveReports.Export.Word.Page.FileFormat.HTML`

To export a report in .Docx file format

VB code. Paste INSIDE the Form Load event.

```
' Provide the Page report you want to render.

Dim rptPath As New IO.FileInfo("..\..\PageReport1.rdlx")

Dim pageReport As New GrapeCity.ActiveReports.PageReport(rptPath))

' Create an output directory.

Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyWord")
outputDirectory.Create()

' Provide settings for your rendering output.

Dim wordSetting As New GrapeCity.ActiveReports.Export.Word.Page.Settings()
```

```
' Set the FileFormat property to .OOXML.

wordSetting.FileFormat = GrapeCity.ActiveReports.Export.Word.Page.FileFormat.OOXML

' Set the rendering extension and render the report.

Dim wordRenderingExtension As New
GrapeCity.ActiveReports.Export.Word.Page.WordRenderingExtension()
Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.

outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(wordRenderingExtension, outputProvider, wordSetting)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.

System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

 // Create an output directory.

System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyWord");
outputDirectory.Create();

// Provide settings for your rendering output.

GrapeCity.ActiveReports.Export.Word.Page.Settings wordSetting = new
GrapeCity.ActiveReports.Export.Word.Page.Settings();

// Set the FileFormat property to .OOXML.

wordSetting.FileFormat = GrapeCity.ActiveReports.Export.Word.Page.FileFormat.OOXML;

// Set the rendering extension and render the report.

GrapeCity.ActiveReports.Export.Word.Page.WordRenderingExtension wordRenderingExtension = new
GrapeCity.ActiveReports.Export.Word.Page.WordRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));
```

```
// Overwrite output file if it already exists.

outputProvider.OverwriteOutputFile = true;

pageReport.Document.Render(wordRenderingExtension, outputProvider, wordSetting);
```

CSV Export

Comma-Separated Values (CSV) is a form of structured data in plain text. The text in a CSV file is saved as series of values separated by commas. You can use the **CsvRenderingExtension ('CsvRenderingExtension Class' in the on-line documentation)** to render your report in this format.

CSV Rendering Properties

ActiveReports offers several options to control how reports render to CSV.

Property	Description
ColumnsDelimiter ('ColumnsDelimiter Property' in the on-line documentation)	Sets or returns the text inserted between columns.
DateTimeFormat ('DateTimeFormat Property' in the on-line documentation)	Specifies the default format for date values, for example, 'yyyy-MM-dd'.
Encoding ('Encoding Property' in the on-line documentation)	Specifies the encoding schema for output.
NoHeader ('NoHeader Property' in the on-line documentation)	Specifies whether to omit the CSV Header.
NumericFormat ('NumericFormat Property' in the on-line documentation)	Specifies the format for numeric values, for example, '0.####'.
QuotationMode	Specifies whether to add double quotes to the exported data.

('QuotationMode Property' in the on-line documentation)	• **AutoQuote** – Simple values are exported without quotes. The quotes are added only when the data contains column or row delimiters. This is the default export behavior. • **AlwaysQuote** – Exported values are always quoted.
QuotationSymbol ('QuotationSymbol Property' in the on-line documentation)	Sets or returns the qualifier character to put around results.
RowsDelimiter ('RowsDelimiter Property' in the on-line documentation)	Sets or returns the text inserted between rows.

Export Report using CSV Rendering Extension

The following steps provide an example of rendering a report in the Csv format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default, a Page report is added to the project.
4. Add a reference to **MESCIUS.ActiveReports.Export.Xml** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

VB code. Paste INSIDE the Form Load event.

```
' Provide the Page report you want to render.

Dim rptPath As System.IO.FileInfo = New System.IO.FileInfo("..\..\PageReport1.rdlx")
Dim pageReport As GrapeCity.ActiveReports.PageReport = New
GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyCSV")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim csvSettings As New
GrapeCity.ActiveReports.Export.Text.Page.CsvRenderingExtension.Settings()

csvSettings.ColumnsDelimiter = ","
csvSettings.Encoding = System.Text.Encoding.UTF8
```

```
csvSettings.NoHeader = True
csvSettings.QuotationMode = GrapeCity.ActiveReports.Export.Text.Page.QuotationMode.AlwaysQuote
csvSettings.QuotationSymbol = """c
csvSettings.RowsDelimiter = vbCr & vbLf

' Set the rendering extension and render the report.
Dim csvRenderingExtension As New
GrapeCity.ActiveReports.Export.Text.Page.CsvRenderingExtension()
Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(csvRenderingExtension, outputProvider, csvSettings)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");
GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyCSV");
outputDirectory.Create();

// Provide settings for your rendering output.
GrapeCity.ActiveReports.Export.Text.Page.CsvRenderingExtension.Settings csvSettings = new
GrapeCity.ActiveReports.Export.Text.Page.CsvRenderingExtension.Settings();
csvSettings.ColumnsDelimiter = ",";
csvSettings.Encoding = Encoding.UTF8;
csvSettings.NoHeader = true;
csvSettings.QuotationMode =
GrapeCity.ActiveReports.Export.Text.Page.QuotationMode.AlwaysQuote;
csvSettings.QuotationSymbol = '"';
csvSettings.RowsDelimiter = "\r\n";

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Text.Page.CsvRenderingExtension csvRenderingExtension = new
GrapeCity.ActiveReports.Export.Text.Page.CsvRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;
pageReport.Document.Render(csvRenderingExtension, outputProvider, csvSettings);
```

JSON Export

JavaScript Object Notation (JSON) is a text-based data format in which the data is stored in the hierarchical form. You can use the **JsonRenderingExtension ('JsonRenderingExtension Class' in the on-line documentation)** to render your report in this format.

JSON Rendering Properties

ActiveReports offers several options to control how reports render to JSON.

Property	Description
Formatted ('Formatted Property' in the on-line documentation)	Specifies whether to format the file with tabs and spaces for readability.
QuotePropertyNames ('QuotePropertyNames Property' in the on-line documentation)	Specifies whether to enclose property names in quotation marks.

Export Report using JSON Rendering Extension

The following steps provide an example of rendering a report in JSON format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default, a Page report is added to the project.
4. Add a reference to **MESCIUS.ActiveReports.Export.Xml** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

VB code. Paste INSIDE the Form Load event.

```
' Provide the Page report you want to render.
Dim rptPath As System.IO.FileInfo = New System.IO.FileInfo("..\..\PageReport1.rdlx")
Dim pageReport As GrapeCity.ActiveReports.PageReport = New
GrapeCity.ActiveReports.PageReport(rptPath)



' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyJSON")
outputDirectory.Create()

' Provide settings for your rendering output.
```

```
Dim jsonSettings As New
GrapeCity.ActiveReports.Export.Text.Page.JsonRenderingExtension.Settings()
jsonSettings.Formatted = True

' Set the rendering extension and render the report.
Dim jsonRenderingExtension As New
GrapeCity.ActiveReports.Export.Text.Page.JsonRenderingExtension()

Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))
' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True

pageReport.Document.Render(jsonRenderingExtension, outputProvider, jsonSettings)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");
GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyJSON");
outputDirectory.Create();

// Provide settings for your rendering output.
eReports.Export.Text.Page.JsonRenderingExtension.Settings jsonSettings = new
        GrapeCity.ActiveReports.Export.Text.Page.JsonRenderingExtension.Settings();

jsonSettings.Formatted = true;

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Text.Page.JsonRenderingExtension jsonRenderingExtension = new
        GrapeCity.ActiveReports.Export.Text.Page.JsonRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
        System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;

pageReport.Document.Render(jsonRenderingExtension, outputProvider, jsonSettings);
```

Excel Data Export

Excel Data is one of the formats to which you can render your report using **ExcelTransformationDevice**

('ExcelTransformationDevice Class' in the on-line documentation). You can export excel files in two formats, Xlsx and Csv.

Excel Data exports only data from Tablix, Table, and Matrix data regions, preserving the data region structure and ignoring layout-related features (page break, cumulative total, etc). Other controls and data regions of the original report are ignored at this export. The **FrozenRows** and **FrozenColumns** settings of Table and Tablix are handled at the report export and can be applied in the exported file.

> **Note**: If a report does not contain any data region (Table or Tablix), a Csv file is not generated.

For the Xlsx format, when a report has multiple data regions, each data region is exported to a separate excel sheet.

For the Csv format, a separate CSV file is created for each data region, available in the report.

The following steps provide an example of rendering a report in the Microsoft Excel format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Page report Application**. By default a Page Report is added to the project.
4. Add **MESCIUS.ActiveReports.Export.Excel** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

Csv Data

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

```
' Provide the Page Report you want to render.
Dim rptPath As System.IO.FileInfo = New System.IO.FileInfo("..\..\PageReport1.rdlx")

Dim pageReport As GrapeCity.ActiveReports.PageReport = New
GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyCsvData")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim settings = New GrapeCity.ActiveReports.Export.Excel.Page.Settings()

settings.FileFormat = GrapeCity.ActiveReports.Export.Excel.Page.ExcelDataFileFormat.Csv
settings.Csv.ColumnsDelimiter = ","
settings.Csv.Encoding = System.Text.Encoding.UTF8
settings.Csv.NoHeader = False
settings.Csv.QuotationSymbol = """c
settings.Csv.RowsDelimiter = vbCrLf

' Set the rendering extension and render the report.
Dim outputProvider As New
```

```
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True
pageReport.Document.Render(New
GrapeCity.ActiveReports.Export.Excel.Page.ExcelTransformationDevice(), outputProvider,
settings)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page Report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new System.IO.DirectoryInfo(@"C:\MyCsvData");
outputDirectory.Create();

// Provide settings for your rendering output.
var settings = new GrapeCity.ActiveReports.Export.Excel.Page.Settings();

settings.FileFormat = GrapeCity.ActiveReports.Export.Excel.Page.ExcelDataFileFormat.Csv;
settings.Csv.ColumnsDelimiter = ",";
settings.Csv.Encoding = Encoding.UTF8;
settings.Csv.NoHeader = false;
settings.Csv.QuotationSymbol = '"';
settings.Csv.RowsDelimiter = "\r\n";

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;
pageReport.Document.Render(new
GrapeCity.ActiveReports.Export.Excel.Page.ExcelTransformationDevice(), outputProvider,
settings);
```

Excel Data

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

```
' Provide the Page Report you want to render.
Dim rptPath As System.IO.FileInfo = New System.IO.FileInfo("..\..\PageReport1.rdlx")
```

```
Dim pageReport As GrapeCity.ActiveReports.PageReport = New
GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyExcelData")
outputDirectory.Create()

' Provide settings for your rendering output.

Dim settings = New GrapeCity.ActiveReports.Export.Excel.Page.Settings()

settings.FileFormat = GrapeCity.ActiveReports.Export.Excel.Page.ExcelDataFileFormat.Xlsx
settings.Xlsx.AllowImages = True
settings.Xlsx.UseCompression = True
settings.Xlsx.OpenXmlStandard =
GrapeCity.ActiveReports.Export.Excel.Page.OpenXmlStandard.Transitional
settings.Xlsx.Security.ReadOnlyRecommended = True

' Set the rendering extension and render the report.
Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True
pageReport.Document.Render(New
GrapeCity.ActiveReports.Export.Excel.Page.ExcelTransformationDevice(), outputProvider,
settings)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page Report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new
System.IO.DirectoryInfo(@"C:\MyExcelData");
outputDirectory.Create();

// Provide settings for your renderidng output.
var settings = new GrapeCity.ActiveReports.Export.Excel.Page.Settings();

settings.FileFormat =
GrapeCity.ActiveReports.Export.Excel.Page.ExcelDataFileFormat.Xlsx;
settings.Xlsx.AllowImages = true;
settings.Xlsx.UseCompression = true;
```

```
settings.Xlsx.OpenXmlStandard =
GrapeCity.ActiveReports.Export.Excel.Page.OpenXmlStandard.Transitional;
settings.Xlsx.Security.ReadOnlyRecommended = true;

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;
pageReport.Document.Render(new
GrapeCity.ActiveReports.Export.Excel.Page.ExcelTransformationDevice(), outputProvider,
settings);
```

Excel Data Rendering Properties

ActiveReports offers several options to control how reports render to Excel Data.

Property	Description
Csv	Csv related properties. See **Csv Rendering Properties** below.
FileFormat ('FileFormat Property' in the on-line documentation)	Indicates whether to use Csv or OpenXml format for the output file.
Xlsx	OpenXml related properties. See **Xlsx Rendering Properties** below.

Csv Rendering Properties

Property	Description
ColumnsDelimiter ('ColumnsDelimiter Property' in the on-line documentation)	Sets or returns the text inserted between columns.
Encoding ('Encoding Property' in the on-line documentation)	Specifies the encoding schema for output.
NoHeader ('NoHeader Property' in the	Specifies whether to omit the CSV Header.

on-line documentation) ('MultiSheet Property' in the on-line documentation)	
QuotationSymbol ('QuotationSymbol Property' in the on-line documentation)	Sets or returns the qualifier character to put around results.
RowsDelimiter ('RowsDelimiter Property' in the on-line documentation)	Sets or returns the text inserted between rows.

Xlsx Rendering Properties

Property	Description
AllowImages ('AllowImages Property' in the on-line documentation)	Indicates whether to allow images or just plain data content.
Author ('Author Property' in the on-line documentation)	Sets the name of the author that appears in the Author field in the Properties of the exported Excel document.
AutoRowsHeight ('AutoRowsHeight Property' in the on-line documentation)	Indicates whether to export rows height or specify auto height.
Categories ('Categories Property' in the on-line documentation)	Sets the name of the categories that appears in the Categories field in the Properties of the exported Excel document.
OpenXmlStandard ('OpenXmlStandard Property' in the on-line documentation)	Specifies the level of Open XML document conformance on exporting in Xlsx file format. You can choose from the following values: • **Transitional:** The default value. • **Strict:** The Excel file generated using Strict cannot be viewed on iOS devices.
('MultiSheet Property' in the on-line documentation)RightToLeft ('RightToLeft Property' in	Shows direction of sheets from right to left.

the on-line documentation)	
Security ('Security Property' in the on-line documentation)	Returns an **ExcelRenderingExtensionSecurity ('ExcelRenderingExtensionSecurity Class' in the on-line documentation)** object for initializing document security.
Title ('Title Property' in the on-line documentation)	Sets the name of the title for a document that appears in the Title field in the Properties of the exported Excel document.
UseCompression ('UseCompression Property' in the on-line documentation)	Indicates whether to use compression on exporting to an Xlsx file.

Text Print Export

The Text Print is a format recommended for printing the RDLX reports with Table and Tablix data regions in the tabular format, preserving the grouping, layout, and formatting. The export is a cross-platform replacement for Text Only printers and is suitable for printing reports in ASCII format, especially on the Dot Matrix printers.

Use the **TxtRenderingExtension ('TxtRenderingExtension Class' in the on-line documentation)** to render your report in Text Print format.

The following steps provide an example of rendering a report in the Text Print format.

1. Create a new Visual Studio project.
2. In the **New Project** dialog that appears, select **ActiveReports 18 Page report Application** and specify a name for the project in the Name field.
3. Click **OK** to create a new **ActiveReports 18 Fixed Page Layout Application**. By default, a Page Report is added to the project.
4. Add reference to **MESCIUS.ActiveReports.Export.Xml** package in the project.
5. On the Form.cs or Form.vb that opens, double-click the title bar to create the Form_Load event.
6. Add the following code inside the Form_Load event.

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

```
' Provide the Page Report you want to render.
Dim rptPath As System.IO.FileInfo = New System.IO.FileInfo("..\..\PageReport1.rdlx")

Dim pageReport As GrapeCity.ActiveReports.PageReport = New
GrapeCity.ActiveReports.PageReport(rptPath)

' Create an output directory.
Dim outputDirectory As New System.IO.DirectoryInfo("C:\MyTextPrint")
outputDirectory.Create()

' Provide settings for your rendering output.
Dim txtSettings = New GrapeCity.ActiveReports.Export.Text.Page.Settings()
txtSettings.HorizontalPaddings =
GrapeCity.ActiveReports.Export.Text.Page.PaddingsType.Keep
txtSettings.LineEnding = "\r\n"
txtSettings.CharHeight = 13
```

```
txtSettings.CharWidth = 7

' Set the rendering extension and render the report.

Dim txtRenderingExtension As
GrapeCity.ActiveReports.Export.Text.Page.TxtRenderingExtension = New
GrapeCity.ActiveReports.Export.Text.Page.TxtRenderingExtension()

Dim outputProvider As New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name))

' Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = True
pageReport.Document.Render(txtRenderingExtension, outputProvider, txtSettings)
```

C# code. Paste INSIDE the Form Load event.

```
// Provide the Page Report you want to render.
System.IO.FileInfo rptPath = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

GrapeCity.ActiveReports.PageReport pageReport = new
GrapeCity.ActiveReports.PageReport(rptPath);

// Create an output directory.
System.IO.DirectoryInfo outputDirectory = new
System.IO.DirectoryInfo(@"C:\MyTextPrint");
outputDirectory.Create();

// Provide settings for your rendering output.
var txtSettings = new GrapeCity.ActiveReports.Export.Text.Page.Settings();
txtSettings.HorizontalPaddings =
GrapeCity.ActiveReports.Export.Text.Page.PaddingsType.Keep;
txtSettings.LineEnding = "\r\n";
txtSettings.CharHeight = 13;
txtSettings.CharWidth = 7;

// Set the rendering extension and render the report.
GrapeCity.ActiveReports.Export.Text.Page.TxtRenderingExtension txtRenderingExtension =
new GrapeCity.ActiveReports.Export.Text.Page.TxtRenderingExtension();
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider outputProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputDirectory,
System.IO.Path.GetFileNameWithoutExtension(outputDirectory.Name));

// Overwrite output file if it already exists.
outputProvider.OverwriteOutputFile = true;
pageReport.Document.Render(txtRenderingExtension, outputProvider, txtSettings);
```

Text Print Rendering Properties

The following options are available while rendering to Text Print format.

Property	Description
CharHeight ('CharHeight Property' in the on-line documentation)	Specifies the character height in points.
CharWidth ('CharWidth Property' in the on-line documentation)	Specifies the character width in points.
FontFamily ('FontFamily Property' in the on-line documentation)	Specifies the monospace font family. The available options are: • Consolas • Courier New • Lucida Console
FontSize ('FontSize Property' in the on-line documentation)	Specifies the font height in points. Recommended values are Consolas 11, Courier New 10, or Lucida Console 10.
HorizontalPaddings ('HorizontalPaddings Property' in the on-line documentation)	Specifies the horizontal padding: • Keep: Retain the original padding • Adjust: Adjust the padding • Remove: Remove the padding
LineEnding ('LineEnding Property' in the on-line documentation)	Specifies the end of a line.

Limitations

- Ignores graphics; the output is only plain text.
- Pagination is not preserved, the output is rendered on a single page

Export Section Reports

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. Explore the following export abilities available in Section reports, as well as for Page/RDLX reports.

- HTML Export
- PDF Export

HTML Export

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 **HTMLExport ('HtmlExport Class' in the on-line documentation)** object after adding reference to **MESCIUS.ActiveReports.Export.Html** package in your project.

HTML Export Filter Properties

Property	Valid Values	Description
BookmarkStyle ('BookmarkStyle Property' in the on-line documentation)	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 ('CharacterSet Property' in the on-line documentation)	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 ('CreateFramesetPage Property' in the on-line documentation)	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 ('IncludeHtmlHeader Property' in the on-line documentation)	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 ('IncludePageMargins Property' in the on-line documentation)	True or False (default)	Set to True to include the report's margins in the HTML output.
MultiPage ('MultiPage Property' in the on-line documentation)	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 ('OutputType Property' in the on-line documentation)	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 CSS.

RemoveVerticalSpace ('RemoveVerticalSpace Property' in the on-line documentation)	True or False (default)	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 ('Title Property' in the on-line documentation)	Any String	Enter the text to use in the header section's title. This is displayed in the title bar of the browser.

Additional Information on Output Types

By default, the report is exported as DynamicHtml (DHTML), with cascading style sheets (CSS). Using the **OutputType ('OutputType Property' in the on-line documentation)** 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 formats. Following is the usage of each output type and controls to avoid in each.

DynamicHtml (DHTML) usage and limitations

Usage:

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

Does not support:

- Diagonal line control
- Control borders

LegacyHtml (HTML) usage and limitations

Usage:

- Create archival reports
- Open in Web browsers

Does not support:

- Line control
- Control borders
- CrossSectionLine controls
- Overlapping controls
- MinCondenseRate property

HTML Export Filter Limitations

- Line spacing in exported HTML can be different from the line spacing in Viewer.
- There may be space between the borders of each control in exported file.
- Text in RichTextBox may appear overlapped.
- Vertical Text and Bookmarks are not supported.
- Any text may appear overlapped.

- Borders are not supported.

Export Report using HTML Export Filter

Use the following steps to export reports through HTML export filter.

1. Create a new or open an existing Visual Studio project.
    - If you are creating a new project,
      - select **ActiveReports 18 Section Report (xml-based) Application** in **Create a New Project** dialog, and then
      - specify a name for the project, and click **OK**.
    - If you are using an existing project,
      - go to the Solution Explorer and right-click the project and select **Add** > **New Item**, and then
      - select **ActiveReports 18 Section Report (xml-based)** and click **Add**.
2. Add a reference to **MESCIUS.ActiveReports.Export.Html** package in the project. See Manage ActiveReports Dependencies for more information.
3. In your project's **Bin>Debug** folder, place the **report.rpx** (Section Report).
4. On the Form.cs or Form.vb, double-click the title bar to create the Form_Load event.
5. In **Form_Load event**, add the following code to export Section Reports .

Visual Basic.NETcode. Paste INSIDE the Form_Load event

```
' Create a Section report.

Dim rpt As New GrapeCity.ActiveReports.SectionReport()
' For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath + "\report.rpx")
rpt.LoadLayout(xtr)
rpt.Run()

' Export the report in HTML format.
Dim HtmlExport1 As New GrapeCity.ActiveReports.Export.Html.Section.HtmlExport()
HtmlExport1.Export(rpt.Document, Application.StartupPath + "\HTMLExpt.html")
```

C# code. Paste INSIDE the Form_Load event.

```
// Create a Section Report
GrapeCity.ActiveReports.SectionReport rpt = new GrapeCity.ActiveReports.SectionReport();

// For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
"\\report.rpx");
rpt.LoadLayout(xtr);
rpt.Run();

// Export the report in HTML format.
GrapeCity.ActiveReports.Export.Html.Section.HtmlExport HtmlExport1 = new
GrapeCity.ActiveReports.Export.Html.Section.HtmlExport();
```

```
HtmlExport1.Export(rpt.Document, Application.StartupPath + "\\HTMLExpt.html");
```

PDF Export

PDF, or portable document format, opens in the Adobe 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 **PDFExport ('PdfExport Class' in the on-line documentation)** object after adding reference to **MESCIUS.ActiveReports.Export.Pdf** package in your project

Note: PDF export and font linking features are only available in the Professional Edition of ActiveReports.

PDF Export Properties

Property	Valid Values	Description
ConvertMetaToPng ('ConvertMetaToPng Property' in the on-line documentation)	True or False (default)	Set to True to change any Windows metafile images to PNG format to keep the file size down. If the report has no metafiles, this setting is ignored.
ExportBookmarks ('ExportBookmarks Property' in the on-line documentation)	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.
FontFallback ('FontFallback Property' in the on-line documentation)	String of font families	Set a comma-delimited string of font families to be used to lookup glyphs missing in the original font.
ImageQuality ('ImageQuality Property' in the on-line documentation)	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). Set to Lowest to keep the file size down. If the report has no metafiles, this setting is ignored.
ImageResolution ('ImageResolution Property' in the on-line documentation)	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 ('NeverEmbedFonts Property' in the on-line documentation)	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 ('Options Property' in the on-line documentation)	See below	Expand this property to see a group of sub properties. These settings control how the Adobe Reader displays the output PDF file when it is first opened. See the table below for details.
Security ('Security	See below	Expand this property to see a group of sub properties. These

Property' in the on-line documentation)		settings control encryption and permissions on the output PDF file. See the table below for details.
Signature ('Signature Property' in the on-line documentation)	A valid PdfSignature object	This must be set up in code. For more information, see Digital Signature Pro.
Version ('Version Property' in the on-line documentation)	Pdf12, Pdf13, Pdf14, Pdf15, Pdf16, Pdf17, PdfA1a, PdfA1b, PdfA2a, PdfA2b, PdfA2u, PdfA3a, PdfA3b, PdfA3u, and PdfUA1	Sets the version of the PDF format the exported document is saved in.
Watermark ('Watermark Property' in the on-line documentation)	See below. **For CrossPlatform compatibility mode only.**	Expand this property to see a group of sub properties. These settings control whether a report is exported to PDF with a watermark on it. See the table below for details.

PDF Export usage and limitations

Usage:

- Create printable reports whose formats do not change from machine to machine.
- Open in Adobe Reader.
- WYSWYG in both compatibility modes - a legacy GDI and a new CrossPlatform mode (section report).

Does not support:

- Multiple lines of vertical text is not supported in Page and RDLX reports (GDI compatibility mode).
- Transparent background-color in charts is not supported in the GDI compatibility mode if **ConvertMetaToPng** is set to True.

Options, Security, and Watermark

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

PDF Options Properties

Property	Valid Values	Description
Application ('Application Property' in the on-line documentation)	String	Set to the string value that you want to display in the Adobe Document Properties dialog, Description tab, Application field.
Author ('Author Property' in the on-line	String	Set to the string value that you want to display in the Adobe Document Properties dialog, Description tab, Author field.

documentation)		
CenterWindow ('CenterWindow Property' in the on-line documentation)	True or False (default)	Set to True to position the Adobe Reader window in the center of the screen when the document is first opened.
DisplayMode ('DisplayMode Property' in the on-line documentation)	None (default), Outlines, Thumbs, or FullScreen	Select how to display bookmarks when the document is first opened. • 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 ('DisplayTitle Property' in the on-line documentation)	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 ('FitWindow Property' in the on-line documentation)	True or False (default)	Set to True to expand the window to fit the size of the first displayed page.
HideMenubar ('HideMenubar Property' in the on-line documentation)	True or False (default)	Set to True to hide the menu in the Adobe Reader when the document is first opened.
HideToolbar ('HideToolbar Property' in the on-line documentation)	True or False (default)	Set to True to hide the toolbars in the Adobe Reader when the document is first opened.
HideWindowUi	True or False (default)	Set to True to hide the scrollbars and navigation controls in the Adobe Reader when the document is first opened, displaying only the document.
Keywords ('Keywords Property' in the on-line documentation)	String	Enter keywords to display in the Adobe Document Properties dialog, Description tab, Keywords field.
OnlyForPrint	True or False (default)	Set to indicate whether the PDF is only for print.

('OnlyForPrint Property' in the on-line documentation)		
Subject ('Subject Property' in the on-line documentation)	String	Enter a subject to display in the Adobe Document Properties dialog, Description tab, Subject field.
Title ('Title Property' in the on-line documentation)	String	Enter a title to display in the Adobe Document Properties dialog, Description tab, Title field. Set DisplayTitle to True to display this text in the title bar of the Adobe Reader when the document is opened.

PDF Security Properties

Property	Valid Values	Description
Encrypt ('Encrypt Property' in the on-line documentation)	True or False (default)	Sets or returns a value indicating whether the document is encrypted.
OwnerPassword ('OwnerPassword Property' in the on-line documentation)	String	Enter the string to use as a password that unlocks the document regardless of specified permissions.
Permissions ('Permissions Property' in the on-line documentation)	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.
UserPassword ('UserPassword Property' in the on-line documentation)	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.

PDF Watermark Properties (CrossPlatform Compatibility Mode only)

Property	Valid Values	Description
Angle ('Angle Property' in the on-line documentation)	String	Specify the degree of angle for the watermark text on the PDF document. Valid values range from 0 to 359, where 0 is horizontal, left to right.
Color ('Color Property' in the on-line documentation)	String	Select a color for the watermark text of the PDF document.
FontName ('FontName	String	Specify the font of the PDF document watermark text.

Property' in the on-line documentation)		
FontSize ('FontSize Property' in the on-line documentation)	String	Specify the font size of the PDF document watermark text.
FontStyle ('FontStyle Property' in the on-line documentation)	Regular, Bold, Italic, Underline, or Strikeout.	Select the font style of the PDF document watermark text.
PrintOnly ('PrintOnly Property' in the on-line documentation)	True or False (default)	Specify whether the watermark should appear on printed pages only.
Text ('Text Property' in the on-line documentation)	String	Enter text to be used as the watermark on the PDF document.

PDF Print Presets Properties

ActiveReports allows you to preset the printing properties for PDF report exports using the **PrintPresets ('PrintPresets Class' in the on-line documentation)** class. This pre-populates the print settings in the Print dialog box. Please see Print Presets for more information.

> **Note:** The print preset properties are only available with the Professional Edition license. An evaluation message is displayed when used with the Standard Edition license.

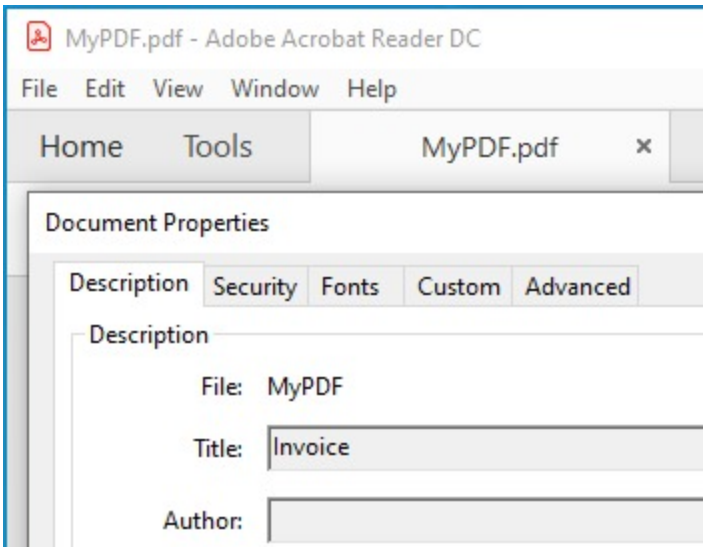
Property	Description
PageScaling ('PageScaling Property' in the on-line documentation)	Specify scaling for the printable area.You can select Default to shrink to the printable area, or you can select None for the actual size.
DuplexMode ('DuplexMode Property' in the on-line documentation)	Specify the duplex mode of the printer. For the best results with the duplex option, the selected printer should support duplex printing. You can choose from the following values, • Simplex: Prints on one side of the paper. This is the default value. • Duplex (Flip on long edge): Prints on both sides of the paper with paper flip on the long edge. • Duplex (Flip on short edge): Prints on both sides of the paper with paper flip on the short edge.
PaperSourceByPageSize ('PaperSourceByPageSize Property' in the on-line documentation)	Determines the output tray based on PDF page size, rather than page setting options. This option is useful when printing PDFs with multiple page sizes, where different sized output trays are available. By default, this option is set to False.

PrintPageRange ('PrintPageRange Property' in the on-line documentation)	Specify the range of page numbers as 1-3 or 1, 2, 3.
NumberOfCopies ('NumberOfCopies Property' in the on-line documentation)	Specify the number of copies to print. You can select any number of copies from 2 to 5, or select Default to specify a single copy.

Note: These properties are available in PDF version 1.7 or higher. The **PageScaling** property is supported in PDF version 1.6.

Metadata in PDFs

Adding Metadata



Metadata such as keywords, descriptions are used by the search engines to narrow down the searches. You can add a number of predefined accessors, such as title, contributors, creators, copyright, description, etc.
using **AdditionalMetadata ('AdditionalMetadata Property' in the on-line documentation)** property. The allowed namespaces are:

- Dublin Core Properties
- XMP Core Properties
- PDF Properties

VB code. Paste INSIDE the Form Load event.

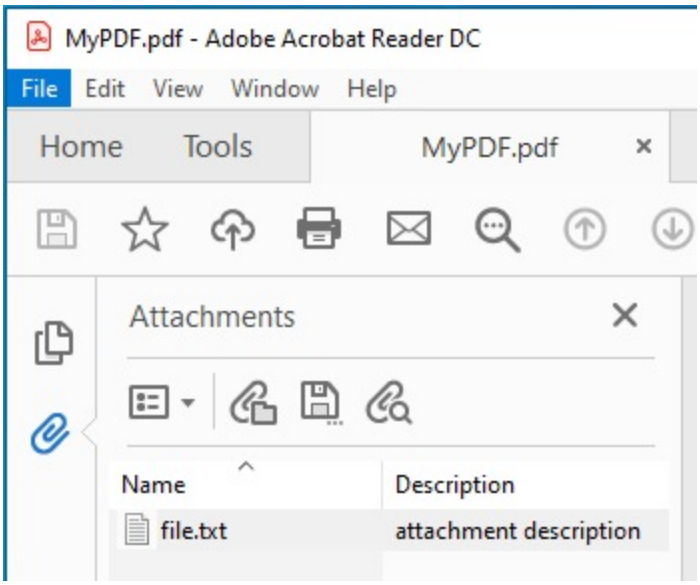
```
Dim sectionReport As GrapeCity.ActiveReports.SectionReport = New
GrapeCity.ActiveReports.SectionReport()
Dim xtr As XmlReader = XmlReader.Create(Application.StartupPath & "\SectionReport1.rpx")
```

```
sectionReport.LoadLayout(xtr)
sectionReport.Run()
Dim pdfExport As GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport = New
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport()
Dim metadata1 = New AdditionalMetadataInfo With {
    .[Namespace] = AdditionalMetadataNamespace.PurlOrg, ' Dublin Core Properties
    .Key = "title",
    .Value = "Invoice"
}
pdfExport.Options.AdditionalMetadata.Add(metadata1)
pdfExport.Export(sectionReport.Document, Application.StartupPath & "\MyPDF.pdf")
```

C# code. Paste INSIDE the Form Load event.

```
GrapeCity.ActiveReports.SectionReport sectionReport = new
GrapeCity.ActiveReports.SectionReport();
XmlReader xtr = XmlReader.Create(Application.StartupPath + "\\SectionReport1.rpx");
sectionReport.LoadLayout(xtr);
sectionReport.Run();
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport pdfExport = new
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport();
// Add meta data
var metadata1 = new AdditionalMetadataInfo
{
    Namespace = AdditionalMetadataNamespace.PurlOrg, //Dublin Core Properties
    Key = "title",
    Value = "Invoice"
 };

pdfExport.Options.AdditionalMetadata.Add(metadata1);
pdfExport.Export(sectionReport.Document, Application.StartupPath + "\\MyPDF.pdf");
```

Adding Attachment



You can include an attachment as metadata (such as invoices) to exported PDFs using **Attachments ('Attachments Property' in the on-line documentation)** property. This property allows to attach files such as a .xml or a .txt file in PDF. Below is example to export Section reports to PDF and attach a file to the exported PDF.

VB code. Paste INSIDE the Form Load event.

```
Dim sectionReport As GrapeCity.ActiveReports.SectionReport = New
GrapeCity.ActiveReports.SectionReport()
Dim xtr As XmlReader = XmlReader.Create(Application.StartupPath + _, SectionReport1.rpx)
sectionReport.LoadLayout(xtr)
sectionReport.Run()
Dim pdfExport As GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport = New
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport()
Dim attachment = New GrapeCity.ActiveReports.Export.Pdf.AttachmentInfo With {
    .Name = "file.txt",
    .Content = System.IO.File.ReadAllBytes("D:\Reports\file.txt"),
    .Description = "attachment description"
}
pdfExport.Options.Attachments.Add(attachment)
pdfExport.Export(sectionReport.Document, Application.StartupPath & "\MyPDF.pdf")
```

C# code. Paste INSIDE the Form Load event.

```
GrapeCity.ActiveReports.SectionReport sectionReport = new
GrapeCity.ActiveReports.SectionReport();
XmlReader xtr = XmlReader.Create(Application.StartupPath + \\SectionReport1.rpx);
sectionReport.LoadLayout(xtr);
sectionReport.Run();
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport pdfExport = new
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport();
// Add attachment
var attachment = new GrapeCity.ActiveReports.Export.Pdf.AttachmentInfo
    {
```

```
            Name = "file.txt",
            Content = System.IO.File.ReadAllBytes(@"D:\Reports\file.txt"),
            Description = "attachment description" //optional
        };
pdfExport.Options.Attachments.Add(attachment);
pdfExport.Export(sectionReport.Document, Application.StartupPath + "\\MyPDF.pdf");
```

Open the exported PDF and you should see the attachment. Check the left sidebar in Adobe Acrobat Reader DC.

> **Note:** Metadata in PDFs is part of the Professional Edition. It is supported with the PDF version PDF/A-3b (or higher).

PDF/A Support Limitations

- The **NeverEmbedFonts** property is ignored, so all fonts of a report are embedded into the PDF document.
- The **Security.Encrypt** property is ignored and the PDF export behaves as if this property is always set to **False**.
- The **OnlyForPrint** property is ignored and the PDF export behaves as if this property is always set to **False**.
- **Transparent images** lose their transparency when exported to PDF/A-1.
- **External hyperlinks** are exported as plain text.

Export Report using PDF Export Filter

Use the following steps to export reports through PDF export filters.

1. Create a new or open an existing Visual Studio project.
    - If you are creating a new project,
      - select **ActiveReports 18 Section Report (xml-based) Application** in **Create a New Project** dialog, and then
      - specify a name for the project, and click **OK**.
    - If you are using an existing project,
      - go to the Solution Explorer and right-click the project and select **Add** > **New Item**, and then
      - select **ActiveReports 18 Section Report (xml-based)** and click **Add**.
2. Add a reference to **MESCIUS.ActiveReports.Export.Pdf** package in the project. See Manage ActiveReports Dependencies for more information.
3. In your project's **Bin>Debug** folder, place the **report.rpx** (Section Report).
4. On the Form.cs or Form.vb, double-click the title bar to create the Form_Load event.
5. In **Form_Load event**, add the following code to export Section Reports .

Visual Basic.NETcode. Paste INSIDE the Form_Load event

```
' Create a Section report.

Dim rpt As New GrapeCity.ActiveReports.SectionReport()
' For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath + "\report.rpx")
rpt.LoadLayout(xtr)
rpt.Run()

' Export the report in HTML format.
```

```
Dim PdfExport1 As New GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport()
PdfExport1.Export(rpt.Document, Application.StartupPath + "\PDFExpt.pdf")
```

C# code. Paste INSIDE the Form_Load event.

```
// Create a Section Report
GrapeCity.ActiveReports.SectionReport rpt = new GrapeCity.ActiveReports.SectionReport();

// For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
"\\report.rpx");
rpt.LoadLayout(xtr);
rpt.Run();

// Export the report in PDF format.
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport PdfExport1 = new
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport();
PdfExport1.Export(rpt.Document, Application.StartupPath + "\\PDFExpt.pdf");
```

Text Export

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 ('Export Method' in the on-line documentation)** method. Use the extension **.txt** to open files in Notepad, or use **.csv** to open comma separated value files 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 **TextExport ('TextExport Class' in the on-line documentation)** object after adding reference to **MESCIUS.ActiveReports.Export.Xml** package in your project.

Text Export Properties

Property	Valid Values	Description
Encoding ('Encoding Property' in the on-line documentation)	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 ('PageDelimiter Property' in the on-line documentation)	String	Enter a character or sequence of characters to mark the end of each page.
QuotationMode ('QuotationMode Property' in the on-line documentation)		Specifies whether to add double quotes to the exported data. • **AutoQuote** – Simple values are exported without quotes. The quotes are added only when the data contains column or

		row delimiters. This is the default export behavior. • **AlwaysQuote** – Exported values are always quoted.
QuotationSymbol ('QuotationSymbol Property' in the on-line documentation)	Char	Enter a character to use as quotation mark in the exported text file. Only fields with delimiter characters are quoted.
SuppressEmptyLines ('SuppressEmptyLines Property' in the on-line documentation)	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 ('TextDelimiter Property' in the on-line documentation)	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 Export usage and limitations

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

Export Report using Text Export Filter

Use the following steps to export reports through Text export filters.

1. Create a new or open an existing Visual Studio project.
    - If you are creating a new project,
      - select **ActiveReports 18 Section Report (xml-based) Application** in **Create a New Project** dialog, and then
      - specify a name for the project, and click **OK**.
    - If you are using an existing project,
      - go to the Solution Explorer and right-click the project and select **Add** > **New Item**, and then
      - select **ActiveReports 18 Section Report (xml-based)** and click **Add**.
2. Add a reference to **MESCIUS.ActiveReports.Export.Xml** package in the project. See Manage ActiveReports Dependencies for more information.
3. In your project's **Bin > Debug** folder, place the **report.rpx** (Section Report).

4. On the Form.cs or Form.vb, double-click the title bar to create the Form_Load event.
5. In **Form_Load event**, add the following code to export Section Reports .

Visual Basic.NETcode. Paste INSIDE the Form_Load event

```
' Create a Section report.
Dim rpt As New GrapeCity.ActiveReports.SectionReport()

' For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath + "\report.rpx")
rpt.LoadLayout(xtr)
rpt.Run()

' Export the report in Text format.
Dim TextExport1 As New GrapeCity.ActiveReports.Export.Xml.Section.TextExport()
TextExport1.Export(rpt.Document, Application.StartupPath + "\TextExpt.txt")
```

C# code. Paste INSIDE the Form_Load event.

```
// Create a Section Report
GrapeCity.ActiveReports.SectionReport rpt = new GrapeCity.ActiveReports.SectionReport();

// For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
"\\report.rpx");
rpt.LoadLayout(xtr);
rpt.Run();

// Export the report in Text format.
GrapeCity.ActiveReports.Export.Xml.Section.TextExport TextExport1 = new
GrapeCity.ActiveReports.Export.Xml.Section.TextExport();
TextExport1.Export(rpt.Document, Application.StartupPath + "\\TextExpt.txt");
```

RTF Export

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. You can set the property either in code using the **RTFExport ('RtfExport Class' in the on-line documentation)** object after adding reference to **MESCIUS.ActiveReports.Export.Word** package in your project.

The RTF export now supports the **CrossPlatform** compatibility mode.

RTF Export usage and limitations

Usage:

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

Does not support:

- Section or Page back colors
- Angled text
- This export is not WYSIWYG and thus does not support many features

Export Report using RTF Export Filter

Use the following steps to export reports through RTF export filters.

1. Create a new or open an existing Visual Studio project.
    - If you are creating a new project,
      - select **ActiveReports 18 Section Report (xml-based) Application** in **Create a New Project** dialog, and then
      - specify a name for the project, and click **OK**.
    - If you are using an existing project,
      - go to the Solution Explorer and right-click the project and select **Add** > **New Item**, and then
      - select **ActiveReports 18 Section Report (xml-based)** and click **Add**.
2. Add a reference to **MESCIUS.ActiveReports.Export.Word** package in the project. See Manage ActiveReports Dependencies for more information.
3. In your project's **Bin>Debug** folder, place the **report.rpx** (Section Report).
4. On the Form.cs or Form.vb, double-click the title bar to create the Form_Load event.
5. In **Form_Load event**, add the following code to export Section Reports .

Visual Basic.NETcode. Paste INSIDE the Form_Load event

```
' Create a Section report.
Dim rpt As New GrapeCity.ActiveReports.SectionReport()

' For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath + "\report.rpx")
rpt.LoadLayout(xtr)
rpt.Run()

' Export the report in RTF format.
Dim RtfExport1 As New GrapeCity.ActiveReports.Export.Word.Section.RtfExport()
RtfExport1.Export(rpt.Document, Application.StartupPath + "\RTFExpt.rtf")
```

C# code. Paste INSIDE the Form_Load event.

```
// Create a Section Report
GrapeCity.ActiveReports.SectionReport rpt = new GrapeCity.ActiveReports.SectionReport();

// For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
"\\report.rpx");
rpt.LoadLayout(xtr);
rpt.Run();
```

```
// Export the report in RTF format.
GrapeCity.ActiveReports.Export.Word.Section.RtfExport RtfExport1 = new
GrapeCity.ActiveReports.Export.Word.Section.RtfExport();
RtfExport1.Export(rpt.Document, Application.StartupPath + "\\RTFExpt.rtf");
```

Excel Export

XLSX 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 XLSX 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 **XLSExport ('XlsExport Class' in the on-line documentation)** object after adding reference to **MESCIUS.ActiveReports.Export.Excel** package in your project.

The Excel export now supports the **CrossPlatform** compatibility mode.

Excel Export Properties

Property	Valid Values	Description
AutoRowHeight ('AutoRowHeight Property' in the on-line documentation)	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 ('DisplayGridLines Property' in the on-line documentation)	True (default) or False	Set to False to hide grid lines in Excel.
FileFormat ('FileFormat Property' in the on-line documentation)	Xls97Plus (default) or Xls95 or Xlsx	Set to Xls95 to use Microsoft Excel 95, Xls95Plus to use Microsoft Excel 97and Xlsx to use Microsoft Excel 2007 or newer.
MinColumnWidth ('MinColumnWidth Property' in the on-line documentation)	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 ('MinRowHeight Property' in the on-line documentation)	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 ('MultiSheet Property' in the on-line	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

documentation)		between page layouts. In general, use False for reports with more than 30 pages.
PageSettings ('PageSettings Property' in the on-line documentation)		Set a print orientation and paper size of Excel sheet.
RemoveVerticalSpace ('RemoveVerticalSpace Property' in the on-line documentation)	True or False (default)	Set to True to remove vertical empty spaces from the spreadsheet. This may improve pagination for printing.
Security ('Security Property' in the on-line documentation)		Set a password and username to protect the excel spreadsheet.
UseCellMerging ('UseCellMerging Property' in the on-line documentation)	True or False (default)	Set to True to merge cells where applicable.
UseDefaultPalette ('UseDefaultPalette Property' in the on-line documentation)	True or False (default)	Set to True to export document with Excel default palette.

Excel Export usage and limitations

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)

Export Report using Excel Export Filter

Use the following steps to export reports through Excel export filters.

1. Create a new or open an existing Visual Studio project.
    - If you are creating a new project,
    - select **ActiveReports 18 Section Report (xml-based) Application** in **Create a New Project** dialog, and then

- specify a name for the project, and click **OK**.
  - If you are using an existing project,
    - go to the Solution Explorer and right-click the project and select **Add** > **New Item**, and then
    - select **ActiveReports 18 Section Report (xml-based)** and click **Add**.
2. Add a reference to **MESCIUS.ActiveReports.Export.Excel** package in the project. See Manage ActiveReports Dependencies for more information.
3. In your project's **Bin>Debug** folder, place the **report.rpx** (Section Report).
4. On the Form.cs or Form.vb, double-click the title bar to create the Form_Load event.
5. In **Form_Load event**, add the following code to export Section Reports .

Visual Basic.NETcode. Paste INSIDE the Form_Load event

```
' Create a Section report.
Dim rpt As New GrapeCity.ActiveReports.SectionReport()

' For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath + "\report.rpx")
rpt.LoadLayout(xtr)
rpt.Run()

' Export the report in Excel format.
Dim XlsExport1 As New GrapeCity.ActiveReports.Export.Excel.Section.XlsExport()

' Set a file format of the exported excel file to Xlsx to support Microsoft Excel 2007
and newer versions.
XlsExport1.FileFormat = GrapeCity.ActiveReports.Export.Excel.Section.FileFormat.Xlsx
XlsExport1.Export(rpt.Document, Application.StartupPath + "\XLSExpt.xlsx")
```

C# code. Paste INSIDE the Form_Load event.

```
// Create a Section Report
GrapeCity.ActiveReports.SectionReport rpt = new GrapeCity.ActiveReports.SectionReport();

// For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
"\\report.rpx");
rpt.LoadLayout(xtr);
rpt.Run();

// Export the report in XLSX format.
GrapeCity.ActiveReports.Export.Excel.Section.XlsExport XlsExport1 = new
GrapeCity.ActiveReports.Export.Excel.Section.XlsExport();

// Set a file format of the exported excel file to Xlsx to support Microsoft Excel 2007
and newer versions.
XlsExport1.FileFormat = GrapeCity.ActiveReports.Export.Excel.Section.FileFormat.Xlsx;
XlsExport1.Export(rpt.Document, Application.StartupPath + "\\XLSExpt.xlsx");
```

TIFF Export

TIFF, or tagged 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 **TIFFExport ('TiffExport Class' in the on-line documentation)** object after adding reference to **MESCIUS.ActiveReports.Export.Image** package in your projet.

TIFF Export Properties

Property	Valid Values	Description
CompressionScheme ('CompressionScheme Property' in the on-line documentation)	None, Rle, Ccitt3 (default), Ccitt4 or Lzw	Select an enumerated value to use for color output control: • 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 ('Dither Property' in the on-line documentation)	True or False (default)	Set to True to dither the image when you save it to a black and white format (Ccitt3, Ccitt4 or Rle). This property has no effect if the CompressionScheme is set to Lzw or None.
DpiX ('DpiX Property' in the on-line documentation)	Integer (VB) or int (C#) greater than 0	Set the horizontal resolution of a report when exporting to TIFF format. The default value is 200. 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.
DpiY ('DpiY Property' in the on-line documentation)	Integer (VB) or int (C#) greater than 0	Set the vertical resolution of a report when exporting to TIFF format. The default value is 196. 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.

TIFF Export usage:

- Create optical archive reports
- Send reports via fax machines
- Open in image viewers
- Generates an image of each page. 100% WYSIWYG. For section report, WYSWYG in both compatibility modes - a legacy GDI and a new CrossPlatform mode.

Export Report using TIFF Export Filter

Use the following steps to export reports through TIFF export filters.

1. Create a new or open an existing Visual Studio project.
    - If you are creating a new project,
      - select **ActiveReports 18 Section Report (xml-based) Application** in **Create a New Project** dialog, and then
      - specify a name for the project, and click **OK**.
    - If you are using an existing project,
      - go to the Solution Explorer and right-click the project and select **Add** > **New Item**, and then
      - select **ActiveReports 18 Section Report (xml-based)** and click **Add**.
2. Add a reference to **MESCIUS.ActiveReports.Export.Image** package in the project. See Manage ActiveReports Dependencies for more information.
3. In your project's **Bin>Debug** folder, place the **report.rpx** (Section Report).
4. On the Form.cs or Form.vb, double-click the title bar to create the Form_Load event.
5. In **Form_Load event**, add the following code to export Section Reports .

Visual Basic.NETcode. Paste INSIDE the Form_Load event

```
' Create a Section report.
Dim rpt As New GrapeCity.ActiveReports.SectionReport()

' For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath + "\report.rpx")
rpt.LoadLayout(xtr)
rpt.Run()

' Export the report in TIFF format.
Dim TiffExport1 As New GrapeCity.ActiveReports.Export.Image.Tiff.Section.TiffExport()
TiffExport1.Export(rpt.Document, Application.StartupPath + "\TIFFExpt.tiff")
```

C# code. Paste INSIDE the Form_Load event.

```
// Create a Section Report
GrapeCity.ActiveReports.SectionReport rpt = new GrapeCity.ActiveReports.SectionReport();

// For the code to work, report.rpx must be placed in the bin\debug folder of your
project.
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
"\\report.rpx");
rpt.LoadLayout(xtr);
rpt.Run();
```

```
// Export the report in TIFF format.
GrapeCity.ActiveReports.Export.Image.Tiff.Section.TiffExport TiffExport1 = new
GrapeCity.ActiveReports.Export.Image.Tiff.Section.TiffExport();
TiffExport1.Export(rpt.Document, Application.StartupPath + "\\TIFFExpt.tiff");
```

Set PDF Print Presets

The page demonstrates presetting basic print options when exporting a Section Report or a Page/RDLX report in PDF format.

> **Note**: The print preset properties are only available with the Professional Edition license. An evaluation message is displayed when used with the Standard Edition license.

1. From the Visual Studio **File** menu, select **New**, then **Project**.
2. In the **Create New Project** dialog that appears, select any of the **ActiveReports 18** templates and click **Next**.
3. Type a name for your project and click **Create.**
4. Select the type of report that you want to add and click **Finish**:
    - RDLX
    - RDLX Dashboard
    - Page
    - Section
5. In the Design view, double-click the Form title bar to create the Form_Load event.
6. Add the following code to invoke the Export methods and set print presets in the Form_Load event.

Section Report

Visual Basic.NET code. Paste INSIDE the Form_Load event

```
Dim sectionReport As New GrapeCity.ActiveReports.SectionReport()
Dim xtr As New System.Xml.XmlTextReader(Application.StartupPath +
"\..\..\SectionReport1.rpx")
sectionReport.LoadLayout(xtr)
sectionReport.Run()

'Define settings for PDF
Dim p As New GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport()
p.Version = GrapeCity.ActiveReports.Export.Pdf.Section.PdfVersion.Pdf17

'Set default print settings using PrintPresets class
p.PrintPresets.PageScaling = GrapeCity.ActiveReports.Export.Pdf.Enums.PageScaling.None
p.PrintPresets.DuplexMode =
GrapeCity.ActiveReports.Export.Pdf.Enums.DuplexMode.DuplexFlipLongEdge
p.PrintPresets.NumberOfCopies =
GrapeCity.ActiveReports.Export.Pdf.Enums.NumberOfCopies.Two
p.PrintPresets.PaperSourceByPageSize = True
p.PrintPresets.PrintPageRange = "1-3"
p.Export(sectionReport.Document, Application.StartupPath + "\PrintPresets.pdf")
```

C# code. Paste INSIDE the Form_Load event

```
GrapeCity.ActiveReports.SectionReport sectionReport = new
GrapeCity.ActiveReports.SectionReport();
System.Xml.XmlTextReader xtr = new System.Xml.XmlTextReader(Application.StartupPath +
@"\..\..\SectionReport1.rpx");
sectionReport.LoadLayout(xtr);
sectionReport.Run();

//Define settings for PDF
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport p = new
GrapeCity.ActiveReports.Export.Pdf.Section.PdfExport();
p.Version = GrapeCity.ActiveReports.Export.Pdf.Section.PdfVersion.Pdf17;

//Set default print settings using PrintPresets class
p.PrintPresets.PageScaling = GrapeCity.ActiveReports.Export.Pdf.Enums.PageScaling.None;
p.PrintPresets.DuplexMode =
GrapeCity.ActiveReports.Export.Pdf.Enums.DuplexMode.DuplexFlipLongEdge;
p.PrintPresets.NumberOfCopies =
GrapeCity.ActiveReports.Export.Pdf.Enums.NumberOfCopies.Two;
p.PrintPresets.PaperSourceByPageSize = true;
p.PrintPresets.PrintPageRange = "1-3";
p.Export(sectionReport.Document, Application.StartupPath + "\\PrintPresets.pdf");
```

Page/RDLX Report

Visual Basic.NET code. Paste INSIDE the Form_Load event

```
'Set the rendering extension and render the report.
Dim pdfExport = New GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension()

'Define settings for PDF
Dim pdfSettings As New GrapeCity.ActiveReports.Export.Pdf.Page.Settings()
pdfSettings.Version = GrapeCity.ActiveReports.Export.Pdf.Page.PdfVersion.Pdf17
pdfSettings.PrintOnOpen = True

'Set default print settings using PrintPresets class
Dim pdfPresetsSetting As New GrapeCity.ActiveReports.Export.Pdf.PrintPresets()
pdfPresetsSetting.PageScaling =
GrapeCity.ActiveReports.Export.Pdf.Enums.PageScaling.None
pdfPresetsSetting.DuplexMode =
GrapeCity.ActiveReports.Export.Pdf.Enums.DuplexMode.DuplexFlipLongEdge
pdfPresetsSetting.NumberOfCopies =
GrapeCity.ActiveReports.Export.Pdf.Enums.NumberOfCopies.Two
pdfPresetsSetting.PaperSourceByPageSize = True
pdfPresetsSetting.PrintPageRange = "1-3"
```

```
pdfSettings.PrintPresets = pdfPresetsSetting

Dim outputFile = New IO.FileInfo("..\..\PrintPresets.pdf")
Dim reportFile = New IO.FileInfo("..\..\PageReport1.rdlx")

Dim fileStreamProvider = New
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputFile.Directory,
Path.GetFileNameWithoutExtension(outputFile.FullName))

Using pageDocument = New GrapeCity.ActiveReports.PageReport(reportFile).Document
    pageDocument.Render(pdfExport, fileStreamProvider, pdfSettings)
End Using
```

C# code. Paste INSIDE the Form_Load event

```
//Set the rendering extension and render the report.
var pdfExport = new GrapeCity.ActiveReports.Export.Pdf.Page.PdfRenderingExtension();

//Define settings for PDF
GrapeCity.ActiveReports.Export.Pdf.Page.Settings pdfSettings = new
GrapeCity.ActiveReports.Export.Pdf.Page.Settings();
pdfSettings.Version = GrapeCity.ActiveReports.Export.Pdf.Page.PdfVersion.Pdf17;
pdfSettings.PrintOnOpen = true;

//Set default print settings using PrintPresets class
GrapeCity.ActiveReports.Export.Pdf.PrintPresets pdfPresetsSetting = new
GrapeCity.ActiveReports.Export.Pdf.PrintPresets();
pdfPresetsSetting.PageScaling =
GrapeCity.ActiveReports.Export.Pdf.Enums.PageScaling.None;
pdfPresetsSetting.DuplexMode =
GrapeCity.ActiveReports.Export.Pdf.Enums.DuplexMode.DuplexFlipLongEdge;
pdfPresetsSetting.NumberOfCopies =
GrapeCity.ActiveReports.Export.Pdf.Enums.NumberOfCopies.Two;
pdfPresetsSetting.PaperSourceByPageSize = true;
pdfPresetsSetting.PrintPageRange = "1-3";

pdfSettings.PrintPresets = pdfPresetsSetting;

var outputFile = new System.IO.FileInfo(@"..\..\PrintPresets.pdf");
var reportFile = new System.IO.FileInfo(@"..\..\PageReport1.rdlx");

var fileStreamProvider = new
GrapeCity.ActiveReports.Rendering.IO.FileStreamProvider(outputFile.Directory,
System.IO.Path.GetFileNameWithoutExtension(outputFile.FullName));

using (var pageDocument = new GrapeCity.ActiveReports.PageReport(reportFile).Document)
{
    pageDocument.Render(pdfExport, fileStreamProvider, pdfSettings);
}
```

Managing Asynchronous or Long-Running Report Rendering

You can manage asynchronous or long-running report rendering. ActiveReports provides a number of possibilities to control the export process, which can help you create more responsive applications. In this topic, we discuss common use cases of the report rendering:

- Display export progress in a number of pages (ActiveReports can display a total number of pages only once the rendering is complete).
- Create an Export dialog with the Cancel option.
- Log information on the report execution.

Display Progress Information

Let's see how you can have displayed the export rendering progress information, using a WinForms Viewer application. As an example, we will take the **Export** sample that you can access by following this link - https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/Export.

The rendering progress information can be displayed for PDF, Image, HTML, and Excel exports at the report rendering.

> **Note**: The progress information is not shown if the report is exported with the **Pagination** property set to False (Image, Excel Rendering Extensions) or with the **Mode** property set to Galley (HTML Rendering Extension).

1. Open the **Export** sample project.
2. From the Visual Studio toolbox, drag the **Label** control onto the Form.
3. Add this code into the Form.

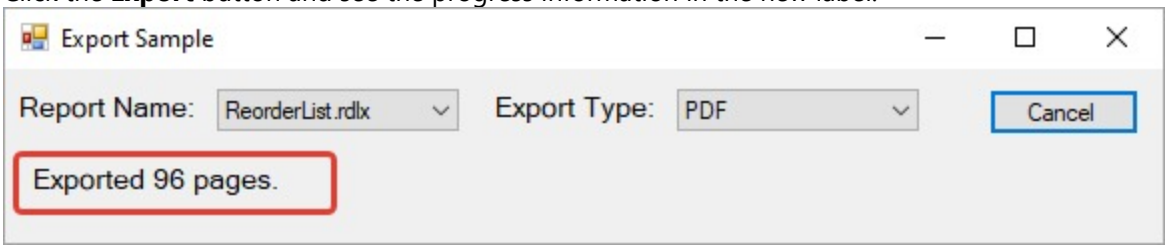
Visual Basic.NET code

```
Private Async Sub ExportAsync(ByVal report As PageReport, ByVal renderingExtension As
IRenderingExtension, ByVal outputProvider As StreamProvider, ByVal settings As
NameValueCollection)
   labelExport.Text = "Export started..."
   Dim control = Me
   Dim progress = New Progress(Of ProgressInfo)(Sub(progressInfo)
      control.BeginInvoke(New MethodInvoker(Sub()
         labelExport.Text = If(progressInfo.IsLast = True, "Export is finished." ,
$"Exported {progressInfo.PageNumber} pages.")
      End Sub))
   End Sub)

   await Task.Run(Sub()
      Try
         report.Document.Render(renderingExtension, outputProvider, settings, False,
False, CancellationToken.None, progress)
      Catch e as Exception
      End Try
   End Sub)
End Sub
```

C# code

```
private async void ExportAsync(PageReport report, IRenderingExtension
renderingExtension, StreamProvider outputProvider, NameValueCollection settings)
{
   labelExport.Text = "Export started...";
   var control = this;
   var progress = new Progress(progressInfo =>
   {
      control.BeginInvoke(new MethodInvoker(() =>
      {
         labelExport.Text = progressInfo.IsLast ? "Export is finished." : $"Exported
{progressInfo.PageNumber} pages.";

      }));
   });
   await Task.Run(() =>
   {
      try
      {
         report.Document.Render(renderingExtension, outputProvider, settings, false,
false, CancellationToken.None, progress);
      }
      catch
      {
         // ignored
      }
   });
}
```

4. Run the project.
5. In the **Export** form, select PDF, Image, HTML, or Excel export type.
6. Click the **Export** button and see the progress information in the new label.



Cancel Report Rendering

You can cancel the report rendering or export by using the following code.

Visual Basic.NET code

```
Dim _cancellationTokenSource
```

```
Private Async Sub ExportAsync(ByVal report As PageReport, ByVal renderingExtension As
IRenderingExtension, ByVal outputProvider As FileStreamProvider, ByVal settings As
NameValueCollection)
   _cancellationTokenSource = New CancellationTokenSource()
   exportButton.Text = "Cancel"
   RemoveHandler exportButton.Click, AddressOf exportButton_Click
   AddHandler exportButton.Click, AddressOf CancelExport
   Await Task.Run(Sub()
      Try
         report.Document.Render(renderingExtension, outputProvider, settings, False, False,
_cancellationTokenSource.Token)
      Catch unusedOperationCanceledException As OperationCanceledException
      End Try
   End Sub)
   If _cancellationTokenSource.IsCancellationRequested Then MessageBox.Show("Export was
cancelled", "Export", MessageBoxButtons.OK, MessageBoxIcon.Information)
   _cancellationTokenSource.Dispose()
   _cancellationTokenSource = Nothing
   RemoveHandler exportButton.Click, AddressOf CancelExport
   AddHandler exportButton.Click, AddressOf exportButton_Click
   exportButton.Text = "Export"
End Sub

Private Sub CancelExport(ByVal sender As Object, ByVal e As EventArgs)
   _cancellationTokenSource?.Cancel()
End Sub
```

C# code

```
private CancellationTokenSource _cancellationTokenSource;
private async void ExportAsync(PageReport report, IRenderingExtension renderingExtension,
FileStreamProvider outputProvider, NameValueCollection settings)
{
   _cancellationTokenSource = new CancellationTokenSource();
   exportButton.Text = "Cancel";
   exportButton.Click -= exportButton_Click;
   exportButton.Click += cancelExport;

   await Task.Run(() =>
   {
      try
      {
         report.Document.Render(renderingExtension, outputProvider, settings, false, false,
_cancellationTokenSource.Token);
      }
      catch (OperationCanceledException)
      {
      }
   });
```

```
   if (_cancellationTokenSource.IsCancellationRequested)
       MessageBox.Show("Export was cancelled", "Export", MessageBoxButtons.OK,
MessageBoxIcon.Information);
   _cancellationTokenSource.Dispose();
   _cancellationTokenSource = null;
   exportButton.Click -= cancelExport;
   exportButton.Click += exportButton_Click;
   exportButton.Text = "Export";
}

private void cancelExport(object sender, EventArgs e)
{
   _cancellationTokenSource?.Cancel();
}
```

Information Logging

To log information on the report rendering, add Trace Listeners (see https://learn.microsoft.com/en-us/dotnet/framework/debug-trace-profile/trace-listeners for more information). As an example, modify the ExportAsync function as follows.

Visual Basic.NET code

```
Private Async Sub ExportAsync(ByVal report As PageReport, ByVal renderingExtension As
IRenderingExtension, ByVal outputProvider As StreamProvider, ByVal settings As
NameValueCollection)
   labelExport.Text = "Export started..."
   Dim control = Me
   Dim progress = New Progress(Of ProgressInfo)(Sub(progressInfo)
      control.BeginInvoke(New MethodInvoker(Function()
         labelExport.Text = If(progressInfo.IsLast = True, "Export is finished." , $"Exported
{progressInfo.PageNumber} pages.")
         Trace.TraceInformation($"Exported {progressInfo.PageNumber} pages.")
      End Function))
   End Sub)
   await Task.Run(Sub()
      Try
         Dim traceWriter = New TextWriterTraceListener("TextWriterOutput.log", "myListener")
         Trace.Listeners.Clear()
         Trace.Listeners.Add(traceWriter)
         report.Document.Render(renderingExtension, outputProvider, settings, False, False,
CancellationToken.None, progress)
         Trace.Flush()
         Trace.Listeners.Clear()
      Catch e as Exception
      End Try
   End Sub)
```

```
End Sub
```

C# code

```
private async void ExportAsync(PageReport report, IRenderingExtension renderingExtension,
StreamProvider outputProvider, NameValueCollection settings)
{
   labelExport.Text = "Export started...";
   var control = this;
   var progress = new Progress(progressInfo =>
   {
      control.BeginInvoke(new MethodInvoker(() =>
      {
         labelExport.Text = progressInfo.IsLast ? "Export is finished." : $"Exported
{progressInfo.PageNumber} pages.";
         Trace.TraceInformation($"Exported {progressInfo.PageNumber} pages.");
      }));
   });
   await Task.Run(() =>
   {
      try
      {
         var traceWriter = new TextWriterTraceListener("TextWriterOutput.log", "myListener");
         Trace.Listeners.Clear();
         Trace.Listeners.Add(traceWriter);
         report.Document.Render(renderingExtension, outputProvider, settings, false, false,
CancellationToken.None, progress);
         Trace.Flush();
         Trace.Listeners.Clear();
      }
      catch
      {
         // ignored
      }
   });
}
```

Print Reports

When you print a report, the report data is first sent to the report generator and then to the printer. You can also preview the report before you actually print it.

This section describes using print methods in desktop viewers, how to print reports on single page, and different scenarios of printing in JSViewer.

ActiveReports provides you with rich Page/RDLX and Section report printing options and settings. In this section, we will look at the basic printing options for each report type, with sample codes.

Printing is implemented using PrintDocument Class (System.Drawing.Printing), so this functionality will only be available for Windows.

> **Note:** It is important to specify the **MESCIUS.ActiveReports.Viewer.Common ('MESCIUS.ActiveReports.Viewer.Common Assembly' in the on-line documentation)** assembly for printing to work correctly.

Print using Print Methods

ActiveReports provides access to Print methods to enable printing of Page and Section Reports. You can access Print methods in any of the following ways:

- **Viewer.Print method (using the Viewer control)**
- **Print methods in SectionDocument or PageDocument**
- **Print methods in the PrintExtension class**

Viewer.Print method

The code sample illustrates how to access the print method using the Viewer control.

You can use the **Print ('Print Method' in the on-line documentation)** method of the **Viewer ('Viewer Class' in the on-line documentation)** class to print a report loaded in the Viewer control. Make sure that the report is loaded completely before Print is executed.

Visual Basic.NET code. Add this code INSIDE the LoadCompleted event of the Viewer

```
Viewer1.Print(True, True, True)
```

C# code. Add this code INSIDE the LoadCompleted event of the Viewer

```
viewer1.Print(true, true, true);
```

Print methods in SectionDocument or PageDocument

SectionDocument and PageDocument types have Print methods that can be used directly on the document object. The following code samples illustrate how to access the print methods that can be used directly on the document object.

> **Note**: The **Print** method is implemented as an extension method of the **PrintExtension.Print ('Print Method' in the on-line documentation)** method, which is present in the GrapeCity.ActiveReports namespace of GrapeCity.ActiveReports.Viewer.Common assembly.
>
> In order to access **Print** method through SectionDocument or PageDocument class, you need to add **GrapeCity.ActiveReports.Viewer.Common ('MESCIUS.ActiveReports.Viewer.Common Assembly' in the on-line documentation)** reference to the project. Also, as mentioned in the code, make sure that you add a reference for the **GrapeCity.ActiveReports ('GrapeCity.ActiveReports Namespace' in the on-line documentation)** namespace in your project using Imports (Visual Basic.NET) or using (C#) statement.

Section Report

Visual Basic.NET code. Paste at the top of the code view.

```
Imports GrapeCity.ActiveReports
```

Visual Basic.NET code. Paste INSIDE the Form_Load event.

```
Dim rpt = New SectionReport1()
rpt.Run(False)
Dim sectionDocument = rpt.Document
sectionDocument.Print(True, True, False)
```

C# code. Paste at the top of the code view.

```
using GrapeCity.ActiveReports;
```

C# code. Paste INSIDE the Form_Load event.

```
var rpt = new SectionReport1();
rpt.Run(false);
var sectionDocument = rpt.Document;
sectionDocument.Print(true, true, false);
```

Page Report

Visual Basic.NET code. Paste at the top of the code view.

```
Imports GrapeCity.ActiveReports
```

Visual Basic.NET code. Paste INSIDE the Form_Load event.

```
Dim file_name As String = "..\..\PageReport1.rdlx"
Dim pageReport As New GrapeCity.ActiveReports.PageReport(New System.IO.FileInfo(file_name))
Dim pageDocument As New GrapeCity.ActiveReports.Document.PageDocument(pageReport)
pageDocument.Print(True, True, False)
```

C# code. Paste at the top of the code view.

```
using GrapeCity.ActiveReports;
```

C# code. Paste INSIDE the Form_Load event.

```
string file_name = @"..\..\PageReport1.rdlx";
GrapeCity.ActiveReports.PageReport pageReport = new GrapeCity.ActiveReports.PageReport(new
System.IO.FileInfo(file_name));
GrapeCity.ActiveReports.Document.PageDocument pageDocument = new
GrapeCity.ActiveReports.Document.PageDocument(pageReport);
pageDocument.Print(true, true, false);
```

Print methods in the PrintExtension class

You can use the **Print ('Print Method' in the on-line documentation)** method of the **PrintExtension ('PrintExtension Class' in the on-line documentation)** class to print a report loaded in the Viewer control. Make sure that the report is loaded completely before print is executed. The following code samples illustrate how to access the print method of the PrintExtension class.

> **Note**: The **Print** method is implemented as an extension method of the **PrintExtension.Print ('Print Method' in the on-line documentation)** method, which is present in the GrapeCity.ActiveReports namespace of GrapeCity.ActiveReports.Viewer.Common assembly.
>
> In order to access **Print** method through SectionDocument or PageDocument class, you need to add **MESCIUS.ActiveReports.Viewer.Common ('MESCIUS.ActiveReports.Viewer.Common Assembly' in the on-line documentation)** reference to the project. Also, as mentioned in the code, make sure that you add a reference for the **GrapeCity.ActiveReports ('GrapeCity.ActiveReports Namespace' in the on-line documentation)** namespace in your project using Imports (Visual Basic.NET) or using (C#) statement.

Section Report

Visual Basic.NET code. Paste INSIDE an event like Button_Click.

```
GrapeCity.ActiveReports.PrintExtension.Print(sectionDocument, True, True)
```

C# code. Paste INSIDE an event like Button_Click.

```
GrapeCity.ActiveReports.PrintExtension.Print(sectionDocument, true, true);
```

Page Report

Visual Basic.NET code. Paste INSIDE an event like Button_Click.

```
GrapeCity.ActiveReports.PrintExtension.Print(pageDocument, True, True)
```

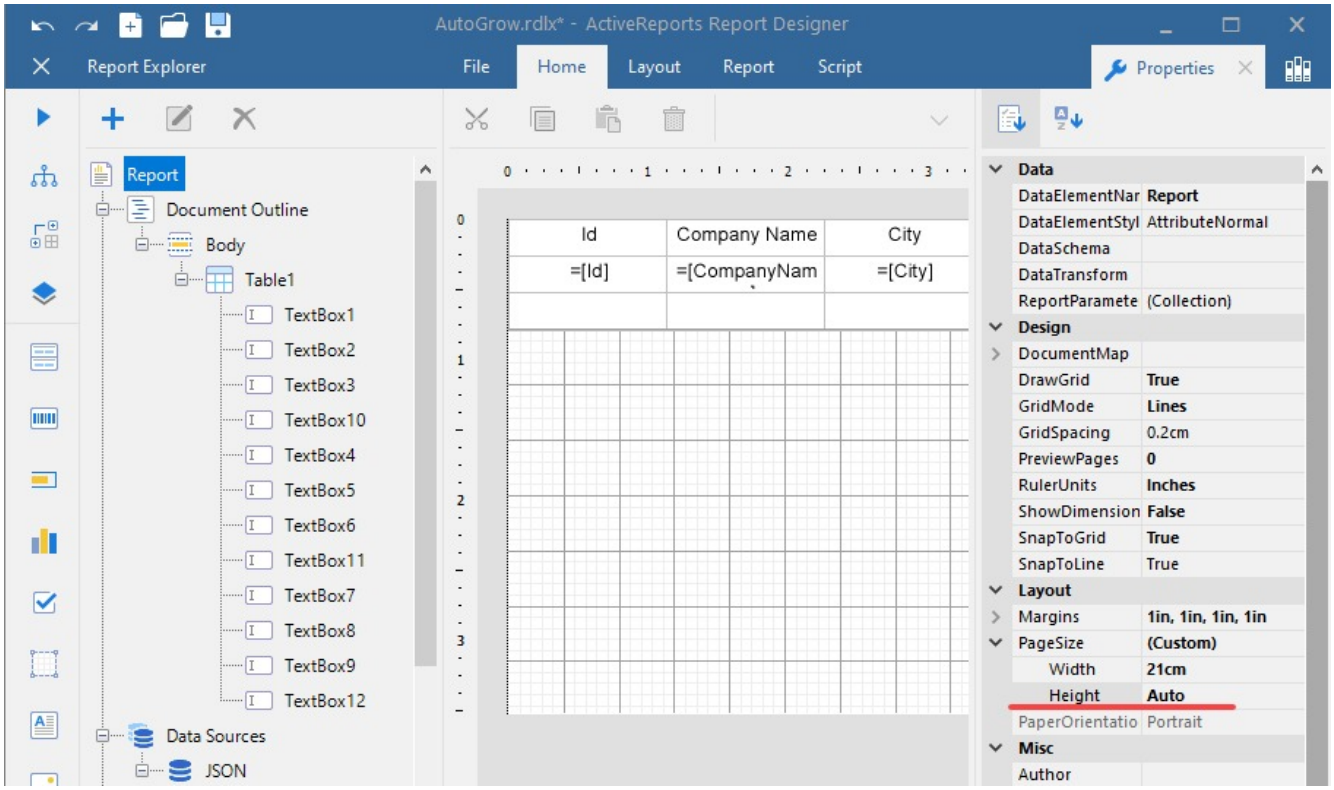
C# code. Paste INSIDE an event like Button_Click.

```
GrapeCity.ActiveReports.PrintExtension.Print(pageDocument, true, true);
```

Print Reports on a Single Page

You can set the printing options for an RDLX report to print it on a single page, despite the report page height. This is useful for printing receipts, bills menu, recharge ticket, etc. where all data is printed in one page.

1. Open an RDLX report in the Designer. This sample uses a report with a Table data region, based on the **Customers** dataset.
2. With a report selected, go to **Properties** and set the **PageSize** > **Height** property to **Auto**. Note that if the **PaperOrientation** property is set to **Portrait,** this property gets disabled when you set **PageSize.Height** to **Auto**.



3. Click **Preview** to see that now the report has one page.
4. Click **Print**.
5. In the **Print** dialog that opens, click **Properties**, and select a page size.
6. Click **OK** to print a report on a single page.

Note:

- If your report has page breaks, then printing on a single page for the report will not work.
- The user should set the correct **Paper Size** in **Printing Preferences** before sending the **Print** command.
- This feature is available in the Visual Studio Integrated Designer and Standalone Report Designer.

Print in Js Viewer

Js Viewer provides several options for printing a report. This topic describes several ways in which a report can be printed in Js Viewer.

Print with Preview

Print report when the report is completely loaded in the viewer, using **print()** method.

index.html

```
var viewer;
function loadViewer() {
    viewer = GrapeCity.ActiveReports.JSViewer.create({
        element: '#viewerContainer',
        reportID: 'Report.rdlx',
        documentLoaded: () => viewer.print()
```

```
    });
}
```

Print without Preview

Print report without previewing using 'global' **print()** method. This is same as the default button in Js Viewer, but without showing report preview.

index.html

```
GrapeCity.ActiveReports.JSViewer.print({ reportID:'Report.rdlx'});
```

Preview Report and Print to PDF

Open the report and export it to PDF with **PrintOnOpen** parameter set to 'true'. In this case, the exported PDF opens in new window of the browser, and the print dialog is displayed.

index.html

```
var viewer;
function loadViewer() {
    viewer = GrapeCity.ActiveReports.JSViewer.create({
        element: '#viewerContainer',
        reportID: 'Report.rdlx',
        documentLoaded: () => viewer.export('Pdf', null, true, { PrintOnOpen: 'true' })
    });
}
```

Print to PDF

Export the report to PDF using 'global' **export()** method and enable the **PrintOnOpen** option. In this case, the report is not opened.

index.html

```
GrapeCity.ActiveReports.JSViewer.export({
    reportID: 'Report.rdlx', exportType: 'Pdf', settings: { PrintOnOpen: 'true' },
    callback: (args) => { window.open(args) }
})
```

Note:

- For Section Reports, use **OnlyForPrint** instead of **PrintOnOpen** (for backward compatibility). For Page and RDLX reports (.rdlx), anyone of PrintOnOpen or OnlyForPrint can be used.
- Use latest versions of Chrome, Firefox, and Chrome-based Edge for the described print features to work correctly.

Plugins Development

ActiveReports components can be used as plugins for third-party applications.

A sample of creating a new application domain, where you can load ActiveReports is shown below:

```
string appDir = Path.GetDirectoryName(GetType().Assembly.CodeBase.Substring(8));
     //grant permission to the untrusted application
     var pset = new PermissionSet(PermissionState.Unrestricted);
     var sandBoxSetup = new AppDomainSetup
          {
               //set the ApplicationBase to a folder
               ApplicationBase = appDir,
               ConfigurationFile = appDir + "....dll.config"

          };
         //create app domain
         AppDomain domain = AppDomain.CreateDomain("Sandboxed Domain", null, sandBoxSetup,
pset);
```

Note: Partial/Medium Trust is obsolete. ActiveReports does not support it anymore.

Samples

The ActiveReports 18 samples are available on GitHub. These samples are categorized under two folders - Samples and WebSamples, and further based on features.

Samples

Sample		Description
Advanced	**Page and RDLX reports**	
	Calendar	This sample demonstrates using Calendar data region in reports.
	Custom Chart	This sample demonstrates using custom report item - Radar chart in a report.
	Custom data Provider	This sample demonstrates how to create a project using custom data provider and how to pull data from a comma separated value (CSV) file.
	Custom Pdf Export	This sample demonstrates exporting reports to PDF format using third-party assemblies.
	Custom Resource Locator	This sample showcases a custom implementation of the resource locator to load pictures from the user's "My Pictures" directory.
	Custom Tile Provider	This sample demonstrates how to create a custom tile provider.
	Oracle Data Provider	This sample illustrates using Oracle Data Provider as data source for designing Page/RDLX reports.
	Section Reports	
	Custom Drill Through	This sample demonstrates using hyperlinks and the viewer hyperlink event to simulate drill-down from one report to another.
	Custom Word Export	This sample demonstrates exporting Section Report to Word format using third-party assemblies.
API	**Page and RDLX reports**	
	Create Report	This sample demonstrates how to create a Page Report layout in code. It further shows creating a table control, adding table rows and table columns inside it, adding cells inside the table rows and columns and adding text boxes inside the cells.
	Digital Signature Pro	This sample demonstrates how to add digital signatures when exporting to PDF format.
	Export	This sample demonstrates how to export Page and RDLX reports to different export formats.
	FontResolver	This sample demonstrates implementing custom fonts.
	Layers	This sample demonstrates how to use Layers in a report.
	Report Wizard	This sample demonstrates how to create a custom Report Wizard that

		allows you to select a report from the list of multiple reports and then allows you to select the data that you want to display in the selected report.
	Stylesheets	This sample demonstrates how to work with embedded and external style sheets in Page and RDLX reports.
	Section Reports	
	Charting	This sample demonstrates chart types used in different scenarios, in both bound and unbound modes.
	Cross Section Controls	This sample demonstrates the use of the cross section lines and boxes.
	Cross Tab Report	This sample demonstrates using unbound data, conditional highlighting and distributing data across columns to create a cross-tab view and data aggregation.
	Custom Annotation	This sample demonstrates adding the Custom Annotation button to the report Viewer toolbar and adding a new annotation to the report.
	Digital Signature Pro	This sample demonstrates adding the Custom Annotation button to the report Viewer toolbar and adding a new annotation to the report.
	Export	This sample demonstrates how to export to different export formats using code.
	Inheritance	This sample demonstrates using the method that inherits a report at run time and design time.
	Print Multiple Pages per Sheet	This sample demonstrates printing a document with multiple pages per sheet by using the common PrintDocument class of the NET.Framework.
	Style Sheets	This sample demonstrates changing styles at run time to provide a different look to a same report.
	Sub Report	This sample demonstrates using subreports in an ActiveReports report.
	Summary	This sample demonstrates how to display summarized data in a Section Report.
Data Binding	**Page and RDLX reports**	
	CSV Data Source	This sample demonstrates how to connect to a CSV data source.
	DataSet DataSource	This sample demonstrates how to use a dataset as a data source for a report.
	Json Data Source	This sample demonstrates how to use the Json data provider at run time and add a web service for authentication.
	Object Data Source	This sample demonstrates how to use Object provider for binding a report.
	OData Data Source	This sample demonstrates how to use OData EndPoint for binding a report.
	OleDb Data Source	This sample demonstrates how to connect to an OleDb data source at run

		time and pass data to the report using LocateDataSource event.
	Xml Data Source	This sample demonstrates how to connect to a XML data source at run time and pass data to the report using LocateDataSource event.
	Section Reports	
	Bound Data	This sample demonstrates binding to ADO.NET Data objects.
	IList Binding	This sample demonstrates creating a custom collection that stores data from the database in the List. The custom collection is displayed by binding data to the DataGridView control by using the DataSource property of this control.
	LINQ	This sample demonstrates how to use LINQ in an ActiveReports report.
	Unbound Data	This sample demonstrates how to create a dataset for a Section Report and use the FetchData event to populate the Fields collection to display the report unbound data.
	XML	This sample demonstrates how to create a report with XML data, using a SubReport or using the XML hierarchical structure.
Designer Pro	Map	This sample demonstrates how to work with Map control in ActiveReports.
	End User Designer	This sample demonstrates a custom end-user report designer that can be integrated in your applications to allow users to modify report layouts.
	FlatEndUserDesigner	This sample demonstrates building the designer using DesignerForm class.
	Reports Gallery	This sample demonstrates customizing End User Designer application to display a list of categorized reports.
	Table of Contents	This sample demonstrates how to use TableofContents control in ActiveReports.
Desktop	WPF Viewer	This sample demonstrates using WPF Viewer in a WPF application.
	Win Viewer	This sample demonstrates using Win Viewer in a Windows Form application.
Web	Custom Preview	The sample demonstrates exporting an ActiveReports report to the HTML or PDF format in your Web application.

Web Samples

Sample	Description
WebDesigner Samples	
Blazor Designer	The samples on Blazor Designer demonstrate the use of ActiveReports Blazor Designer with Server application, WebAssembly application, and remote report service.
WebDesigner MVC(Core)	This sample demonstrates WebDesigner with an ASP.NET MVC Core back end.
WebDesigner	This sample demonstrates the ActiveReports WebDesigner with an Angular 8 app and ASP.NET Core

Angular(Core)	back end.
WebDesigner Blazor	This sample demonstrates the ActiveReports WebDesigner with Blazor framework.
WebDesigner Custom Data Providers	The sample demonstrates the method to use custom data providers (such as SQLite and OData) for supplying data to the report in the ActiveReports WebDesigner.
WebDesigner Custom Store	The sample demonstrates the use of custom resources service for ActiveReports WebDesigner with an ASP.NET Core back end.
JSViewer Samples	
BlazorViewer	The samples on Blazor Viewer demonstrate the use of ActiveReports Blazor Viewer with Server application, WebAssembly application, and remote report service.
JSViewer Angular(Core)	This sample demonstrates the use of the ActiveReports JSViewer with an Angular 8 app and ASP.NET Core back end.
JSViewer CORS	This sample demonstrates the use of ActiveReports JSViewer with an ASP.NET MVC 5 back end when the server is hosted elsewhere using CORS.
JSViewer MVC CORS(Core)	This sample demonstrates the use of ActiveReports JSViewer with an ASP.NET MVC 5 Core back end when the server is hosted elsewhere using CORS.
JSViewer MVC	This sample demonstrates the use of ActiveReports JSViewer with an ASP.NET MVC 5 back end.
JSViewer MVC(Core)	This sample demonstrates the use of ActiveReports JSViewer with an ASP.NET MVC Core back end.
JSViewer React(Core)	This sample demonstrates the use of ActiveReports JSViewer with an ReactJS app and ASP.NET Core back end.
JSViewer Vue(Core)	This sample demonstrates the use of ActiveReports JSViewer with an VueJS app and ASP.NET Core back end.
WebViewer ASP.NET	This sample demonstrates the ActiveReports web control feature and generating a parameterized report.
Silent Print	The SilentPrint sample project consists of three samples - JSViewerBatchPrint_MVC_Core, JSViewerSilentPrint_MVC_Core, and PrintAgent. The JSViewerBatchPrint_MVC_Core sample demonstrates how to print many reports by clicking the Print button, without showing the Print Preview dialog for every report. Silent printing is implemented through a print agent that needs to be started. The JSViewerSilentPrint_MVC_Core sample demonstrates how to print a report by clicking once the JSViewer Print button, without showing the Print Preview dialog. Silent printing is implemented through a print agent that needs to be started. The PrintAgent sample contains a Windows service, hosting an ASP.NET Core API that allows printing PDF files. The print agent uses the GrapeCity.Documents.Pdf library.

Samples

The samples in Samples18 folder demonstrate report designing features that cover desktop, designer, API as well as advanced features. Download these samples from following link:

https://github.com/activereports/Samples18

Each sample has a **C#** and a **Visual Basic.NET** code example for Visual Studio. You can also see the comments within the sample projects throughout code.

Sample		Description
Advanced	**Page and RDLX reports**	
	Calendar	This sample demonstrates using Calendar data region in reports.
	Custom Chart	This sample demonstrates using custom report item - Radar chart in a report.
	Custom data Provider	This sample demonstrates how to create a project using custom data provider and how to pull data from a comma separated value (CSV) file.
	Custom Pdf Export	This sample demonstrates exporting reports to PDF format using third-party assemblies.
	Custom Resource Locator	This sample showcases a custom implementation of the resource locator to load pictures from the user's "My Pictures" directory.
	Custom Tile Provider	This sample demonstrates how to create a custom tile provider.
	Oracle Data Provider	This sample illustrates using Oracle Data Provider as data source for designing Page/RDLX reports.
	Section Reports	
	Custom Drill Through	This sample demonstrates using hyperlinks and the viewer hyperlink event to simulate drill-down from one report to another.
	Custom Word Export	This sample demonstrates exporting Section Report to Word format using third-party assemblies.
API	**Page and RDLX reports**	
	Create Report	This sample demonstrates how to create a Page Report layout in code. It further shows creating a table control, adding table rows and table columns inside it, adding cells inside the table rows and columns and adding text boxes inside the cells.
	Digital Signature Pro	This sample demonstrates how to add digital signatures when exporting to PDF format.
	Export	This sample demonstrates how to export Page and RDLX reports to different export formats.
	Font Resolver	This sample demonstrates implementing custom fonts.
	Layers	This sample demonstrates how to use Layers in a report.

	Report Wizard	This sample demonstrates how to create a custom Report Wizard that allows you to select a report from the list of multiple reports and then allows you to select the data that you want to display in the selected report.
	Stylesheets	This sample demonstrates how to work with embedded and external style sheets in Page and RDLX reports.
	Section Reports	
	Charting	This sample demonstrates chart types used in different scenarios, in both bound and unbound modes.
	Cross Section Controls	This sample demonstrates the use of the cross section lines and boxes.
	Cross Tab Report	This sample demonstrates using unbound data, conditional highlighting and distributing data across columns to create a cross-tab view and data aggregation.
	Custom Annotation	This sample demonstrates adding the Custom Annotation button to the report Viewer toolbar and adding a new annotation to the report.
	Digital Signature Pro	This sample demonstrates how to add digital signatures when exporting to PDF format.
	Export	This sample demonstrates how to export to different export formats using code.
	Inheritance	This sample demonstrates using the method that inherits a report at run time and design time.
	Print Multiple Pages per Sheet	This sample demonstrates printing a document with multiple pages per sheet by using the common PrintDocument class of the NET.Framework.
	Style Sheets	This sample demonstrates changing styles at run time to provide a different look to a same report.
	Sub Report	This sample demonstrates using subreports in an ActiveReports report.
	Summary	This sample demonstrates how to display summarized data in a Section Report.
Data Binding	**Page and RDLX reports**	
	CSV Data Source	This sample demonstrates how to connect to a CSV data source.
	DataSet DataSource	This sample demonstrates how to use a dataset as a data source for a report.
	Json Data Source	This sample demonstrates how to use the Json data provider at run time and add a web service for authentication.
	Object Data Source	This sample demonstrates how to use Object provider for binding a report.
	OData Data Source	This sample demonstrates how to use OData EndPoint for binding a report.

	OleDb Data Source	This sample demonstrates how to connect to an OleDb data source at run time and pass data to the report using LocateDataSource event.
	Xml Data Source	This sample demonstrates how to connect to a XML data source at run time and pass data to the report using LocateDataSource event.
	Section Reports	
	Bound Data	This sample demonstrates binding to ADO.NET Data objects.
	IList Binding	This sample demonstrates creating a custom collection that stores data from the database in the List. The custom collection is displayed by binding data to the DataGridView control by using the DataSource property of this control.
	LINQ	This sample demonstrates how to use LINQ in an ActiveReports report.
	Unbound Data	This sample demonstrates how to create a dataset for a Section Report and use the FetchData event to populate the Fields collection to display the report unbound data.
	XML	This sample demonstrates how to create a report with XML data, using a SubReport or using the XML hierarchical structure.
Designer Pro	Map	This sample demonstrates how to work with Map control in ActiveReports.
	End User Designer	This sample demonstrates a custom end-user report designer that can be integrated in your applications to allow users to modify report layouts.
	FlatEndUserDesigner	This sample demonstrates building the designer using DesignerForm class.
	Reports Gallery	This sample demonstrates customizing End User Designer application to display a list of categorized reports.
	Table of Contents	This sample demonstrates how to use TableofContents control in ActiveReports.
Desktop	WPF Viewer	This sample demonstrates using WPF Viewer in a WPF application.
	Win Viewer	This sample demonstrates using Win Viewer in a Windows Form application.
Web	Custom Preview	The sample demonstrates exporting an ActiveReports report to the HTML or PDF format in your Web application.

Advanced

The samples in the Advanced folder describe advanced features separately for:

- Page and RDLX reports
- Section Reports

Page and RDLX Reports

This section discusses following samples describing various features in Page and RDLX reports:

Calendar
: This sample demonstrates using Calendar data region in reports.

Custom Chart
: This sample demonstrates using custom report item - Radar chart in a report.

Custom Data Provider
: This sample demonstrates how to create a project using custom data provider and how to pull data from a comma separated value (CSV) file.

Custom Pdf Export
: This sample demonstrates exporting reports to PDF format using third-party assemblies.

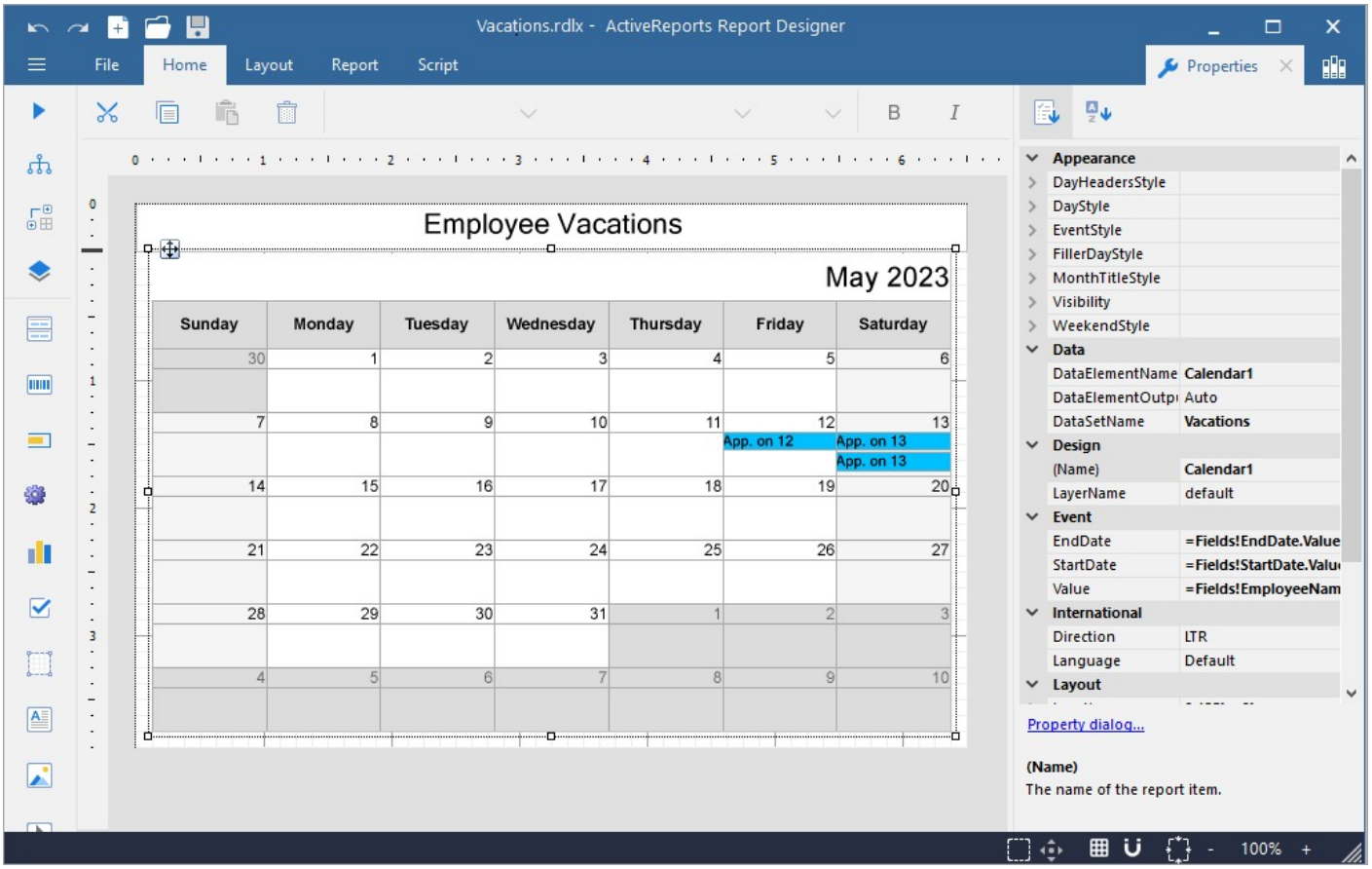
Custom Resource Locator
: This sample showcases a custom implementation of the resource locator to load pictures from the user's "My Pictures" directory.

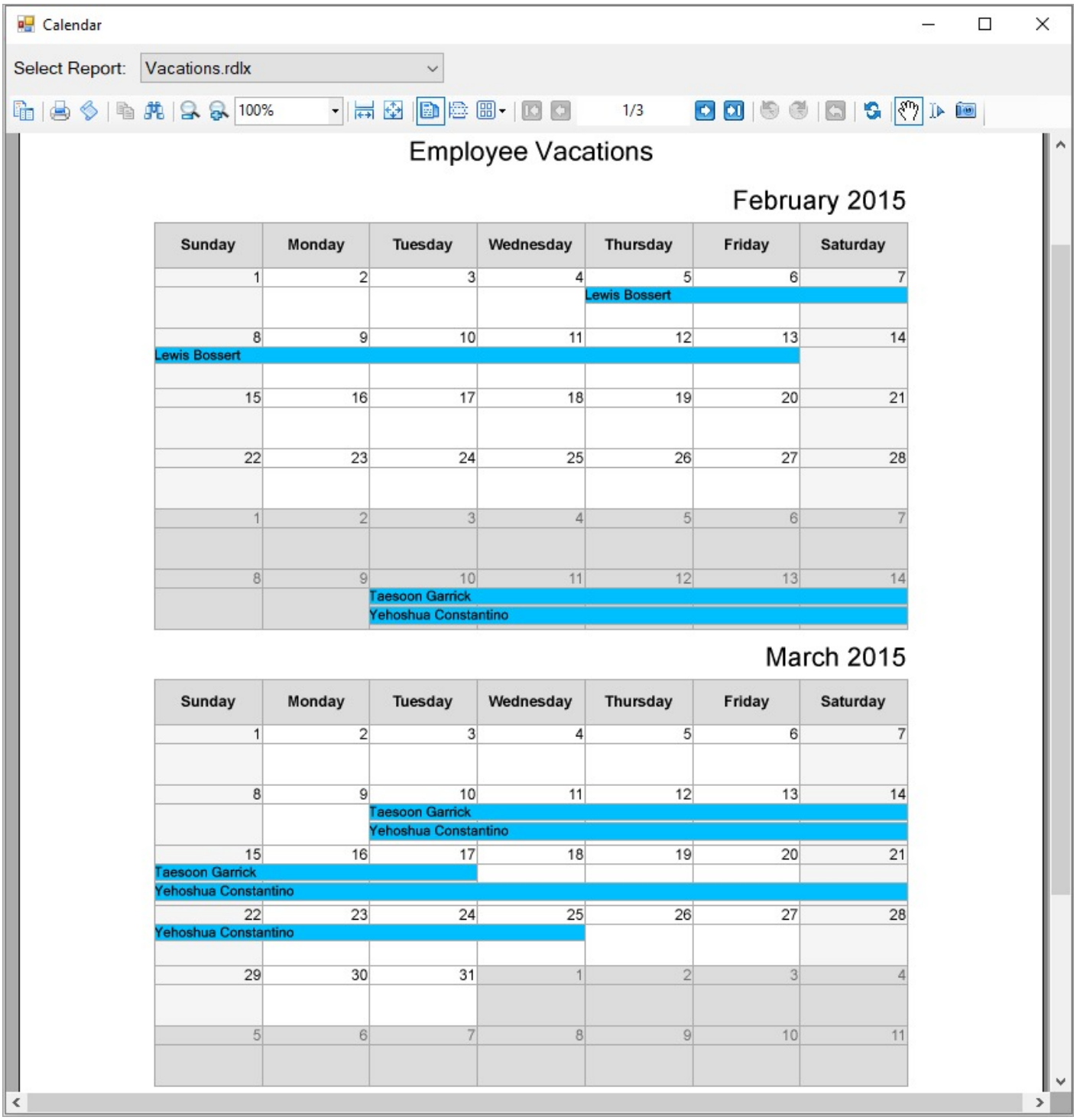
Custom Tile Provider
: This sample demonstrates how to create a custom tile provider.

Oracle Data Provider
: This sample illustrates using Oracle Data Provider as data source for designing Page/RDLX reports.

Calendar

The Calendar control is now moved to samples to allow customers to continue using this data region for displaying date-based data or events in a calendar format.





Sample Location

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/Calendar

Details

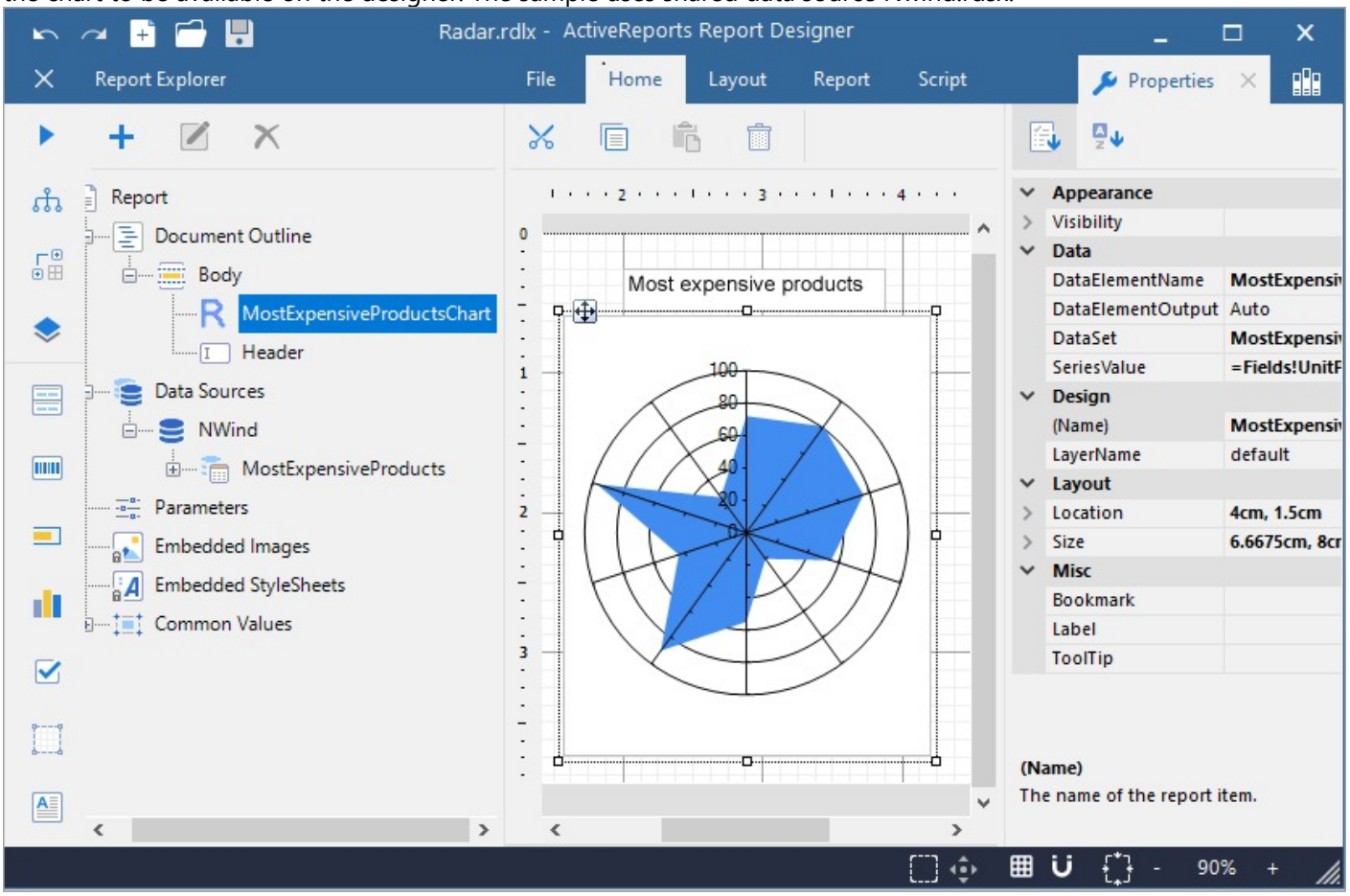
This sample consists of following projects; all data and files are taken from the Calendar data region itself:

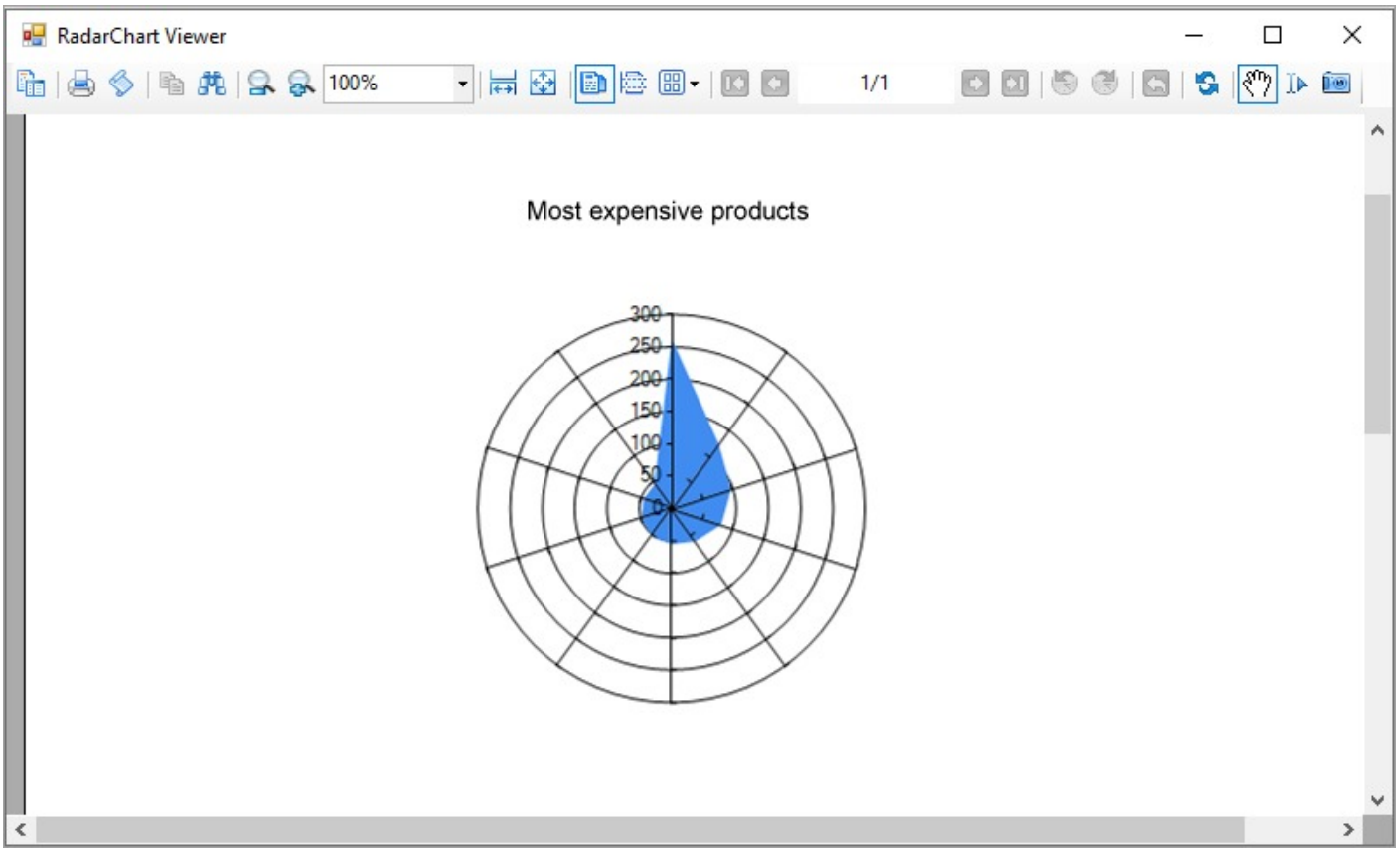
- CalendarComponent: It implements **ICustomReportItem** interface to render Calendar, **IDataRegion** interface for data binding, and **IImageRenderer** interface to render calendar content range to the canvas.
- CalendarDesigner: The designer is inherited from **CustomReportItemDesigner** class. It implements property initialization, glyph drawing, and evaluation utils.
- TestDesignerPro: This is the default start up project. On running this project, an RDLX report with a calendar is displayed on the designer. You can also drag and drop and use the Calendar data region available on the toolbox.
- TestViewer: On running this project, a calendar is rendered on Windows Forms Viewer.
- Tests: It contains code for proper functioning of the sample.

Customers who are required to use calendar should now compile and distribute **CalendarComponent** and **CalendarDesigner** assemblies. Binding should be done through ActiveReports.config file (see test applications).

Custom Chart

This sample illustrates creating custom report item - Radar Chart. The **ICustomReportItem** interface is used to implement custom control, which is radar chart. The designer inherited from **CustomReportItemDesigner** class allows the chart to be available on the designer. The sample uses shared data source Nwind.rdsx.





Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomChart/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomChart/C#

Details

When you run this sample, an RDLX report 'Radar.rdlx' with Radar chart is displayed on the designer. Go to the Preview tab of the designer to view the report with data pulled from Nwind.rdsx.

The sample consists of following projects:

- RadarChart: It implements **ICustomReportItem** interface to render Radar chart and **IDataRegion** for data binding. The **IImageRenderer** interface renders data to image; this renderer accesses custom data grouping (note that series property name should be same as that defined in designer) and reads values for the chart series.

- RadarDesigner: The designer is inherited from **CustomReportItemDesigner** class. To render one series, one data grouping is added in the **Initialize** method. Custom properties called DataSetName and SeriesValue are added which can be changed in the designer, see classes **DataSetNamesConverter** and **RadarValuesConverter**. For design-time rendering, **RadarControlGlyph** class implements overriding **ControlGlyph** property of the designer and rendering stub data. In this glyph, the **MovableBehavior** method

implements moving and resizing of the chart control.

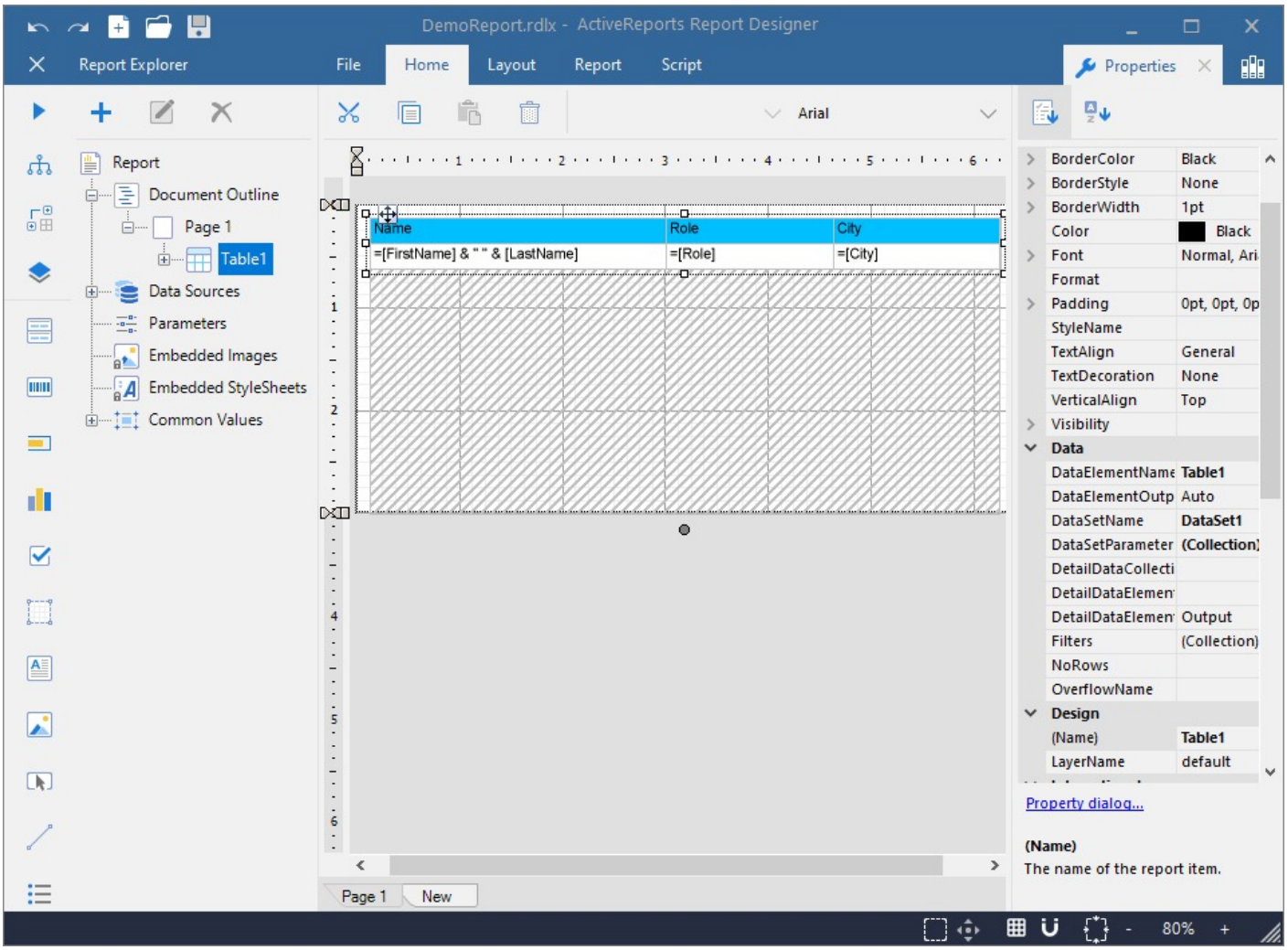
- TestDesignerPro: This is the default start up project. On running this project, an RDLX report with Radar chart is displayed on the designer. You can change the chart properties from the Properties pane. You can also drag and drop and use the Radar Chart control available on the toolbox.

- TestViewer: On running this project, Radar chart is rendered on Windows Forms Viewer.

The custom report item (**RadarChart**) and its designer (**RadarDesigner**) are defined in **ActiveReports.config** file placed in test application projects.

Custom Data Provider

The Custom Data Provider sample demonstrates how to create a project that use a custom data provider and how to pull data from a comma separated value (CSV) file. This sample is part of the ActiveReports Professional Edition.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomDataProvider/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomDataProvider/C#

Details

When you run this sample, a **DesignerForm** displaying the **DemoReport.rdlx** report in ActiveReports Designer and a **HelperForm** explaining the steps to connect a report to the comma separated value (CSV) file appears.

In the ActiveReports Designer, you can add a dataset with a comma separated values (CSV) file. For adding this file,in the Report Explorer, expand the DataSources node, right-click the node for the data source and select **Add DataSet**. In the DataSet dialog that appears, under **Query** section, go to the **Query String** field and click the drop-down arrow to display the custom query editor. In the custom query editor, click the **Select CSV File** button and select the **Categories.csv** file kept within the project.

Go to the Preview tab of the Designer to view the report with the data pulled from the custom data provider.

The sample consists of following three projects:

CustomDataProvider: It contains following items:

- CsvColumn: This class represents information about fields in the data source.
- CsvCommand: This class provides the IDbCommand implementation for the .NET Framework CSV Data Provider.
- CsvConnection: This class provides an implementation of IDbConnection for the .NET Framework CSV Data Provider.
- CsvDataProviderFactory: This class implements the DataProviderFactory for .NET Framework CSV Data Provider.
- CsvDataReader: This class provides an implementation of IDataReader for the .NET Framework CSV Data Provider.

CustomDataProviderUI: It contains following items:

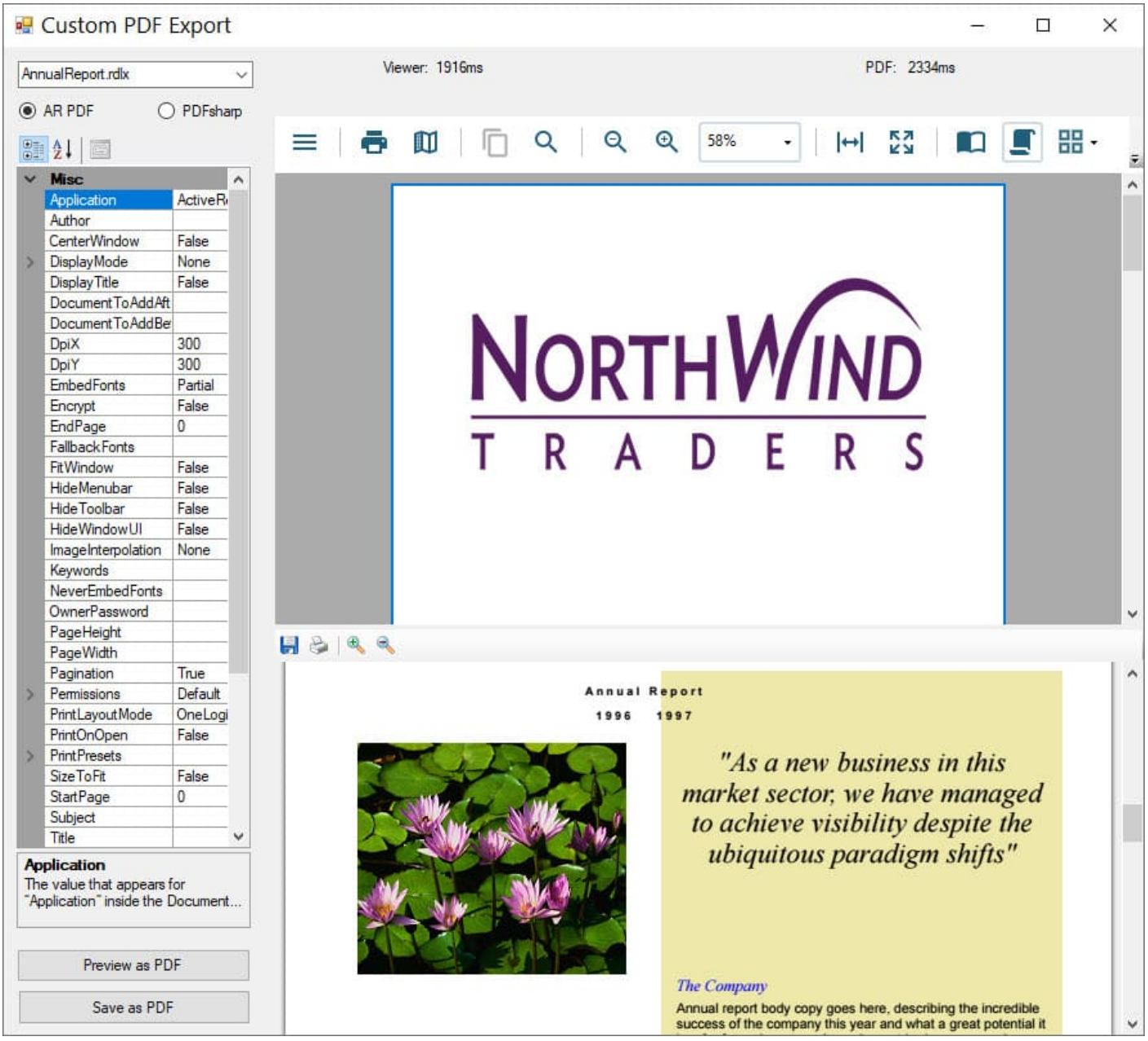
- CSVFileSelector: This is the form that contains the **Select CSV File** button. This button is displayed in the CSV data provider query editor when you open the **DataSet** dialog and under **Query**, in the **Query String** field and click the drop-down arrow to display the custom query editor. Clicking the **Select CSV File** button allows you to select the **Categories.csv** file that is used as a custom data provider for the sample report.
- QueryEditor: This is the class that reads the content of the specified file and builds the CSV data provider query string.

CustomDataProviderUITest: It contains following items:

- Categories.csv: This is the comma separated values (CSV) file that serves as a custom data provider for the sample report. This file is selected in the **Please, select CSV File** dialog that appears when you click the **Select CSV File** button in the **Query String** field under **Query** in the **DataSet** dialog.
- DemoReport.rdlx: The DemoReport.rdlx displays the custom data. This report contains one **Table** data region with the **TextBox** controls, which display the name, the role and the city information of an employee.
- DesignerForm: This is the main form of this sample that appears when you run the sample. On this form, you can connect the sample report to a custom data provider by adding a dataset with a comma separated values (CSV) file. Right-click the form and select **View Code** to see the code implementation for the ActiveReports Designer.
- ActiveReports.config: The configuration file that configures the project to use the custom data provider.
- HelperForm: This form appears on top of the main DesignerForm when you run the sample. This form contains the explanatory text with the steps on how to bind the sample report to the comma separated value (CSV) file. You can close the Help form by clicking the X button in the upper-right corner of the form.

Custom PDF Export

This sample shows how to implement simple exports to custom formats (which is not available in ActiveReports right now). The sample uses third-party library to show export to PDF.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomPdfExport/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomPdfExport/C#

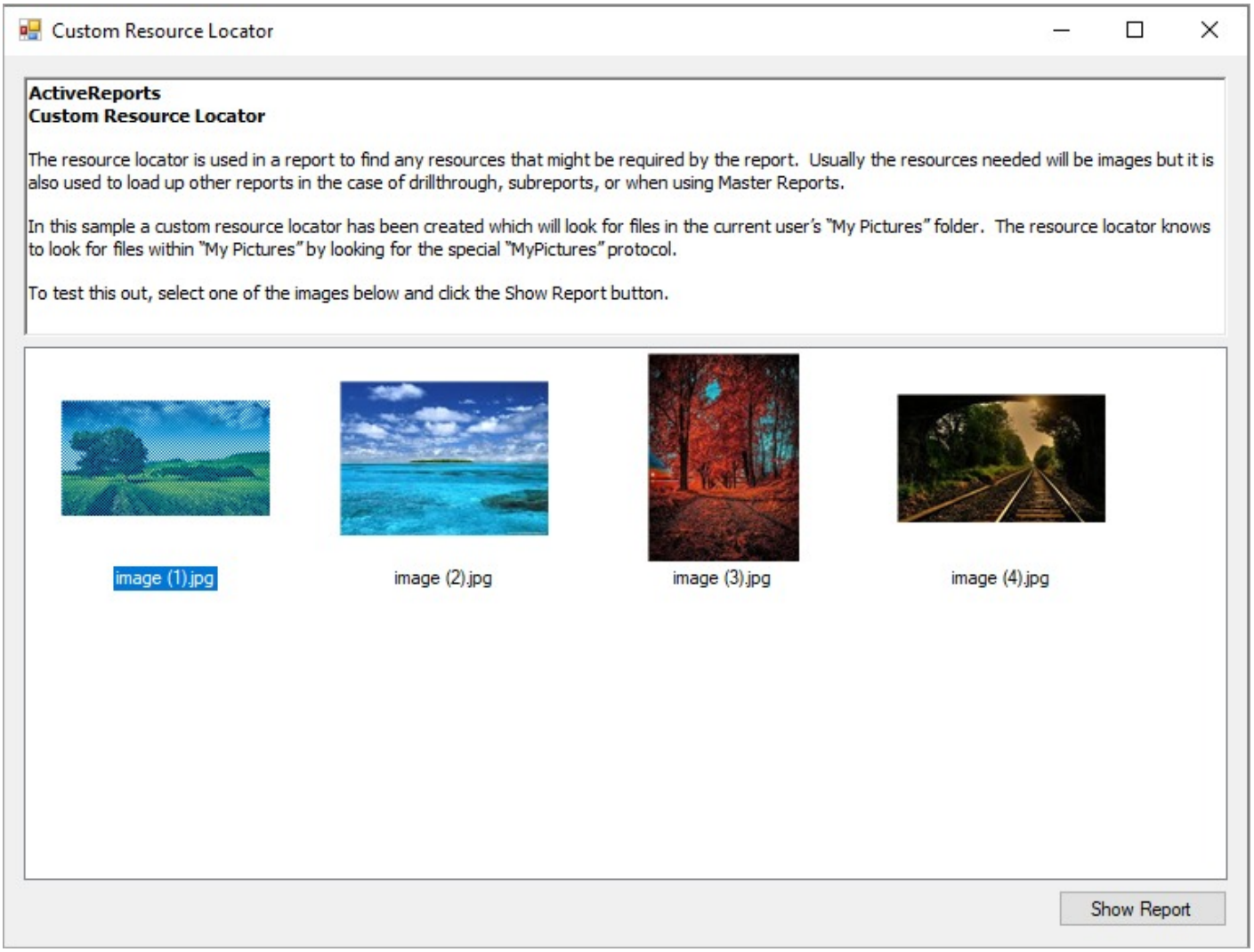
Details

When you run this sample, you see a testing application form, which splits to Windows Forms Viewer and PDF Viewer. You can choose a report and a Pdf export option, click 'Preview as PDF' and view the report in the viewers, and 'Save as PDF' to save the report as Pdf. The sample consists of following projects:

- PdfRendering: It implements **IRenderingExtension** and **IDrawingCanvas** interfaces for customized export to PDF format.
- TestApplication: This is the default start up project to compare and debug custom export.

Custom Resource Locator

The Custom Resource Locator sample demonstrates a custom implementation of the resource locator to load pictures from the user's **Pictures** or **My Pictures** directory. In general, you can use a resource locator in a report to find any resources that a report may require.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomResourceLocator/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomResourceLocator/C#

Details

When you run this sample, you see the **MainForm** with the list of images from the **Pictures** or **My Pictures** directory. Select any image and click the **Show Report** button. A report with the selected image opens in the **PreviewForm**.

> ⚠ **Caution**: To run this sample properly, you must have image files in your pictures directory. If the directory does not contain any pictures, you should add them to the folder manually.

The sample consists of:

- Resources folder: This folder contains the **Description.rtf** file that contains a summarized content of the resource locator that gets displayed inside the RichTextBox control on the **MainForm** at run time. This folder also contains the **NoImage.bmp** image file that is used if there is no image in the pictures directory.

- DemoReport.rdlx: The DemoReport.rdlx displays the selected image. This report contains two TextBox controls and one Image control, which display the image name, the image type and the image at run time after you click the **Show Report** button on the **MainForm**.

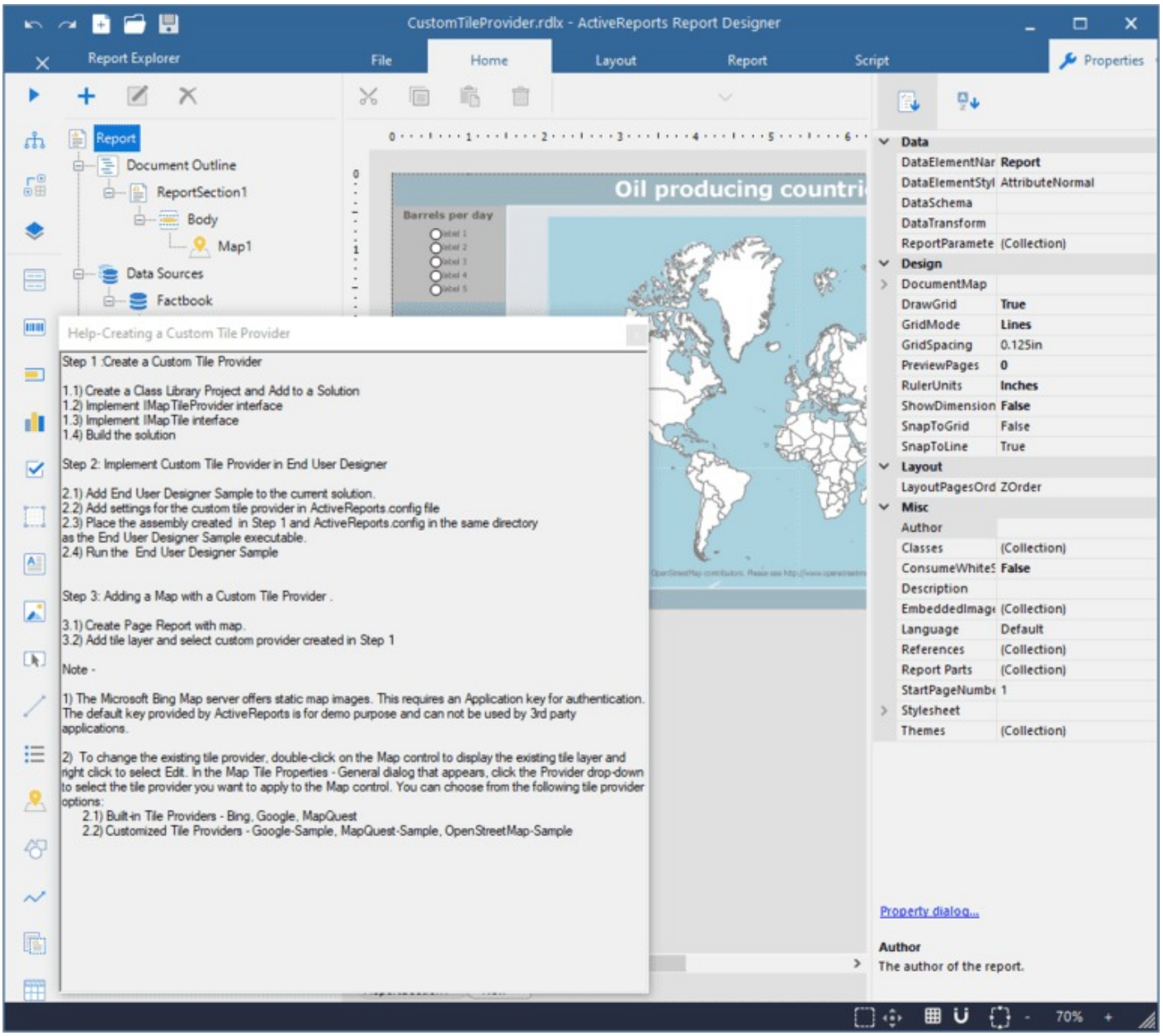
- MainForm: This is the main form of this sample that appears when you run the sample. This form contains the RichTextBox, the ListView and the Button controls. The RichTextBox control displays the summarized information saved in the **Description.rtf** file about the resource locator and the sample. The ListView control gets populated with the images located in the **My Pictures** directory; the Button control is used to generate the report with the selected image. Right-click the form and select **View Code** to see how to load text in the RichTextBox control and images in the ListView control. It also contains code that displays the **DemoReport.rdlx** on the **showReport_Click** event.

- MyPicturesLocator: This file is an internal class that contains code that looks for resources in the **My Pictures** directory.
- PreviewForm: This form uses the ActiveReports **Viewer** control to display the **DemoReport.rdlx** with the selected image. Right-click the form and select **View Code** to see how to load the report into the Viewer.

Custom Tile Provider

The CustomTileProvider sample demonstrates how to create a custom tile provider using **IMapTileProvider ('IMapTileProvider Interface' in the on-line documentation)** and **IMapTile ('IMapTile Interface' in the on-line documentation)** interfaces and configure it in a Map Control which is placed on a RDLX report. This sample uses two projects - CustomTileProviders and TileProviderEndUserDesigner in a single Visual Studio solution. The CustomTileProviders project contains the tile server configurations for the tile providers, whereas the TileProviderEndUserDesigner project references the created CustomTileProviders project assemblies.

> **Note:** CustomTileProvider is for use with the Professional Edition license only. An evaluation message is rendered when used with the Standard Edition license.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomTileProvider/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/CustomTileProvider/C#

Details

When you run this sample, the ActiveReports End User Designer appears with an overlaying **Help-Creating a Custom Tile Provider** dialog. This dialog gives you step-by-step instructions to create a new tile provider for a Map control with custom settings.

The End User Designer displays a RDLX report containing a Map control with MapQuest set as the default tile provider. To change the existing tile provider, double-click on the Map control to display the existing tile layer and right click to select **Edit**. In the **Map Tile Properties - General** dialog that appears, click the **Provider** drop-down to select the tile provider you want to apply to the Map control. Go to the Preview tab to view the data in the selected tile provider. You can choose from the following tile provider options:

- Google-Sample
- MapQuest-Sample
- OpenStreetMap-Sample

> **Note**: The Microsoft **Bing Map** server offers static map images. This requires an Application key for authentication. The default key provided by ActiveReports is for demo purpose and can not be used by 3rd party applications. In order to obtain a Bing Map Key, see HowTo - Create a Bing Map Account and HowTo - Get a Bing Map Key.

The sample consists of two projects:

CustomTileProviders: It contains following classes:

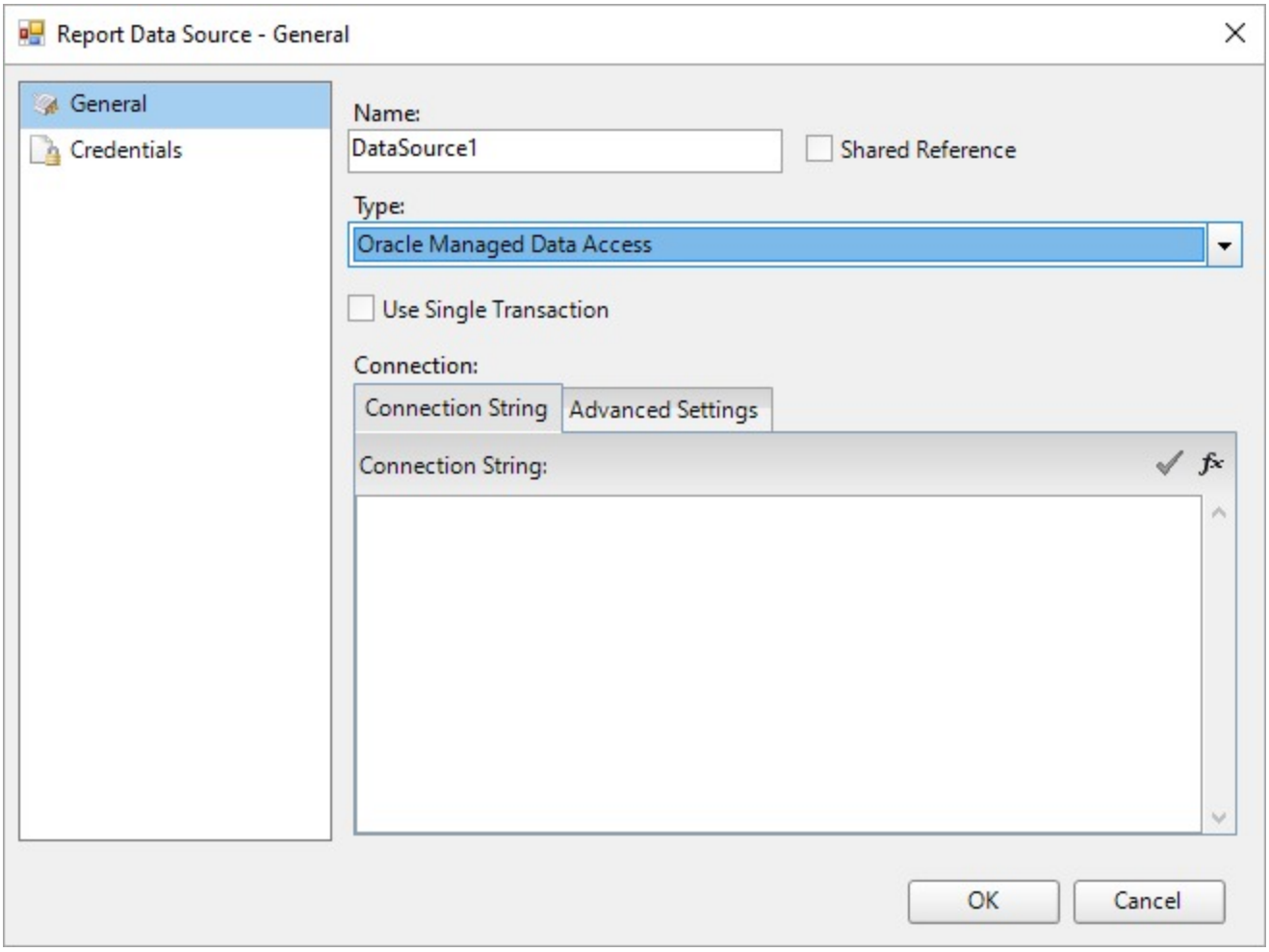
- GoogleMapsTileProvider: This class implements the **IMapTileProvider ('IMapTileProvider Interface' in the on-line documentation)** interface and contains the settings for the map tile images provided from https://www.google.com/maps.
- MapQuestTileProvider: This class implements the **IMapTileProvider** interface and contains the settings for the map tile images provided from https://www.mapquest.com/.
- MapTile: This class represents a single map tile, implementing the **IMapTile ('IMapTile Interface' in the on-line documentation)** interface.
- OpenStreetMapTileProvider: This class implements the **IMapTileProvider** interface and contains the settings for the map tile images provided from https://www.openstreetmap.org.
- WebRequestHelper: This class picks the raw data from the tile providers and loads them into the System.IO.MemoryStream class.

TileProviderEndUserDesigner: It contains following files:

- CustomTileProvider.rdlx: This report contains the Map control that visualizes the oil production in different parts of the world on a virtual earth background. The map control uses the color rule set on a polygon layer to differentiate parts of world as per their oil production capacity. These colors are defined using a color rule which is described in the legend at run time. The report gets the data from **Factbook.rdsx** shared data source.
- DesignerForm.cs: This is the main form that gets displayed when you run the sample. This form uses multiple controls like the ToolStripPanel, ToolStripContainerPanel, SplitContainer, Designer, Toolbox, ReportExplorer and PropertyGrid controls to create a customized End User Designer. It also contains code to load CustomTileProvider.rdlx report into the Designer.
- ActiveReports.config: This configuration file contains the settings for the various tile providers, and is located in the same folder as the EndUserDesigner.exe file for the tile provider settings to work.
- HelperForm.cs: This form uses the HelperForm class to display a screen containing step-by-step instructions for the user to create a custom tile provider.

Oracle Data Provider

Use this sample if you want to connect to the Oracle Data Provider, which is otherwise not available since System.Data.OracleClient is deprecated.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/OracleDataProvider/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/PageAndRDLX/OracleDataProvider/C#

Details

When you run this sample, a blank DesignerForm for RDLX report is displayed. Connect to the Oracle data provider as follows:

1. Add a data source.
2. In the Report Data Source dialog, select **Type** as **Oracle Managed Data Access**.
3. Enter the connection string.

Sample Oracle Connection String

data source=in-data-sql/orcl.grapecity.net;user id=user1;password=password@123

Now, proceed the report designing by pulling the data from Oracle data provider.

The sample consists of following:

TestDesignerPro.csproj: This is the default start up project.

ActiveReports.config: Located inside the startup project, it is a configuration file that contains the settings for using the oracle data provider:

- **DisplayName** to reference data provider and **Type** to use the oracle custom data provider as mandatory fields
- **AdapterType** implemented in 'OracleConnectionAdapter' class and **SchemaProviderType** implemented in 'GeneralOracleSchemaProvider' class for additional features.

OracleConnectionAdapter.cs: This class provides features related to parameters such as handling multi-value paramaters and parameterized queries.

GeneralOracleSchemaProvider.cs: This class generates DataSchema to enable visual query designer support.

Section Report

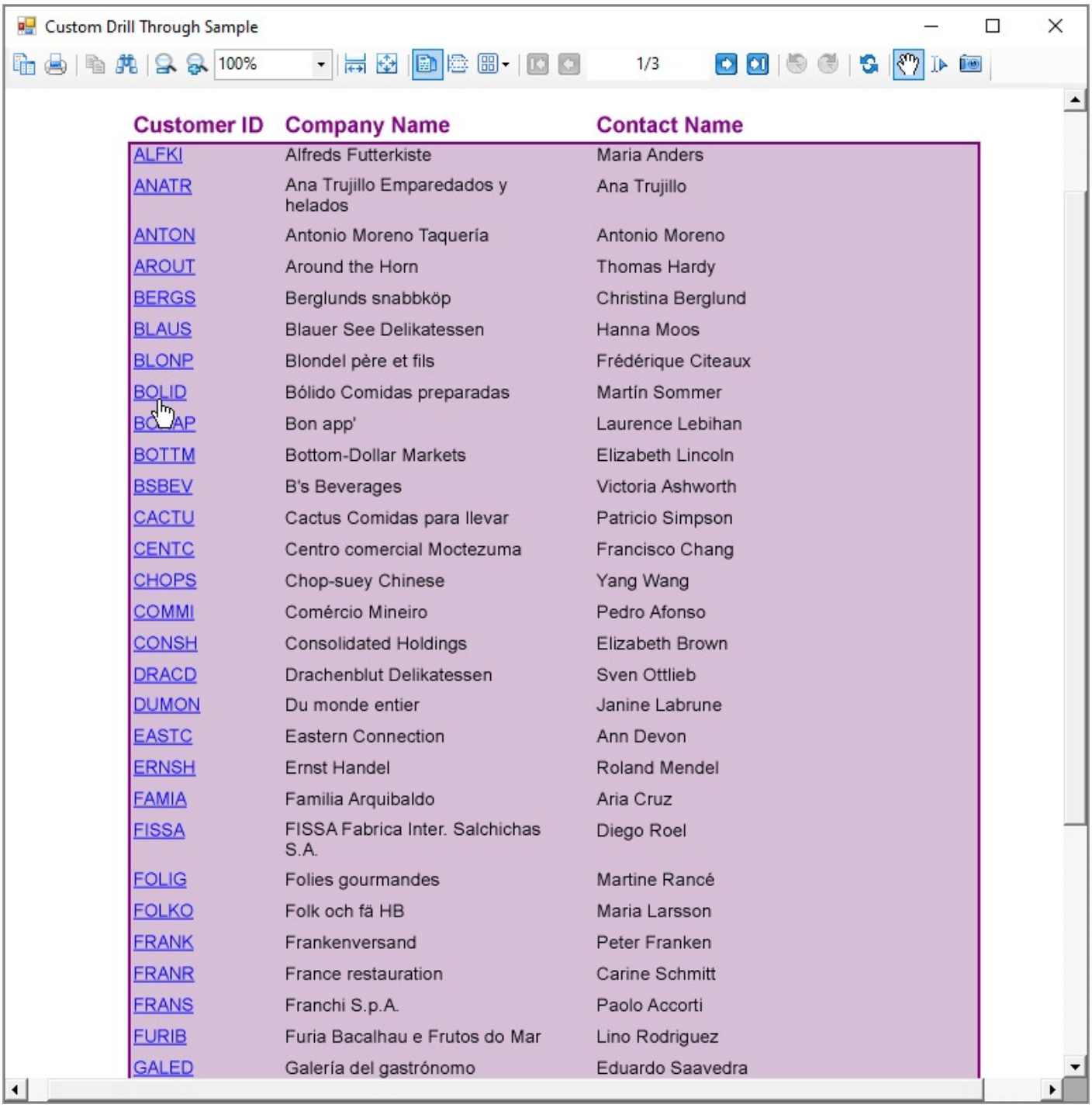
This section discusses following samples describing various features in Section Reports:

CustomDrillThrough
: Demonstrates using hyperlinks and the viewer hyperlink event to simulate drill-down from one report to another.

CustomWordExport
: Demonstrates exporting Section Report to Word format using third-party assemblies.

Custom Drill Through

The Custom Drill Through sample consists of three reports and a ViewerForm. The reports use the Hyperlink event of the Viewer control to pass a value from the Hyperlink property of a TextBox control to a Parameter value in a more detailed report.



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

Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/Section/CustomDrillThrough/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/Section/CustomDrillThrough/C#

Details

When you run this sample, a report displaying bound fields with a link created on CustomerID is displayed in a Viewer control. DrillThrough feature allows users to navigate to another report containing detailed data. Clicking the CustomerID hyperlink takes you to the second report which displays detailed information of the selected CustomerID. On further clicking the OrderID hyperlink the third report displaying the details of the selected order is opened in the Viewer. This feature enables the users to systematically go through the detailed data of the desired CustomerID.

> **Note:** To run this sample, you must have Nwind.db downloaded from GitHub in ..*Samples18**Data**NWIND.db*.

The sample consists of:

ViewerForm: This form contains only the Viewer control. Right-click the form and select **View Code** to see the code that allows multiple ViewerForms to display for the reports, 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 **DrillThrough1**, or collects a numeric value and passes it to the orderID Parameter of the report **DrillThrough2**. This code then runs the report with the parameter value and displays it in another instance of the ViewerForm.

DrillThroughMainReport: 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.

- Detail section: The Detail section has the **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. Right click on the form, select **View code** to 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.

- 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.

  > **Note:** This hyperlink does not work in Preview mode, because it relies on code in the ViewerForm to pass the value to DrillThrough1 report's parameter.

DrillThrough1 Report: This report looks similar to the DrillThroughMain 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, the **RepeatStyle** property is set to **OnPage**. This section consists of five label controls to serve as column headers for the Detail section and a CrossSectionBox control.

- Detail section: Right click on the form, select **View code** to see the parameter in the SQL Query that collects the 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.

- 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 it is related GroupHeader section is in use.

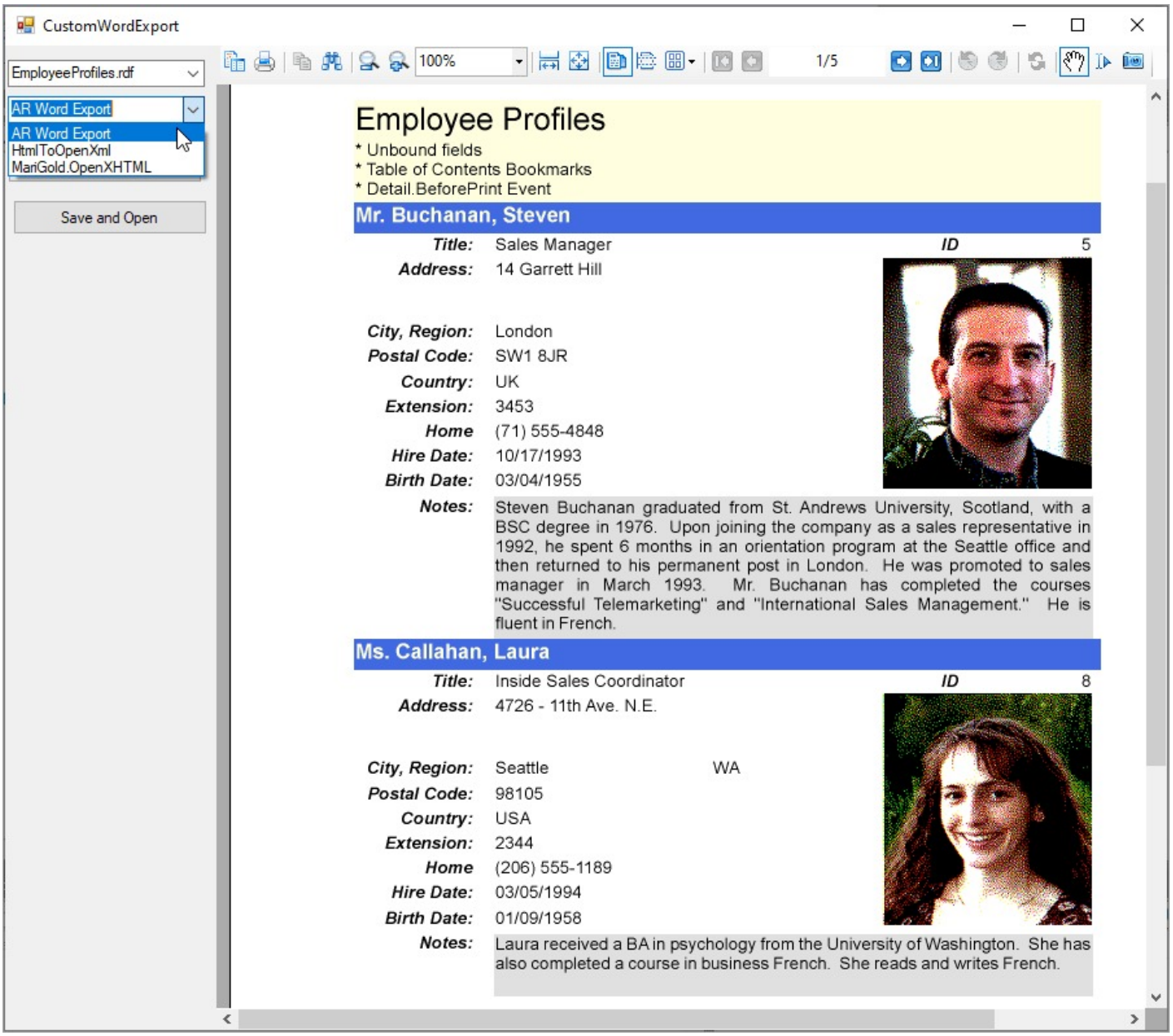
DrillThrough2 Report: Like DrillThrough1, this report has a parameter in a SQL Query, but unlike the other two reports, this one has no hyperlink. It displays order details for the OrderID value passed into it from the clicked hyperlink in DrillThrough1.

- 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: Right click on the form, select **View code** to see the parameter in the SQL Query that collects the value from the ViewerForm.

Custom Word Export

This sample shows how to export Section Report to Word (DOCX) format using third party assemblies.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Advanced/Section/CustomWordExport/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Advanced/Section/CustomWordExport/C#

Details

When you run this sample, a Windows Form is displayed that prompts you to select the report type and the Word export format type. Please note that we are not going to support any third party assemblies or projects. This sample demonstrates the idea on creating custom exports to satisfy different requirements.

API

The samples in the API folder describe creating reports and implementing various features through code in Page, RDLX, and Section reports.

- Page and RDLX Reports
- Section Report

Page and RDLX Reports

This section contains:

CreateReport
: This sample demonstrates how to create a Page Report layout in code. It further shows creating a table control, adding table rows and table columns inside it, adding cells inside the table rows and columns and adding text boxes inside the cells.

DigitalSignaturePro
: This sample demonstrates how to add digital signatures when exporting to PDF format.

Export
: This sample demonstrates how to export Page and RDLX reports to different export formats.

Font Resolver
: This sample demonstrates how to resolve custom fonts in preview and export.

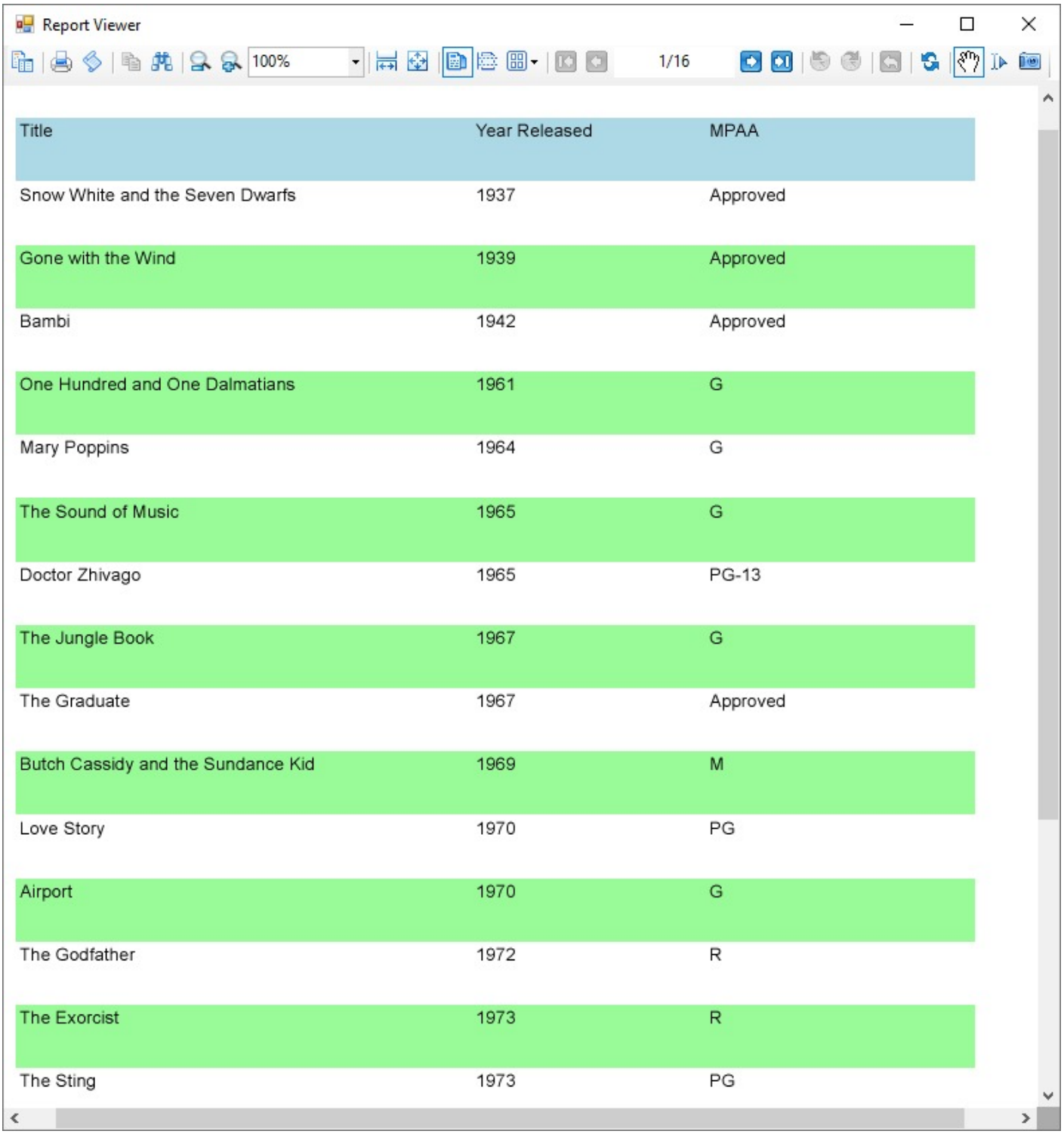
Layer
: This sample demonstrates how to use Layers in a report.

ReportWizard
: This sample demonstrates how to create a custom Report Wizard that allows you to select a report from the list of multiple reports and then allows you to select the data that you want to display in the selected report.

Stylesheets
: This sample demonstrates how to work with embedded and external style sheets in Page and RDLX reports.

Create Report

The Create Report sample demonstrates how to create a Page Report using code and display it in the ActiveReports Viewer.



Title	Year Released	MPAA
Snow White and the Seven Dwarfs	1937	Approved
Gone with the Wind	1939	Approved
Bambi	1942	Approved
One Hundred and One Dalmatians	1961	G
Mary Poppins	1964	G
The Sound of Music	1965	G
Doctor Zhivago	1965	PG-13
The Jungle Book	1967	G
The Graduate	1967	Approved
Butch Cassidy and the Sundance Kid	1969	M
Love Story	1970	PG
Airport	1970	G
The Godfather	1972	R
The Exorcist	1973	R
The Sting	1973	PG

Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/CreateReport/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/CreateReport/C#

Details

When you run this sample, the ActiveReports Viewer appears with a Page Report that is bound to a database.

> **Note:** To run this sample, you must have access to the **Reels.db**. The Reels.db file can be downloaded from GitHub.

The sample consists of:

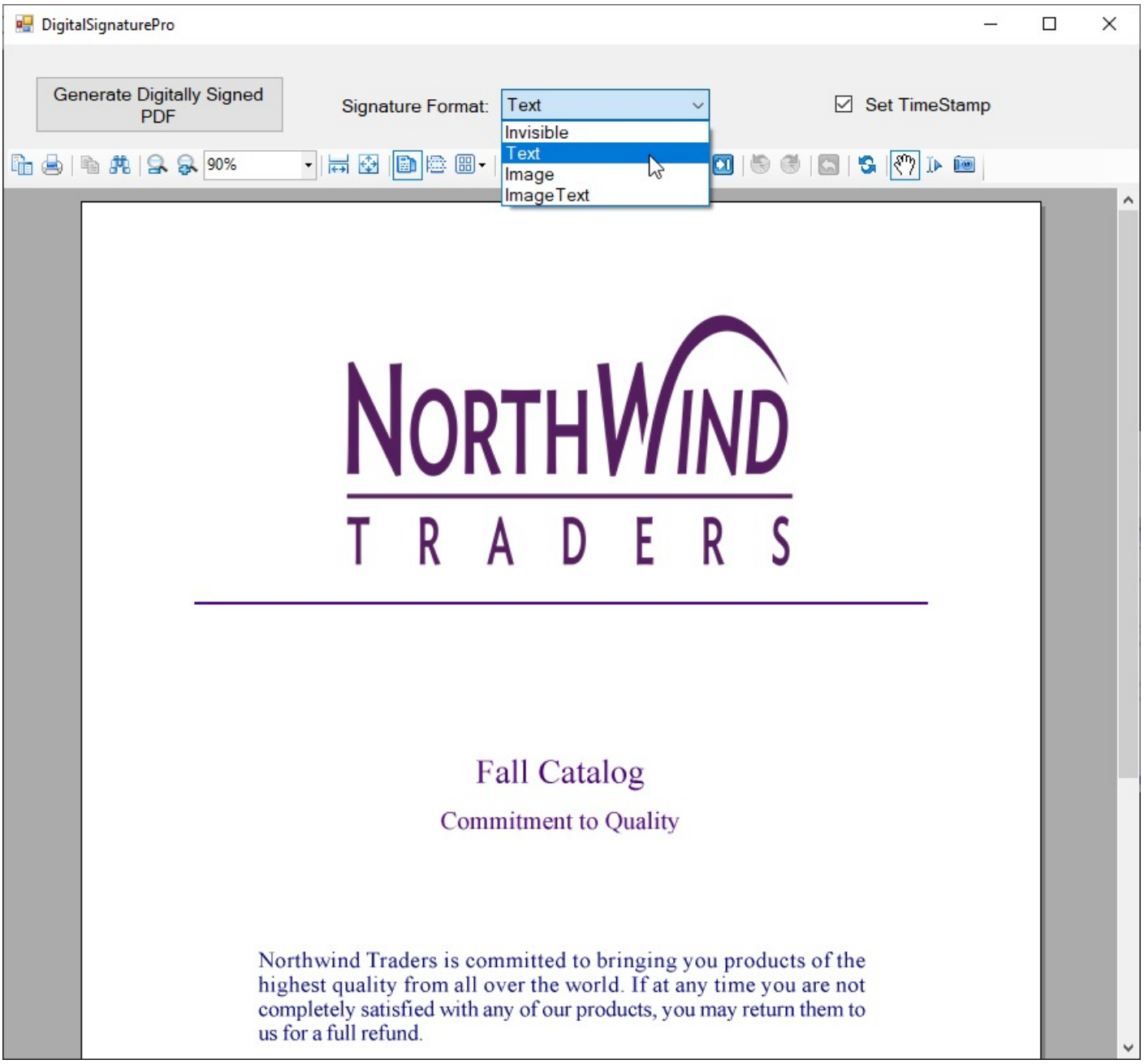
ReportsForm: This is the main form of the sample that contains the Viewer, the Sidebar, and Toolbox controls, used to create the ActiveReports Viewer at run time. Right-click the form and select View Code to see how to set up the Viewer. It also contains code that loads a layout created in the LayoutBuilder class to a Page Report object; then loads the Page Report object to a stream, which is loaded to the Viewer.

Constants: This file is an internal class that contains string values that are required for creating a dataset of the report.

LayoutBuilder: This file is an internal class that contains code for creating a Page Report layout and adding a data source and a dataset to it.

Digital Signature Pro

This sample demonstrates how you can digitally sign or set time stamp for a Page/RDLX Report when exporting it to PDF format using the PDF Rendering Extension.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/DigitalSignaturePro/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/DigitalSignaturePro/C#

Details

When you run this sample, the Invoice report is displayed in the Viewer control.

Clicking the **Generate Digitally Signed PDF** button in the Viewer toolbar creates a PDF file with a time stamp or digital signatures, based on the settings you have specified in the Viewer toolbar. You can change the content of signatures in the **Signature Format** box and you can add the time stamp to the generated pdf file by checking the **Set TimeStamp** checkbox, in the Viewer toolbar.

When you click the **Generate Digitally Signed PDF** button, a dialog for saving the destination file appears. After you indicate the location for a new PDF file, the PDF report file is created. Digital signature certificates dynamically reference and use GrapeCity.pfx, included in the project. Also, digital signatures dynamically load and use the gc.bmp file that you can find in the Image folder of this sample project.

> **Note:** To run this sample, you must have Nwind.db downloaded from GitHub in ..*Samples18**Data**NWIND.db*.

- Image folder: This folder stores the gc.bmp file with the company logo that digital signatures dynamically load and use.
- Catalog report: It shows a multi page layout spread over four pages in the report. The layout in Page1 and Page2 contains Image, Label and Textbox controls to display introductory text. The layout on Page3 contains a List data region with TextBox controls and a Table to display product details for each product category. The layout on Page4 uses Textbox, Shape and Line controls amongst others to create an Order Form, which a user is to fill manually.
- GrapeCity.pfx: In order to create a digital signature, you must have a valid PKCS#12 certificate (*.pfx) file. 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 https://www.source-code.biz/snippets/vbasic/3.htm.
- PDFDigitalSignature form: This is the main form of the Sample that uses the ActiveReports **Viewer** control in the bottom section of the form, and a panel docked to the top contains the **Create Digitally Signed PDF** button, the **Signature Format** box with the drop-down list and the **Set TimeStamp** checkbox.

  Clicking the **Generate Digitally Signed PDF** button opens a dialog for saving the destination file. After you indicate the location for a new PDF file, the PDF report file is created.

  The drop-down list of the **Signature Format** box contains the following options.

  - Invisible - the invisible pdf digital signature.
  - Text - the pdf digital signature that contains text only.
  - Image - the pdf digital signature that contains graphics only.
  - ImageText - the pdf digital signature that contains text and graphics.

  Checking the **Set TimeStamp** checkbox allows you to add the time stamp to the signature of the generated pdf file. The time stamp contains the Time Stamp Server address, its login and password information.

  Right-click **PDFDigitalSignature** in the Solution Explorer and select **View Code** to see the code implementation for the pdf digital signature options.

- Resource.resx: This file contains the string for the message box that appears after the PDF file is generated and the string for the message box that appears when the free service limitation is exceeded.
- PAdES support. You can use the **SignatureFormat ('SignatureFormat Property' in the on-line documentation)** setting, allowing to use PAdES (PDF Advanced Electronic Signatures), available in PDF2.0 (ISO 32000-2).

  Visual Basic.NET code

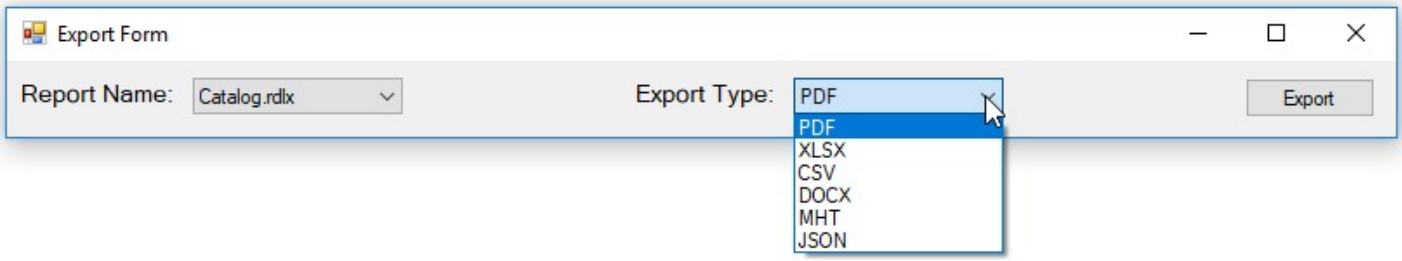
```
settings.SignatureFormat =
GrapeCity.ActiveReports.Export.Pdf.Section.Signing.SignatureFormat.ETSI_CAdES_detached
```

C# code

```
settings.SignatureFormat =
GrapeCity.ActiveReports.Export.Pdf.Section.Signing.SignatureFormat.ETSI_CAdES_detached;
```

Export

This sample illustrates how to export to different export formats using code. The available export formats are PDF, XLSX, CSV, DOCX, MHT and JSON.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/Export/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/Export/C#

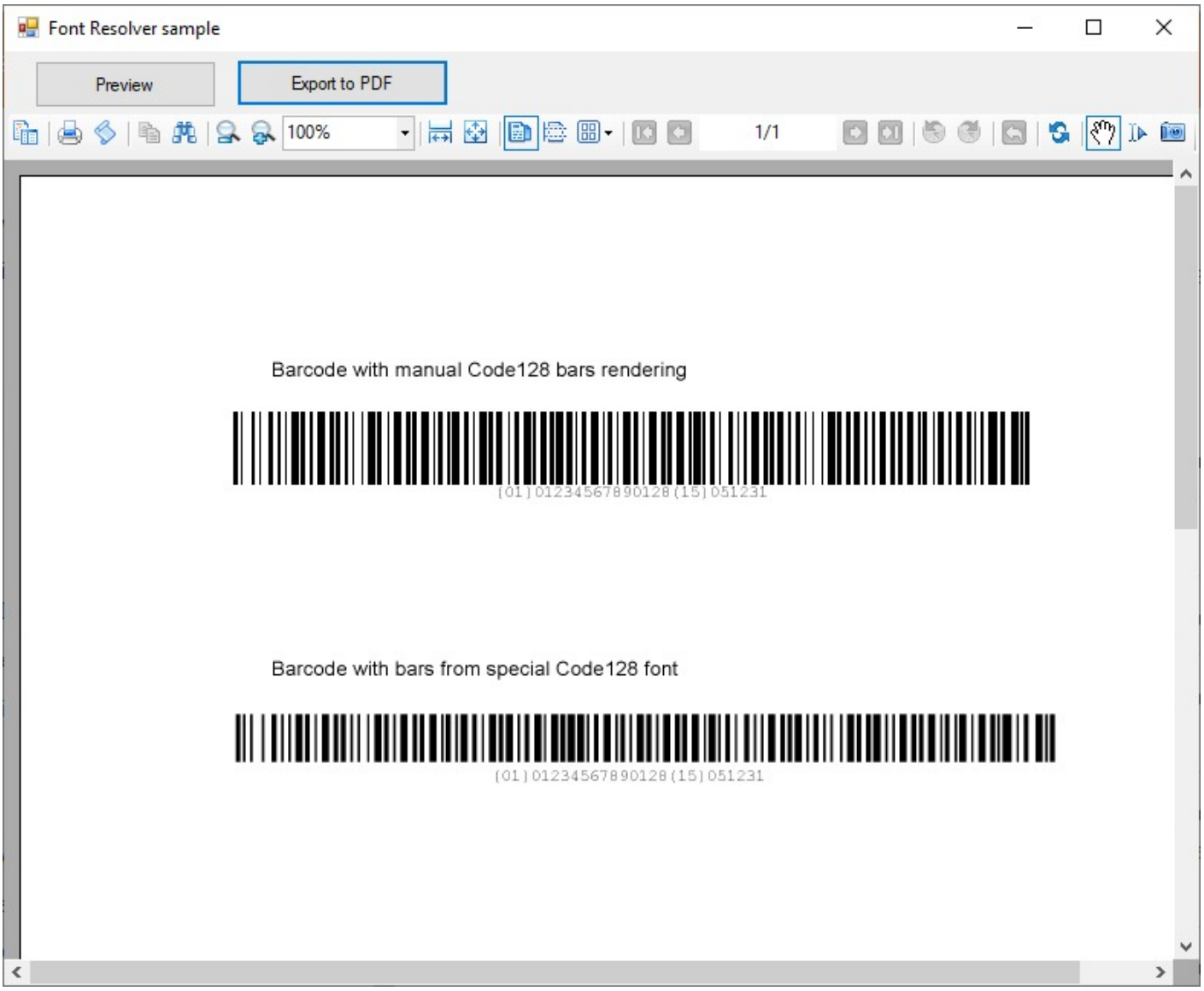
Details

When you run this sample, the Export Form is displayed. The Export Form contains the **Report Name** and **Export Type** combo boxes along with the **Export** button.

The sample consists of **ExportForm**. It exports the selected report in the selected export format.
On running the application, select a report for export in the **Report Name** combo box. In the **Export Type** combo box, you can select one of the following formats for export - PDF, XLSX, CSV, DOCX, MHT and JSON. Clicking the **Export** button opens the **Save As** dialog where you can specify the name of the exported file. By default, the exported file is saved in the **Documents** folder.

Font Resolver

This sample demonstrates how use custom font resolver to configure fonts for preview and export on all platforms without installation them.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/FontResolver/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/FontResolver/C#

Details

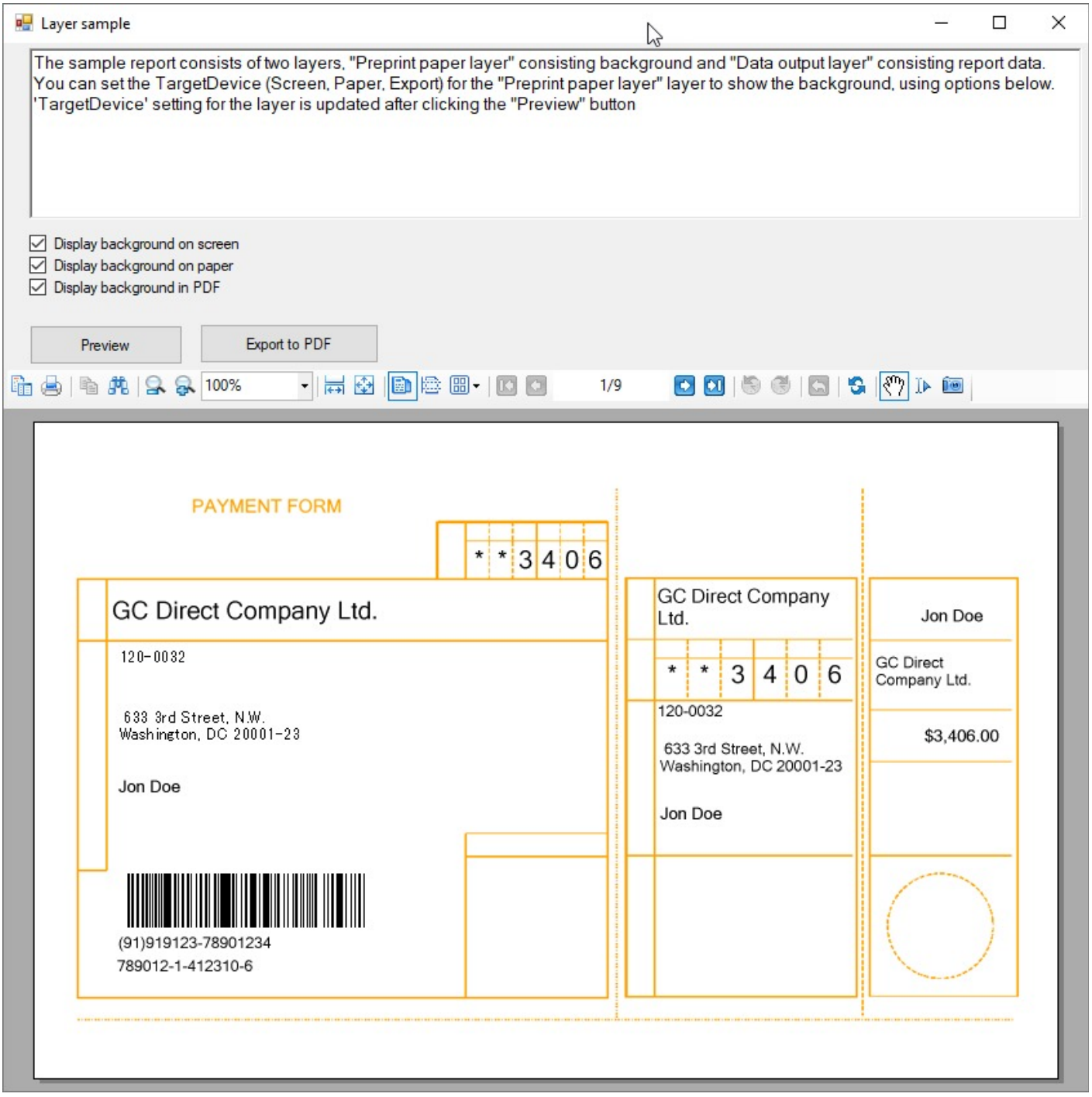
When you run this sample, the WinForms Viewer appears. You need to click the **Preview** button to load an RDLX report with resolved fonts (when the font is not installed). The Export to PDF button exports the report with resolved fonts.

The same code works for the Page or Section report too.

The sample consists of **MainForm** which contains code to resolve windows and barcode fonts.

Layers

The Layers sample demonstrates how to work with Layers.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/Layers/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/Layers/C#

Details

The sample report consists of two layers, **Preprint paper layer** and **Data output layer**. The **Preprint paper layer** contains the report background and the **Data output layer** contains report data.

For the **Preprint paper layer**, you can set the TargetDevice (Screen, Paper, Export) to show the background, using the options in the checkboxes.
The TargetDevice setting for the layer is updated once you click the **Preview** button.

See Layers for further details.

LayersForm

This is the main form that appears when you run the Layers sample.

Right-click the form and select **View Code** to see how to set up the report designer. It also contains code to preview the Layer report and set the target device for the layer to the report, and export the report displayed in the Viewer to PDF.

Layer.rdlx

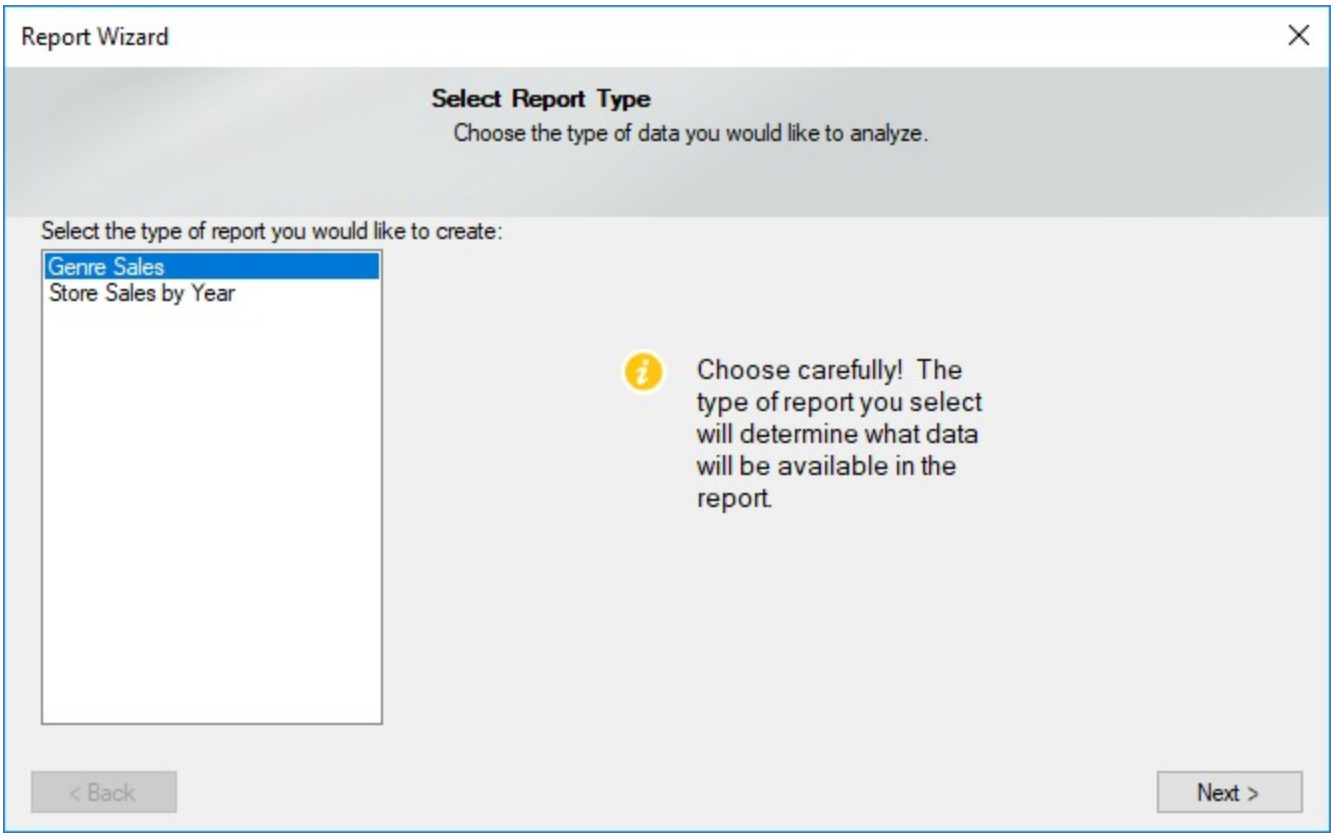
Layer.rdlx: This report uses the BILL.db data source. The TextBox controls are used to display the Payment Form fields.

This report is contains two Layers with the following controls and TargetDevice settings.

Layer Name	Target Device	Description
default	All	This is a default layer that contains all report data.
PrePrintPaper	All	Contains the report background.

Report Wizard

The Report Wizard sample demonstrates how to create and customize a report, using the report wizard.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/ReportWizard/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/ReportWizard/C#

Details

When you run this sample, the form with the Report Wizard appears. On the **Select Report Type** page, select the report type you want to analyze and click the **Next** button.

On the next **Choose grouping options** page of the wizard that appears, you are asked to choose a field for grouping the report data and click the **Next** button. You can also enable the checkbox at the bottom of the page if you like to include the last detail of the report as separate group. If you leave it unchecked, the checkbox automatically adds the last detail to a previous group.

On the next **Select output fields** page of the wizard that appears, you are asked to choose the fields you want to display in your report and click the **Next** button. On the next **Summarization and Review** page, you can review the settings you have selected previously. You can also set the summary options if you want to display a grand total or a sub-total in the report. Finally, when you click the **Finish** button, the **Unified ReportDesigner** appears and displays the report.

The sample consists of following items:

MetaData folder: This folder contains two internal classes, **FieldMetaData** and **ReportMetaData**. The FieldMetaData

class contains information on the fields used in the report string values. This file provides this information when required by the application. Similarly, the ReportMetaData class contains information on the reports used in this sample. This file provides this information when required by the application.

Resources folder: This folder contains images and icons used by the Report Wizard API.

UI folder:

- WizardSteps: This folder contains templates of the Report Wizard pages, which get displayed inside the Panel control on the WizardForm at run time.
- DesignerForm: This form appears when you click the **Finish** button on the last page of the Report Wizard. This form uses the ToolStripPanel, ToolStripContentPanel, Designer, Toolbox, ReportExplorer and a PropertyGrid controls to create the Unified ReportDesigner.
- Right-click the form and select **View Code** to see how to set the designer and create a blank Page Report. It also contains code that attaches the Toolbox, ReportExplorer and PropertyGrid controls to the Designer, inserts DropDown items to the ToolStripDropDownItem, sets their functions, and checks for any modifications that have been made to the report in the designer.
- DragDropListBox: This file contains code to override methods that enable and handle the drag-and-drop feature in the ListBox, which appears on the wizard page where you select the grouping and the output fields. Thus, instead of selecting a field and clicking the **Add** button in the ListBox control, you can directly drag fields as well.
- TipControl: This file contains the design of the Tip that appears on the first page of the report wizard.
- WizardDialog: This is the main form that appears when you run the sample. It uses the PictureBox, two Labels, Panel and two Button controls to create the Report Wizard. The PictureBox displays the logo while the two Label controls display the Page Title and Page Description respectively. The Panel control loads the design of different pages of the Report Wizard. The two Button controls handle the last page and next page functions. Right-click the form and select **View Code** to see how to define functions of different controls used on the form.

Constants: This is an internal class that contains fields in string values that can be summarized.

GenreSales.rdlx-master: This is the master report for the **Genre Sales** report. It uses the Image, two Textbox and ContentPlaceholder controls to design the report. The Image control displays the logo on the top of the report while the two Textbox controls display the report execution time and page number information respectively. The ContentPlaceholder control displays its content report.

LayoutBuilder: This is the internal class file that contains code for creating the layout of both child reports and loading data in it.

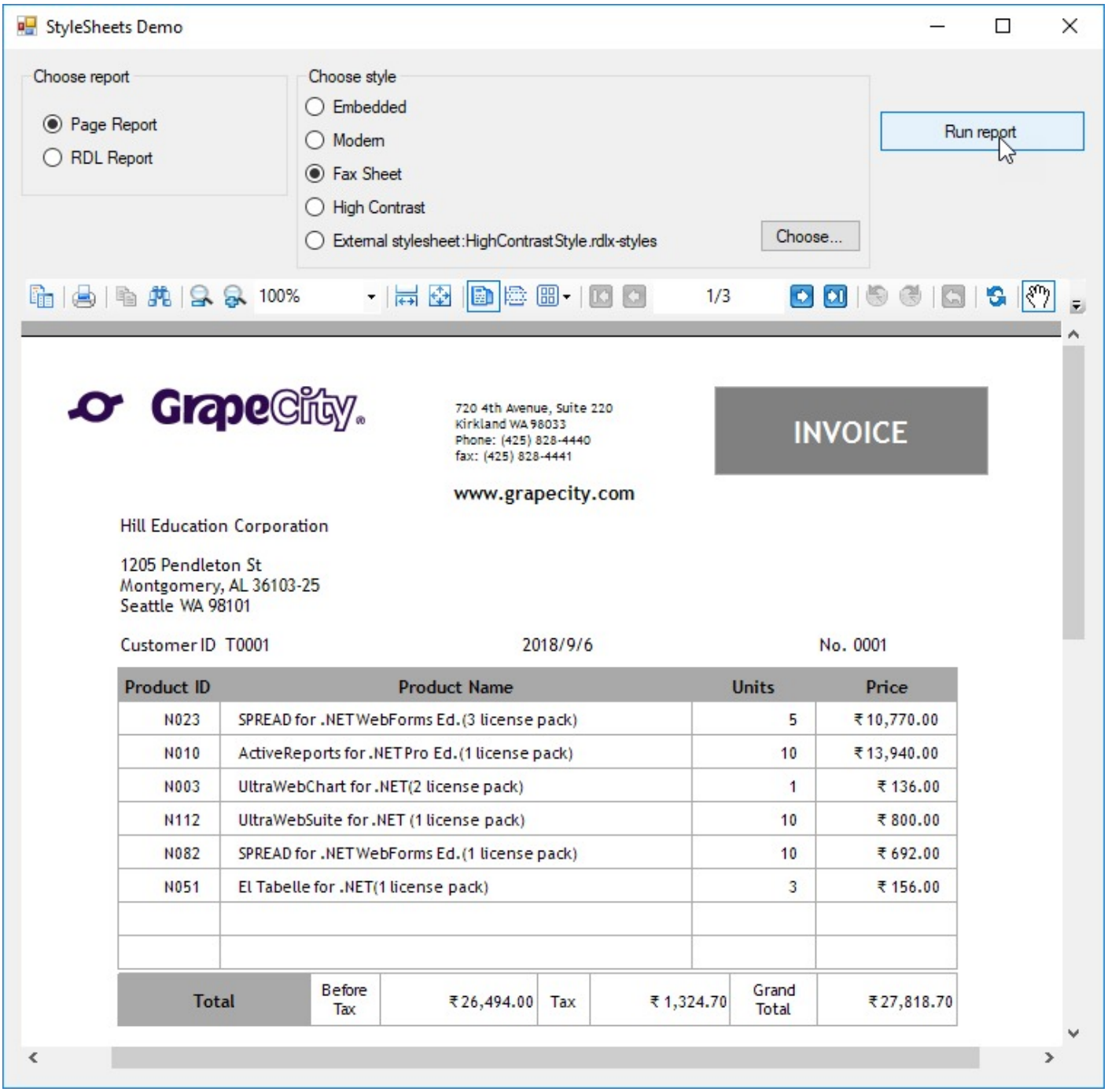
Reports.xml: This XML file is used as a database to provide data for the reports in this sample.

ReportWizardState: This is the internal class file that contains code for handling the UI of the Report Wizard.

StoreSales.rdlx-master: This is the master report for the **Store Sales** report. It uses the Image, two Textbox and ContentPlaceholder controls to design the report. The Image control displays the logo on the top of the report while the two Textbox controls display the report execution time and page number information respectively. The ContentPlaceholder control displays its content report.

Style Sheets

This sample demonstrates how to work with embedded and external style sheets in Page and RDLX reports.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/Stylesheets/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/PageAndRDLX/Stylesheets/C#

Details

When you run this sample, the main **StyleSheetsForm** interface displaying the following options appears:

1. **Choose Report:** Select the type of report you want to display in the Viewer.
2. **Choose Style:** Select a style sheet; embedded or external to apply to the report.

Click on **Run Report** button to load the report with the selected options in the Viewer. See Styles for more information about style sheets.
StyleSheetsForm: This is the main form of the sample that displays reports in the ActiveReports **Viewer** control in the bottom section of the form, and options such as Choose report, Choose style and Run report button at the top of the form.

DeliverySlip.rdlx: This is a Page Report, that contains TextBox, Label, Container controls and two Table data regions to display the invoice information. The report uses Theme1.rdlx-theme file and BaseStyle as an embedded style sheet. Some TextBox controls on the report also use the Sum function to display the total price information for each invoice. This report uses the **Seikyu2** shared data source.

ReorderList.rdlx: This is an RDLX report, that contains Table data region to display data from **Reels** shared data source. The Reels logo in the report is embedded within the Reels.rdlx-theme.

External Stylesheet: Following external style sheets are provided in this folder:

External Stylesheet Location - ..\Samples18\API\PageAndRDL\Stylesheets\Reports

- BaseStyle.rdlx-styles
- FaxSheetStyle.rdlx-styles
- HighContrastStyle.rdlx-styles
- ModernStyle.rdlx-styles

Section Report

This section discusses following samples:

Charting
: This sample demonstrates chart types used in different scenarios, in both bound and unbound modes.

Cross Section Controls
: This sample demonstrates the use of the cross section lines and boxes.

Cross Tab Report
: This sample demonstrates using unbound data, conditional highlighting and distributing data across columns to create a cross-tab view and data aggregation.

Custom Annotation
: This sample demonstrates adding the Custom Annotation button to the report Viewer toolbar and adding a new annotation to the report.

Digital Signature Pro
: This sample demonstrates how to sign digitally or set time stamp for a Section Report when exporting it to PDF format.

Export
: This sample demonstrates how to export to different export formats using code.

Inheritance
: This sample demonstrates using the method that inherits a report at run time and design time.

Print Multiple Pages per Sheet
: This sample demonstrates printing a document with multiple pages per sheet by using the common PrintDocument class of the NET.Framework.

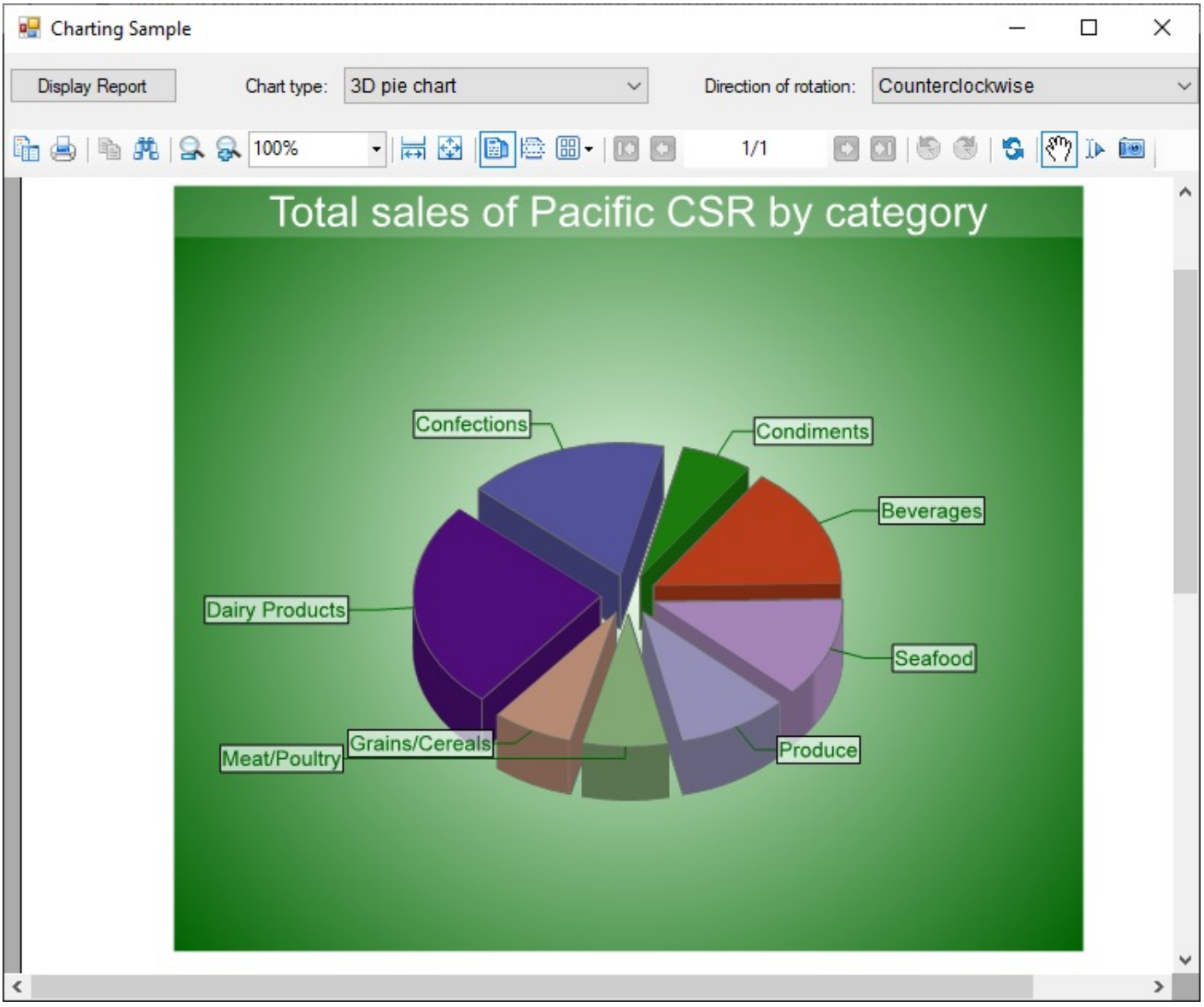
Style Sheets
: This sample demonstrates changing styles at run time to provide a different look to a same report.

Sub Report
: This sample demonstrates using subreports in an ActiveReports report.

Summary
: This sample demonstrates how to display summarized data in a Section Report.

Charting

The Charting sample provides an option to choose from various chart types and a button to display the selected chart in a Viewer control.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/Charting/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/Charting/C#

Details

Chart type combobox

Select from the following ChartType options.

- 2D Bar Chart - Use this chart to compare values of items across categories.
- 3D Pie Chart - Use this chart to display data in 3D format to depict how percentage of each data item contributes to a total percentage. Selecting this ChartType provides an option to set the Direction of rotation of 3D Pie chart to Clockwise or Counterclockwise.
- 3D Bar Chart - Use this chart to compare values of items across categories and to display the data in a 3D format.
- Finance Chart -Use this chart to display the 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.
- Stacked Area Chart - Use this chart to demonstrate how each value contributes to the total.

Report Display button
Click this button to display the selected chart type in a Viewer control.

> **Note:** To run rpt2DBar, rpt3DPie and rpt3DBar report, you must have access to the Nwind database. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

ViewerForm
The ViewerForm contains the **Viewer** control, with the **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 code used to run the report and display it in the viewer.

rpt2DBar report
Displays bar chart on a report. Retrieves the data to be displayed in a chart from Orders table in **Nwind.db** database. Settings for chart data source can be done using the **Chart Data Source** dialog.

rpt3DBar report
Displays 3D bar chart on a report. Retrieves the data to be displayed in a chart from Orders table in **Nwind.db** database. Generates a DataSet for the chart in ReportStart event and sets it in DataSource property of Chart control.

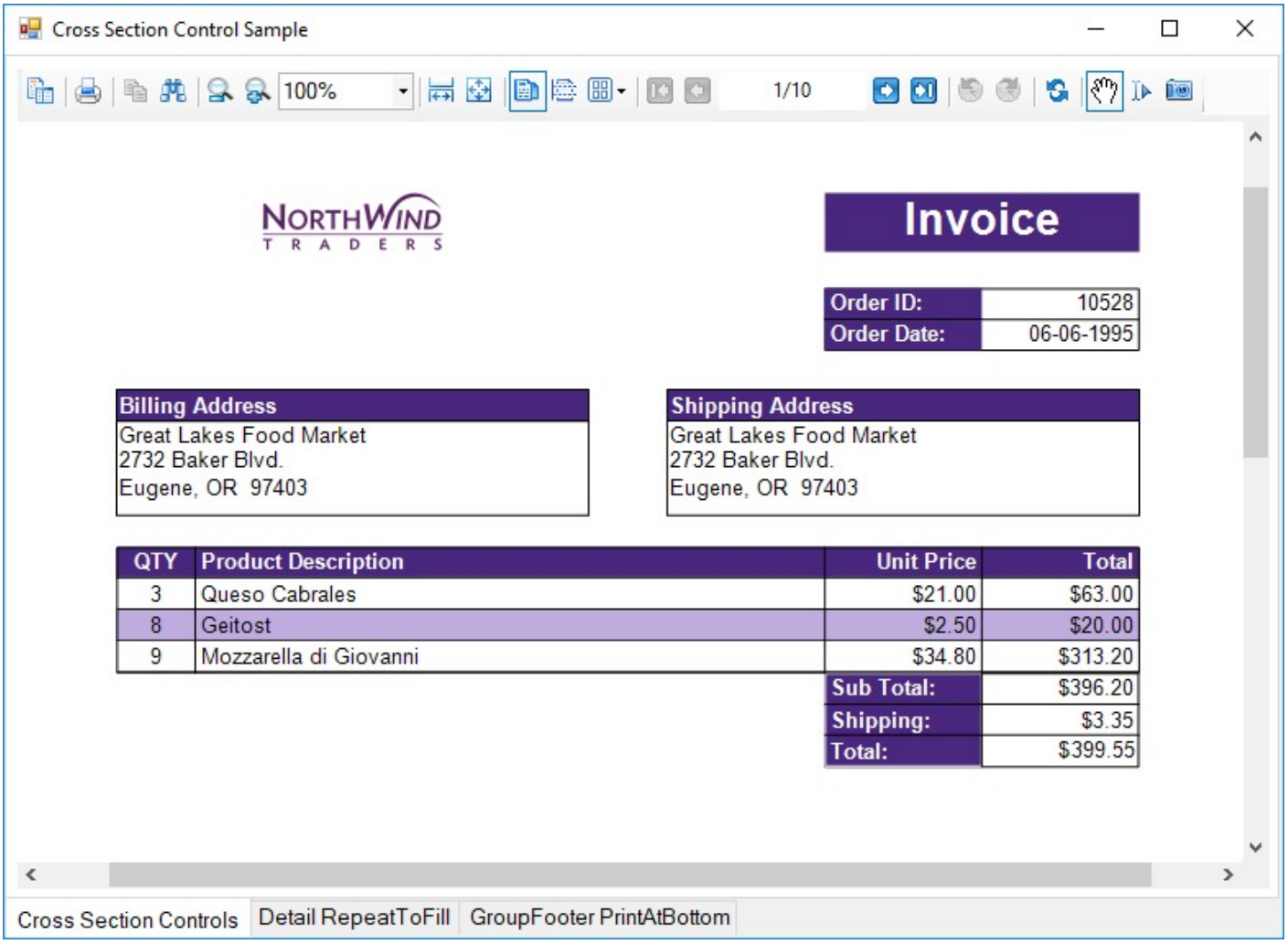
rpt3DPie report
Displays 3D pie chart on a report. Retrieves the data to be displayed in a chart from each of the Employees, Categories, Products, Orders, Order Details tables in **Nwind.db** database. Generates a DataTable for the chart in a ReportStart event and sets it in DataSource property of Chart control. Rotational direction of 3D pie chart can be set to **Clockwise** or **Counterclockwise**.

rptCandle report
Displays candle chart on a report. Chart data is set at design time using **DataPoint Collection Editor**. DataSource property is not used for this chart.

rptStackedArea report
Displays stacked area chart on a report. Chart data is set at design time using **DataPoint Collection Editor**. DataSource property is not used for this chart.

Cross Section Controls

This CrossSectionControls sample displays the invoice report of a company in detail. This sample uses the CrossSectionBox and CrossSectionLine controls to demonstrate the lines and display the Invoice report in a tabular form. It includes a **ViewerForm** with three tabs, three **Viewer** controls and an **Invoice** report to highlight several report features. Run the project and click the tab to see these features in action.



Sample

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/CrossSectionControls/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/CrossSectionControls/C#

Details

When you run this sample, a form with three different tab options and each tab option displaying a report in a Viewer control is displayed. Click any tab option at the bottom of the form to display the selected tab feature applied to the

report in a Viewer control. Select from the following tab options.

Cross Section Controls
Draws table style gridlines easily through multiple sections without any gap.

Detail RepeatToFill
The **Detail RepeatToFill** tab has the **RepeatToFill** property set to **True**. This ensures that the formatting (alternating purple and white rows and CrossSection controls) fills space as needed to keep the same layout between pages.

GroupFooter PrintAtBottom
The GroupFooter section has the PrintAtBottom property set to **True**. This pulls the GroupFooter section to the bottom of the page, just above the PageFooter section.

> **Note:** To run this sample, you must have access to the **Nwind.db**. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

- ViewerForm: The ViewerForm has three tabs-**Cross Section Controls, Detail RepeatToFill, GroupFooter PrintAtBottom**, 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.
- Invoice Report: The Invoice report demonstrates the usage of the following features.
  - PageHeader Section: This section contains Shape, Label and TextBox controls. The **Shape** control provides a border around the **Order ID** and **Order Date** fields and labels. The orderDateTextBox 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.
  - customerGroupHeader: The 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 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 CrossSectionBox or CrossSectionLine 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 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 customerGroupHeader section is set to the **OrderID** field, so that the section (followed by related details and GroupFooter) prints once per order.

  - Detail section: This section contains four bound TextBox controls. The 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 TextBox controls 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 colored bar (in this case, purple bar) report by alternating the **BackColor** property of the section. Click the Data Source Icon on the **Detail**

band to view the **Connection String** used in the report.

- customerGroupFooter section: This section has the **NewPage** property set to **After** so that a new page is printed for each OrderID (the associated GroupHeader's DataField). The Subtotal TextBox uses the following properties.
  - The **DataField** property uses a **Total** 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 **customerGroupHeader**, 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**.

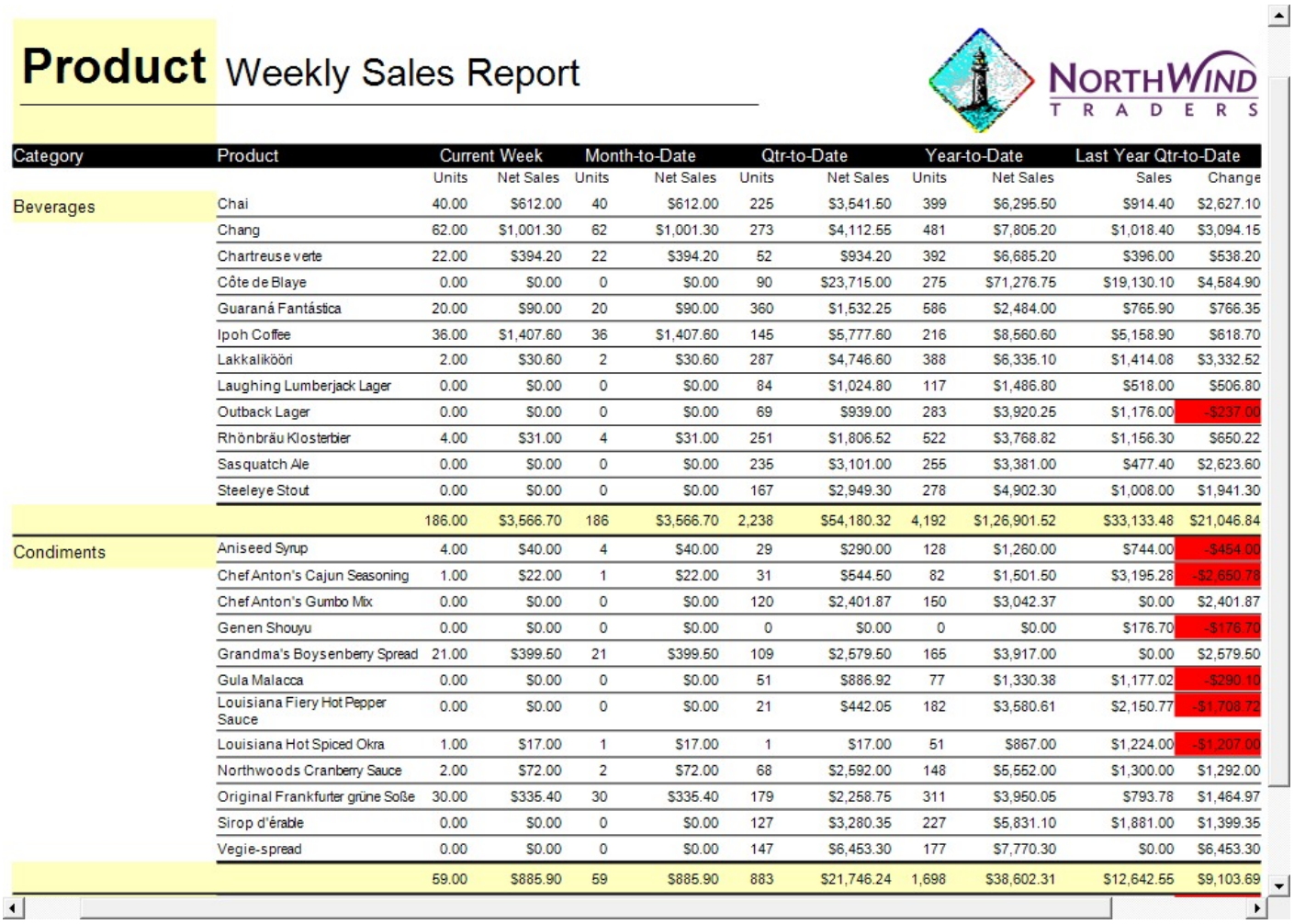
  Right-click the report and select **View Code** to see the code used in the **customerGroupFooter Format** event to calculate the value for the Grand Total TextBox, and to format it as currency.

- 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.

> **Tip:** In order 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.

Cross Tab Report

The CrossTabReport sample displays hierarchical information in a **cross tabular** structure. This sample consists of a StartForm with an ActiveReports Viewer control and a ProductWeeklySales report.



Product Weekly Sales Report

NorthWind Traders

Category	Product	Current Week		Month-to-Date		Qtr-to-Date		Year-to-Date		Last Year Qtr-to-Date	
		Units	Net Sales	Units	Net Sales	Units	Net Sales	Units	Net Sales	Sales	Change
Beverages	Chai	40.00	$612.00	40	$612.00	225	$3,541.50	399	$6,295.50	$914.40	$2,627.10
	Chang	62.00	$1,001.30	62	$1,001.30	273	$4,112.55	481	$7,805.20	$1,018.40	$3,094.15
	Chartreuse verte	22.00	$394.20	22	$394.20	52	$934.20	392	$6,685.20	$396.00	$538.20
	Côte de Blaye	0.00	$0.00	0	$0.00	90	$23,715.00	275	$71,276.75	$19,130.10	$4,584.90
	Guaraná Fantástica	20.00	$90.00	20	$90.00	360	$1,532.25	586	$2,484.00	$765.90	$766.35
	Ipoh Coffee	36.00	$1,407.60	36	$1,407.60	145	$5,777.60	216	$8,560.60	$5,158.90	$618.70
	Lakkalikööri	2.00	$30.60	2	$30.60	287	$4,746.60	388	$6,335.10	$1,414.08	$3,332.52
	Laughing Lumberjack Lager	0.00	$0.00	0	$0.00	84	$1,024.80	117	$1,486.80	$518.00	$506.80
	Outback Lager	0.00	$0.00	0	$0.00	69	$939.00	283	$3,920.25	$1,176.00	-$237.00
	Rhönbräu Klosterbier	4.00	$31.00	4	$31.00	251	$1,806.52	522	$3,768.82	$1,156.30	$650.22
	Sasquatch Ale	0.00	$0.00	0	$0.00	235	$3,101.00	255	$3,381.00	$477.40	$2,623.60
	Steeleye Stout	0.00	$0.00	0	$0.00	167	$2,949.30	278	$4,902.30	$1,008.00	$1,941.30
		186.00	$3,566.70	186	$3,566.70	2,238	$54,180.32	4,192	$1,26,901.52	$33,133.48	$21,046.84
Condiments	Aniseed Syrup	4.00	$40.00	4	$40.00	29	$290.00	128	$1,260.00	$744.00	-$454.00
	Chef Anton's Cajun Seasoning	1.00	$22.00	1	$22.00	31	$544.50	82	$1,501.50	$3,195.28	-$2,650.78
	Chef Anton's Gumbo Mix	0.00	$0.00	0	$0.00	120	$2,401.87	150	$3,042.37	$0.00	$2,401.87
	Genen Shouyu	0.00	$0.00	0	$0.00	0	$0.00	0	$0.00	$176.70	-$176.70
	Grandma's Boysenberry Spread	21.00	$399.50	21	$399.50	109	$2,579.50	165	$3,917.00	$0.00	$2,579.50
	Gula Malacca	0.00	$0.00	0	$0.00	51	$886.92	77	$1,330.38	$1,177.02	-$290.10
	Louisiana Fiery Hot Pepper Sauce	0.00	$0.00	0	$0.00	21	$442.05	182	$3,580.61	$2,150.77	-$1,708.72
	Louisiana Hot Spiced Okra	1.00	$17.00	1	$17.00	1	$17.00	51	$867.00	$1,224.00	-$1,207.00
	Northwoods Cranberry Sauce	2.00	$72.00	2	$72.00	68	$2,592.00	148	$5,552.00	$1,300.00	$1,292.00
	Original Frankfurter grüne Soße	30.00	$335.40	30	$335.40	179	$2,258.75	311	$3,950.05	$793.78	$1,464.97
	Sirop d'érable	0.00	$0.00	0	$0.00	127	$3,280.35	227	$5,831.10	$1,881.00	$1,399.35
	Vegie-spread	0.00	$0.00	0	$0.00	147	$6,453.30	177	$7,770.30	$0.00	$6,453.30
		59.00	$885.90	59	$885.90	883	$21,746.24	1,698	$38,602.31	$12,642.55	$9,103.69

Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/CrossTabReport/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/CrossTabReport/C#

Details

When you run the sample, a report displaying a list of weekly sales for current week, month, quarter or year for each product category is displayed in the Viewer control. The values highlighted in red represent negative values.

> **Note:** To run this sample, you must have access to the **Nwind.db**. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

StartForm

The Viewer control has the **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 Report:

ProductWeeklySales Report

This report features a number of accumulated values using summary function property settings and calculated values in the code.The summary function displays a list of summary values only for weekly sales total in group footer without displaying the Detail section. By setting the **Height** property of Detail section to **0** and/or hiding the Detail section by setting **Visible** property to **False**, you can create a summary report that only displays sections other than the Detail section.
Based on the order date of Detail section data, you can determine whether it will be included in a week, month, quarter, year or previous year's quarter and set the values by sorting in unbound fields. Value of each unbound field is automatically summarized by each Field placed in the group footer section.

- ReportHeader section: This section of the report features static controls including Labels, a Picture, a Line, and a Shape control. The report header prints only once on the first page of the report, thus report title, company information, and a logo are added in this section.
- PageHeader section: The page header section contains static Label controls that print at the top of each page and serve as column headers for the group header sections.
- ghCategory section: This group header section has the 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 section: Although this group header contains no controls and is hidden by setting the Height property to 0 and Visible property to False, it still performs two important functions. First, the DataField property is set to ProductName, to sort the data inside each category by product, and second, the related group footer section displays the bulk of the data for the report.
- Detail section: The detail section of this report is hidden by setting the Height property to 0 and Visible property to False, but it does contain four bound fields whose values are used in the code. In the Detail Format event, the value from the hidden txtDetProduct TextBox is collected and passed to the _sProductName variable. For more information on section events, see Section Report Events.
- gfProduct section: 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. 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** TextBox. The **Value** for the **txtPQTDChange** TextBox 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** TextBox is set to **Red** if the value is negative. 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 section: 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)

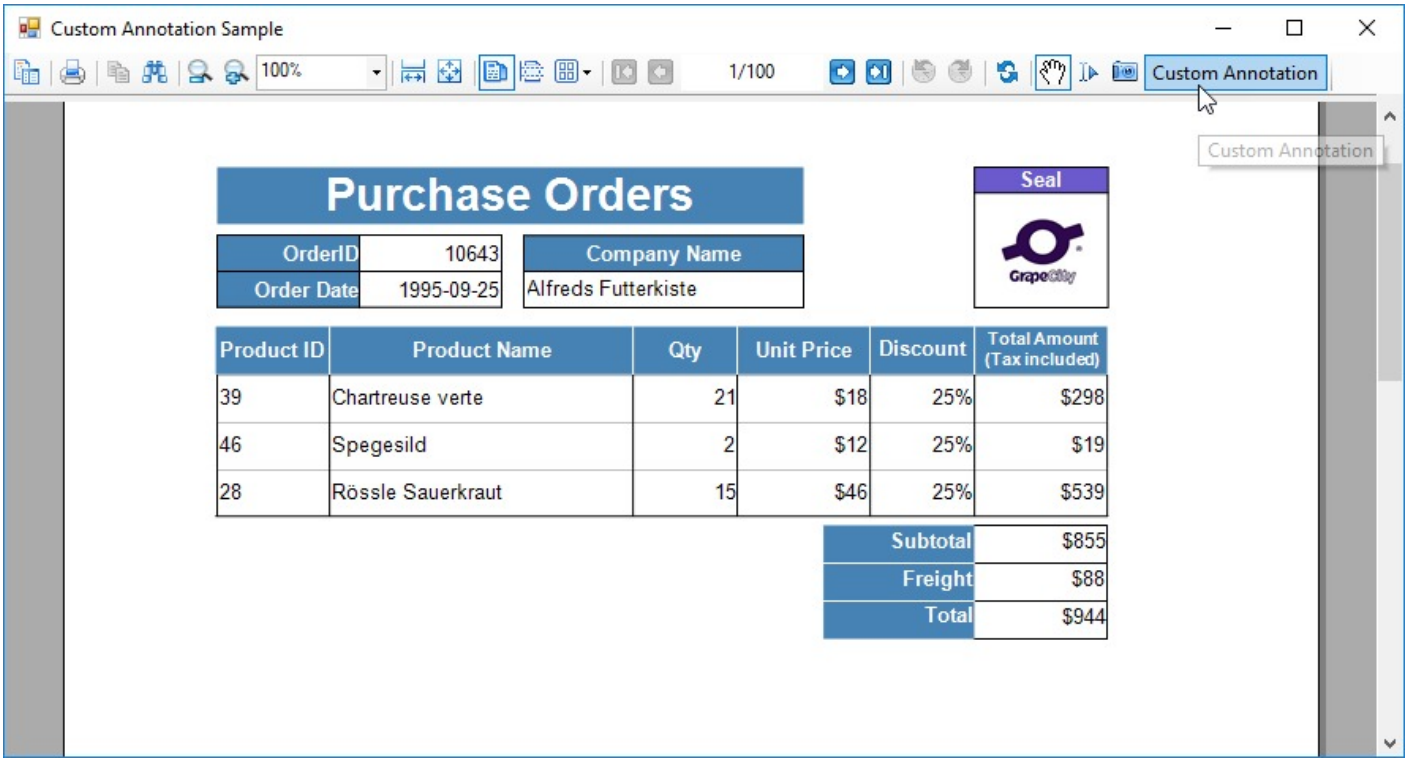
Adds values rather than counting or averaging them.
  - **SummaryGroup**: ghCategory
    Summarizes the values that fall within the current category group.
  - **SummaryRunning**: None (the default value)
    Ensures that this value is reset each time the category group changes.
  - **SummaryType**: SubTotal
    Summarizes the current group rather than a page or report total.

- PageFooter section: This section is not used, so it is hidden by setting the **Height** property to **0** and/or **Visible** property to **False**. Otherwise, it would print at the bottom of each page. The section cannot be deleted, because the related PageHeader section is in use.

- ReportFooter section: This section is not used, so it is hidden by setting the **Height** property to **0** and/or **Visible** property to **False**. 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 Annotation

The Custom Annotation sample demonstrates how to add the **Custom Annotation** button to the Viewer toolbar and depicts the method to add any annotation (seal image in this case) to the report. Only one annotation can be used per page.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/CustomAnnotation/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/CustomAnnotation/C#

Details

When you run the sample, the report appears in the Viewer control. The Viewer control toolbar contains the **Custom Annotation** button that opens the report annotation.

> **Note:** To run this sample, you must have access to the **Nwind.db**. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

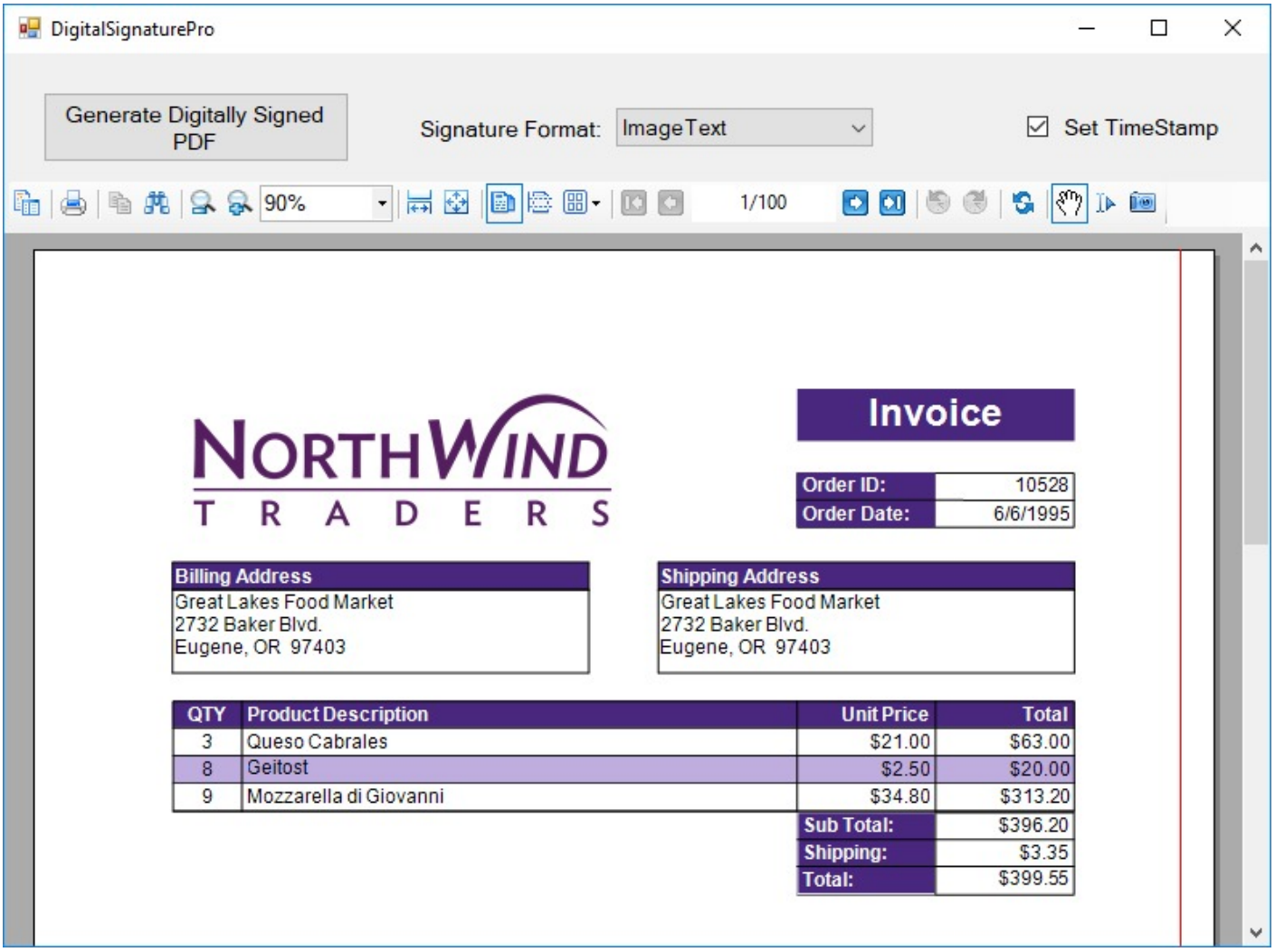
- AnnotationForm: Add **Custom Annotation** button in Form_Load event. The behavior on clicking the Custom Annotation button is mentioned in the description of the Click event.
- Annotation report:
  - **ghOrderID section**
    Product order receipt is grouped according to OrderID.
  - **Detail section**
    Use RepeatToFill property to output empty rows till the end and perform page break group wise.
  - **GFOrderID section**
    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 are summarized using the following properties:
    - **SummaryFunc:** Sum (the default value) adds values rather than counting or averaging them.

      > **Caution:** SummaryFunc has no effect unless the SummaryType property is set to either SubTotal or GrandTotal.

    - **SummaryGroup:** ghOrderID summarizes the values that fall within the current order id.
    - **SummaryRunning:** None (the default value) ensures that this value is reset each time the order id changes.
    - **SummaryType:** SubTotal summarizes the current group rather than a page or report total.
- Resources folder: This folder holds the icon used for adding annotation (seal image) to the report.
- Resource1.resx: This file contains the string for the message box that appears when **Custom Annotation** button is clicked again to add the annotation.

Digital Signature Pro

This sample demonstrates how you can digitally sign or set time stamp for a Section Report when exporting it to PDF format.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/DigitalSignaturePro/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/DigitalSignaturePro/C#

Details

When you run this sample, the Invoice report is displayed in the Viewer control.

Clicking the **Generate Digitally Signed PDF** button in the Viewer toolbar creates a PDF file with a time stamp or digital signatures, based on the settings you have specified in the Viewer toolbar. You can change the content of signatures in the **Signature Format** box and you can add the time stamp to the generated pdf file by checking the **Set TimeStamp** checkbox, in the Viewer toolbar.

When you click the **Generate Digitally Signed PDF** button, a dialog for saving the destination file appears. After you indicate the location for a new PDF file, the PDF report file is created. Digital signature certificates dynamically reference and use GrapeCity.pfx, included in the project. Also, digital signatures dynamically load and use the gc.bmp

file that you can find in the Image folder of this sample project.

> **Note:** To run this sample, you must have access to the Nwind database. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

- Image folder: This folder stores the gc.bmp file with the Mescius logo that digital signatures dynamically load and use.
- Invoice report: The Invoice report is the Sample report that uses three GroupHeader sections, a Detail section and a GroupFooter section as well as a label in the PageFooter section to display data. See The Bound Data Sample topic for details on the Invoice report.
- GrapeCity.pfx: In order to create a digital signature, you must have a valid PKCS#12 certificate (*.pfx) file. 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 https://www.source-code.biz/snippets/vbasic/3.htm.
- PDFDigitalSignature form: This is the main form of the Sample that uses the ActiveReports **Viewer** control in the bottom section of the form, and a panel docked to the top contains the **Create Digitally Signed PDF** button, the **Signature Format** box with the drop-down list and the **Set TimeStamp** checkbox.

  Clicking the **Generate Digitally Signed PDF** button opens a dialog for saving the destination file. After you indicate the location for a new PDF file, the PDF report file is created.

  The drop-down list of the **Signature Format** box contains the following options.

  - Invisible - the invisible pdf digital signature.
  - Text - the pdf digital signature that contains text only.
  - Image - the pdf digital signature that contains graphics only.
  - ImageText - the pdf digital signature that contains text and graphics.

  Checking the **Set TimeStamp** checkbox allows you to add the time stamp to the signature of the generated pdf file. The time stamp contains the Time Stamp Server address, its login and password information.

  Right-click **PDFDigitalSignature** in the Solution Explorer and select **View Code** to see the code implementation for the pdf digital signature options.

- Resource.resx: This file contains the string for the message box that appears after the PDF file is generated and the string for the message box that appears when the free service limitation is exceeded.
- PAdES support. You can use the **SignatureFormat ('SignatureFormat Property' in the on-line documentation)** setting, allowing to use PAdES (PDF Advanced Electronic Signatures), available in PDF2.0 (ISO 32000-2). The code samples are as follows.

  Visual Basic.NET code

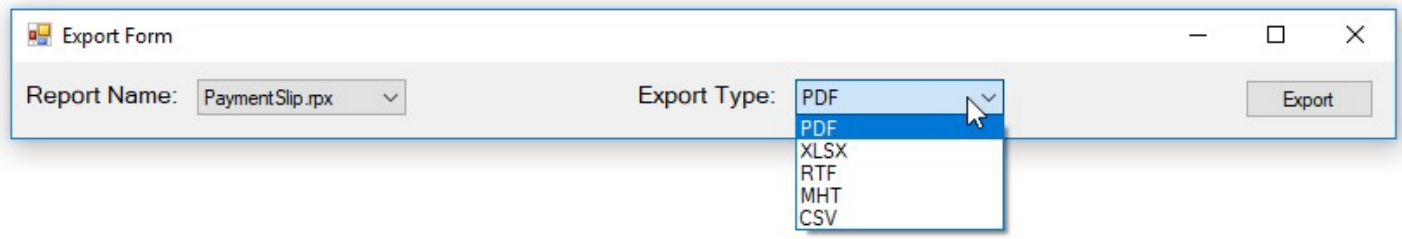
  ```
  oPDFExport.Signature.SignatureFormat =
  GrapeCity.ActiveReports.Export.Pdf.Section.Signing.SignatureFormat.ETSI_CAdES_detached
  ```

  C# code

  ```
  oPDFExport.Signature.SignatureFormat =
  GrapeCity.ActiveReports.Export.Pdf.Section.Signing.SignatureFormat.ETSI_CAdES_detached;
  ```

Export

This sample illustrates how to export to different export formats using code. The available export formats are PDF, XLSX, RTF, MHT and CSV.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/Export/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/Export/C#

Details

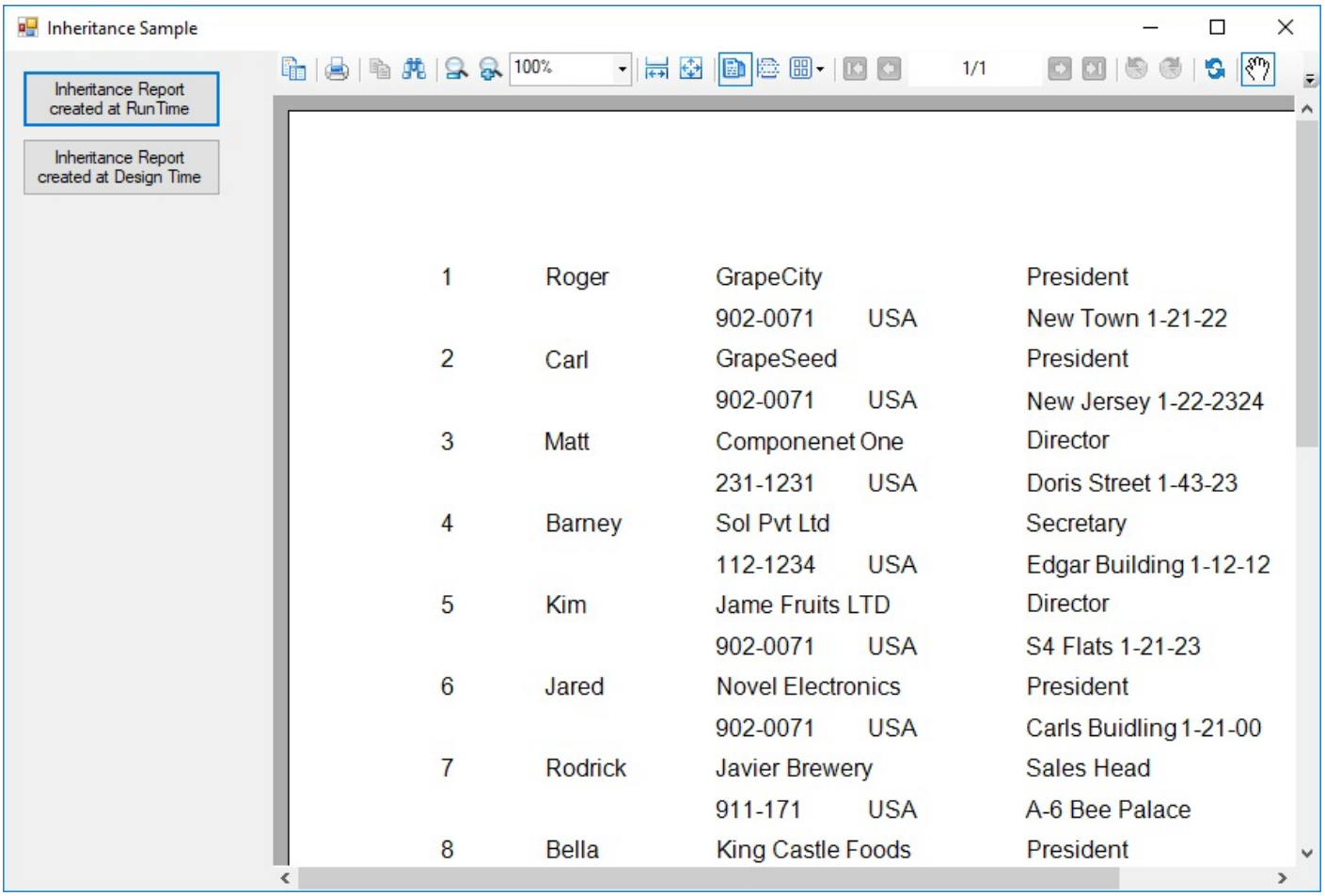
When you run this sample, the Export Form is displayed. The Export Form contains the **Report Name** and **Export Type** combo boxes along with the **Export** button.

The sample consists of **ExportForm**. It exports the selected report in the selected export format.

On running the application, select a report for export in the **Report Name** combo box. In the **Export Type** combo box, you can select one of the following formats for export - PDF, XLSX, RTF, MHT and CSV. Clicking the **Export** button opens the **Save As** dialog where you can specify the name of the exported file. By default, the exported file is saved in the **Documents** folder.

Inheritance

This sample explains the method to inherit a report at run time and design time. The Inheritance sample solution is composed of two classes - the parent class and the child class for both inheritance at run time and design time.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/Inheritance/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/Inheritance/C#

Details

When you run the sample, the report is created and displayed, providing you with the choice of two options - **Inheritance Report created at RunTime** and **Inheritance Report created at Design Time**. By clicking a button on the form, you get a report that inherits another class at run time or at design time.

Inheritance Report created at RunTime
The rptInheritBase class is the inheritance class for the generated report when the **Inheritance Report created at RunTime** button is clicked on the form. The rptInheritBase class inherits the SectionReport class of the GrapeCity.ActiveReports namespace as parent class. It is possible to use DataInitialize event or FetchData event in this class and also possible to load csv file and set values for csv files. It defines the CsvPath property which gets csv file path.
The rptInheritChild class inherits the rptInheritBase class and is a class which only defines the report design. By adding

the event handler for inheritance in the constructor and setting for csv file in CsvPath property, it is possible to perform rendering of data executed by event of BaseReport which is the inheritance class.

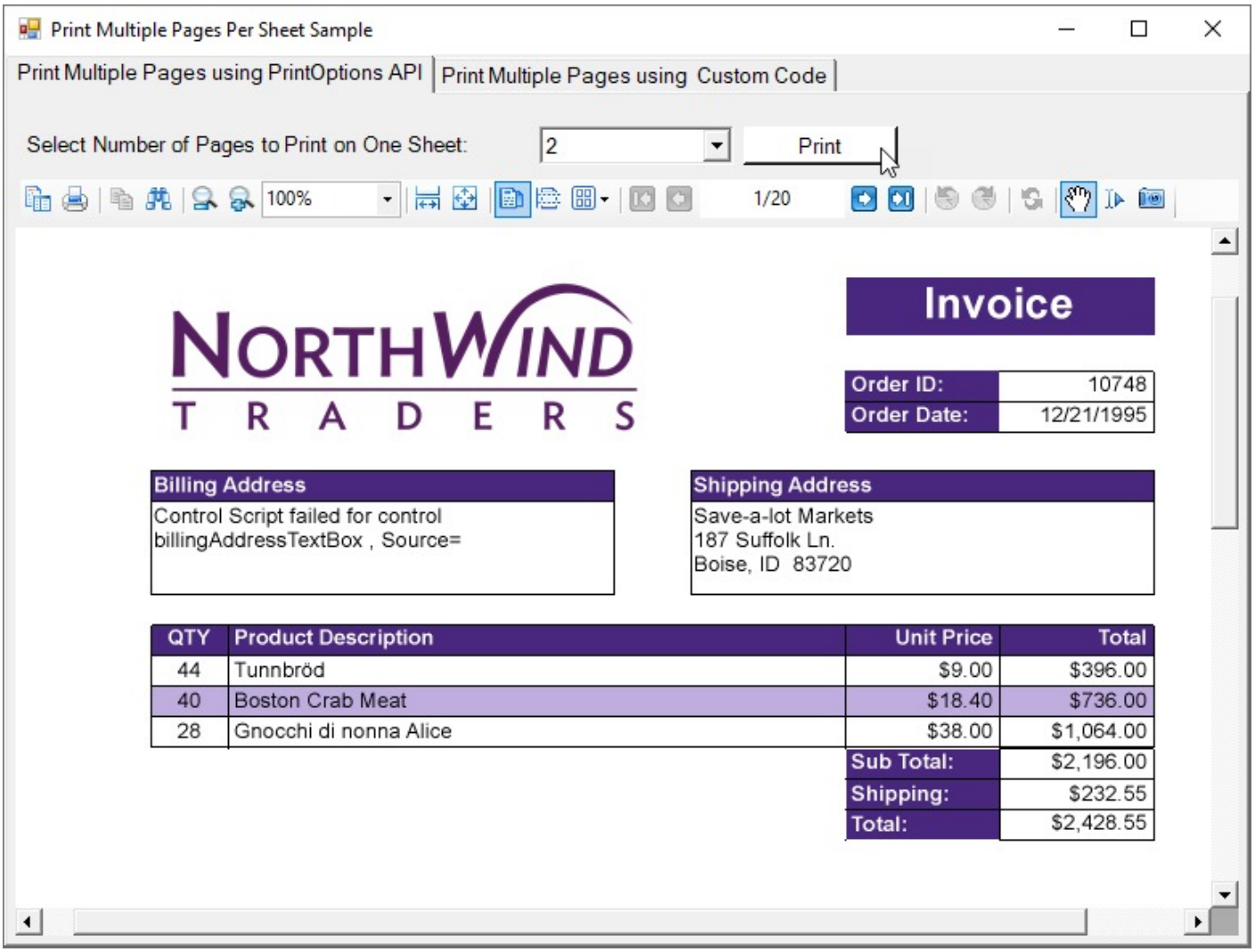
Inheritance Report created at Design Time
The rptDesignBase class is the inheritance class for the generated report when the **Inheritance Report created at Design Time** button is clicked on the form. The rptDesignBase class inherits SectionReport class of the GrapeCity.ActiveReports namespace as parent class. Using this class, you can place any control (ReportInfo controls etc. to display report title, page number, page count) you wish to inherit in PageHeader section and PageFooter section. The rptDesignChild class is inherited from rptDesignBase class. It only defines the design of Detail section. PageHeader section and PageFooter section use the design of rptDesignBase class which is an inherited class. DataSource setting can be performed from rptDesignChild class.

> ⚠ **Caution:** If you have not run the project even once, an error occurs when you try to open the report designers of the inherited classes rptInheritChild or rptDesignChild from the solution explorer. In case this error occurs, **Build** the project once before opening the report.

- ViewerForm: Creates an instance of specified report and display the report in Viewer control.
- rptDesignBase: The rptDesignBase class that defines the layout of PageHeader section and PageFooter section.
- rptDesignChild: The rptDesignChild designer defines the layout on Detail section and sets the value of DataField property of the controls placed in Detail section. Also set the data source to output using **Report Data Source** dialog.
- rptInheritBase: The class to set values for data field and rendering of csv file using DataInitialize event FetchData event.
- rptInheritChild: The designer that sets layout for each control and it's DataField property.

Print Multiple Pages per Sheet

The PrintMultiplePagesPerSheet sample demonstrates how you can print a document with multiple pages per sheet using the common PrintDocument class or PrintOptions class from .NET Framework. This sample project consists of the PrintMultiplePagesForm and the Invoice report.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/PrintMultiplePagesPerSheet/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/PrintMultiplePagesPerSheet/C#

Details

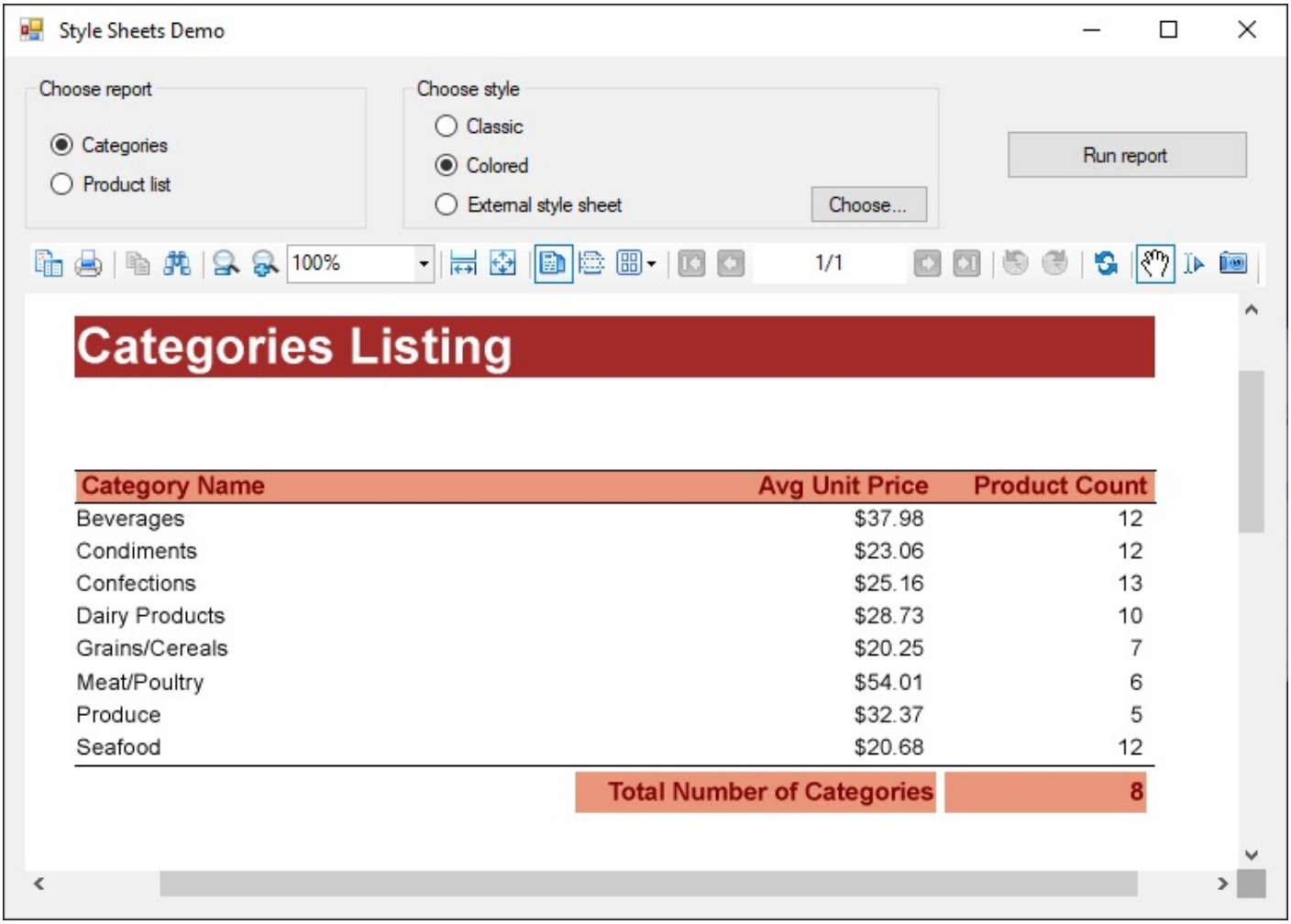
When you run this sample, you will see the PrintMultiplePagesForm with the Invoice report. On this form, you can select the number of pages to be printed on each sheet using the **Select Numer of Pages to Print on One Sheet** ComboBox. You can also select from **PrintMultiple Pages using PrintOptions API** and **PrintMultiple Pages using Custom Code** tabs options and click the **Print** button on each of these tab to print the selected number of pages in one sheet.

> **Note:** To run this sample, you must have access to the **Nwind.db**. The NWIND.db file can be downloaded from GitHub.

- PrintMultiplePagesForm: 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. The top section of Viewer contains a panel in which two tabs, ComboBox control, Label and two Print buttons are placed. **ComboBox** control lets you select the number of pages per sheet (2,4 or 8)and the **Print** button in **PrintMultiple Pages using PrintOptions API** and **PrintMultiple Pages using Custom Code** tab, print the selected number of pages in one sheet. The form also has two dialogs - dlgPrint and PrintDocument which assist in displaying the Print dialog box and printing the document.
  Right-click and select **View Code** to see the code that displays the Invoice report when the form loads. Also the code demonstrates the different ways of printing a document - the **Print** button in **PrintMultiple Pages using Custom Code** tab uses the PrintDocument class and the **Print** button in **PrintMultiple Pages using PrintOptions API** tab uses the PrintOptions class.
- Invoice report: The Invoice report uses a PageHeader section, GroupHeader section, Detail section, GroupFooter section as well as a PageFooter section to display data in a Label control.

Style Sheets

This sample demonstrates how you can change styles at run time to provide a different look to a same report. The project includes two reports, three report styles and a form containing the ActiveReports Viewer control and other controls that allow you to select any combination of styles and reports.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/Stylesheets/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/Stylesheets/C#

Details

Choose Report
Choose between the type of report, Categories and Product List, you want to display in the Viewer control.

Choose Style
Choose between **Classic**, **Colored** and **External style sheet** options to apply the style to the selected report.
Clicking the **Choose** button option for External style sheet displays the **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 this button to display the selected report with the applied style in a Viewer control. Clicking this button creates an SectionReport 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.

The sample consists of:

- 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 Working with External Style Sheets.

- Reports: Two reports, **CategoryReport** and **ProductsReport**, are included in this sample so that you can apply styles 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

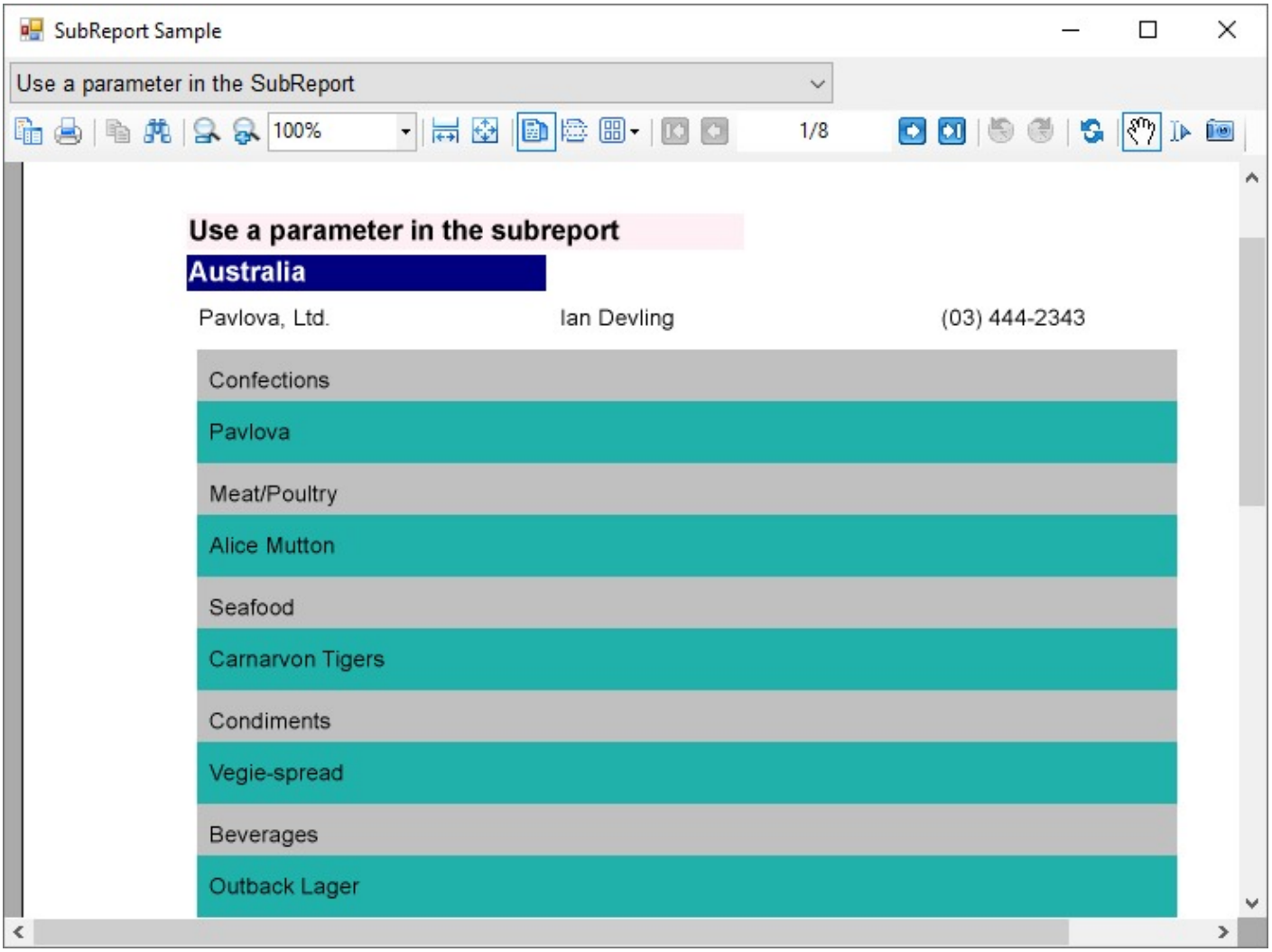
  This event contains code that sets up an Open dialog that shows only ***.reportstyle** files, and passes the selected reportstyle path and file name string to the externalStyleSheet variable.

Run Report Button Click Event

This event contains code that creates an empty SectionReport 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.

Sub Report

The SubReports sample demonstrates how the SubReport control can be used to generate nested and hierarchical reports.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/SubReport/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/SubReport/C#

Details

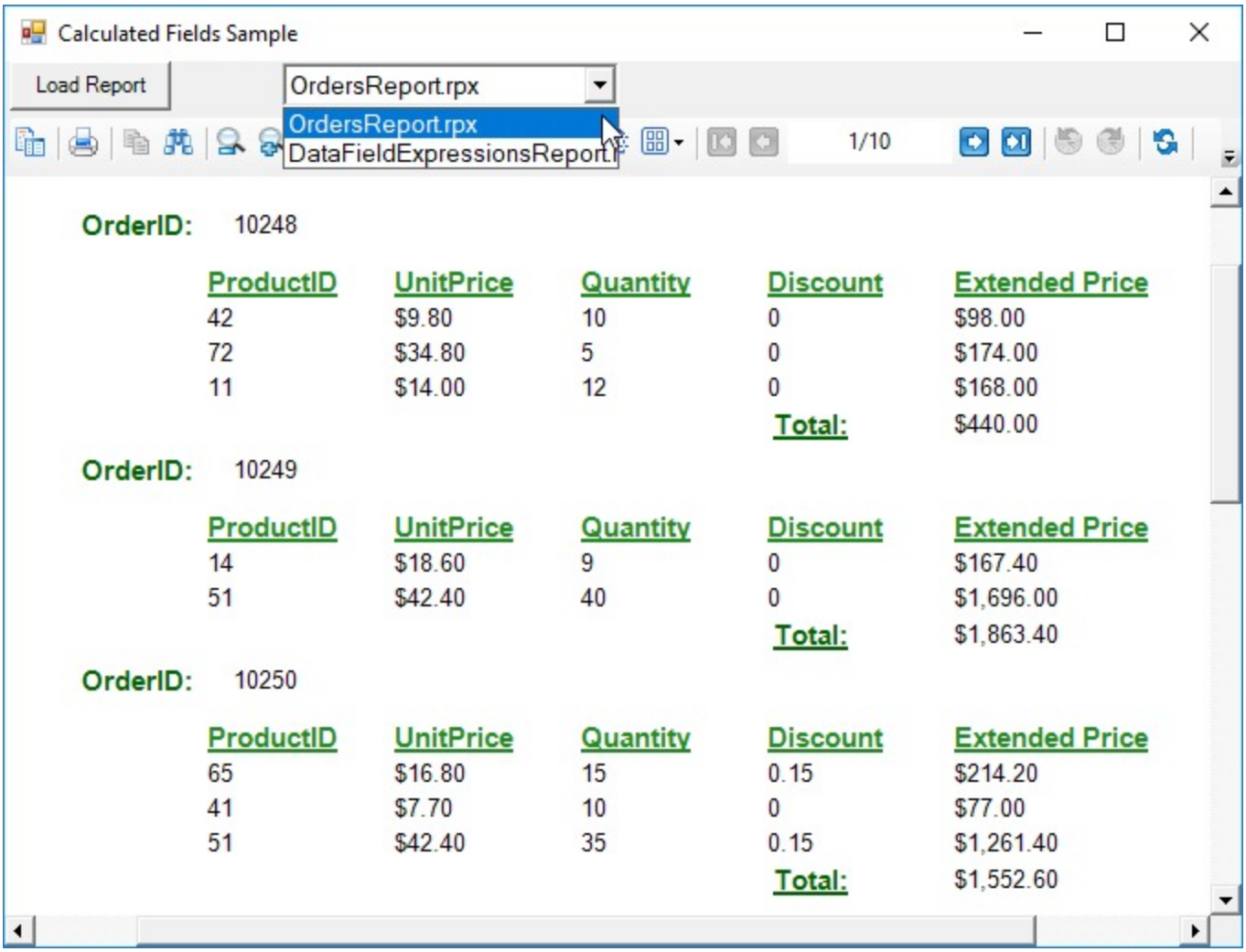
When you run this sample, the blank Viewer form appears, with the drop-down list of the sample reports on the top of the form. Select the report from the drop-down list to have it displayed in the Viewer control. You can select from the following options.

- **Simple SubReport** - the basic sample report that demonstrates how to embed a report into another report. On selecting this report, rptSimpleMain report is displayed. The Detail section of this report contains the bound Textbox control to display the Category Name information and the Subreport control to display data from rptSimpleSub.
- **Nested SubReport** - the report demonstrates how to nest subreports to display main, child, and grandchild levels in a report. It uses rptNestedParent, rptNestedChildMain, and rptNestedChildSub reports.
- **Hierarchical SubReport** - the main report dataset with the SHAPE statement defines the hierarchical structure of the report that uses a subreport. It uses rptHierarchicalMain and rptHierarchicalSub reports.
- **SubReport using the data set that contains relationship** - the main report having dataset with the relation that is defined in code, in the **DataSet.Relations** property of the main rptDSRelationParent report. It uses rptDSRelationParent, rptDSRelationChildMain, and rptDSRelationChildSub reports.
- **Master-detail report containing a SubReport** - the sample report that demonstrates how to create a master detail report that uses a subreport. It uses rptMasterMain and rptMasterSub reports.
- **Bookmark in SubReport** - the sample report that uses bookmarks from the subreport. It uses rptBookmarkMain and rptBookmarkSub reports.
- **Use a parameter in the SubReport** - the sample report demonstrates how to set up a parameter in the data source of the subreport. See rptParamMain and rptParamSub for details.
- **To view the Dataset with multiple tables using SubReports** - the sample report with the dataset that contains multiple data tables. The main report uses subreports to output multiple tables in a single report. See rptUnboundDSMain and rptUnboundDSSub for details.

> **Note:** To run this sample, you must have access to the **Nwind.db**. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

Summary

This sample shows how to use calculated fields and data field expressions for simple calculations in a report.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/API/Section/Summary/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/API/Section/Summary/C#

Details

When you run this sample, The Viewer control with the **Select Report** drop-down list is displayed. There, you can select one of the two reports - **OrdersReport** or **DataFieldExpressionsReport**, and click the **Load Report** button to display the report in the Viewer.

The OrdersReport shows how to use calculated fields, where the field values are calculated in code. A custom field is added to the Fields collection in the DataInitialize event and the field value is calculated in the FetchData event.

The DataFieldExpressionsReport demonstrates the use of data field expressions for simple calculations within the same section of the unbound report using known Fields collection values. These data field expressions cannot be used with the built in summary functions.

> **Note:** To run this sample, you must have access to the **Nwind.db**. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

The sample consists of the **StartForm** and two reports - **OrdersReport** and **DataFieldExpressionsReport**.

- StartForm: The **Viewer** control has the **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.
- OrdersReport: The report shows the ProductID, UnitPrice, Quantity, Discount, Extended Price and Total value for each OrderID. The **Extended Price** value is a calculated field that displays the result of the formula specified in FetchData event.
  The OrdersReport uses a GroupHeader section, a Detail section and a GroupFooter section as well as a Label in the PageFooter section to display data.

  > **Note:** Except for the Detail section, all sections come in header and footer pairs. Unused sections have their **Height** properties set to **0** and their **Visible** properties set to **False**.

  ghOrderID section

  This group header section has the **DataField** property set to OrderID. This setting, along with data sorted by the same field, displays a report grouped by OrderID. The section contains one bound TextBox control to display the OrderID at the beginning of each group.

  Detail section

  The Detail section of this report contains 5 bound TextBox controls that render for each row of data of the OrderID.

  gfOrderID section

  This group footer section displays total of the gfOrderID data in TextBox controls that have values passed in code, or are bound to fields from the report's Fields collection using the DataField property. The total extended price for the OrderID is summarized using the following properties:
  SummaryFunc: Sum (the default value)
  Adds values rather than counting or averaging them.
  SummaryGroup: ghOrderID
  Summarizes the values that fall within the current OrderID group.
  SummaryRunning: Group
  Calculates a running summary (each value is the sum of the current value and all preceding values) within the same group level.
  SummaryType: SubTotal
  Summarizes the current group rather than a page or report total.

  PageFooter section

  The page footer section contains a static Label control that prints at the bottom of each page and contains the note on the number of pages in the report.

- DataFieldExpressionsReport: The DataFieldExpressionsReport displays data, using the Fields collection from the OrderDetail class. The report data under **ExtendedPrice** is calculated by the data field expression, specified in the **DataField** property of the **txtExtendedPrice** TextBox at the design time. The DataFieldExpressionsReport is an unbound report that uses the field values from the OrderDetail class for displaying the report data.

PageHeader section

The PageHeader section contains one Label control with the note text and four labels with the names of the report data fields.

Detail section

The Detail section contains four textboxes that use the Fields collection values to display the report data. The **DataField** property of the **txtExtendedPrice** textbox in the Detail section demonstrates how to format your data field expression.

OrderDetail class

The OrderDetail class contains data that is used in the fields of the report. When the report is run, the values of these fields are used to display data of the report. The field values are bound to the fields in the **DataInitilize** event and the data is bound to the field values in the **FetchData** event of the DataFieldExpressionsReport.

Data Binding

The samples in the Data Binding folder demonstrate data binding with various data providers separately for Page/RDLX and Section Reports.

- Page and RDLX Reports
- Section Report

Page and RDLX Reports

This section contains:

CSV Data Source
: This sample demonstrates how to connect to a CSV data source.

DataSet DataSource
: This sample demonstrates how to use a dataset as a data source for a report.

Json Data Source
: This sample demonstrates how to use the Json data provider at run time and add a web service for authentication.

Object Data Source
: This sample demonstrates how to use Object provider for binding a report.

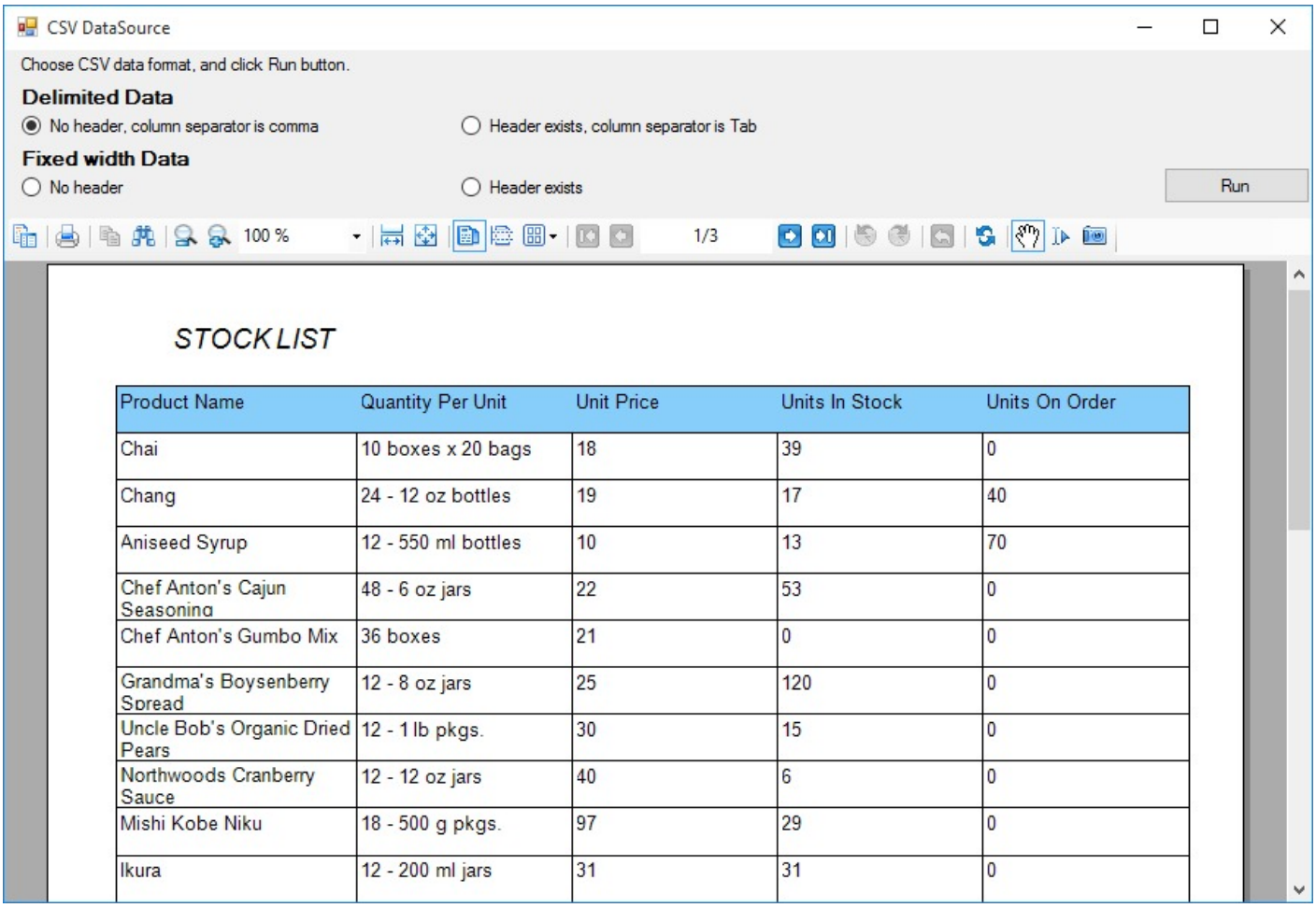
OData Data Source
: This sample demonstrates how to use OData EndPoint for binding a report.

OleDb Data Source
: This sample demonstrates how to connect to an OleDb data source at run time and pass data to the report using LocateDataSource event.

XML Data Source
: This sample demonstrates how to connect to a XML data source at run time and pass data to the report using LocateDataSource event.

CSV Data Source

The CSV Data Source sample demonstrates how to use the CSV data provider in a Page Report.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/CSVDataSource/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/CSVDataSource/C#

Run-Time Features

When you run this sample, the **CSV Data Source** window appears. Select a radio button to specify the data format of the CSV file to use for the data source, and click the Run button to show the report in the viewer. You can choose from the following CSV data formats:

- Delimited Data
  - No header, column separator is comma
  - Header exists, column separator is Tab
- Fixed width Data

- No header
- Header exists

Project Details

MainForm

This is the main form that appears when you run this sample. It uses the ActiveReports Viewer control to display the report at run time, and radio buttons to select the data source settings.

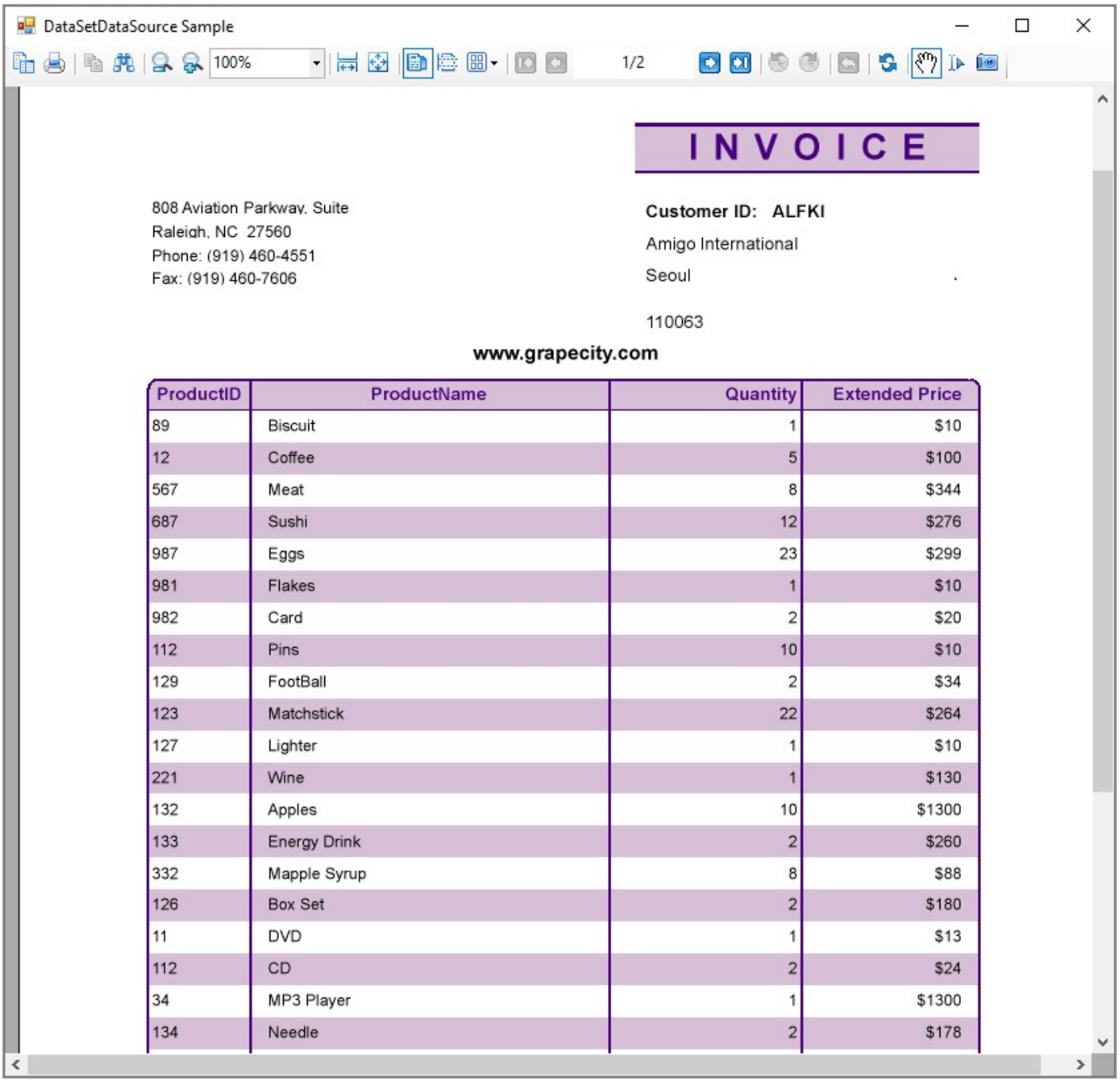
Right-click the form and select **View Code** to see how to set the data source settings in the connection string, and how to show the report at run time.

StockList.rdlx

This is the report that gets displayed in the Viewer at run time. The report is bound to the StockList dataset of the CSV data provider and uses a Table data region and TextBox controls to display the stock list. The stock list is grouped by CustomerID value.

DataSet DataSource

The DataSetDataSource sample demonstrates how to connect a Page Report to an unbound data source at run time, using the DataSet provider with the LocateDataSource event. The reporting engine raises the LocateDataSource event when it needs input on the data to use.



ProductID	ProductName	Quantity	Extended Price
89	Biscuit	1	$10
12	Coffee	5	$100
567	Meat	8	$344
687	Sushi	12	$276
987	Eggs	23	$299
981	Flakes	1	$10
982	Card	2	$20
112	Pins	10	$10
129	FootBall	2	$34
123	Matchstick	22	$264
127	Lighter	1	$10
221	Wine	1	$130
132	Apples	10	$1300
133	Energy Drink	2	$260
332	Mapple Syrup	8	$88
126	Box Set	2	$180
11	DVD	1	$13
112	CD	2	$24
34	MP3 Player	1	$1300
134	Needle	2	$178

Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/DataSetDataSource/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/DataSetDataSource/C#

Run-Time Features

When you run this sample, the Invoice2.rdlx report is displayed in the Viewer control. The report displays the Invoice form with the list of products along with the product ID, quantity and price of the products.

The report connects to an unbound data source at run time using the LocateDataSource event and the DataSet provider.

Project Details

Invoice.rdlx

In the Report Data Source dialog, the type of the report data source is set to DataSetProvider and the ConnectionString is left blank. In the DataSet dialog, data fields used on the report are added to the DataSet on the Fields page.

This report uses the Table, TextBox, Label and Shape controls to create an Invoice layout for displaying the customer transactions. The Container control at the bottom of the report contains a label, a textbox and line control. The textbox uses the Sum function to display the sum of the values returned by the expression indicated in the Value property.

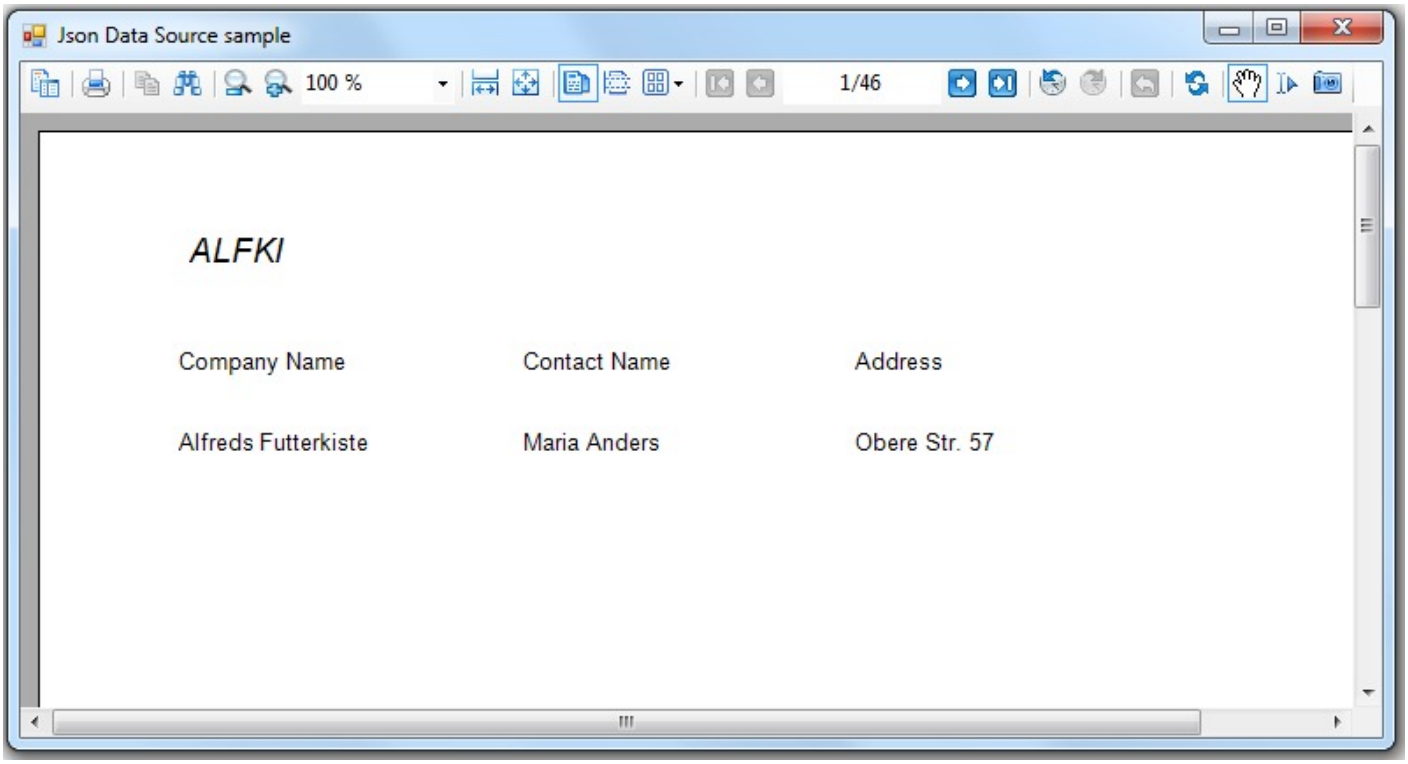
ViewerForm

This is the main form that appears when you run this sample. It uses the ActiveReports Viewer control to display the report at run time. The code-behind the form contains the code with the LocateDataSource event that connects the Invoice2.rdlx report to unbound data and the code that populates fields for the data table.

Json Data Source

The Json Data Source sample demonstrates how to use the Json data provider. The sample uses a web service that requires authentication to access the Json data source at run time.

You must have IIS Express installed on your machine for the Json Data Source sample to run.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/JsonDataSource/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/JsonDataSource/C#

Run-Time Features

The sample consists of two projects: the Windows Application project with a report, MainForm, and DataLayer class that provides report data; and the Web Application project with a web service that provides access authentication for the report data at run time.

When you run this sample, you will see the MainForm appear. The MainForm contains the ActiveReports **Viewer** that displays a report with a list of Customers from the Json data provider.

> **Note:** To run this sample, you must have access to the **customers.json** data file. The customers.json file can be downloaded from GitHub: ..\Samples18\Data\customers.json.

Project Details

JsonDataSource

This is the Windows Application project that contains the MainForm, the sample Page Report, and the DataLayer file

with the report data.

DataLayer

This file is an internal class that contains code to create the data connection. It provides interaction with the server containing Json data using HTTP, including the access credentials.

MainForm

This is the main form that appears when you run this sample. It uses the ActiveReports Viewer control to display the report at run time.

Right-click the form and select **View Code** to see how to load and show the report at run time.

testReport.rdlx

This is the report that gets displayed in the Viewer at run time. The report contains the embedded Json schema as the data source.

It uses the Table data region to display data from the Json data source, the **Customers** sample data file.

WebService

This is the Web Application project that contains the web service used to retrieve data from the Json data source for the sample report at run time.

BasicAuthHttpModule

This file is the public class that provides access authentication when the report connects to the Json data source at run time.

Service.asmx

This is the web service required to access the Json data provider.

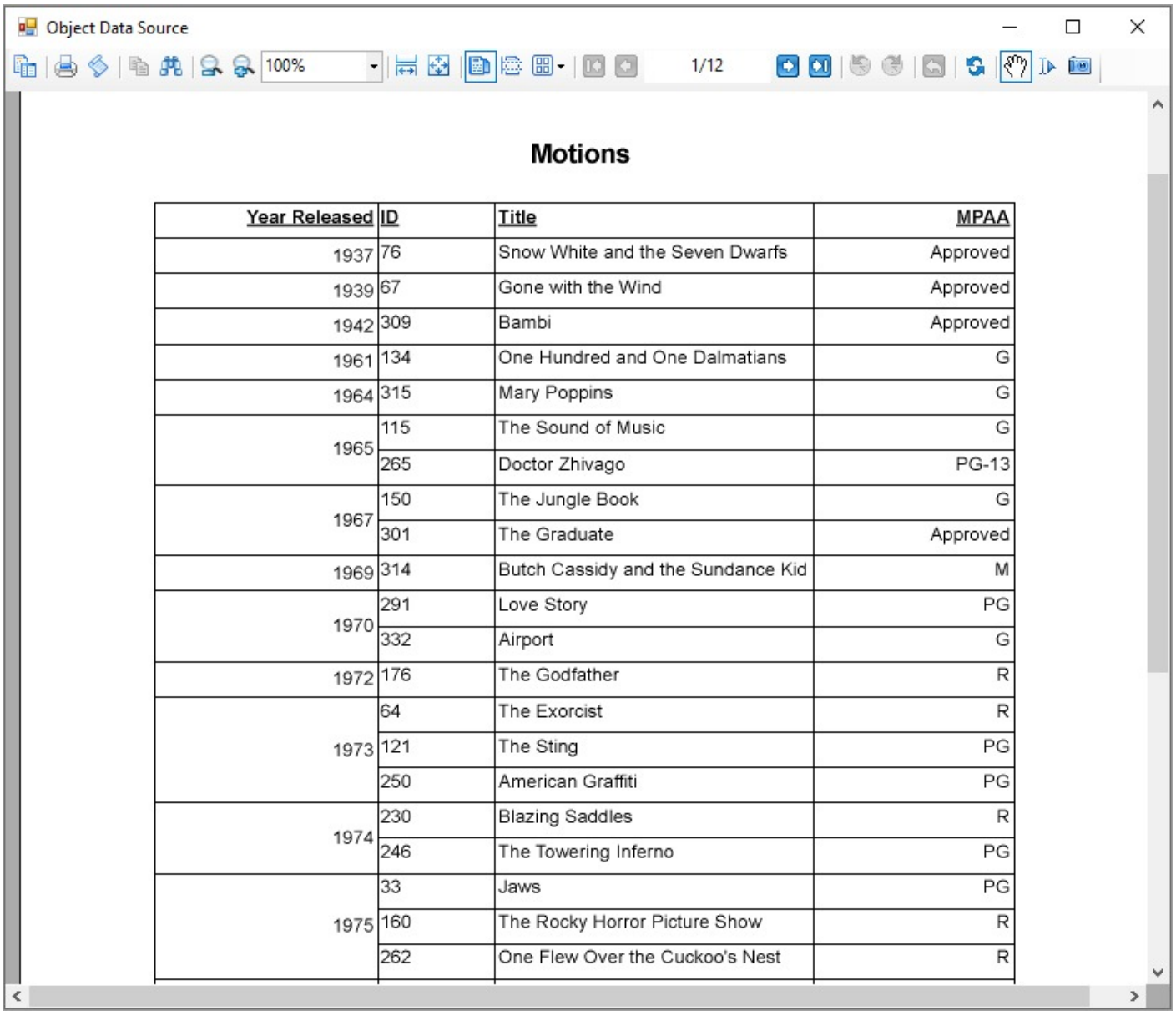
Web.config

This is the web configuration file for configuring your web application.

Object Data Source

The Object Data Source sample demonstrates how to use Object provider for binding a report that contains a subreport.

- **Sample Location**
- **Run-Time Features**
- **Project Details**



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/ObjectDataSource/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/ObjectDataSource/C#

Run-Time Features

When you run this sample, the MainForm with the ActiveReports Viewer appears. The viewer displays the report where **YearReleased** column is a part of the main report (ObjectsReport.rdlx) whereas other columns such as **ID**, **Title**

and **MPAA** are part of the subreport (SubObjectsReport.rdlx).

Project Details

DataLayer

This file is an internal class that contains code to provide data for the report.

MainForm

This is the main form that appears when you run this sample. It uses the ActiveReports Viewer control to display the report at run time.

Right-click the form and select **View Code** to see how to load and show the report at run time.

ObjectsReport.rdlx

This is the report that gets displayed in the Viewer at run time.

The report uses the header with the Textbox to display the report heading and the footer with a Textbox to display the page number information. The body of the report has the Table data region to display data where only first column is obtained from the Objects data provider. For that, the Textbox controls of the Table are bound to the Objects data source in the **Value** property. For rest three columns, subreport control is used.

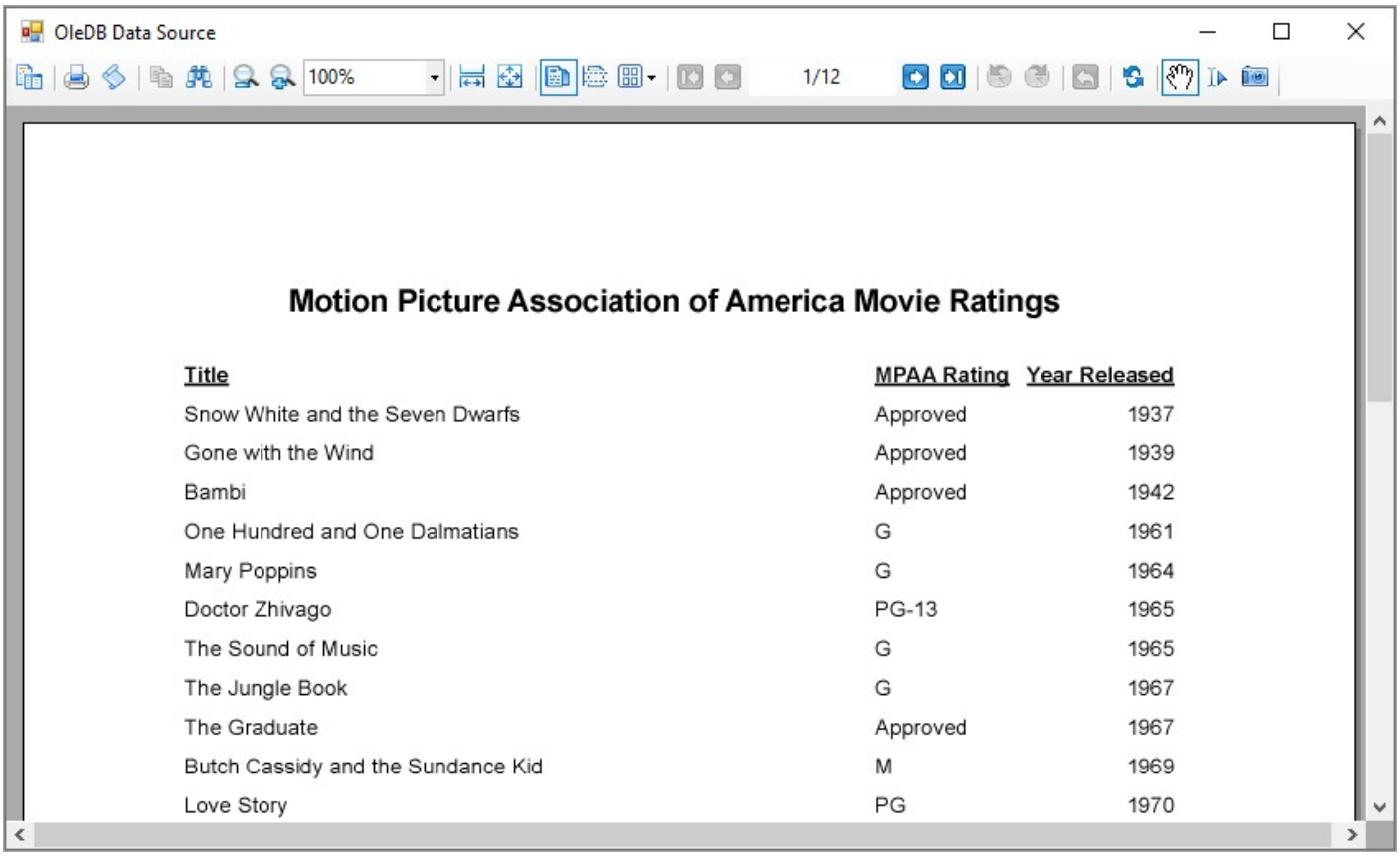
SubObjectsReport.rdlx

This is the subreport that is placed on Table data region of the main report.

The report has a Table data region to display data obtained from the Objects data provider. Here also, the Textbox controls of the Table are bound to the Objects data source in the **Value** property.

OleDb Data Source

The OleDB Data Source sample demonstrates how to use the OleDb data provider for binding a report to data.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/OleDbDataSource/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/OleDbDataSource/C#

Run-Time Features

When you run this sample, the MainForm with the ActiveReports Viewer appears. The Viewer displays the report with the list of movies, their ratings and the release year information.

> **Note:** To run this sample, you must have access to the **Reels.mdb**. The Reels.mdb file can be downloaded from GitHub: ..\Samples18\Data\Reels.mdb.

Project Details

DataLayer

This file is an internal class that contains code to create a data connection. It creates the OleDb data reader to read data from the Reels database and add it to an array to provide data for the report.

MainForm

This is the main form that appears when you run this sample. It uses the ActiveReports Viewer control to display the report at run time.

Right-click the form and select **View Code** to see how to load and show the report at run time.

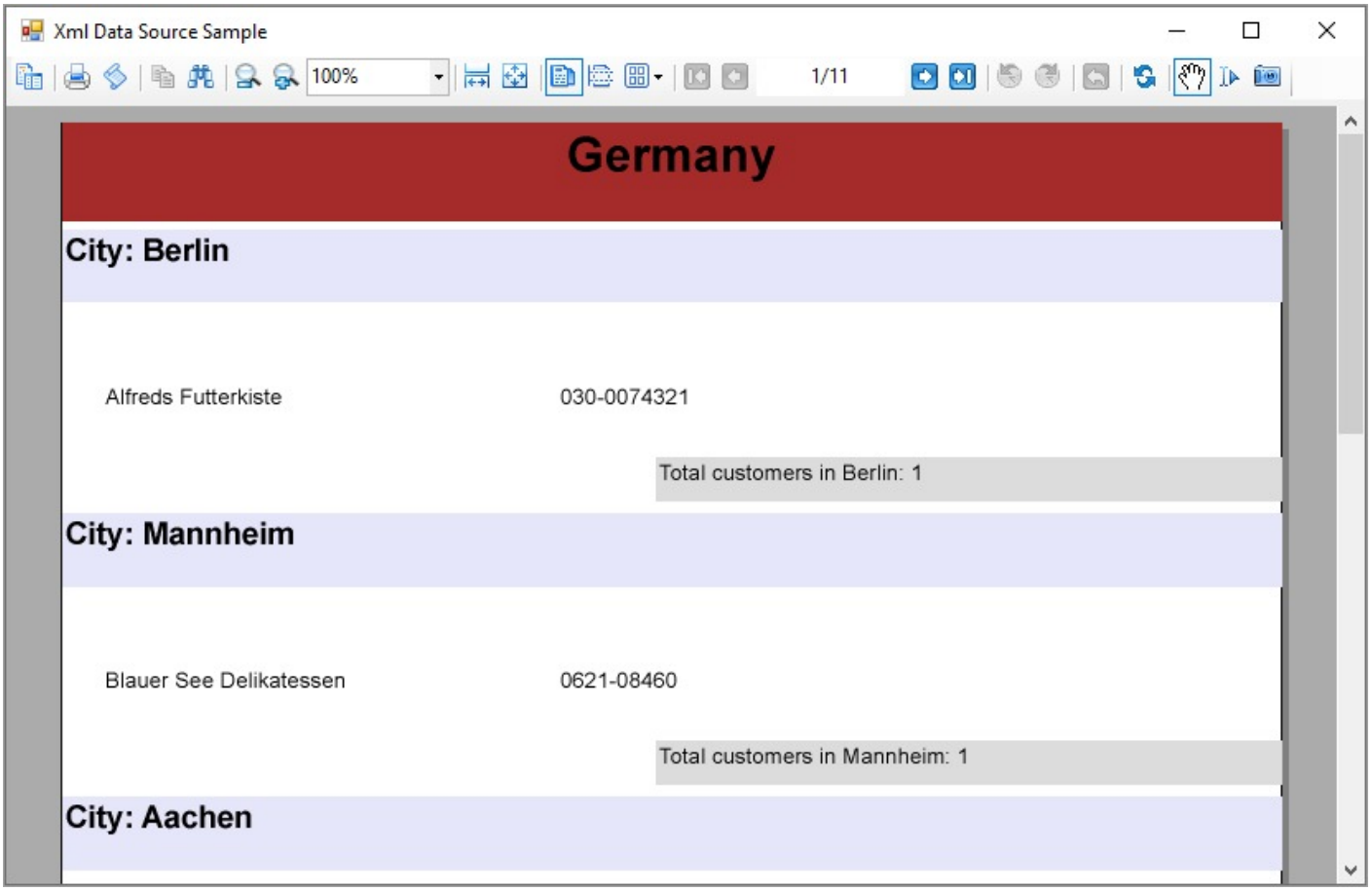
OleDBReport.rdlx

This is the report that gets displayed in the Viewer at run time.

The report uses the header with the Textbox to display the report heading and the footer with the Textbox to display the page number information. The body of the report has the Table data region to display data obtained from the OleDb data provider. For that, the Textbox controls of the Table are bound to the OleDB data source in the **Value** property.

Xml Data Source

The XML Data Source sample demonstrates how to use the XML data provider for supplying data to the report.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/XmlDataSource/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/XmlDataSource/C#

Run-Time Features

When you run this sample, you will see the MainForm appear. The MainForm contains the ActiveReports **Viewer** that displays a report with data from the xml data provider.

Project Details

BandedListXML.rdlx

This is the main report that gets displayed in the Viewer at run time. It uses the BandedList and Textbox controls, and the Subreport control inside the BandedList data region to display data.

The BandedList data region uses two groups to group the report data by the fields **City** and **Country**.

The Subreport control, placed in the GroupFooter section of the BandedList, displays the **CountrySales** report.

CountrySales.rdlx

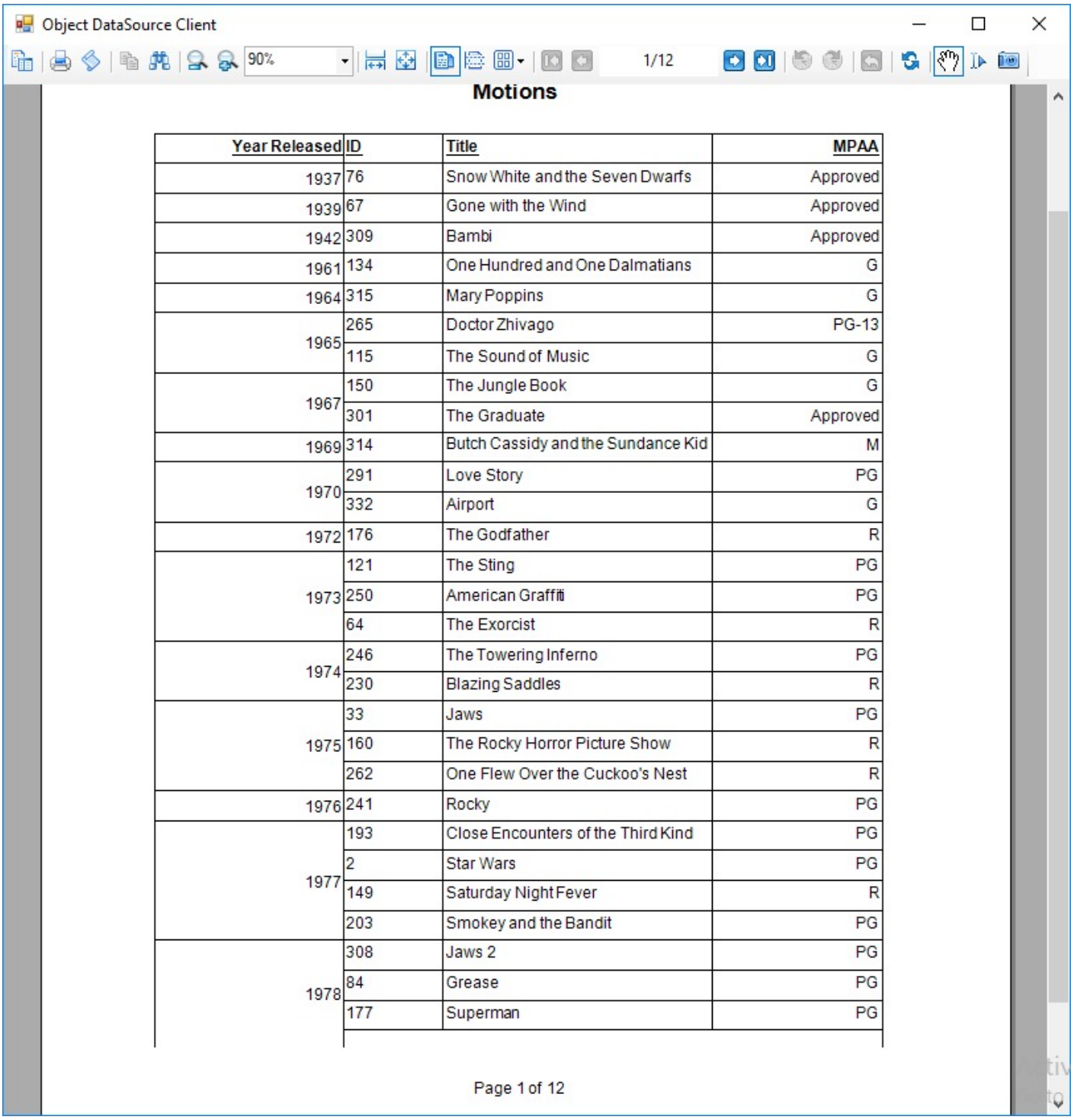
This is the report that gets displayed by the Subreport control of the BandedListXML report.

It uses the Chart data region to display data. The **Chart Type** property is set to **Doughnut (Pie Exploded Doughnut)**, which shows the analysis of companies sales amount for different countries.

OData Data Source

The OData Data Source sample demonstrates how to use OData EndPoint for binding a report.

- **Sample Location**
- **Run-Time Features**
- **Project Details**



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/ODataDataSource/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/PageAndRDLX/ODataDataSource/C#

Run-Time Features

The ODataDataSource sample needs a running ODataEndPoint to obtain data. To run the sample, please perform the following steps:

1. Right-click solution in **Solution Explorer** and select **Properties**.
2. Select **Multiple startup projects** radio button and Start actions in **ODataEndPoint** and **ObjectDataSourceClient**, or **ODataEndPoint** and **JsonDataSourceClient** projects in Startup Project tab.
3. Run the sample again.

Project Details

The sample consists of JsonDataSourceClient and ObjectDataSourceClient projects to render data and ODataEndPoint to query data.

JsonDataSourceClient

This folder contains the DataLayer, Program and Service classes required for the data connection.

The MainForm is the form that appears when you run this sample if you have previously selected **ODataEndPoint** and **JsonDataSourceClient** projects as startup projects in the sample **Properties**.

ObjectDataSourceClient

This folder contains the DataLayer, Program and Service classes required for the data connection.

The MainForm is the form that appears when you run this sample if you have previously selected **ODataEndPoint** and **ObjectDataSourceClient** projects as startup projects in the sample **Properties.**

The **Models** subfolder contains Movie and Year classes.

ODataEndPoint

This folder contains the **AppData** and **AppStart** subfolders required to run the application.

The **Controllers** subfolder contains the **MoviesController** and **CustomersController** files. The **MoviesController** handles the user interaction and returns the main view. The **CustomersController** handles the customer details information that is displayed when a customer is selected.

The **Models** subfolder contains the Customer and Movie classes providing data for the report.

Global.asax is the default class that sets global URL routing values for this web application.

Web.config is the configuration file that contains the httpHandlers that allow ActiveReports to process this web application. Note that you need to manually update version information here when you update your version of ActiveReports.

Section Report

This section contains:

Bound Data

Demonstrates binding to ADO.NET Data objects.

IList Binding

Demonstrates creating a custom collection that stores data from the database in the List. The custom collection is displayed by binding data to the DataGridView control by using the DataSource property of this control.

LINQ

The LINQ sample demonstrates how to use LINQ in an ActiveReports report.

Unbound Data

The Unbound Data sample demonstrates how to create a dataset for a Section Report and use the FetchData event to populate the Fields collection to display the report unbound data.

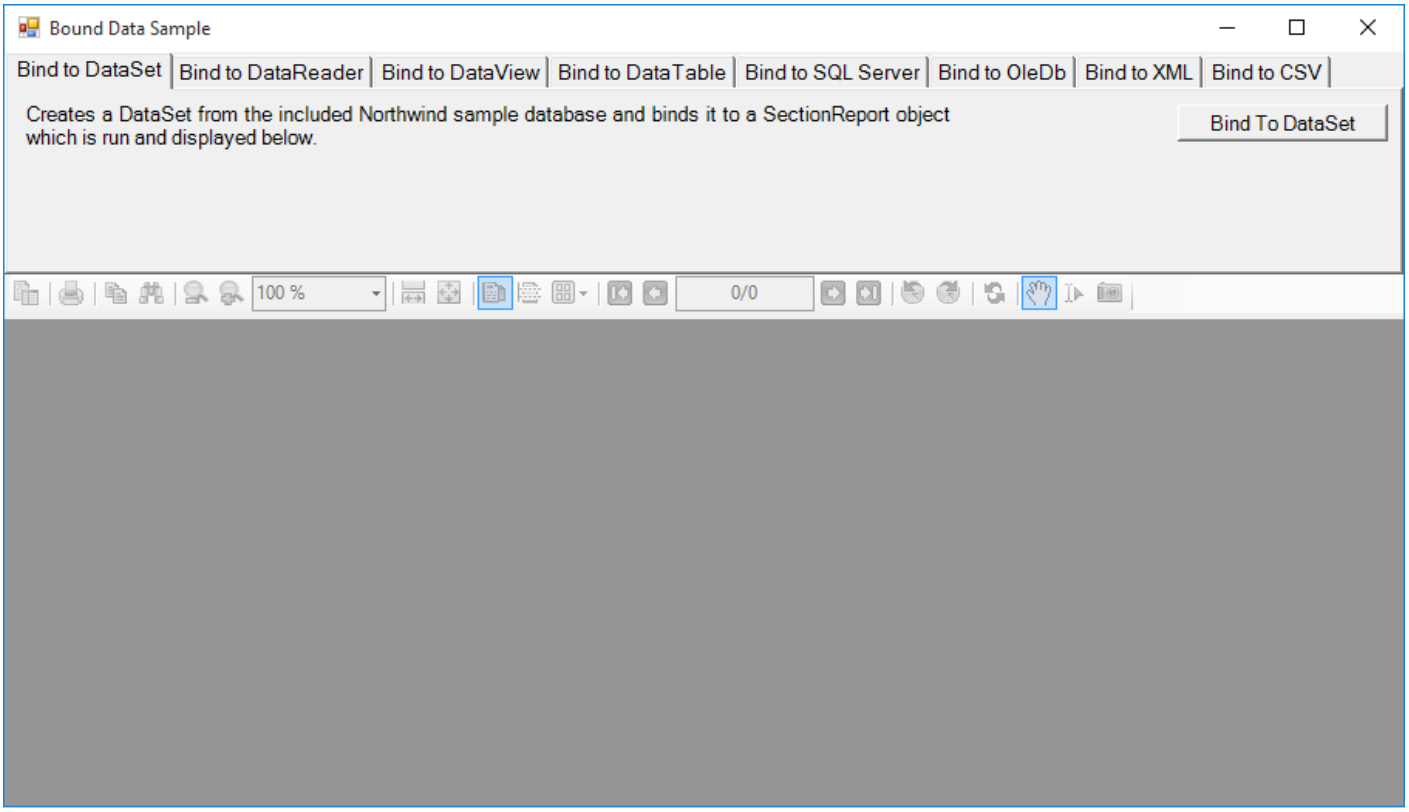
XML

This sample demonstrates how to create a report with XML data, using a SubReport or using the XML hierarchical structure.

Bound Data

The Bound Data sample demonstrates various ways to bind data in a Section Report.

When you run the sample, the Viewer control displays the form with eight tabs, each with a different data binding technique. Click to select a tab, and then click the **Bind To** button to create the report.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/BoundData/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/BoundData/C#

Run-Time Features

The top panel of the MainForm is composed of eight tabs:

Bind to DataSet
Creates a DataSet from the sample database and binds it to a SectionReport object.

Bind to DataReader
Creates a DataReader from the sample database and binds it to a SectionReport object.

Bind to DataView
Creates a DataView from the sample database and binds it to a SectionReport object. This tab contains a ComboBox which lets you choose the company name from the NWind database.

Bind to DataTable
Creates a DataTable from the sample database and binds it to a SectionReport object.

Bind to SQL Server
Creates a SQL Server DataSource from a SQL server instance and binds it to a SectionReport object. The ComboBox present in this tab lets you populate the dropdown list with the existing SQL servers on the network.

Bind to OleDb
Creates an Oledb DataSource and binds it to a SectionReport object.

Bind to XML
Creates a XML DataSource from a file and binds it to a SectionReport object. The XML tab also features a **Generate XML** button that generates a DataSet and saves it as an XML data file. The generated file is then used as a data source for the report.

Bind to CSV
Creates a CSV DataSource from a file and binds it to a SectionReport object. You can select the data type of the file from the following options:

- Delimited Data (with or without header)
- Fixed width Data (with or without header)

Project Details

MainForm

The MainForm uses the ActiveReports **Viewer** control in the bottom section of the form, and a panel docked to the top contains 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.

Invoice Report

The Invoice report uses three GroupHeader sections, a Detail section and a GroupFooter section as well as a label in the PageFooter section to display data.

> **Note:** Except for the Detail section, all sections come in header and footer pairs. Unused sections 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.

This section also 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 GFOrderID 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 TextBoxes render once for each row of data found in the current OrderID before the report moves on to the GroupFooter sections.

GFOrderID

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 TextBoxes. Two of the TextBox controls use the following properties to summarize the detail data: **SummaryFunc**, **SummaryGroup**, and **SummaryType**. For more information, Create a Summary Report.

The **Total** TextBox does not use the DataField property or any of the summary properties, or even any code. To find the functionality of this TextBox, in design view, click the **Script** tab at the bottom of the report.

PageFooter

This section has one simple Label control. For more information about report sections and the order in which they print, see Section Report Structure and Report Events.

ProductList Report

The ProductList report uses the Header and Detail sections to display data.

The Header section contains a number of Label controls to display column names for the product list.

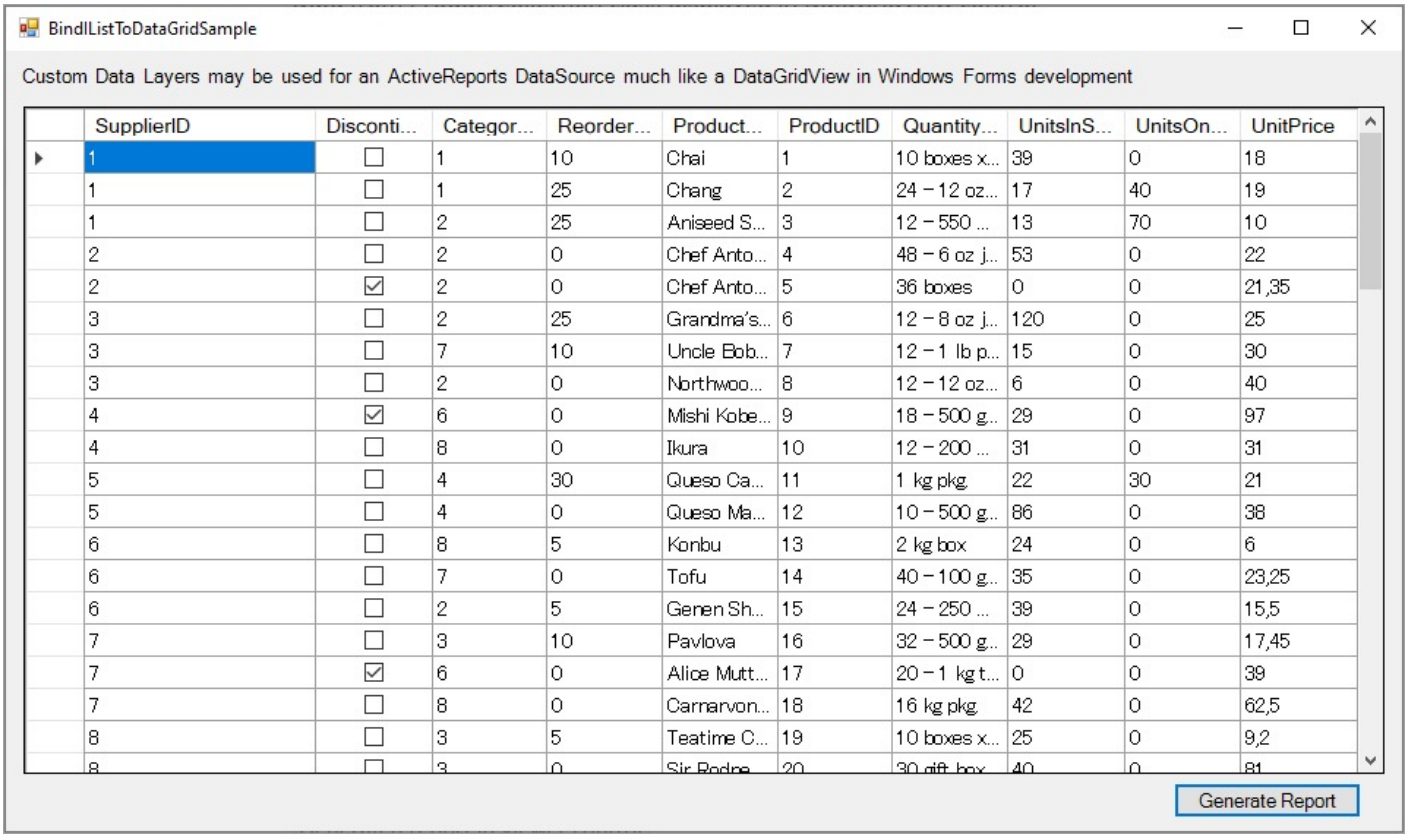
The Detail section contains four TextBox controls to fetch the product data.

IList Binding

The IList Binding sample uses **CollectionBase** class, with implementation of IList interface to create a **ProductCollection** class which gets populated from the Products table of the NWind database. The created ProductCollection is used as a database for binding data to the DataGridView control.The data from ProductCollection class gets displayed using the DataSource property of DataGridView control. On clicking the Generate Report button,

the ProductCollection class is again used to display the data of the generated report in a Viewer control. Similarly, you can display a report by binding to the DataSource property of a report.

Data from ProductCollection class displayed in DataGridView control

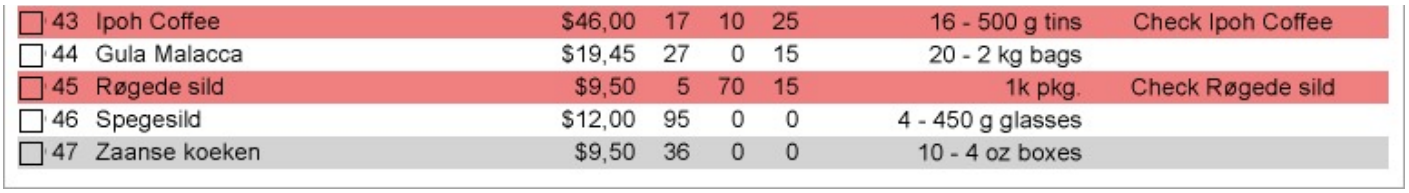


SupplierID	Disconti...	Categor...	Reorder...	Product...	ProductID	Quantity...	UnitsInS...	UnitsOn...	UnitPrice
1	☐	1	10	Chai	1	10 boxes x...	39	0	18
1	☐	1	25	Chang	2	24 – 12 oz...	17	40	19
1	☐	2	25	Aniseed S...	3	12 – 550 ...	13	70	10
2	☐	2	0	Chef Anto...	4	48 – 6 oz j...	53	0	22
2	☑	2	0	Chef Anto...	5	36 boxes	0	0	21,35
3	☐	2	25	Grandma's...	6	12 – 8 oz j...	120	0	25
3	☐	7	10	Uncle Bob...	7	12 – 1 lb p...	15	0	30
3	☐	2	0	Northwoo...	8	12 – 12 oz...	6	0	40
4	☑	6	0	Mishi Kobe...	9	18 – 500 g...	29	0	97
4	☐	8	0	Ikura	10	12 – 200 ...	31	0	31
5	☐	4	30	Queso Ca...	11	1 kg pkg.	22	30	21
5	☐	4	0	Queso Ma...	12	10 – 500 g...	86	0	38
6	☐	8	5	Konbu	13	2 kg box	24	0	6
6	☐	7	0	Tofu	14	40 – 100 g...	35	0	23,25
6	☐	2	5	Genen Sh...	15	24 – 250 ...	39	0	15,5
7	☐	3	10	Pavlova	16	32 – 500 g...	29	0	17,45
7	☑	6	0	Alice Mutt...	17	20 – 1 kg t...	0	0	39
7	☐	8	0	Carnarvon...	18	16 kg pkg.	42	0	62,5
8	☐	3	5	Teatime C...	19	10 boxes x...	25	0	9,2
8	☐	3	0	Sir Rodne...	20	30 gift box...	40	0	81

Generated report displayed in Viewer control

Product Listing for Northwind Traders

Discontinued	Product ID	Product Name	Price per Unit	Stocked Units	Ordered Units	Reorder Level	Quantity Per Unit	Comments
☐	1	Chai	$18,00	39	0	10	10 boxes x 20 bags	
☐	2	Chang	$19,00	17	40	25	24 - 12 oz bottles	Check Chang
☐	3	Aniseed Syrup	$10,00	13	70	25	12 - 550 ml bottles	Check Aniseed Syrup
☐	4	Chef Anton's Cajun Seasoning	$22,00	53	0	0	48 - 6 oz jars	
☑	5	Chef Anton's Gumbo Mix	$21,35	0	0	0	36 boxes	
☐	6	Grandma's Boysenberry Spread	$25,00	120	0	25	12 - 8 oz jars	
☐	7	Uncle Bob's Organic Dried Pears	$30,00	15	0	10	12 - 1 lb pkgs.	
☐	8	Northwoods Cranberry Sauce	$40,00	6	0	0	12 - 12 oz jars	
☑	9	Mishi Kobe Niku	$97,00	29	0	0	18 - 500 g pkgs.	
☐	10	Ikura	$31,00	31	0	0	12 - 200 ml jars	
☐	11	Queso Cabrales	$21,00	22	30	30	1 kg pkg.	Check Queso Cabrales
☐	12	Queso Manchego La Pastora	$38,00	86	0	0	10 - 500 g pkgs.	
☐	13	Konbu	$6,00	24	0	5	2 kg box	
☐	14	Tofu	$23,25	35	0	0	40 - 100 g pkgs.	
☐	15	Genen Shouyu	$15,50	39	0	5	24 - 250 ml bottles	
☐	16	Pavlova	$17,45	29	0	10	32 - 500 g boxes	
☑	17	Alice Mutton	$39,00	0	0	0	20 - 1 kg tins	
☐	18	Carnarvon Tigers	$62,50	42	0	0	16 kg pkg.	
☐	19	Teatime Chocolate Biscuits	$9,20	25	0	5	10 boxes x 12 pieces	
☐	20	Sir Rodney's Marmalade	$81,00	40	0	0	30 gift boxes	
☐	21	Sir Rodney's Scones	$10,00	3	40	5	24 pkgs. x 4 pieces	Check Sir Rodney's Scones
☐	22	Gustaf's Knäckebröd	$21,00	104	0	25	24 - 500 g pkgs.	
☐	23	Tunnbröd	$9,00	61	0	25	12 - 250 g pkgs.	
☑	24	Guaraná Fantástica	$4,50	20	0	0	12 - 355 ml cans	
☐	25	NuNuCa Nuß-Nougat-Creme	$14,00	76	0	30	20 - 450 g glasses	
☐	26	Gumbär Gummibärchen	$31,23	15	0	0	100 - 250 g bags	
☐	27	Schoggi Schokolade	$43,90	49	0	30	100 - 100 g pieces	
☑	28	Rössle Sauerkraut	$45,60	26	0	0	25 - 825 g cans	
☑	29	Thüringer Rostbratwurst	$123,79	0	0	0	50 bags x 30 sausgs.	
☐	30	Nord-Ost Matjeshering	$25,89	10	0	15	10 - 200 g glasses	Check Nord-Ost Matjeshering
☐	31	Gorgonzola Telino	$12,50	0	70	20	12 - 100 g pkgs	Check Gorgonzola Telino
☐	32	Mascarpone Fabioli	$32,00	9	40	25	24 - 200 g pkgs.	Check Mascarpone Fabioli
☐	33	Geitost	$2,50	112	0	20	500 g	
☐	34	Sasquatch Ale	$14,00	111	0	15	24 - 12 oz bottles	
☐	35	Steeleye Stout	$18,00	20	0	15	24 - 12 oz bottles	
☐	36	Inlagd Sill	$19,00	112	0	20	24 - 250 g jars	
☐	37	Gravad lax	$26,00	11	50	25	12 - 500 g pkgs.	Check Gravad lax
☐	38	Côte de Blaye	$263,50	17	0	15	12 - 75 cl bottles	
☐	39	Chartreuse verte	$18,00	69	0	5	750 cc per bottle	
☐	40	Boston Crab Meat	$18,40	123	0	30	24 - 4 oz tins	
☐	41	Jack's New England Clam Chowder	$9,65	85	0	10	12 - 12 oz cans	
☑	42	Singaporean Hokkien Fried Mee	$14,00	26	0	0	32 - 1 kg pkgs.	



☐	43	Ipoh Coffee	$46,00	17	10	25	16 - 500 g tins	Check Ipoh Coffee
☐	44	Gula Malacca	$19,45	27	0	15	20 - 2 kg bags	
☐	45	Røgede sild	$9,50	5	70	15	1k pkg.	Check Røgede sild
☐	46	Spegesild	$12,00	95	0	0	4 - 450 g glasses	
☐	47	Zaanse koeken	$9,50	36	0	0	10 - 4 oz boxes	

Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/IListBinding/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/IListBinding/C#

Run-Time Features

When you run this sample, DataGridView, which is a standard Windows Forms control displays the custom collection. To display a report with the bound custom collection, click the **Generate Report** button.

> **Note:** To run this sample, you must have access to the **Nwind.db**. The NWIND.db file can be downloaded from GitHub: ..\Samples18\Data\NWIND.db.

Project Details

The IList Binding sample consists of two projects: **IListBinding** and **IListBinding.DataLayer**.

IListBinding Project

BindIListToDataGridSample

This form contains a DataGridView control, a Label control and a Generate Report button. It displays output results by binding data of a custom collection to the DataGridView control. On clicking the **Generate Report** button, ViewerForm displays a report bound to this custom collection.

IlistReportSample report

The report uses the ReportHeader, GroupHeader1 and Detail sections for the report output.

ReportHeader section
The ReportHeader section contains a Label that displays the title of the report.

GroupHeader1 section
The GroupHeader1 section contains nine Label controls that define the layout of the report data.

Detail section
The Detail section contains TextBox controls to display the report data. Following settings have been performed to enhance the appearance of the report output.

- Change the background color of alternate rows
  Use the **BackColor** property of the Detail section (set in the Format event of the Detail section) to change

the background color of each row for better visibility of the table.

- Change the background color for selected rows
  Use the **BackColor** property of the Detail section (set in the Format event of the Detail section) to change the background color of selected rows. The background color changes when Reorder Level is below Ordered Units.

ViewerForm

Setting the **Dock** property of the Viewer control to **Fill** 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.

IListBinding.DataLayer project

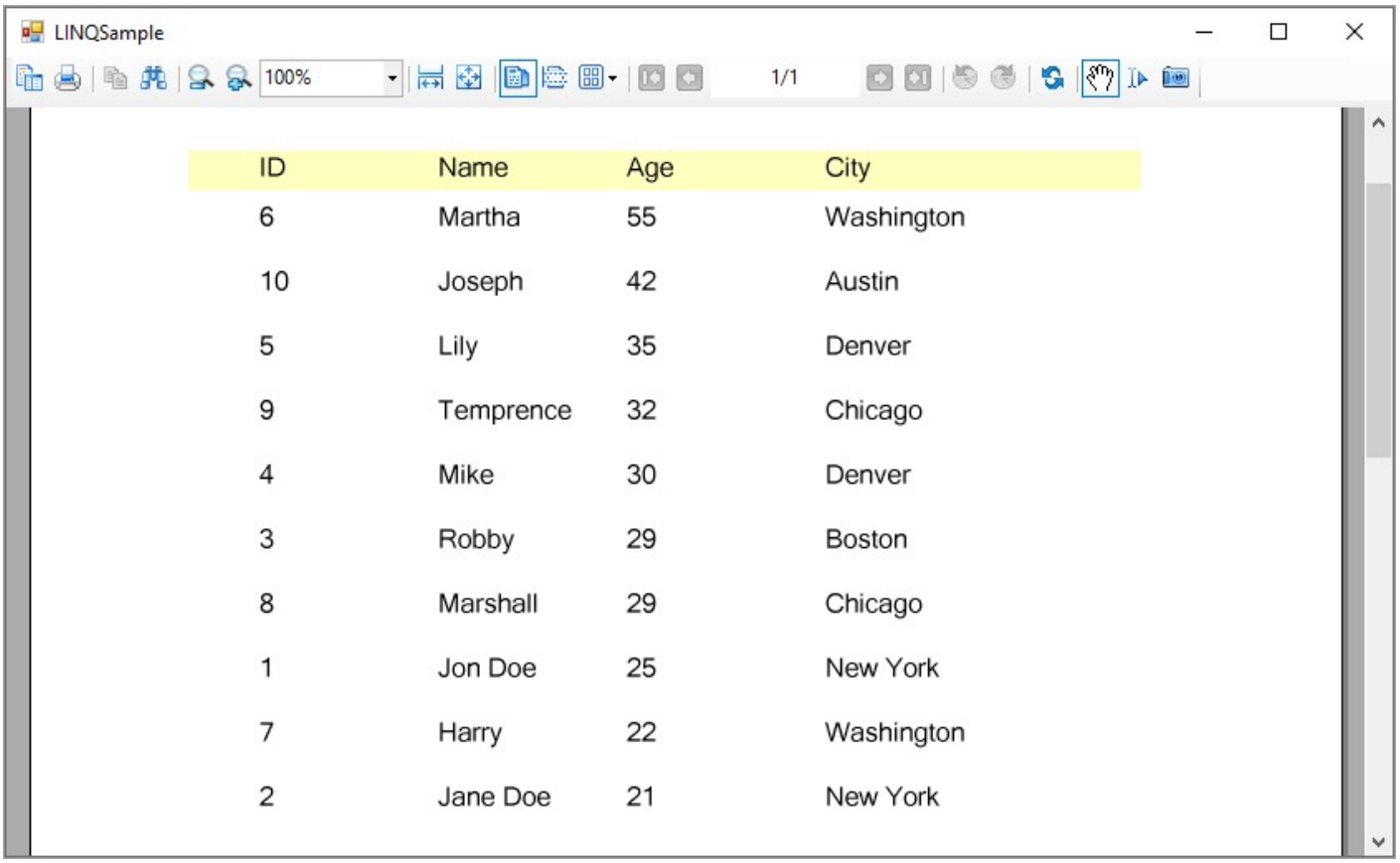
DataProvider Class
Implements the connection to the data base.

Product Class
Defines custom collection class.

ProductCollection Class
Implements the CollectionBase class to create a collection of Product class. The list of this collection stores data from the Products table.

LINQ

The LINQ sample demonstrates how to use LINQ in an ActiveReports report.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/LINQ/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/LINQ/C#

Run-Time Features

This sample uses a LINQ Query to sort recordsets in descending order of age. The resultant recordsets are converted to an IList and used as a data source for the report which is displayed in Viewer control.

Project Details

ViewerForm

Displays the report output results. ToList method is set in **DataSource** property of the report to extract objects that use LINQ.

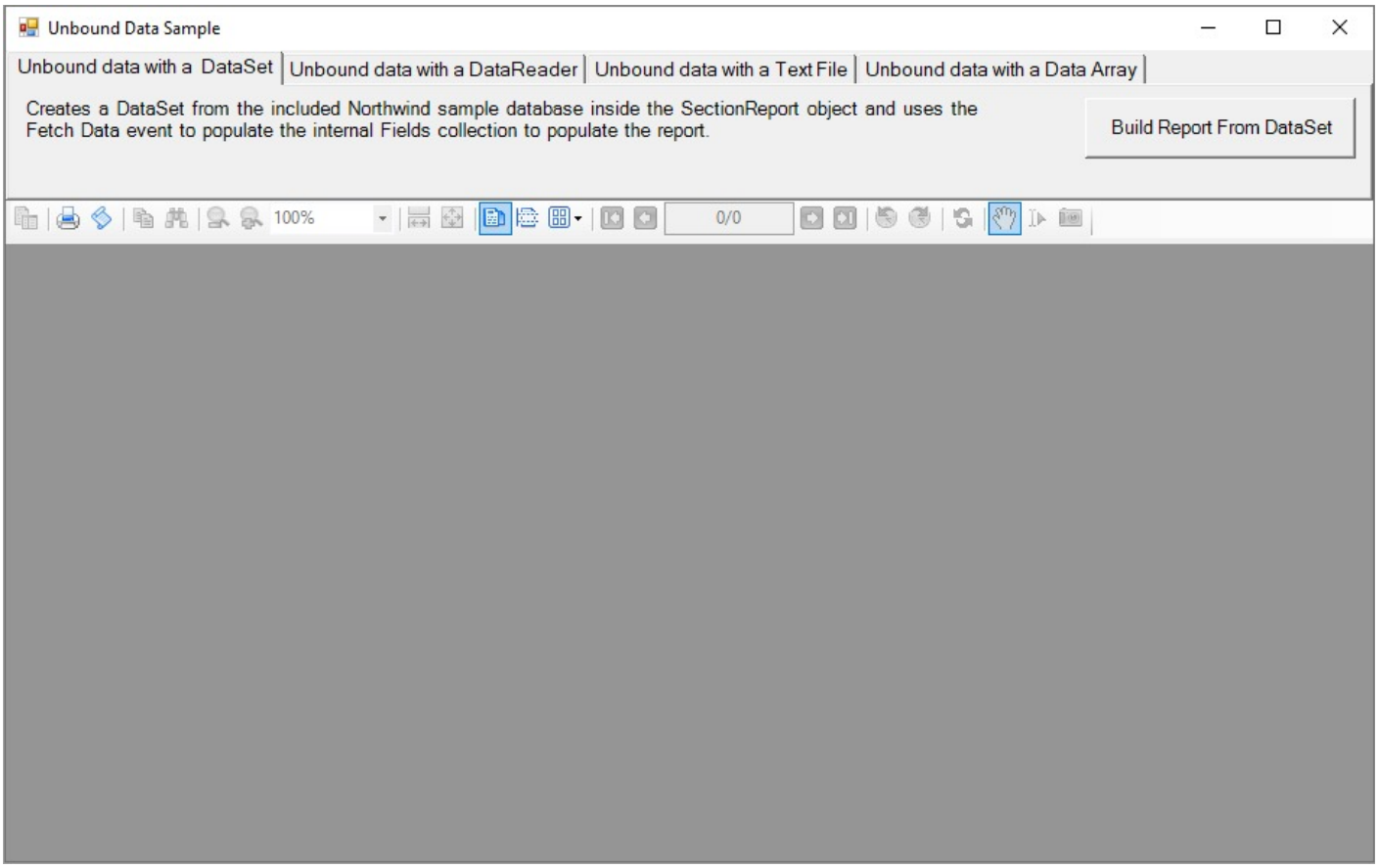
rptLINQtoObject report

The report DataSource is a list of Person constructor created from generic class. Creates a query to sort in descending order of Age.

Unbound Data

The Unbound Data sample demonstrates how to create a dataset for a Section Report and use the FetchData event to populate the Fields collection to display the report unbound data.

When you run the sample, the Viewer control displays the form with four tabs, each with a different dataset binding technique. Click to select a tab, and then click the **Build Report From** button to create the report with unbound data.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/UnboundData/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/UnboundData/C#

Run-Time Features

- **Unbound data with a DataSet**
  Creates a data set from the included Northwind sample database inside the SectionReport object and uses the FetchData event to populate the internal Fields collection to display the report data.
- **Unbound data with a DataReader**
  Creates a data reader from the included Northwind sample database inside the SectionReport object and uses the FetchData event to populate the internal Fields collection to display the report data.
- **Unbound data with a Text File**
  Sets the Invoice.txt file as a data source for the SectionReport object and uses the FetchData event to populate the internal Fields collection to display the report data.
- **Unbound data with a Data Array**
  Creates a data array from the included sample text file inside the SectionReport object and uses the FetchData event to populate the internal Fields collection to display the report data.

Project Details

MainForm

This is the main form of the sample that uses the ActiveReports **Viewer** control in the bottom section of the form, and a panel docked to the top contains four tabs, each with a different data binding technique. Click to select a tab, and then click the button on the tab to display the report with unbound data.

UnboundDAInvoice report

The Invoice report for the **Unbound data with a Data Array** option. The report consists of the page header, group header, detail, group footer and page footer sections. The detail section contains information on the order details, the group header provides grouping data functions by using its **DataField** property.

For the details on the Invoice report, see the Bound Data Sample topic.

UnboundDRInvoice report

The Invoice report for the **Unbound data with a DataReader** option. The report consists of the page header, group header, detail, group footer and page footer sections. The detail section contains information on the order details, the group header provides grouping data functions by using its **DataField** property.

For the details on the Invoice report, see the Bound Data Sample topic.

UnboundDSInvoice report

The Invoice report for the **Unbound data with a DataSet** option. The report consists of the page header, group header, detail, group footer and page footer sections. The detail section contains information on the order details, the group header provides grouping data functions by using its **DataField** property.

For the details on the Invoice report, see the Bound Data Sample topic.

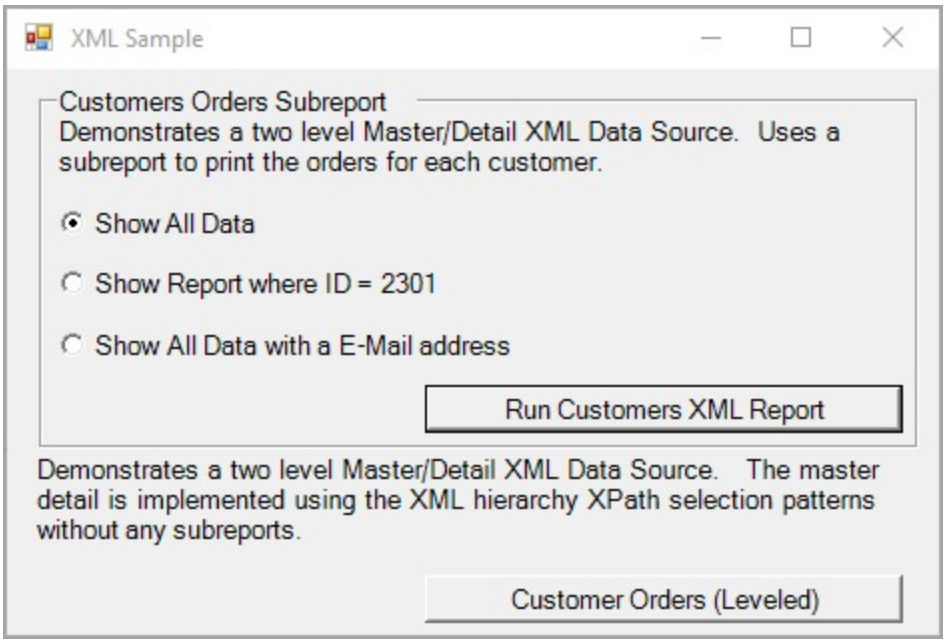
UnboundTFInvoice report

The Invoice report for the **Unbound data with a Text File** option. The report consists of the page header, group header, detail, group footer and page footer sections. The detail section contains information on the order details, the group header provides grouping data functions by using its **DataField** property.

For the details on the Invoice report, see the Bound Data Sample topic.

XML

The XML sample displays customer order list using the XML data source. The sample demonstrates how to create a report with XML data, using a SubReport or using the XML hierarchical structure.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/XML/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DataBinding/Section/XML/C#

Run-Time Features

When you run the sample, you will be asked to select between the following.

Run Customers XML Report
Demonstrates a two level Master/Detail XML Data Source. Uses a SubReport to print the orders of each customer. By selecting any of the radio buttons - **Show All Data**, **Show Report where ID = 2301**, or **Show All Data with an E-Mail address**, you can change the creation option of the generated report.

Customer Orders (Leveled)
Displays customer's orders by using the XML hierarchical structure. The Master/Detail is implemented using the XML hierarchy XPath selection patterns without any SubReports.

Project Details

StartForm

This is the main form of the sample. You see this form after you run the project and where you can select how to display XML data.

- Run Customers XML Report
  Displays the customers order list by using SubReports.To bind the CustomersOrders report to the XML data source, the valid XPath expression is entered in the **RecordsetPattern** field on the XML tab of the **Report Data Source** dialog.

- Customer Orders (Leveled)
  Displays customer's orders by using the XML hierarchical structure, which is demonstrated by the OrdersLeveled report. To bind this report to XML data, the path to the XML file is entered in the **File URL** field on the XML tab of the **Report Data Source** dialog and also the XML hierarchical structure like "../../@email" is specified in each field.

CustomersOrders report

This report embeds the srptOrders SubReport. Following settings are performed in this report.

- Embed the SubReport control
  Places the SubReport control in the Detail section and connects it to the Orders SubReport for displaying the orders list.

OrderItems SubReport

This report binds to the Subreport control of Orders report.

Orders SubReport

This is the report with a SubReport control that is bound to the OrderItems report. Following settings are performed in this report.

- Embed the SubReport control
  Places the SubReport control in the Detail section and connects it to the OrderItems report for displaying the list of orders.
  The Report property of the Subreport control is specified in the Orders_ReportStart event.
- Set the XML data source included in the SubReport
  Indicates a method to retrieve the record set using the NodeList property instead of the RecordsetPattern property of the XML data source on OrderItems report at design time.
  The NodeList property of the XML data source is set in the Detail_Format event.

OrdersLeveled report

Displays the list of customer's orders. Following settings are performed in this report.

- Groups data
  Groups data using the XPath pattern that represents the `hierarchical` structure of XML. DataField property of ghCustomers section and ghOrders section is set at design time.

ViewerForm

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.

Designer Pro

The samples in the Designer Pro folder demonstrate features provided with the ActiveReports professional edition.

Map
: This sample demonstrates how to work with Map control in ActiveReports.

End User Designer
: This sample demonstrates a custom end-user report designer that can be integrated in your applications to allow

users to modify report layouts.

Table of Contents

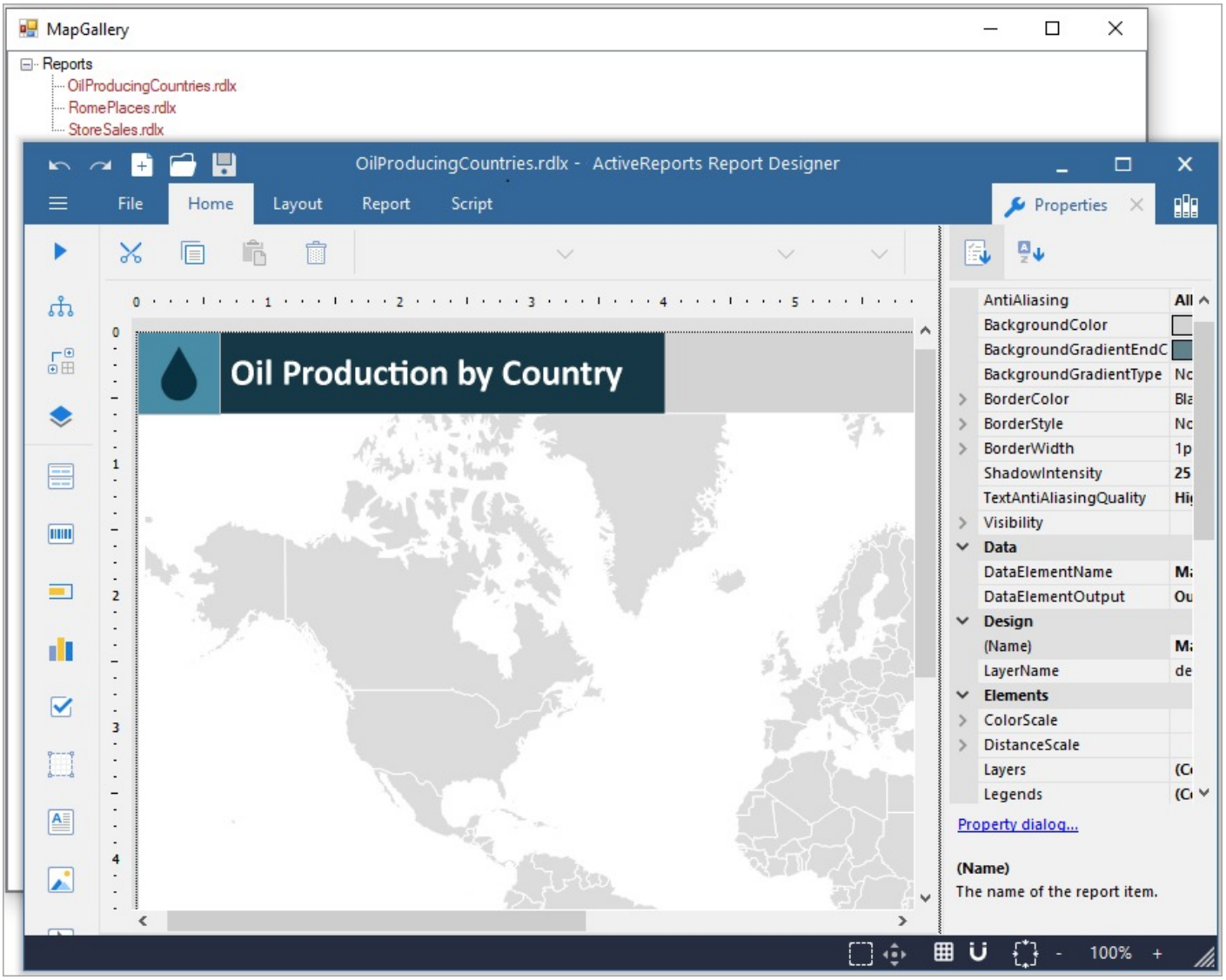
This sample demonstrates how to use TableofContents control in ActiveReports.

Reports Gallery

This sample demonstrates customizing End User Designer application.

Map

The Map sample demonstrates the basic functioning of the Map control with the help of four reports that explain the different features of the control. This sample is part of the ActiveReports Professional Edition.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DesignerPro/Map/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DesignerPro/Map/C#

Details

When you run this sample, the End User Designer shows a list of .rdlx reports at the bottom left of the form. Expand the **Reports** node to view reports under it and double-click a report to load it into the designer.

Report Form

This is the main form that appears when you run the sample. This form uses the ToolStripPanel, ToolStripContentPanel, Designer, Toolbox, ReportExplorer and PropertyGrid controls to create a customized ReportDesigner.
Right-click the form and select **View Code** to see how to set up the designer and create a blank Page Report. It also contains code that adds the reports to the TreeView control, loads a report into the Designer when the report is double-clicked, and checks for any modifications that have been made to the report in the designer.

Reports Folder

OilProducingCountries.rdlx: This report uses the FactBook shared data source connection to provide data.

It contains a Map control that visually displays the oil production in different countries of the world on a virtual earth background. The map control uses the color rule set on a polygon layer to differentiate parts of world according to their oil production capacity. These colors are defined using a color rule which is described in the legend at run time.

RomePlaces.rdlx: This report contains a Map control that visually displays famous places in Rome.

The map control uses Google maps in a Tile layer to provide a virtual earth background and a Point layer to plot famous places on the map using image markers. Clicking the image marker opens the web page for the selected place in Wikipedia.

StoreSales.rdlx: This report contains a Map control that visually displays the sales of different stores in the US.

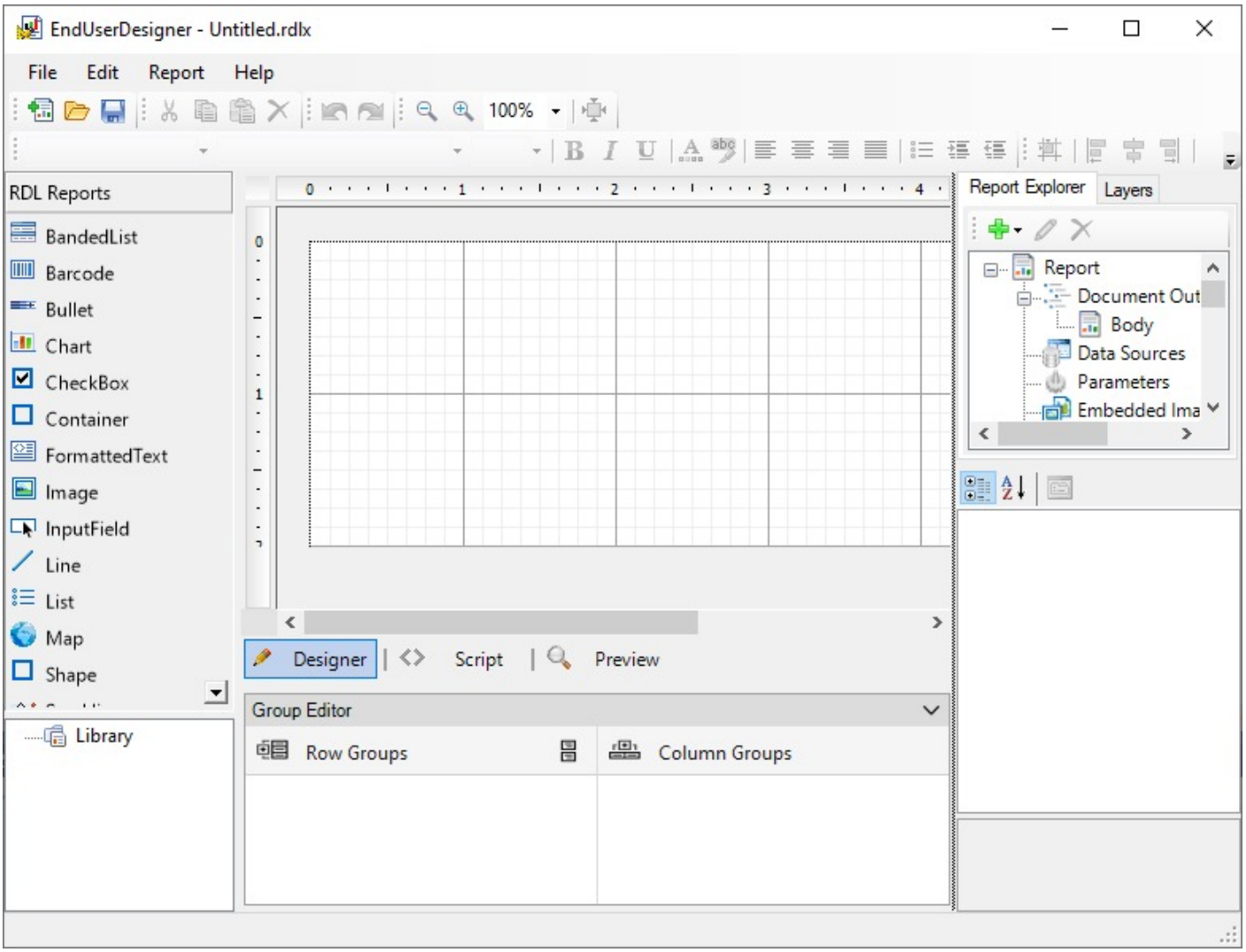
The Map control uses the built-in USA map template. The polygon layer defines the country and state boundaries while the point layer is used to plot store locations. The Point layer uses the marker size rule to differentiate stores according to their profits. The different sized markers used for plotting stores are defined in a legend that appears on the map at run time. The point layer also uses the drill-through link (the Action is set to Jump to report) that opens a specific store report (StoreReport2.rdlx) when the marker is clicked.

StoreReport2.rdlx: This report uses the **Reels** shared data source connection to provide data.

It opens on clicking a specific store location that is indicated with a marker. The Table data region in the report uses a drill-down link to display the list of employees in the selected store. The report uses the chart data region to display the sales for the store. The Reels logo that appears on the report in the Image control is embedded within the Reels theme.

End User Designer

The EndUserDesigner Sample demonstrates how to set up a custom end-user report designer using the Designer, ReportExplorer, Layer List, ToolBox, ReportsLibrary, and GroupEditor controls. This Sample is part of the ActiveReports Professional Edition.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DesignerPro/EndUserDesigner/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DesignerPro/EndUserDesigner/C#

Run-Time Features

When you run the sample, the End User Designer appears in the Viewer control. This report designer provides the functionality of the ActiveReports Designer and supports the report layouts as elaborated in Report Types page.

The End User Designer lets you create report layouts and edit them at design time or runtime. The Designer includes the Property Window with extensive properties for each element of the report, the Toolbox is filled with report controls, the Report Explorer with a tree view of report controls, the Reports Library displaying report parts (group of controls) in a report, and a Group Editor displaying row and column groups for Tablix data region. Page reports and RDLX reports provide the Layer List in a tabbed window with the Report Explorer. The Layer List window displays a list of layers in the report along with their visibility and lock options.

See Report Parts, Layers, and Tablix for more information.

The project consists of following forms:

ExportForm

This is the form with the **Export** dialog for Page Report, RDLX report and Section Report.

A user sees the **Export** dialog under the **Preview** tab in the **File** menu > **Export**. This dialog allows to select the export type and to browse for the file location in local folders where the report is exported. See Export in Desktop Viewers for details on the type of export formats supported in Section Report, Page Report, and RDLX report.

Control	Name	Description
ComboBox	cmbExportFormat	The **Export Format** combo box that allows to select options for the report export type.
Button	btnOK	The **OK** button in the lower part of the ExportForm.
Button	btnCancel	The **Cancel** button in the lower part of the ExportForm.
PropertyGrid	exportPropertyGrid	Provides interface for export options of each export type.
SaveFileDialog	exportSaveFileDialog	The **Save File** dialog that allows to specify the file name for saving an exported report file.
Label	lblExport	The **Export** label in the header of the ExportForm.
Label	lblExportFormat	The **Export Format** label of the Export Format combo box.
Label	lblExportOptions	The **Export Options** label of the Export property grid.
Label	lblSelectExportTxt	The **Select Export** text that describes the purpose of the ExportForm.
SplitContainer	exportSettings SplitContainer	Represents a movable bar that divides the display area of the ExportForm into two resizable panels - the ExportForm header panel and the ExportForm panel with the Export Format combo box and the Property Grid.
SplitContainer	exportHeader SplitContainer	Represents a movable bar that divides the Export Format section consisting of the Export Format combo box with the Export Format label and the Property Grid of ExportForm.

Right-click the ExportForm in the Solution Explorer and select **View Code** to see the code implementation for the Export form.

EndUserDesigner form

This is the form with a basic end-user report designer that contains the following elements. These elements are dragged from the Visual Studio toolbox onto the form.

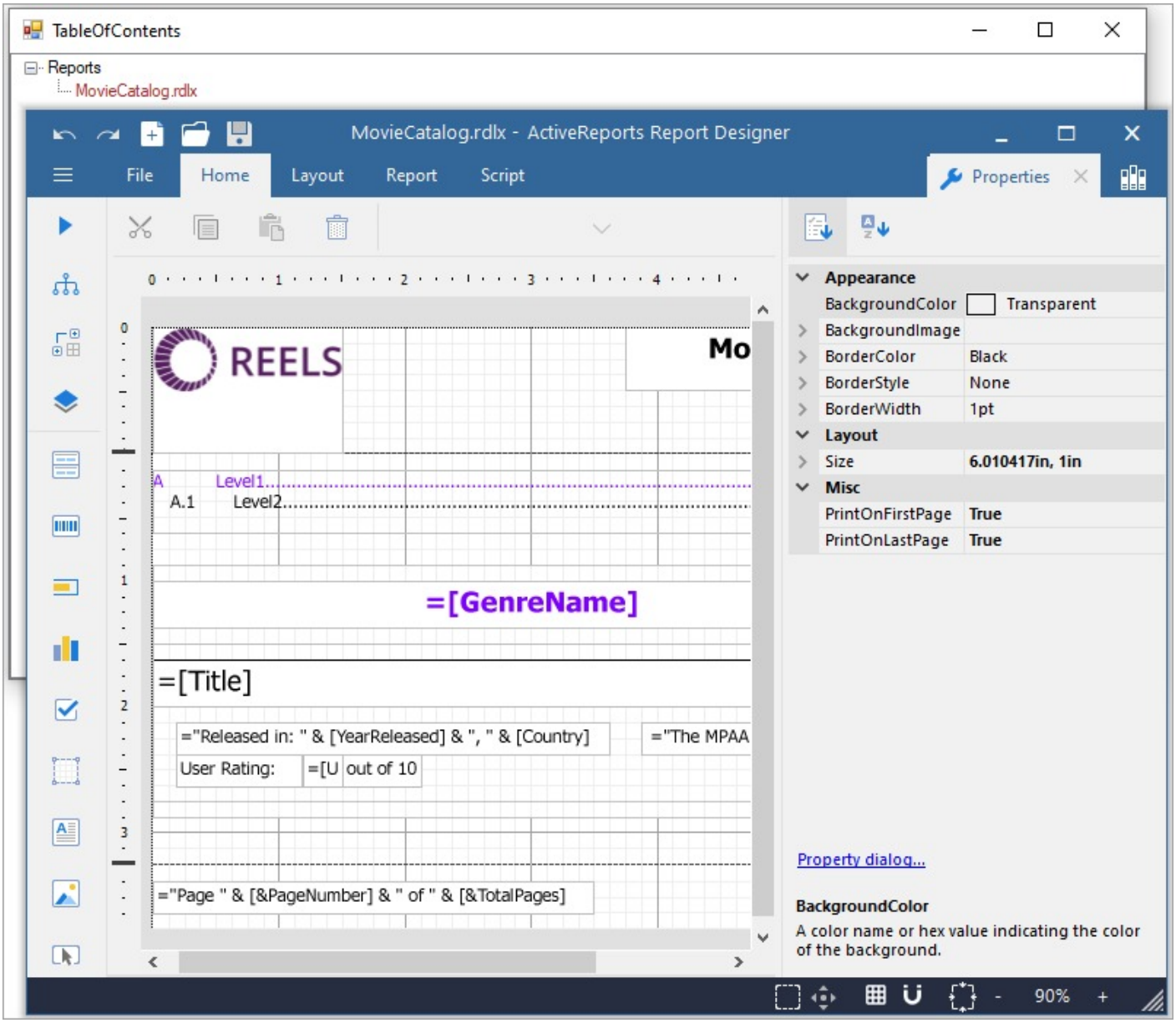
Control	Name	Description
Designer	reportDesigner	The Designer control that allows you to create and modify a report.
ReportExplorer	reportExplorer	Gives you a visual overview of the report elements in the form of a tree view where each node represents a report element.

TabControl	reportExplorerTabControl	Represents a movable bar that divides the display area of the designer into two tabs - the Report Explorer and the Layer List
PropertyGrid	reportPropertyGrid	Provides an interface for each element of the report.
Toolbox	reportToolbox	Displays all of the controls specific to the type of report that has focus.
LayerList	layerList	Represents a list of Layers in the report along with their visibility and lock options.
SplitContainer	mainContainer	Represents a movable bar that divides the display area of the Designer into two resizable panels.
SplitContainer	designerExplorerPropertyGridContainer	Represents a movable bar that divides the display area of the designer into two resizable panels -the toolbox and the toolstrip.
SplitContainer	bodyContainer	Represents a movable bar that divides the display area of the Viewer into two resizable panels.
SplitContainer	explorerPropertyGridContainer	Represents a movable bar that divides the display area of the designer into two resizable panels - the report explorer and the property grid.
ToolStripContainer	toolStripContainer	Provides a central panel on top of the Designer to hold the Toolstrip element with the menu items.
ReportsLibrary	reportsLibrary	Displays all reports and included report parts.
GroupEditor	groupEditor	Shows the row and column groups in a Tablix data region.

Right-click the EndUserDesigner form in the Solution Explorer and select **View Code** to see the code implementation for the End User Designer.

Table of Contents

The TableofContents sample demonstrates the basic features of the TableofContents control. This sample is part of the ActiveReports Professional Edition.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DesignerPro/TableOfContents/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DesignerPro/TableOfContents/C#

Details

When you run this sample, the End User Designer appears with a MovieCatalog.rdlx under the Reports node. The report contains a TableOfContents control which displays a list of movie titles along with their page numbers under each genre. On clicking the movie title, the details on the selected movie are displayed.

Report Form

This is the main form that appears when you run this sample. This form uses the ToolStripPanel, LayerList, TreeView, ToolStripContentPanel, Designer, Toolbox, ReportExplorer and PropertyGrid controls to create a customized ReportDesigner.

Right-click the form and select **View Code** to see how to set up the designer. It also contains code that adds the reports to the TreeView control, loads a report into the Designer when it is double-clicked, and checks for any modifications that have been made to the report in the designer.

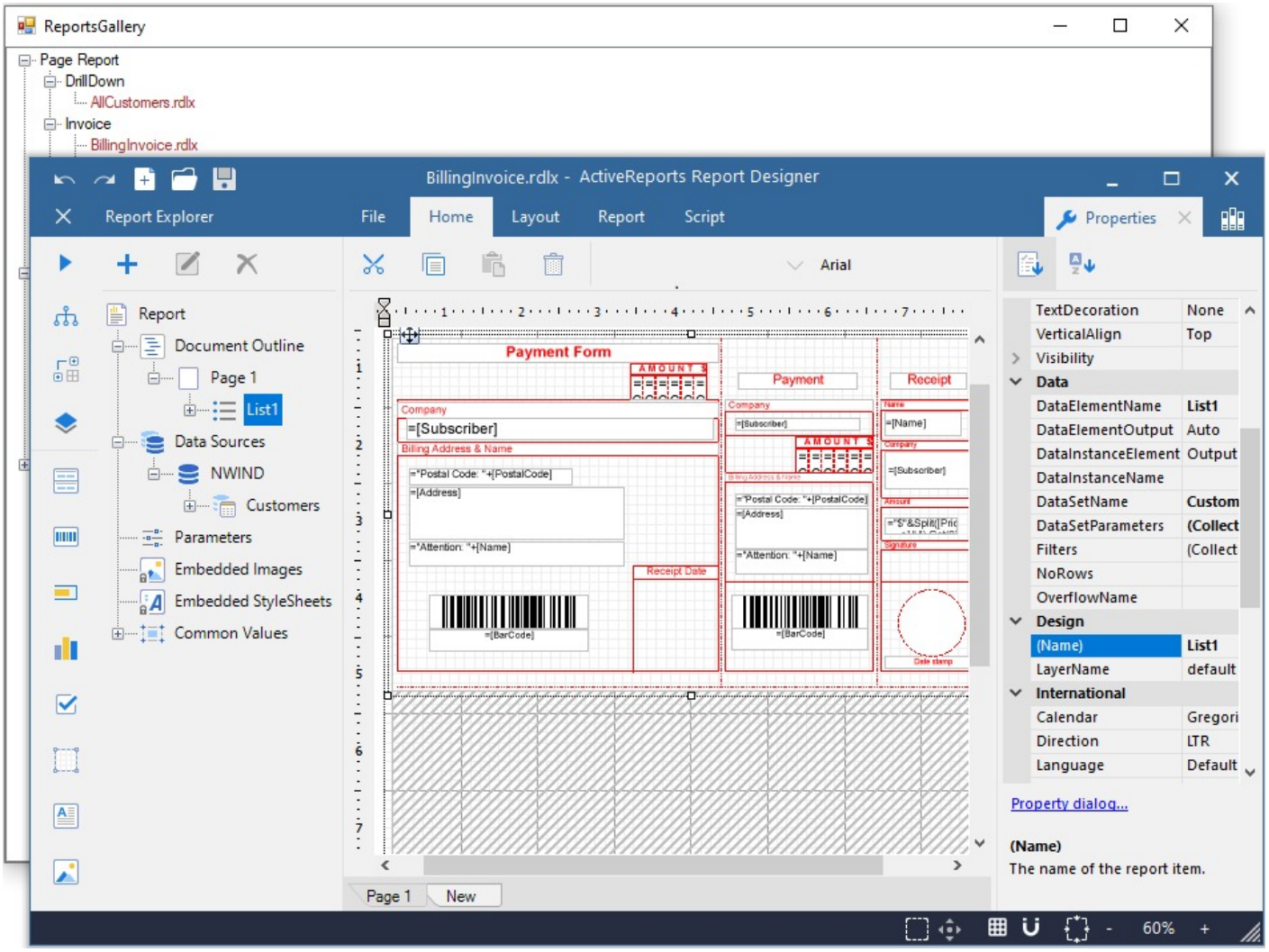
Reports Folder

MovieCatalog.rdlx: This report uses the Reels data source connection to provide data.

The report makes use of TableOfContents, Image, TextBox, Label and List controls to display the layout of the report. TableofContents control displays an organized hierarchy of movie titles under each genre with two heading levels. Clicking the genre name or the movie title takes you to the corresponding page number that contains the details. The Reels logo that appears on the report is embedded within the Reels theme.

Reports Gallery

The Reports Gallery sample demonstrates how to customize the End User Designer application by adding the TreeView control to display a list of categorized reports. At run time, the user can double-click any report out of the three categories: Page, RDLX, and Section, and load it into the designer.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/DesignerPro/DesignerPro/ReportsGallery/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/DesignerPro/DesignerPro/ReportsGallery/C#

Run-Time Features

When you run this sample, the End User Designer appears with a list of report categories at the bottom left of the Form. Expand each category to view the reports under it and double-click any report to load it into the designer. You can also go to the Preview Tab to view the report.

Project Details

ReportsForm

This is the main form that appears when you run this sample. This form uses the ToolStripPanel, ToolStripContentPanel, Designer, Toolbox, ReportExplorer, TreeView, PropertyGrid and LayerList controls to create a customized unified ReportDesigner.

Right-click the Form and select **View Code** to see how to set up the designer and create a blank Page Report. It also contains code that attaches the Toolbox, ReportExplorer, PropertyGrid and LayerList controls to the Designer, inserts DropDown items to the ToolStripDropDownItem and sets their functions. Finally, it also contains code that adds the reports to the TreeView control, loads a report into the Designer when the report is double-clicked, and checks for any modifications that have been made to the report in the designer.

Reports folder

Reports folder consists of three subfolders - Page Report, RDLX report and Section Report, each containing a set of reports that highlight the major features of the corresponding report types.

Page Report folder

This folder contains several subfolders that illustrate the use of Page reports in different scenarios.

DrillDown

AllCustomers report: This report uses the NWind shared data source. It uses the **Table** data region to display customer's contact details. The **Textbox** displaying the **CustomerID** field in the detail row of the Table is used to set a drill-through link to navigate to the **CustomerDetails** report.

Invoice folder

BillingInvoice report: This report uses the NWind shared data source. It showcases a billing invoice layout commonly used in convenience stores. The report mostly contains Label, TextBox and Line controls in its layout. It also includes an **EAN128FNC1** barcode due to its high reading accuracy.

Invoice1 report: This report uses the NWind shared data source. It includes a BandedList and few TextBox controls to create the Invoice layout for displaying customer transactions. Both the page and the BandedList control are grouped by the **OrderID** field. One of the text boxes in the footer section of the BandedList control uses the Sum function to display the grand total of all transactions.

Invoice2 report: This report uses the NWind shared data source. It uses a Table, few TextBoxes and Shape controls to create the Invoice layout for displaying customer transactions. The report page is grouped by the **CustomerID** field, therefore all transactions made by a customer appear together based on the ID. One of the text boxes placed inside the Container control at the bottom of the report uses the Sum function to display the grand total of all transactions.

Invoice_Grouped report: This report uses the NWind shared data source. It uses the Table, few TextBox and Label controls to display customer transactions in an Invoice. The data is grouped on the **CustomerID** field, so that all transactions made by a customer appear together based on their ID.

Invoice_Parameters report: This report uses the NWind shared data source connection and two datasets to provide data. It is similar to the **Invoice_Grouped** report with an additional parameters feature. It uses parameters set on the **CompanyName** field to filter data on report preview.

Other

BarCode report: This report demonstrates all barcode types that are supported by ActiveReports. The barcode types are presented in the Table data region, using a single page layout. The rows of the table use alternate background colors (grey and white). At run time, the Barcode report displays one page that fits the table with all the sample barcodes.

Catalog report: This report uses the NWind shared data source. It shows a multi page layout spread over four pages in the report. The layout in **Page1** and **Page2** contains Image, Label and Textbox controls to display introductory text. The layout on **Page3** contains a List data region with TextBox controls and a Table to display product details for each product category. The List is grouped by the **CategoryID** field to filter products by their category and its FixedSize property is set to fit in excess data. The layout on **Page4** (that appears as page 9 at run time) uses Textbox, Shape and Line controls amongst others to create an Order Form, which a user is to fill manually.

CellMerging report: This report uses the NWind shared data source. This report demonstrates cell merging in Tablix data region, where cells with same values are merged automatically to avoid showing duplicate values. The row group area contains three groups that are nested in a parent/child relationship to display the row group data. The Country (parent) and City group (child) values are merged automatically to remove duplicate data values.

DeliverySlip_theme report: This report uses the **Seikyu2** shared data source. It uses the TextBox, Label, Container controls and two Table data regions to display the invoice information. The report uses two themes and has its **CollateBy** property set to **ValueIndex** to determine the order in which the report pages are rendered. Some TextBox controls on the report also use the Sum function to display the total price information for each Invoice. The page is grouped on the **EstimateID** field, so the invoices are sorted by **EstimateID**.

EmployeeSales report: This report uses the NWind shared data source. It contains the Chart and Table data regions to display sales by each employee for the year 1997. A column chart shows the graphical representation of sales by each employee while the Table lists down the exact sales figures. One of the TextBox controls on the report also uses the Sum function to display the grand total of all sales.

Enterprise Reports - Marketing Plan Data: This report displays information for the Marketing Plan. The report contains embedded XML data. Table, TextBox, Line, Image, and Shape controls are used to display data. The status of each task is represented through color coding using expressions.

IRS-W4 report: This report uses the IRS XML data source to provide data to the report. It mainly contains Textbox, CheckBox, Shape and Line controls to create the layout of a tax form used in the US.

Letter report: This report uses the NWind shared data source. It contains an Image, FormattedTextBox, Table and an OverflowPlaceholder controls to create a layout for a letter. The FormattedTextBox control uses text with HTML tags to display content. The Table displays OrderID with order dates and order amount and is linked with the OverflowPlaceholder control to display excess data on the same page. The Reels logo, displayed on the report inside the Image control, is embedded within the Reels theme and the TextBox placed below the Image control uses a Global expression to display the current date. The layout page is grouped by the **CustomerID** field to filter data on each report page according to individual customers.

MyOrdersReport: This report uses MyOrders.csv as a data source to demonstrate the native support for CSV data providers. It contains a Table data region that displays orders with data fetched from the CSV data source.

PurchaseReport: This report uses the NWind shared data source. It contains the Textbox and Table controls to display purchase details for each company. The layout page is grouped by the Company_CD field to filter data on each report page according to the company. The Textbox control placed below each table column uses the **Sum** function to display totals. The FixedSize property of the Table is set to fit in excess data.

ReelsTablix report: This report uses the Reels shared data source. It contains the Tablix data region to display a sales report for each country, city and media type by year. The Tablix data region uses row grouping to group the data by Country, City and MediaType**,** and column grouping to group the data by Years and Quarters.

ResourceConsumptionByYear report: This report uses MostPopulatedCountriesEnergyUsageByYear shared data source. It illustrates composite charts by using a Chart data region that shows annual Natural Resource Consumption - Oil and Electricity - versus Population size. Natural Resource Consumption and Population size are plotted on the two Y axes represented by two different chart types - Column and Line. The report also contains a parameter which lets users choose the country for which they want to view the data.

SalesReport: This report uses the Reels data source connection and two datasets that fetch data through a Stored

Procedure. The layout of the report uses the Chart to display sales and profit for the selected date range and the Table to display the numeric values of the same. The table also uses the **DataBar** function to plot profits graphically. The Reels logo, displayed on the report inside the Image control, is embedded within the Reels theme. This report also uses two nullable parameters to select the range of dates.

TablixSample report: This report uses the NWind shared data source. It contains the Tablix data region that displays an orders report with product category and names showing quarterly and yearly orders. The Tablix uses a nested row grouping to group the Tablix data region by CategoryName and ProductName, and nested column grouping to group the Tablix data region by Years and Quarter. The tablix body area displays the aggregate Sum for each category and the grand total of order amount.

TackSeal report: This report uses the NWind shared data source. It contains the OverflowPlaceholder control and the List data region to create a columnar display of tack seals with postal barcodes. The List data region has its **OverflowName** property set to **OverflowPlaceHolder1** and the OverflowPlaceHolder1 control has its **OverflowName** property set to **OverflowPlaceHolder2** to assure correct display of columnar data display within the report page.

RDLX report folder

This folder contains several subfolders that illustrate the use of RDLX Reports in different scenarios.

Dashboard

CallCenterDashBoard report: This report uses the CallCenter shared data source. It contains uses the Bullet control to indicate when the data is approaching or past a warning range and the Sparkline controls to indicate daily trends in key pieces of performance and sales data. It also uses the Icon Set data visualizer to indicate the warning levels for key performance data.

MarketDashBoard report: This report uses the MarketData shared data source. It contains the Sparkline control to display stock price trends over the last 30 days. The Sparkline allows investors to see trends without knowing what actual values are associated with each point.

TeamList report: This report uses the FootballStatistics shared data source. It contains the List control that displays a list of team names with drill-through links to navigate to the **TeamStatisticsDashboard.rdlx** report that displays the selected team's statistics.

SalesDashboard report: This report uses the SalesResult shared data source. It consists of two datasets to display multiple Chart controls and a Tablix data region to visualize the sales performance data. This report illustrates the Galley-mode feature where all of the report contents can be previewed in a single scrollable page.

Factbook

CountryFacts report: This report uses the Factbook shared data source. It contains a List data region grouped on CountryID that groups data by **Country**. The report also contains a map image whose **Value** property is set to the expression with the **MapCode** data field from the XML data source. It contains a hidden parameter that is used to accept the **Country ID**. This report is called in other reports by drill-through links or as a subreport, so the Country ID is passed silently. The default ID is the World. The dataset has a filter that retrieves only countries, specified by the report parameter. From each textbox with category under **Energy Production / Consumption** you can see the Image control that uses the Icon Set data visualizer to flag energy categories where consumption is greater than production.

LifeExpectancyByGdpAndMedianAge report: This report uses the Factbook shared data source. It contains the Tablix data region to compare the average life expectancy based on the category where the Median age and GDP fall. The tablix data region consist of a row group **GDP of country** and a column group **Median Age** to display the data.

Top10CountriesByGdp report: This report uses the FactbookSortedByGdp shared data source and shows the Top 10

countries by GDP. It contains a List data region with an image in the Image control that uses the RangeBar function to create an ad-hoc horizontal bar chart.

Reels

CustomerMailingList report: This report uses the Reels shared data source. It includes a Container control along with TextBox and Barcode controls to display a mailing list of customers. The **Columns** property of the Body section is set to **3** to display mailing labels in three columns.

CustomerOrders report: This report uses the Reels shared data source. It contains the List and Table data regions that group data by **CustomerID** and **SaleID** respectively to display order information. The **SalesAmount** textbox of the Table has its **Value** property set to Sum function to calculate the total for each order (as a table group subtotal). The **YearTotal** textbox of the Table also has its **Value** property set to Sum function to calculate the total of pre-tax sales. The Reels logo that is displayed on the report is embedded within the Reels theme. The page number in the PageHeader section is reset every time a new customer is displayed. This report also contains a subreport, CustomerOrdersCoupon.rdlx.

DistrictReport: This report uses the Reels shared data source. It contains the Chart and Tablix data region to display the number of sold items and the profit for each month during the two year span (2004-2005) for the selected district. The Tablix data region consists of a row group SaleDate and a column group StoreName. The report parameter determines which district to display and uses the available values from the second report dataset **SalesData**.The **StoreName** textbox in the report body uses the drill-through link (the **Action** is set to **Jump to report**) that opens the **StoreReport.rdlx** with more information.

DistrictSales report: This report uses the Reels shared data source. It contains the BandedList data region that groups data by **SaleYear**, **DistrictID** and **RegionID** to display district sales details. **Sum** function is used to get the total sales at the District, Region, Year levels and also to display a grand total. The Reels logo that is displayed on the report is embedded within the Reels theme.

Filmography report: This report uses the Reels shared data source. It contains nested List data regions that group data by **MovieID** and **MoviePersonID** to return a distinct number of movies with the selected actors. The report contains two cascading parameters that narrow down the number of actors displayed in the list first based on the alphabet with which the actor's name begins and then based on the actor's name. The Reels logo that is displayed on the report is embedded within the Reels theme.

GenreSales report: This report uses the Reels shared data source. It contains a Tablix data region to display the units sold for each genre, in each year and each quarter. The Tablix report data consist of a row group **GenreName** and a column group **SaleDate.Year** to display the data. The report also uses the plain column **Chart** to display the number of titles sold for each genre. This report uses the multi value parameter that allows you to select more than one genre for displaying sales data.

GenreStatistics report: This report uses the Reels shared data source. Median and Mode aggregate functions are used to show the middle values in a set of data as well as the most commonly occurring value. It also uses the **ReelsConfidential.rdlx-master** report to provide standard page headers and footers. The Reels logo that is displayed on the report is embedded within the Reels theme.

MonthlySalesReport: This report uses the Reels shared data source. It contains a Table data region that groups data by **DistrictID**. The Textbox controls in the Table have their **Value** property set to expressions to display the total of sales for each district and region as well as the totals of all districts within a region for a given month. It also uses the Plain Line Chart with the data grouped and sorted by **Day** for each **SaleDate** to display sales and profit for the selected month. This report uses query based parameters to select the month and region for displaying data. The Reels logo

that is displayed on the report is embedded within the Reels theme.

MovieCatalog report: This report uses the Reels shared data source. It contains the Image, TextBox, TableofContents and List controls to display the list of movies in a catalog. This report uses the TableofContents control to display, an organized hierarchy of the report heading levels and labels along with their page numbers, in the body of a report. It also uses the TextBox and Image control to display the layout of the report. The Reels logo that is displayed on the report is embedded within the Reels theme.

MovieReport: This report uses the Reels shared data source. It contains four List data regions with groupings. The **MovieList** is grouped by **MovieID**, the **GenreList** is used to display the genre names and is grouped by **GenreID**, the **CrewList** is used to display the title and is grouped by **CrewTitleID**, and finally the **CastNameList** is used to display the cast and crew and is grouped by **MoviePersonID**. This report uses cascading parameters. The first parameter asks to select which letter the movie titles starts with, and then the second parameter asks to select a movie to display. The **CrewName** textbox in the report Body uses the drill-through link (the **Action** is set to **Jump to report**) that opens **Filmography.rdlx** with more information on the selected person. The parameters of this report are passed to the Filmography.rdlx report. The Reels logo that is displayed on the report is embedded within the Reels theme.

RegionPerformance report: This report uses Reels.db database. It contains the Table data region that uses the **Region** value to group the report data and the **SalesAmount** value to sort the report data. It also uses filtering to filter the report data by the **RegionID** value. Textbox controls in the Table have their **Value** properties set to Sum functions to display the total of sales amount for each region. The Reels logo that is displayed on the report is embedded within the Reels theme.

ReorderList report: This report uses Reels shared data source. It contains the Table data region without any grouping. The Table detail row has its **BackgroundColor** property set to the expression to create a light yellow bar report. The Reels logo that appears on the report is embedded within the Reels theme.

SalesByMediaType report: This report uses the Reels shared data source. It contains the List data region that is grouped by Image to display data. The list also includes the Table data region that groups its data by **MediaID**. The report also contains the Plain Column Chart to display sales and profit by media type and embedded images for each category. This report uses the **ReelsConfidential.rdlx-master** report to render the report page header and footer.

SalesByRegion report: This report uses the Reels shared data source. It contains the Tablix data region and subtotals to display the number of units sold and profit for each region by year and quarter. The Tablix report data consist of a row group **Region** and a column group **SaleDate.Year** to display the data. The report also uses the Plain Column Chart to display the annual profit for each region. Data in the Chart is also grouped by **Region** and **SaleDate.Year**.

SalesReceipt report: This report uses the Reels shared data source. It contains a Table data region and three Container controls nested in the List data region. The List is grouped by **SaleID** to produce the body of the receipt. The **salesTaxLabel** and **totalSalesTax** text boxes use the Sum function that totals the amount due in the list and adds tax to the sum of a field for the grand total due. The Reels logo that is displayed on the report is embedded within the Reels theme.

SalesReport: This report uses the Reels shared data source. It contains a Table and Chart data regions to display totals of sales and profit for the selected date range. The Table also uses the Data Bar function to plot the profits. The Chart and Table data is grouped by **Month** and **Year**. The **Month** textbox uses the drill-through link to display **MonthlySalesReport.rdlx** with more information on the selected month. This report uses parameters that allow the null value to select the range of dates. The Reels logo that is displayed on the report is embedded within the Reels theme.

StorePerformance report: This report uses the Reels shared data source. It identifies stores with profits above or below expectations. It also has two Image controls that display the database images and use the IconSet function. This report uses the **ReelsConfidential.rdlx-master** report to render page headers and footers.

StoreReport: This report uses the Reels shared data source. It contains the Table, List and Chart data regions to display sales for each employee. The Table and Chart data is grouped by **EmployeeID**. The Table uses hierarchical grouping to show the relationship between employee and supervisor for each store. The **FirstName** textbox has its **Padding> Left** property set to the Level function that leaves a space 15 pixels wide to the left of the control. This is the drill-down report where the **Visibility > Hidden** property of the Table detail row is set to **=Fields!Supervisor.Value <> 0** and the **Visibility > ToggleItem** property is set to the **FirstName** data field. The expression in the **Visibility > Hidden** property calculates whether the supervisor field returns 0, so only the supervisor's name is displayed initially. By clicking the toggle image next to the supervisor's name, the rows with details about employees are displayed. The report uses a cascade of parameter values - **Region**, **District** and **StoreNumber**. Each parameter depends on the value of the previous parameter and each comes from a separate dataset. The Reels logo that is displayed on the report is embedded within the Reels theme.

TopPerformers report: This report uses the Reels shared data source. It contains two Table data regions to display the top and bottom performers based on the movies sales. Each Table data is grouped by **MoviePersonID** and **MovieID** (nested grouping). The TopN filter is applied to one table and the BottomN filter is applied to another. The number of items returned by each of the filters is specified in report parameters. This report also uses two integer parameters to alter the number of items displayed in each table. The parameters use default values, which are passed to textboxes of the Table Headers. This is the drill-down report where the second Table Group Header in each Table has its **Visibility > Hidden** property set to **True** and the **ToggleItem** property set to the **PerformerName** textbox. By clicking the toggle image next to the name, the rows containing details about performers are displayed.

Others

AnnualPortfolioChart report: This report uses the MostPopulatedCountriesEnergyUsageByYear shared data source. It illustrates composite charts by using a Chart data region that shows Annual Stock Performance. The Trading Volume and Trading Value are plotted on two Y axes represented by two different chart types, Column and Line.

Financial Reports - BalanceSheetReport: This report displays the company’s assets and liabilities. The report contains embedded JSON data. Table, Textbox, and Image controls are used to display data. The total assets and liabilities are calculated using expressions.

Financial Reports - CashFlowReport: This report displays the company’s cash flow. The report contains embedded JSON data. Table, Textbox, Line, Image, and Shape controls are used to display data. The VB code in the script transforms the numerical values in the accounting format.

Financial Reports - IncomeStatementReport: This report displays the income statement for the Year End (December 2017). The report contains embedded JSON data. Table, Textbox, and Image controls are used to display data. The net income and total revenue are calculated using expressions.

Financial Reports - IncomeStatementReport2: This report displays the company’s sales and expenses. The report contains embedded JSON data. Table, Textbox, Image, and Shape controls are used to display data. Total sales, Gross profit, and Net profit are calculated using expressions.

Flight On-time Performance Report: This report shows on-time performance of US airlines. The report uses nested data regions bound to different datasets from the following data sources:

- States.xml - XML file stores information about geographical data of regions and states US.
- FlightDetails.csv - CSV file stores flight information - flight date, arrival time, departure time, etc.
- Airlines.json - JSON file stores unique airlines names corresponding to an airlineID.

The controls that are included in this report are List, Container, Table, and Tablix, where Container, Table, and Tablix are nested inside List. These controls show data as described below:

- List displays geographical information - Region, State Name, and Number of Airports, from States.xml data file through Tablix data region and Textbox control inside Container control.
- Table displays flight information - Number of flights, on-time arrival and departure details, from FlightDetails.csv.

The mapping for 'DataSet Joins' is created by adding a filter in Table on Region fields. This lists the flight information in Table control that flew in a particular region displayed in the List.

Medical Reports - BloodTestReport: This report displays the patient's blood test results. The report contains embedded XML data. Table, Textbox, and Image controls are used to display data. Results and Reference Intervals are calculated using expressions.

Medical Reports - PatientDiseaseSummaryReport: This report displays a summary of the patient's diseases and allergies. The report contains embedded JSON data. Table, Textbox, and Image controls are used to display data.

Telecom Reports - TelephoneBillSample: This report displays the company's telephone bill. The report contains embedded XML data. Table, Checkbox, Image, and Textbox controls are used to display data. Expressions are used to calculate totals.

Section Report folder

BarCode report: This report displays all barcode types that are supported by ActiveReports.

- **PageHeader section**: This section contains the table columns to display the report header. The **Text** property of the two Textbox controls, define the name of the two table columns.
- **Detail section**: This section contains a list of all the **Barcode** types that are supported by ActiveReports. The barcodes are presented in a table that contains two columns, the one with the **Textbox** control displays the barcode name, and the other one with the Barcode control displays the barcode image.

Invoice1 report: This report calculates the total amount for each customer along with purchase details. The report also displays the average of the previous invoice amount, the current invoice amount and the consumption tax.

- **PageHeader section**: This section contains the **Label** and **Textbox** controls. It uses the bound data fields to calculate the invoice information in the header. The values of the **Excise** and **BillTotal** Textbox controls are set in the **Script** tab and are calculated in the BeforePrint event.

> **Note**: This sample uses bound fields on the PageHeader section, assuming that the value of all records to be displayed on a page does not change. However, it is not recommend that you place bound fields in the PageHeader or PageFooter sections because these sections are displayed on a page once. For the same reason it is not recommended that you place bound fields on the GroupHeader and GroupFooter sections.

- **GroupHeader section**: This section (ghColumnCaption) contains the Label controls that are captions for the information displayed in the Detail section. This section also uses the CrossSectionBox and the Shape controls.
- **Detail section**: This section contains the bound TextBox controls that display each row of data associated with the current ghColumnCaption. The Shape control is used to alternate row colors in the Detail section. Go to the **Script** tab and see the shpDetailBack.BackColor property in the Detail_Format event.
- **PageFooter**: This section contains the Label controls that explain codes used in the Category column of the invoice details.

Invoice2 report: This report shows another invoice layout and uses **Seikyu2.db** database to provide data.

- **GroupHeader1 section**: This section contains the Label and Textbox controls. It uses the bound data fields to calculate the invoice information in the header. The values of the **Year, Month** and **Day** Textbox controls are set under the **Script** tab and are calculated in the ReportStart event. It also uses the **CrossSectionLine** and the **CrossSectionBox** controls that span the GroupHeader1 section to Detail section. The CrossSection Box ends in the GroupFooter1 section. These controls form vertical lines between columns of the invoice details and a rectangle around the details of the invoice at run time.
- **Detail section**: This section uses the data table **tb_Main** for the main report data and the data table **tb_Count** to retrieve the number of data items within a group. The Detail data in each group is retrieved beforehand to calculate the required number of empty rows. For the required empty row count, substitute data with "" in the FetchData event. The **Shape** control is used to alternate row colors in the Detail section. Go to the **Script** tab and see the shpDetailBack.BackColor property in the Detail_Format event.
- **GroupFooter1 section**: This section contains the Label and Textbox controls that display the totals of the invoice. The values of the **Tax** and **Pretax** Textbox controls are set under the **Script** tab and are calculated in the Format event.

LabelReport: This report displays the tack seal, which is commonly used in postal services. This multi-column report uses the customer barcode that is bound to data by using the **DataField** property. This report uses **Nwind.db** database to provide data.

- **Detail1 section**: This section contains the bound data fields to display the contact information and the Barcode control that is bound to data by using the **DataField** property. The **ColumnCount** property is set to 3, which allows the report to display the tack seal in 3 columns.

PaymentSlip Report: The report displays the salary payment slip where the EAN128FNC1 barcode is used for barcode bound to a data field. This report uses **Bill.db** data source connection to provide data.

- **Detail section**: This section contains bound data fields to display the payment information and the Barcode control that is bound to data by using the **DataField** property.

PurchaseOrder report: The report displays purchase slips created with bound data. It groups data for each purchase slip. The detail count is not fixed but the number of rows in the report layout is fixed. In this case, when the **RepeatToFill** property of the Detail section is set to True, the detail data is repeatedly displayed on the entire page. The report uses the **RepeatToFill** property and calculated fields to demonstrate how to create the report layout without any code. Calculated fields are used to calculate the total cost and the total selling price. This report uses **Shiire.db** data source.

- **groupHeader1 section**: This section contains the Label and Textbox controls. It uses the bound data fields to calculate the company information in the header.
- **Detail section**: This section contains Textbox controls. It uses bound data fields to calculate the product information in the body of the report. The **detail.BackColor** property is used to alternate row colors in the Detail section. Go to the **Script** tab and see the detailBack.BackColor property in the detail_Format event.
- **groupFooter1 section**: This section contains the Label and Textbox controls. It uses the bound data fields to calculate the total information in the footer of the report group.

Schedule report: The sample report displays the weekly schedule of employees in the Gantt chart format. The report displays six day report schedules on a single page in the Viewer control.

- **GroupHeader1 section**: This section contains the Label and Textbox controls. It uses the bound data fields to calculate the employee information in the header.
- **Detail section**: The Gantt chart view is created by using the Label, Shape and Line control. You can display the horizontal bars like in the Gantt chart by modifying the size of the Label control. To add horizontal bars, you can dynamically add twenty four (12 x 2) Label controls within the 9-20 hours time frame that will become the

horizontal bars of Gantt chart in the ReportStart event. Once they are added, set the **Visible** property to False to hide them.

You can adjust the width of horizontal bars by setting calculated values from the time width in the **Tag** property of the Label control within Detail_Format event. For the horizontal adjustment of bars, change the width of the Label control to match with the value in the **Tag** property.
UnderlayNext report: This report displays the bound data. It uses **Nwind.db** database to provide data.

- **PageHeader section**: This section contains the Label and Line controls to create the header for report data.
- **GroupHeader1 section**: This section contains a TextBox control bound to the Country field and the CrossSectionLine controls to create the table for the body data of the report. **UnderlayNext Property (on-line documentation)** of this section is set to True to align the beginning position of the report data in the Detail section with the GroupHeader1 section.
- **Detail section**: This section contains the bound Textbox controls to display report data. This section displays the name of the city, contact name and postal code for each customer according to their countries.

Desktop

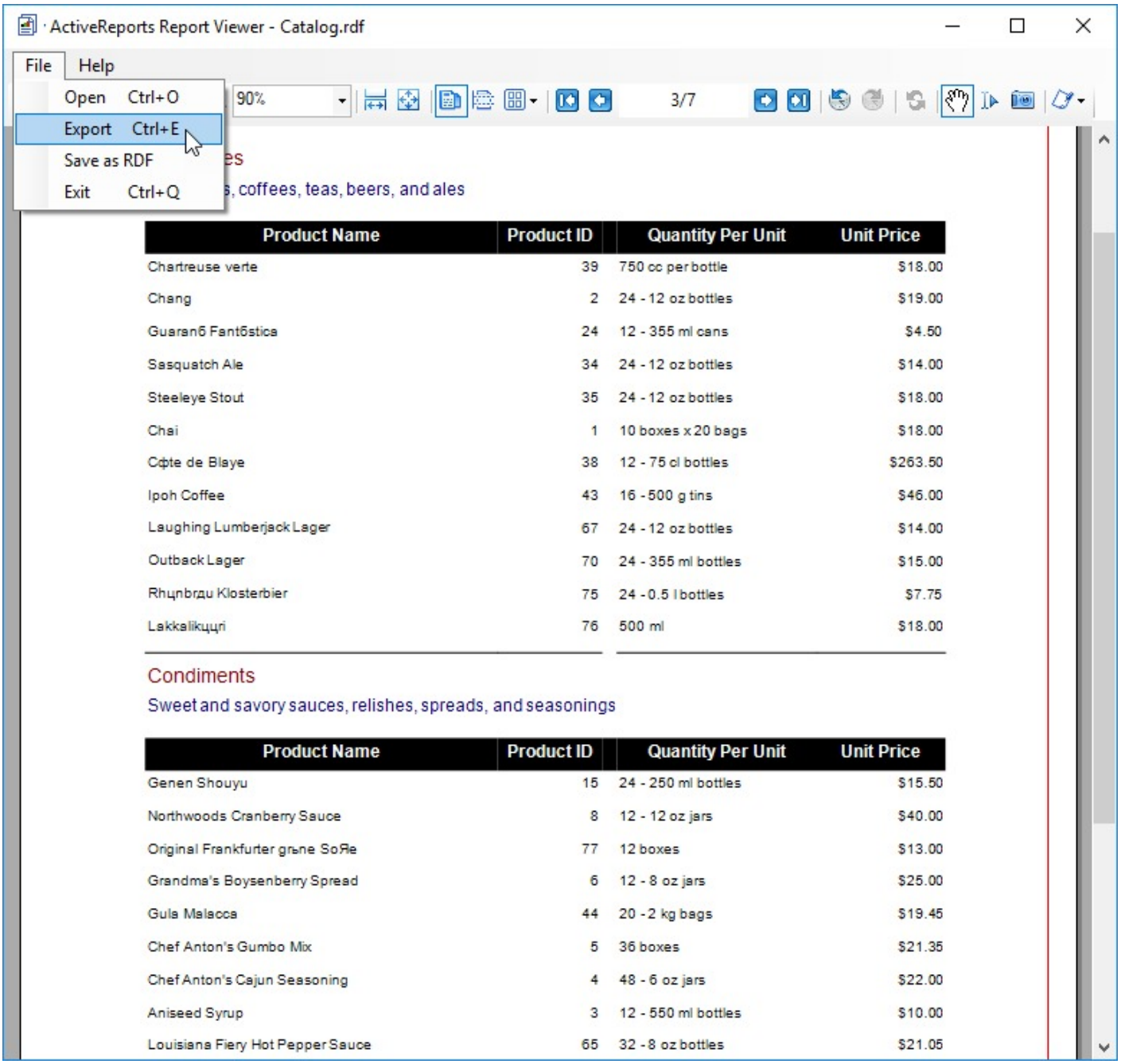
The samples in Desktop folder demonstrate features of WPF and Win Viewers.

WPF Viewer
: This sample demonstrates using WPF Viewer in a WPF application.

Win Viewer
: This sample demonstrates using Win Viewer in a Windows Form application.

Win Viewer

This sample demonstrates using Win Viewer to load RPX, RDLX, JSON or RDF report formats or save reports to report document file (RDF) format.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Desktop/WinViewer/VB.NET

C#

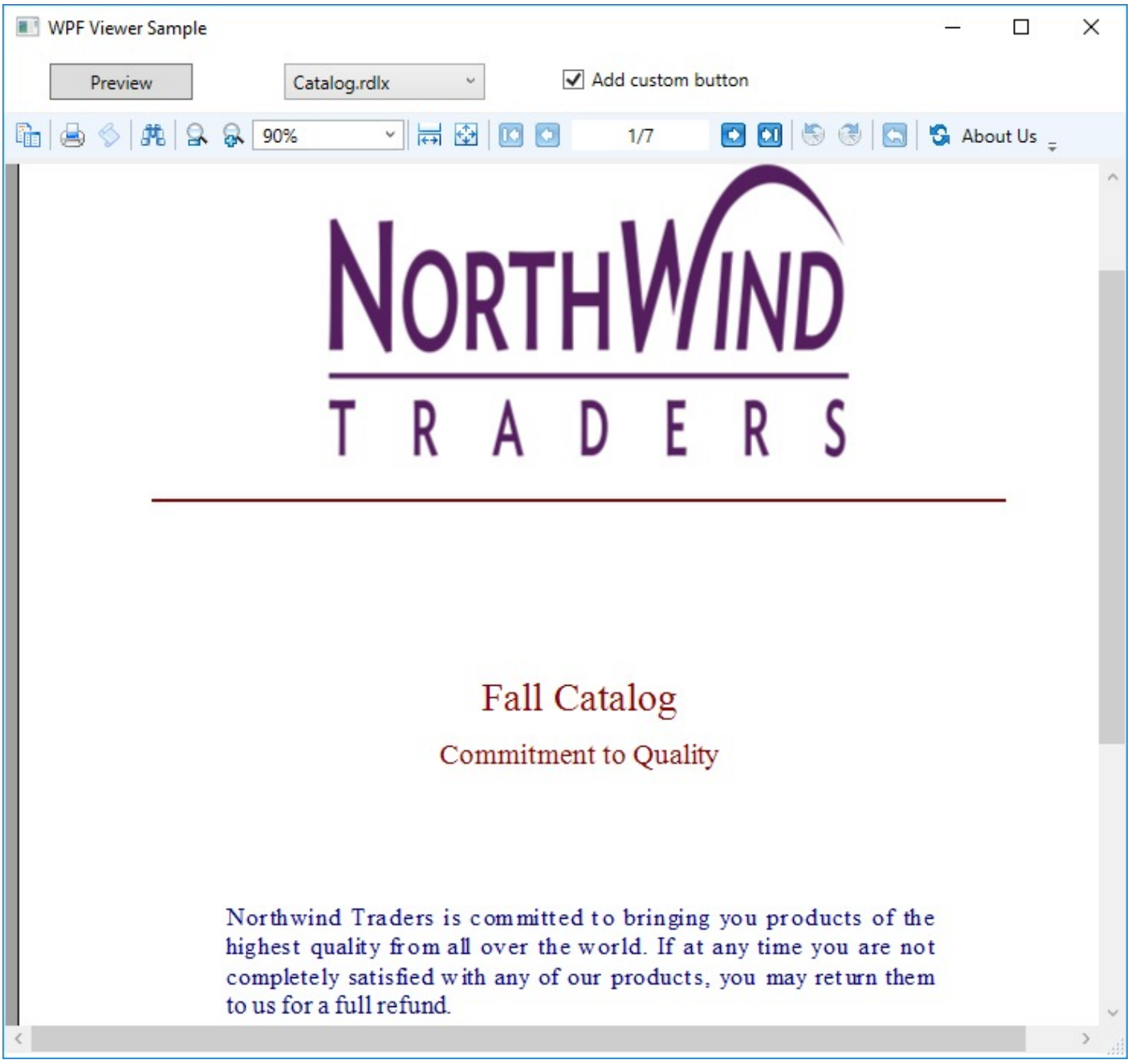
https://github.com/activereports/Samples18/tree/main/Desktop/WinViewer/C#

Details

When you run this sample, a Viewer control containing File and Help menu appears on the top. You can load any of the RDF files from **RDFs** folder using File > Open. An RDF file is a static copy of a report saved to the native Report Document Format. This can be loaded into the Viewer control without running it or accessing data. You can use **Save as RDF** option to save other report formats to RDF format. For more information, see Save and Load RDF Files. You can also export to other file formats using File > Export option.

WPF Viewer

The WPF Viewer samples demonstrates the use of WPF Viewer and its options to view the rdlx and rpx reports.



Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Desktop/WpfViewer/VB.NET

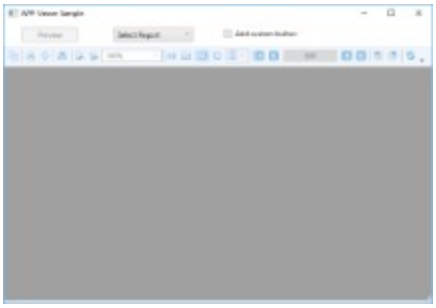
C#

https://github.com/activereports/Samples18/tree/main/Desktop/WpfViewer/C#

Details

Run-Time Features

When you run the sample, MainWindow.xaml containing the WPF Viewer, **Select Report** ComboBox, **Preview** button and **Add Custom Button** CheckBox appears.



Select Report

In the **Select Report** ComboBox, you can select from a list of 6 sample reports. The ComboBox contains the following reports - Catalog.rdlx, EmployeeSales.rdlx, Invoice1.rdlx, Invoice2.rpx, LabelReport.rpx, and PaymentSlip.rpx.

Preview

The **Preview** button opens the report selected in **Select Report** ComboBox in the WPF Viewer.

Add custom button

The **Add custom button** CheckBox demonstrates the customization options of the WPF Viewer. To see the **About Us** custom button appear in the WPF Viewer toolbar, select the **Add custom button** CheckBox and click the **Preview** button. To remove the **About Us** custom button from the WPF Viewer toolbar, click to clear the **Add custom button** CheckBox and then click the **Preview** button.

> **Note: Preview** button and **Add custom button** CheckBox are only enabled when a report is selected from the **Select Report** ComboBox.

Project Details

Reports folder

The folder contains the following reports.

Catalog.rdlx: This is a sample layout for the product catalog. This report uses multiple page layouts to create a catalog. The layout of this report is spread over 4 pages, which appear one after another at run time. **Page1** and **Page2** layouts simply display introductive text. The layout on **Page3** contains a List data region with TextBox controls and a Table to display product details for each product category. The List is grouped by the CategoryID field to filter products by their category. The layout on **Page4** uses Textbox, Shape and Line controls to create a Order Form, which a user is to fill manually.

EmployeeSales.rdlx: This is a sample layout to display sales by each employee for the year 1997. The Chart control is used to display a graphical analysis of sales by each employee and the Table data region lists down the exact numbers.

Invoice1.rdlx: This report uses a BandedList and few TextBox controls to create the Invoice layout for displaying customer transactions. Both page and the BandedList control are grouped by the OrderID field. The text boxes in the footer section of the BandedList control use the Sum function to display the total of the transactions. For detailed information on the Invoice2.rpx report, see the Reports Gallery Sample.

Invoice2.rpx: This is a sample layout for the invoice report. The report page is grouped by the EstimateID field. Sum function is used to display the GrandTotal of all transactions. For detailed information on the Invoice2.rpx report, see the Reports Gallery Sample.

LabelReport.rpx: This report displays the tack seal, which is commonly used in postal services. This multi-column report uses the customer barcode that is bound to the data using the **DataField** property. For detailed information on the LabelReport.rpx report, see the Reports Gallery Sample.

PaymentSlip.rpx: This report displays the invoice payment slip with the GS1-128 barcode that is used for payment services in convenience stores. GS1-128 is the convenience store barcode, formerly called UCC/EAN-128. For detailed information on the PaymentSlip.rpx report, see the Reports Gallery Sample.

DefaultWPFViewerTemplates.Xaml

The DefaultWPFViewerTemplates.xaml is used for the WPF Viewer customization. For steps on the WPF Viewer customization, see the WPF Viewer walkthrough.

MainWindow.xaml

The MainWindow.xaml is displayed when you run the sample. It contains the WPF Viewer, the **Select Report** ComboBox, the **Preview** button and the **Add custom button** CheckBox.

The code behind of MainWindow.xaml.vb (or .cs), handles the display of RDLX and RPX reports in the WPF Viewer and the customization of the WPF Viewer application.

MyCommand

This class contains the text that is displayed when you click the **About Us** custom button in the WPF Viewer toolbar.

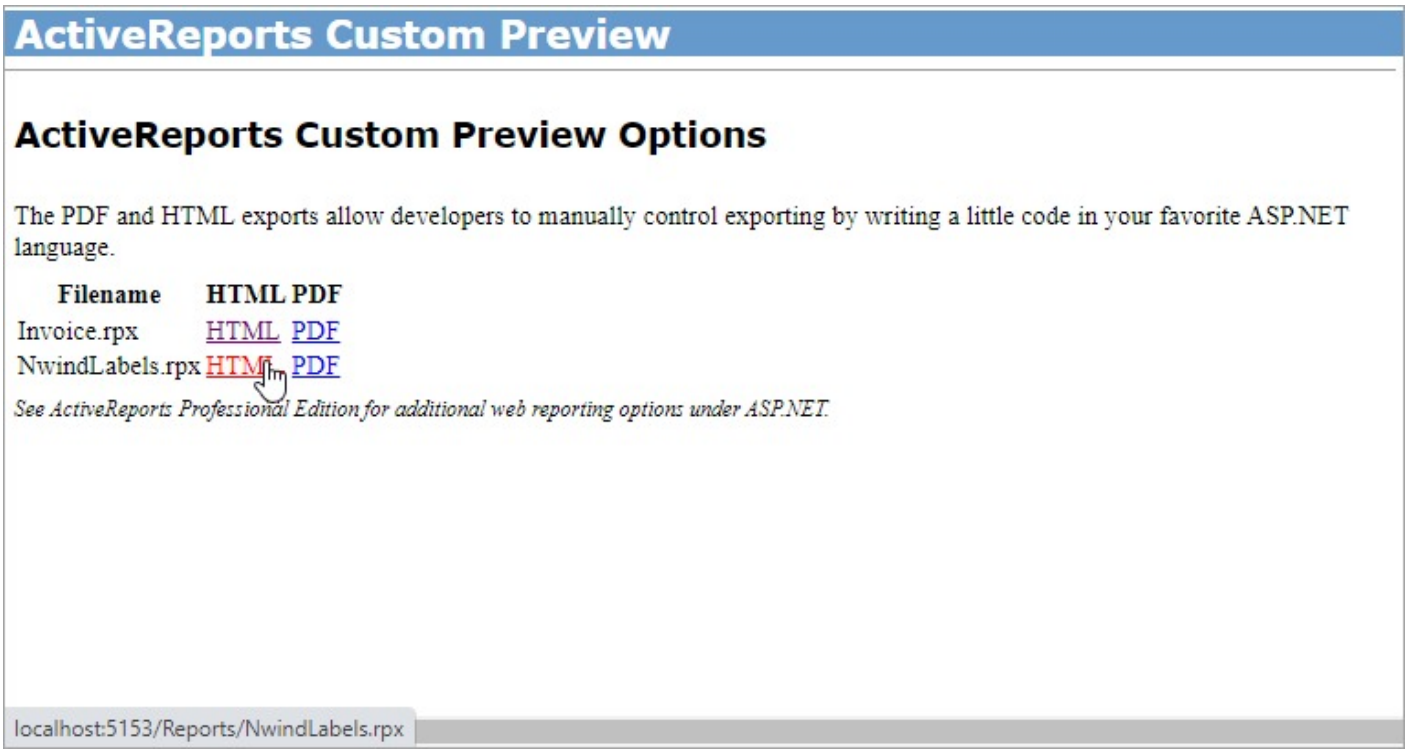
Web

The Web folder contains following sample:

Custom Preview
: The sample demonstrates exporting an ActiveReports report to the HTML or PDF format in your Web application.

Custom Preview

The Custom Preview sample demonstrates a method to view a report at client side in HTML or PDF format. Application structure consists of ASP.NET website, using which the report is streamed to the client as HTML or PDF. This sample describes custom exporting without the Pro Edition server controls or RPX handlers as well as running reports on the server. The PDF and HTML exports allow you to manually control exporting by writing a little code in ASP.NET language. Steps explained in this sample can be used for both Standard and Professional editions.



Note: Before running this sample, in the Solution Explorer, click the licenses.licx file and then, from the **Build** menu, select Build Runtime License. Please see To license Web Forms projects made on the trial version for details.

Sample Location

Visual Basic.NET

https://github.com/activereports/Samples18/tree/main/Web/CustomPreview/VB.NET

C#

https://github.com/activereports/Samples18/tree/main/Web/CustomPreview/C#

Details

When you run the sample, the Default.aspx page appears in your browser. This page provides two links to other reports that demonstrate custom PDF or HTML export options.

Clicking the **Custom Exporting PDF Example** option opens the **Invoice** report and clicking **Custom Exporting HTML Example** option opens **NwindLabels** report in the Default.aspx page.

Note: To run this sample, you must have Nwind.db downloaded from GitHub in *..\Samples18\Data\NWIND.db*

The project consists of the following elements.

- Reports folder: The Reports folder contains two rpx reports - the **Invoice** report and the **NwindLabels** report.

  Invoice Report

  The Invoice report uses three GroupHeader sections, a Detail section and a GroupFooter section as well as a

label in the PageFooter section to display data.

> **Note:** Except for the Detail section, all sections come in header and footer pairs. Unused sections have their **Height** properties set to **0** and their **Visible** properties set to **False**.

ghOrderHeader section

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.

This section also 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 section

The **DataField** property of this section is also set to **OrderID**. This allows subtotal summary functions in the related GFOrderID section to calculate properly.

This section contains a number of labels and bound text boxes, as well as two **Line** controls.

ghTableHeader section

This section contains only labels for the data to follow in the Detail section.

Detail section

This section contains bound TextBox controls. These TextBoxes render once for each row of data found in the current OrderID before the report moves on to the GroupFooter sections.

GFOrderID section

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 TextBoxes. Two of the TextBox controls use the following properties to summarize the detail data: **SummaryFunc**, **SummaryGroup**, and **SummaryType**. For more information, Create a Summary Report.

The **Total** TextBox does not use the DataField property or any of the summary properties, or even any code. To find the functionality of this TextBox, in design view, click the **Script** tab at the bottom of the report.

PageFooter section

This section has one simple Label control. For more information about report sections and the order in which they print, see Section Report Layout/Structure and Report Events.

NwindLabels Report

TheNwindLabels report only uses the Detail section to display the report data.

> **Note:** Except for the Detail section, all sections come in header and footer pairs. Unused sections have their **Height** properties set to **0** and their **Visible** properties set to **False**.

Detail section

This section contains bound TextBox controls and Label controls. This section prints 30 labels per 8½ x 11 sheet.

The Detail section uses the **CanGrow** property set to **False** to maintain the label size and the **ColumnCount**, **ColumnDirection,** and **ColumnSpacing** properties to accommodate multiple labels in a single page.

- CustomExportHtml.aspx: This Web form is displayed by clicking the **Custom Exporting HTML Example** option on the Default.aspx page. In CustomExportHtml.aspx, report is outputted to the ReportOutput folder using the **CustomHtmlOutput** class and the exported HTML is displayed in the browser. The CustomHtmlOutput class implements the required IOutputHtml in the HTML export and saves the output results to a file with a unique name.

  > **Note:** This sample requires write permissions to the **ReportOutput** folder that is located in the web samples directory.

- CustomExportPdf.aspx: The Web form is displayed by clicking the **Custom Exporting PDF Example** option on the Default.aspx page. In CustomExportPdf.aspx, the report is exported to memory stream and then outputted in the browser.

  > **Note:** This sample requires write permissions to the **ReportOutput** folder that is located in the web samples directory.

- Default.aspx: This is the main Web form of the sample that shows the introductory text and links to other sample pages that demonstrate the following web features.
  - **Custom Exporting PDF Example** - This link opens the Invoice report in the PDF Reader by exporting it to memory stream and then outputting it in the browser.
  - **Custom Exporting HTML Example** - This link opens the NWindLabels report. This report is outputted to the ReportOutput folder by using the CustomHtmlOutput class and the exported HTML is displayed in the browser.

- Web.config: This configuration file contains the httpHandlers that allow ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.

Web Samples

The samples in WebSamples folder demonstrate web related features in Js Viewer, WebDesigner, etc. Download these samples from following link:

https://github.com/activereports/WebSamples18

Sample	Description
WebDesigner Samples	
Blazor Designer	The samples on Blazor Designer demonstrate the use of ActiveReports Blazor Designer with Server application, WebAssembly application, and remote report service.
WebDesigner MVC(Core)	This sample demonstrates WebDesigner with an ASP.NET MVC Core back end.
WebDesigner Angular(Core)	This sample demonstrates the ActiveReports WebDesigner with an Angular 8 app and ASP.NET Core back end.

WebDesigner Blazor	This sample demonstrates the ActiveReports WebDesigner with Blazor framework.
WebDesigner Custom Data Providers	The sample demonstrates the method to use custom data providers (such as SQLite and OData) for supplying data to the report in the ActiveReports WebDesigner.
WebDesigner Custom Store	The sample demonstrates the use of custom resources service for ActiveReports WebDesigner with an ASP.NET Core back end.
WebDesigner Custom	The sample shows how to use the WebDesigner API to create a customized UI for ActiveReports WebDesigner with an ASP.NET Core back end.
WebDesigner Blazor Custom	The sample shows how to use the Blazor WebDesigner API to create a customized UI for ActiveReports WebDesigner with Blazor back end.
WebDesigner Custom Shared Data Sources	The sample shows how to enable and use shared data sources in ActiveReports WebDesigner with an ASP.NET Core back end.
Js Viewer Samples	
BlazorViewer	The samples on Blazor Viewer demonstrate the use of ActiveReports Blazor Viewer with Server application, WebAssembly application, and remote report service.
JSViewer Angular(Core)	This sample demonstrates the use of the ActiveReports Js Viewer with an Angular 8 app and ASP.NET Core back end.
JSViewer CORS	This sample demonstrates the use of ActiveReports Js Viewer with an ASP.NET MVC 5 back end when the server is hosted elsewhere using CORS.
JSViewer MVC CORS(Core)	This sample demonstrates the use of ActiveReports Js Viewer with an ASP.NET MVC 5 Core back end when the server is hosted elsewhere using CORS.
JSViewer MVC	This sample demonstrates the use of ActiveReports Js Viewer with an ASP.NET MVC 5 back end.
JSViewer MVC(Core)	This sample demonstrates the use of ActiveReports Js Viewer with an ASP.NET MVC Core back end.
JSViewer React(Core)	This sample demonstrates the use of ActiveReports Js Viewer with an ReactJS app and ASP.NET Core back end.
JSViewer Vue(Core)	This sample demonstrates the use of ActiveReports Js Viewer with an VueJS app and ASP.NET Core back end.
WebViewer ASP.NET	This sample demonstrates the ActiveReports web control feature and generating a parameterized report.
Silent Print	The SilentPrint sample project consists of three samples - JSViewerBatchPrint_MVC_Core, JSViewerSilentPrint_MVC_Core, and PrintAgent. The JSViewerBatchPrint_MVC_Core sample demonstrates how to print many reports by clicking the Print button, without showing the Print Preview dialog for every report. Silent printing is implemented through a print agent that needs to be started. The JSViewerSilentPrint_MVC_Core sample demonstrates how to print a report by clicking once the

	JSViewer Print button, without showing the Print Preview dialog. Silent printing is implemented through a print agent that needs to be started. The PrintAgent sample contains a Windows service, hosting an ASP.NET Core API that allows printing PDF files. The print agent uses the GrapeCity.Documents.Pdf library.

Important: It is recommended to use only that Visual Studio version which is specified in each sample description. For example, for applications running on ASP.NET Core, if you use Visual Studio 2017, you will see 'No License' message because the Visual Studio 2017 does not support core license compilation.

WebDesigner Samples

Blazor Designer

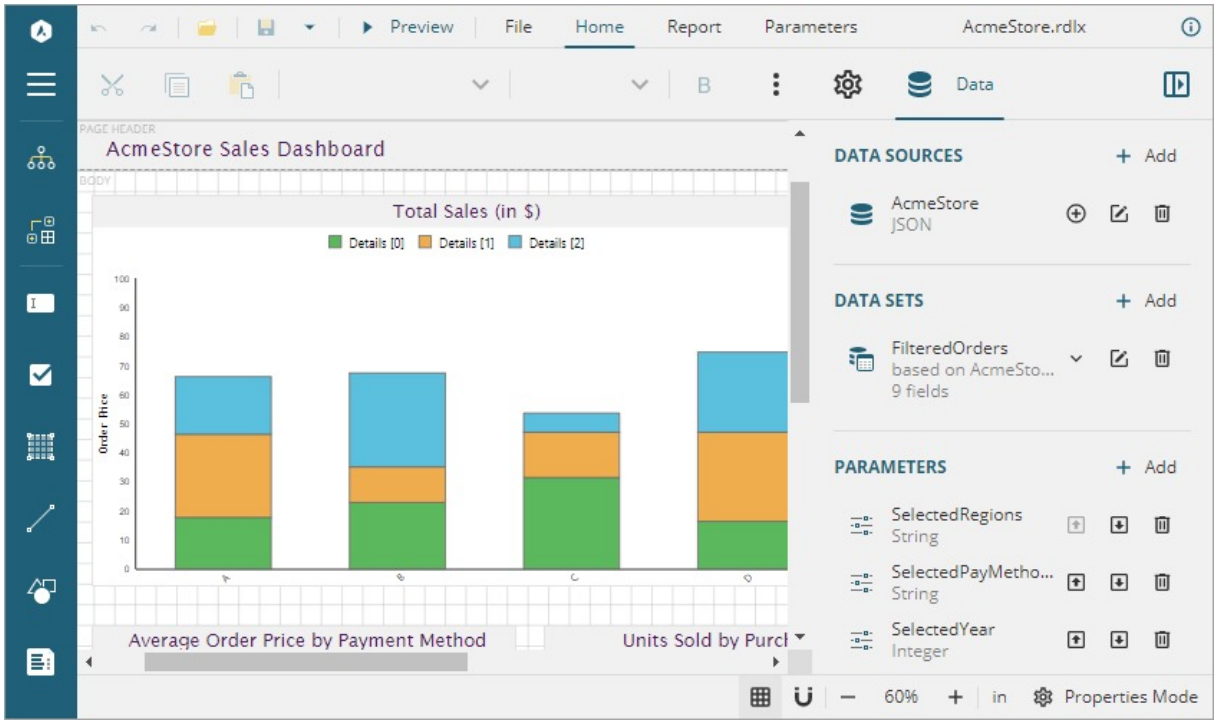
Three samples are available for the Blazor Designer demonstration.

Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)

BlazorDesignerServer

The sample demonstrates creating a Blazor Server Application with the ActiveReports Blazor Designer, using local report service and remote report service.



Sample Location

https://github.com/activereports/WebSamples18/tree/main/BlazorDesigner/BlazorDesignerServer

Details

When you run the sample, the Blazor Designer opens in your browser wherein you can create, edit, or modify your reports.

The project uses the **MESCIUS.ActiveReports.Aspnetcore.Designer, MESCIUS.ActiveReports.Aspnetcore.Viewer, MESCIUS.ActiveReports.Blazor.Designer** and **MESCIUS.ActiveReports.Blazor.Viewer** NuGet packages.

The project consists of the following elements.

- wwwroot: Contains designer CSS file for the Blazor application.
- Pages: This folder contains Razor pages and supporting files.
- _Imports.razor: The Razor template file to include the directives.
- Program.cs file: Add services to container, configure HTTP request, and add CORS middleware.

ReportService

The sample demonstrates creating a remote report server that could be be used by the ActiveReports Blazor Designer.

Sample Location

https://github.com/activereports/WebSamples18/tree/main/BlazorDesigner/ReportService

Details

When you run the sample, the browser displays the report service as running. The project uses the **MESCIUS.ActiveReports.Aspnetcore.Designer** and **MESCIUS.ActiveReports.Aspnetcore.Viewer** NuGet packages.

The project consists of the following elements.

- Program.cs: Contains necessary code to add services, configure the HTTP request and CORS.
- Controllers folder: Contains DataSets and Templates controllers.
- Datasets folder: Contains JSON datasets - Categories, Employees, Products, and Invoice.
- Resources folder: Contains reports, themes, images, etc. that are used by the project to illustrate WebDesigner.
- Services folder: Consists of classes that get datasets and templates information.

BlazorDesignerWebAssembly

The sample demonstrates how to make Blazor WebAssembly Application with the ActiveReports Blazor Designer using remote report service.

Sample Location

https://github.com/activereports/WebSamples18/tree/main/BlazorDesigner/BlazorDesignerWebAssembly

Details

You need to first build and run **ReportService** sample to start the report service if you want to use existing report templates. When you run the **BlazorDesignerWebAssembly** sample, the Blazor Designer opens in your browser and you can design a new repot or open an existing one.

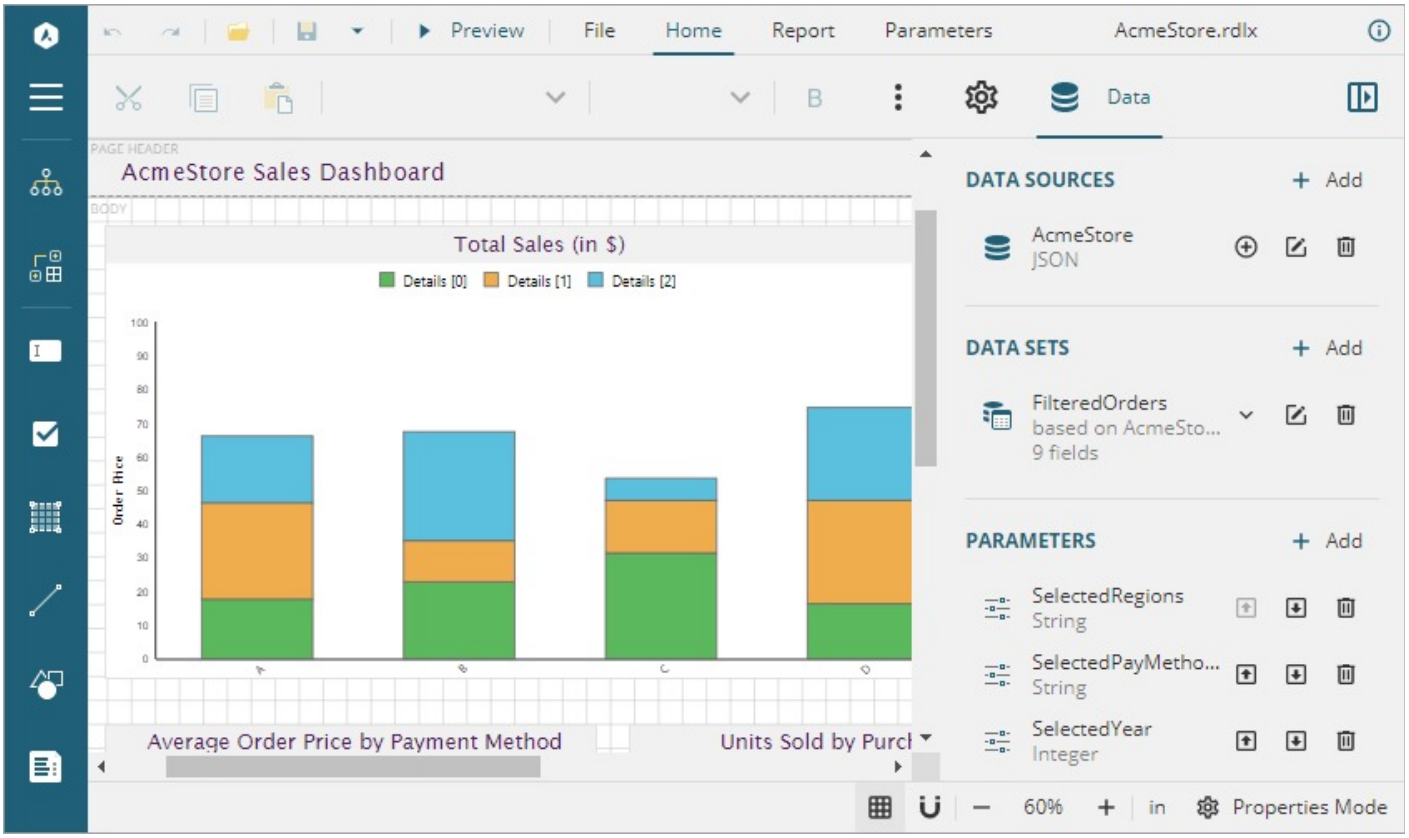
The project uses the **MESCIUS.ActiveReports.Blazor.Designer**, **MESCIUS.ActiveReports.Blazor.Viewer**, **Microsoft.AspNetCore.Components.WebAssembly**, and **Microsoft.AspNetCore.Components.WebAssembly.DevServer** NuGet packages.

The project consists of the following elements.

- wwwroot: Contains index.html file for the Blazor application.
- Pages: This folder contains Razor pages.
- _Imports.razor: The Razor template file to include the directives.
- Program.cs file: Create and run web host instance.

WebDesigner MVC

The WebDesigner_MVC sample demonstrates WebDesigner with an ASP.NET MVC 5 back-end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET Framework 4.6.2 or later

Sample Location

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_MVC

Details

When you run the sample, the WebDesigner opens in your browser wherein you can create, edit, or modify your reports. Following are the main menu options:

- **File**: Contains options to create, open, or save reports. It also contains the version information in About option and help documentation link in Help option.

- **Home**: Consists of report editing options such as cut, copy, paste, and delete. It also provides shortcuts for text formatting such as font, font size, font color, and horizontal and vertical text alignments.
- **Report**: Contains options to add, delete, or move pages (in Page Report) and add or remove header and footer (RDLX report), and change report themes.
- **Parameters**: Contains designer for designing the custom parameter panel.
- **Preview**: Click Preview to preview reports.
- **Properties**: Displays the properties of the selected report element. If more than one element is selected, only their common properties are shown.
- **Data**: Contains options to manage data sources, data sets, and parameters. It also displays common values such as current date and time, page number, total pages, and more.

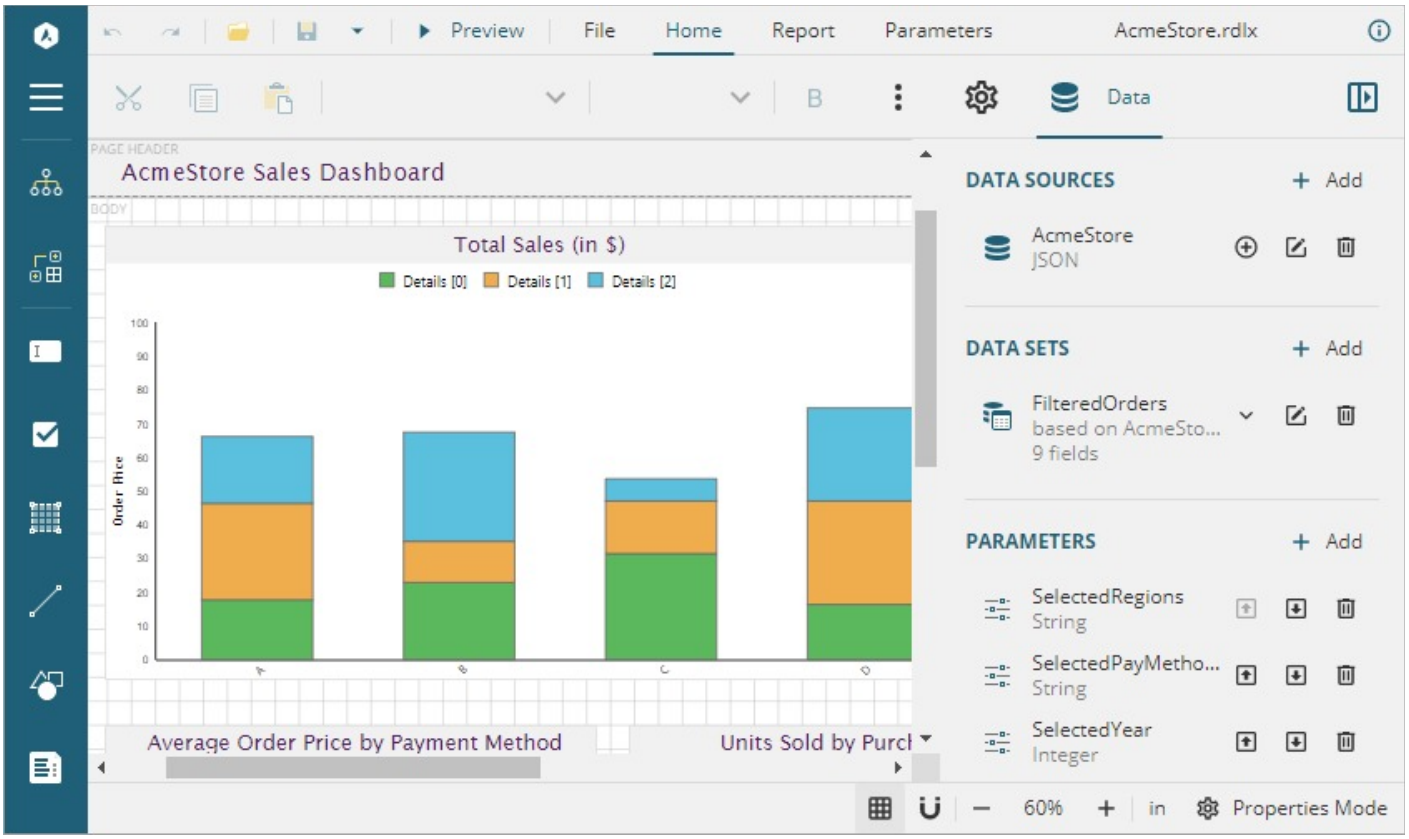
The project consists of the following elements.

- Controllers folder: Contains DataSets, Design, Preview, and Templates controllers.
- Datasets folder: Contains JSON datasets Categories, Employees, and Products.
- Resources folder: Contains reports, themes, images, etc. that are used by the project to illustrate WebDesigner.
- Services folder: Consists of classes that get datasets and templates information.
- wwwroot folder: Contains designer CSS and JavaScript files.

> ⚠ Before publishing the sample, you must copy the sample **datasets**, **resources**, **templates** folders to the **publish** folder.

WebDesigner MVC(Core)

The WebDesigner_MVC(Core) sample demonstrates WebDesigner with an ASP.NET Core back-end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_MVC_Core

Details

When you run the sample, the WebDesigner opens in your browser wherein you can create, edit, or modify your reports. Following are the main menu options:

- **File**: Contains options to create, open, or save reports. It also contains the version information in About option and help documentation link in Help option.
- **Home**: Consists of report editing options such as cut, copy, paste, and delete. It also provides shortcuts for text formatting such as font, font size, font color, and horizontal and vertical text alignments.
- **Report**: Contains options to add, delete, or move pages (in Page Report) and add or remove header and footer (RDLX report), and change report themes.
- **Parameters**: Contains designer for designing the custom parameter panel.

- **Preview**: Click Preview to preview reports.
- **Properties**: Displays the properties of the selected report element. If more than one element is selected, only their common properties are shown.
- **Data**: Contains options to manage data sources, data sets, and parameters. It also displays common values such as current date and time, page number, total pages, and more.

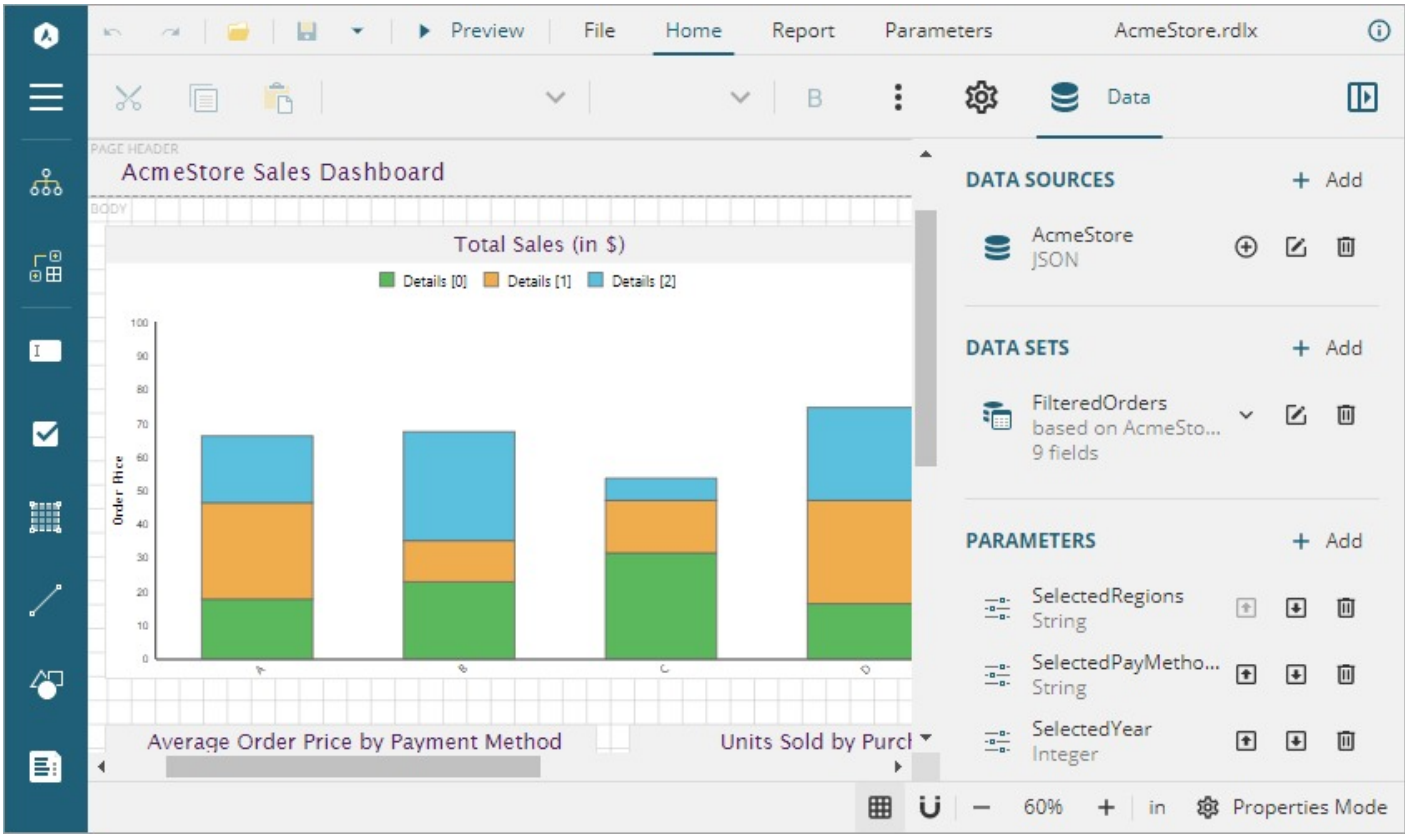
The project consists of the following elements.

- Controllers folder: Contains DataSets, Design, Preview, and Templates controllers.
- Datasets folder: Contains JSON datasets - Categories, Employees, Products, and DataSet with Parameters.
- Resources folder: Contains reports, themes, images, etc. that are used by the project to illustrate WebDesigner.
- Services folder: Consists of classes that get datasets and templates information.
- wwwroot folder: Contains designer CSS and JavaScript files.

> ⚠ Before publishing the sample, you must copy the sample **datasets**, **resources**, **templates** folders to the **publish** folder.

WebDesigner Angular(Core)

The WebDesigner_Angular(Core) sample demonstrates the use of ActiveReports WebDesigner with an Angular 8 app and ASP.NET Core back end.



> **Note**: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)
- Angular 14 requires Node.js 14 or later

Sample Location

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_Angular_Core

Details

When you run the sample, the WebDesigner opens in your browser wherein you can create, edit, or modify your reports. Following are the main menu options:

- **File**: Contains options to create, open, or save reports. It also contains the version information in About option.
- **Home**: Consists of report editing options such as cut, copy, paste, and delete. It also provides shortcuts for text formatting such as font, font size, font color, and horizontal and vertical text alignments.
- **Report**: Contains options to add, delete, or move pages (in Page Report) and add or remove header and footer (RDLX report), and change report themes.
- **Parameters**: Contains designer for designing the custom parameter panel.
- **Preview**: Click Preview to preview reports.
- **Properties**: Displays the properties of the selected report element. If more than one element is selected, only their common properties are shown.
- **Data**: Contains options to manage data sources, data sets, and parameters. It also displays common values such as current date and time, page number, total pages, and more.

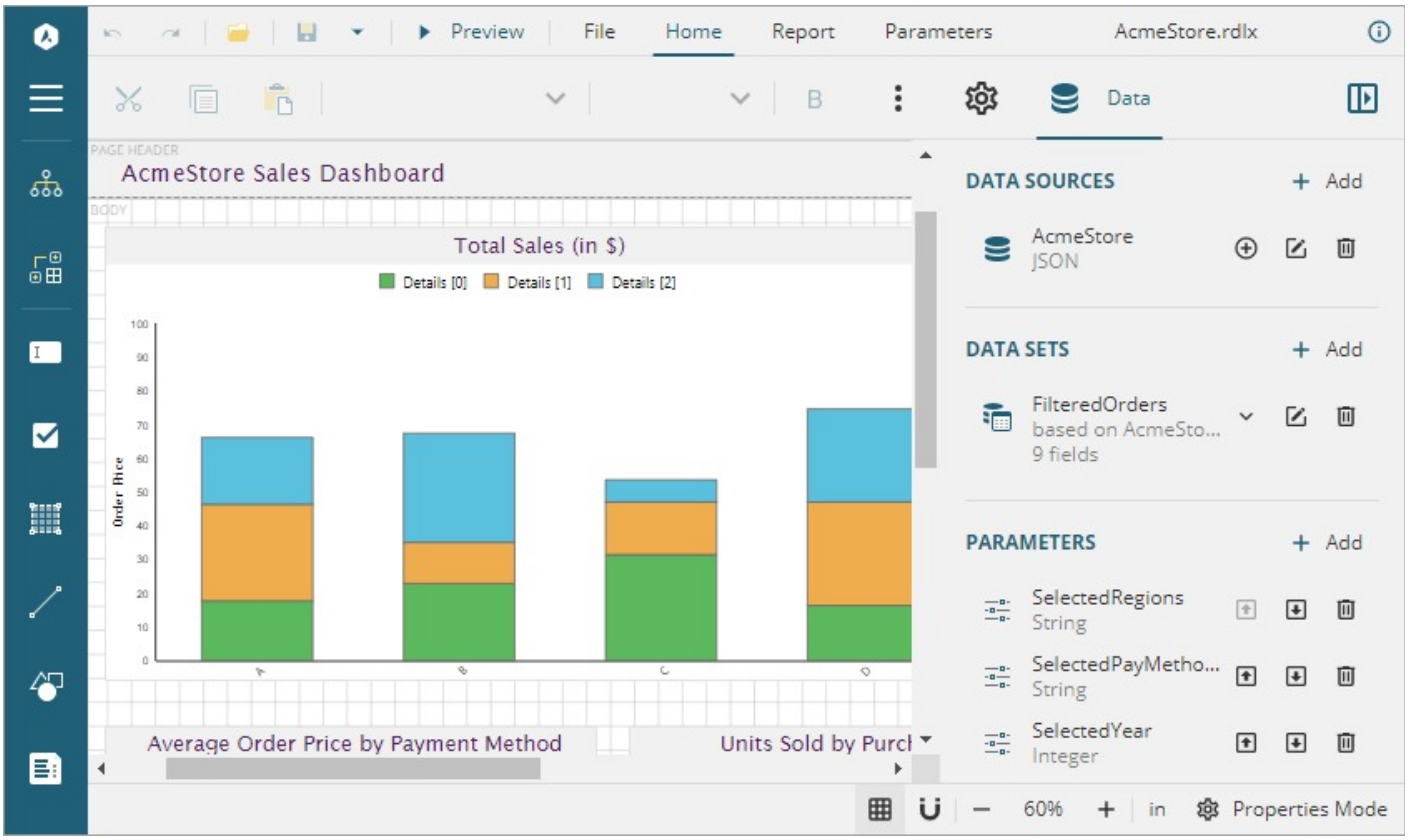
Note: The timeout error sometimes appears on running the WebDesigner_Angular(Core) sample with default settings. In this case, you should increase the connection timeout period. See Troubleshooting for details on how to resolve this issue.

The project consists of the following elements.

- ClientApp folder: This folder contains a standard Angular CLI app that is used for all UI concerns.
- Controllers folder: Contains DataSets and Templates controllers.
- Datasets: Contains JSON datasets - Categories, Employees, Products, and DataSet with Parameters.
- Resources: Contains reports, themes, images, etc that are used by the project to illustrate WebDesigner.
- appsettings.json: The json configuration file.
- readme: This file contains the instructions on how to run the sample project.
- Startup.cs: This is the default startup file.
- Web.config: This configuration file contains the httpHandlers that allow ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.
- Services: Consists of classes that get datasets and templates information.
- wwwroot: Contains designer CSS and JavaScript files.

WebDesigner Blazor

The sample demonstrates how to wrap the ActiveReports WebDesigner in Blazor.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_Blazor

Details

When you run the sample, the WebDesigner opens in your browser wherein you can create, edit, or modify your reports. The project uses the **MESCIUS.ActiveReports.Aspnetcore.Viewer** and the **MESCIUS.ActiveReports.Aspnetcore.Designer** NuGet packages as well as the NPM packages. See Manage ActiveReports Dependencies for details.

Following are the main menu options:

- **File**: Contains options to create, open, or save reports. It also contains the version information in About option.

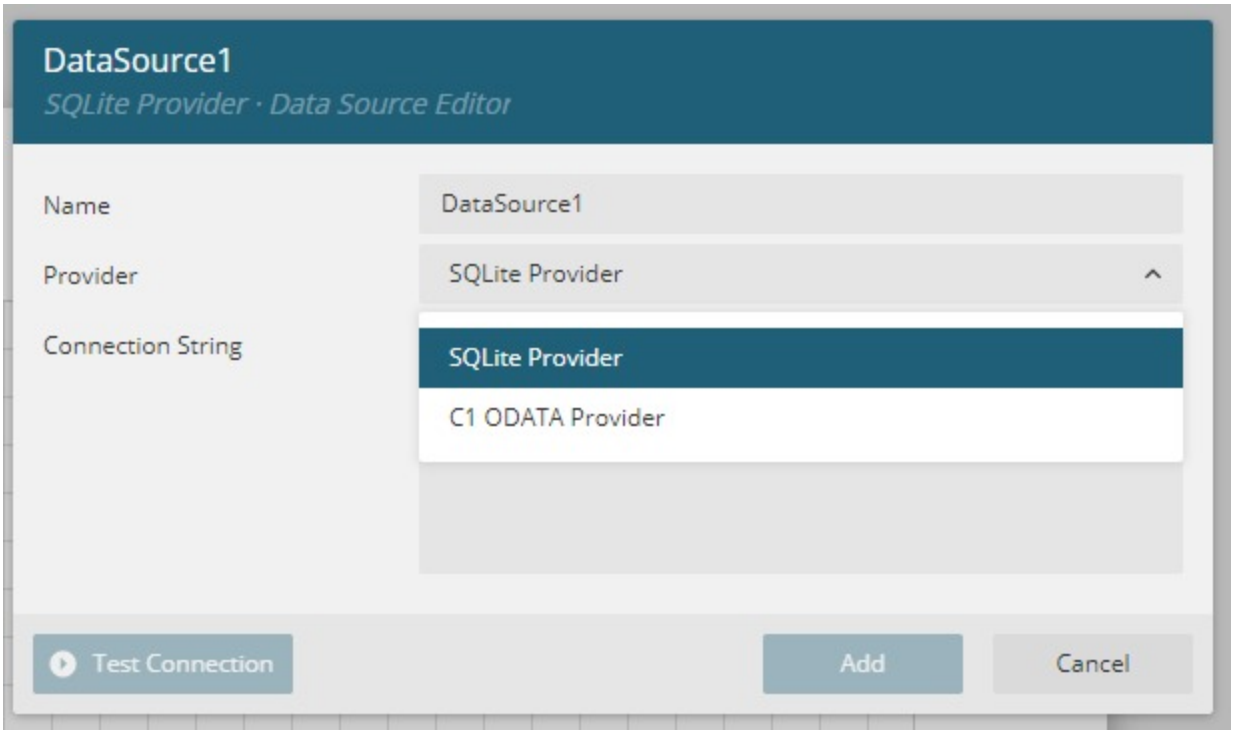
- **Home**: Consists of report editing options such as cut, copy, paste, and delete. It also provides shortcuts for text formatting such as font, font size, font color, and horizontal and vertical text alignments.
- **Report**: Contains options to add, delete, or move pages (in Page Report) and add or remove header and footer (RDLX report), and change report themes.
- **Parameters**: Contains designer for designing the custom parameter panel.
- **Properties**: Displays the properties of the selected report element. If more than one element is selected, only their common properties are shown.
- **Data**: Contains options to manage data sources, data sets, and parameters. It also displays common values such as current date and time, page number, total pages, and more.
- **Preview**: Click Preview to preview reports.

The project consists of the following elements.

- ClientApp folder: This folder contains a standard Angular CLI app that is used for all UI concerns.
- Controllers folder: Contains DataSets and Templates controllers.
- Datasets: Contains JSON datasets.
- Implementation: Contains files to define the template extension and set the datasets.
- Pages: This folder contains Razor pages and supporting files.
- _Imports.razor: The Razor template file.
- appsettings.json: The json configuration file.
- Resources: Contains reports, themes, images, etc that are used by the project to illustrate the WebDesigner.
- Services: Consists of classes that get datasets and templates information.
- wwwroot: Contains designer CSS and JavaScript files of the Blazor application.
- appsettings.json: The json configuration file.
- readme: This file contains the instructions on how to run the sample project.
- Startup.cs: This is the default startup file.
- Web.config: This configuration file contains the httpHandlers that allow ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.

WebDesigner Custom Data Providers

The WebDesigner Custom Data Providers sample demonstrate the method to use custom data providers (such as SQLite and OData) for supplying data to the report in the ActiveReports WebDesigner.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

C#

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_CustomDataProviders

Details

This sample describes the following methods.

- Using SQLite Provider
- Using C1 ODATA Provider
- Using ODATA Datasets

Using SQLite Provider

In order to use SQLite provider to add a data source, go to **Data** tab and click **Add** next to **Data Sources**. In the Data Source dialog, select the 'SQLite Provider' option in the **Provider**, set the path of the desired sqlite file in the **Connection String**, and click **Add** button. This will add a data source using the SQLite provider to the report.

Using C1 ODATA Provider

In order to use C1 ODATA Provider to add a data source, go to **Data** tab and click **Add** next to **Data Sources**. In the Data Source dialog, select the 'C1 ODATA Provider' option in the **Provider**, set the URL of the desired data from where the data will be fetched in the **Connection String**, and click **Add** button. This will add a data source using the C1 ODATA Provider to the report.

Using ODATA Datasets

In order to add ODATA Datasets, go to **Data** tab and click Add next to Data Sets. Select NWWIND dataset, an ODATA whose definition is available in the **datasets** folder.

> **Note:**
>
> - In case you face any problem related to C1 ODATA Provider, kindly register and install the trial version of the ComponentOne's 'Service Components'.
> - This sample does not have a built-in license for C1 Data Connector, so you may need to buy this license separately.

The project consists of the following elements.

Datasets Folder

This folder contains two JSON files named as Employees and Invoices.

Northwind.sqlite

This is the sample SQLite data source added in the project for reference. Retrieves the data to be displayed in the report.

Resources Folder

This folder consists of three files - Cosmo.rdlx-theme, InvoiceOData.rdlx and ListSqlite.rdlx. You can load the sample report 'InvoiceOData.rdlx' and 'ListSqlite.rdlx' that uses OData Data Source and Sqlite Data Source.

Implementation Folder

This folder consists of two files - FileSystemTemplates.cs and ODataDataSets.cs. FileSystemTemplates.cs is used to define the template extension whereas 'ODataDataSets.cs' is used to set the ODATA datasets.

WebDesigner Custom Store

The WebDesigner Custom Store sample demonstrates the use of custom resources service for ActiveReports WebDesigner with an ASP.NET Core back end. This sample contains two different implementations which is based on LiteDB and CosmosDB. By default, LiteDB implementation is used. To be able to use CosmosDB, you should have an Azure Cosmos DB account. For more details regarding the method to use CosmosDB, please refer to the 'howto.md' file placed in the 'resources\CosmosDB' folder.

> **Note**: To run this sample, you must have
>
> - Visual Studio 2022 (version 17.0 or later)
> - .NET 6.0 SDK

- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

C#

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_CustomStore

Details

This sample implements the IResourcesService interface. If you are using any unmanaged resources, you should implement 'IDisposable' interface also. The project consists of the following elements.

Implementation Folder

This folder contains the following files and folders.

- **Custom Store** folder – This folder contains many folders named DataSets, Images, Reports, Templates, and Themes. Each folder has some specific cs files.
- **Storage** folder – This folder consists of three files CosmoDB.cs, ICustomStorage.cs, and LiteDB.cs.
- **CosmoDB.cs, LiteDB.cs** – These files implement the 'GetReport' and 'GetSectionReport' methods from ICustomStorage interface.

Resources Folder

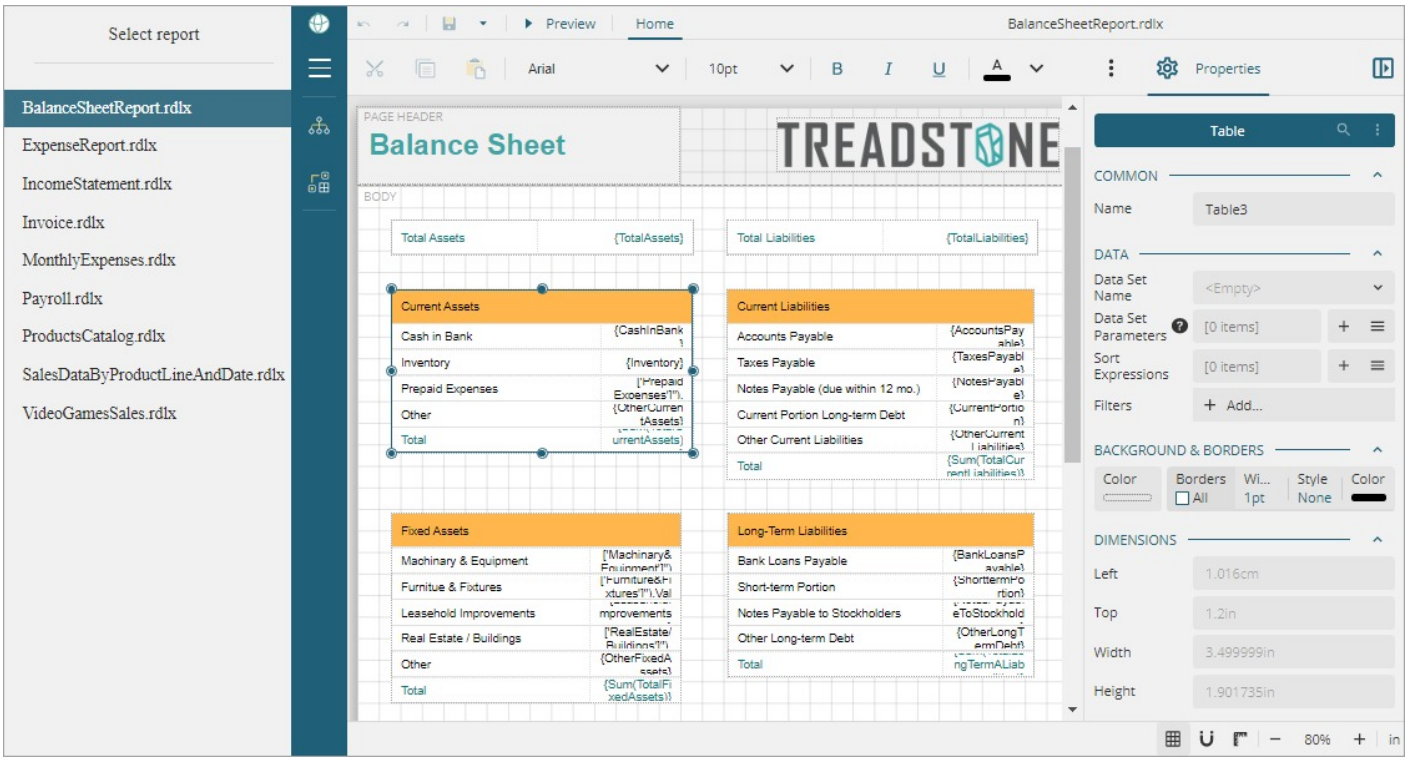
This folder consists of a 'CosmosDB' folder and 'lite.db' file, which is a database file. CosmosDB folder includes information regarding how to use the CosmosDB in the project. This sample already contains all the necessary code for working with CosmosDB.

Startup.cs

In this file, the implemented 'IResourcesService' service is registered. Also, 'UseCustomStore' method for viewer (app.UseReportViewer) with 'GetReport', 'GetSectionReport' methods and 'UseCustomStore' method for designer (app.UseReportDesigner) with 'IResourcesService' implementation are called as an argument.

WebDesigner Custom

This sample shows how to use the WebDesigner API to create a customized UI for ActiveReports WebDesigner with an ASP.NET Core back end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

C#

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_Custom

Details

Some of the customizations demonstrated in the sample are shown below. You can check the **Index.cshtml** page of the sample for complete implementation.

- Use a **custom logo** (white labeling)

```
menu: { logo: { custom: { type: 'css', class: 'example-icon' } }
```

- Hide **About** button

```
appBar: { aboutButton: { visible: false } }
```

- Hide **File** menu

```
documents: { fileView: { visible: false }
```

- **Lock the report layout**, that is, disable the operations that modify the report layout structure

```
lockLayout: true
```

- Hide **ToolBox**

```
menu: { toolBox: { visible: false }}
```

- Hide **Parameters tab**, the design area for designing a custom parameter pane

```
appBar: { parametersTab : { visible: false }}
```

- Hide **Data tab**

```
data: { dataTab: { visible: false }}
```

Hiding **Data tab** hides the options such as data sources, data sets, parameters, and common values.

- Hide **Properties Mode** button that helps switch between the Advanced and Basic properties of the report elements

```
statusBar: { propertiesModeButton: { visible: false }}
```

- Limit font list to limited font families

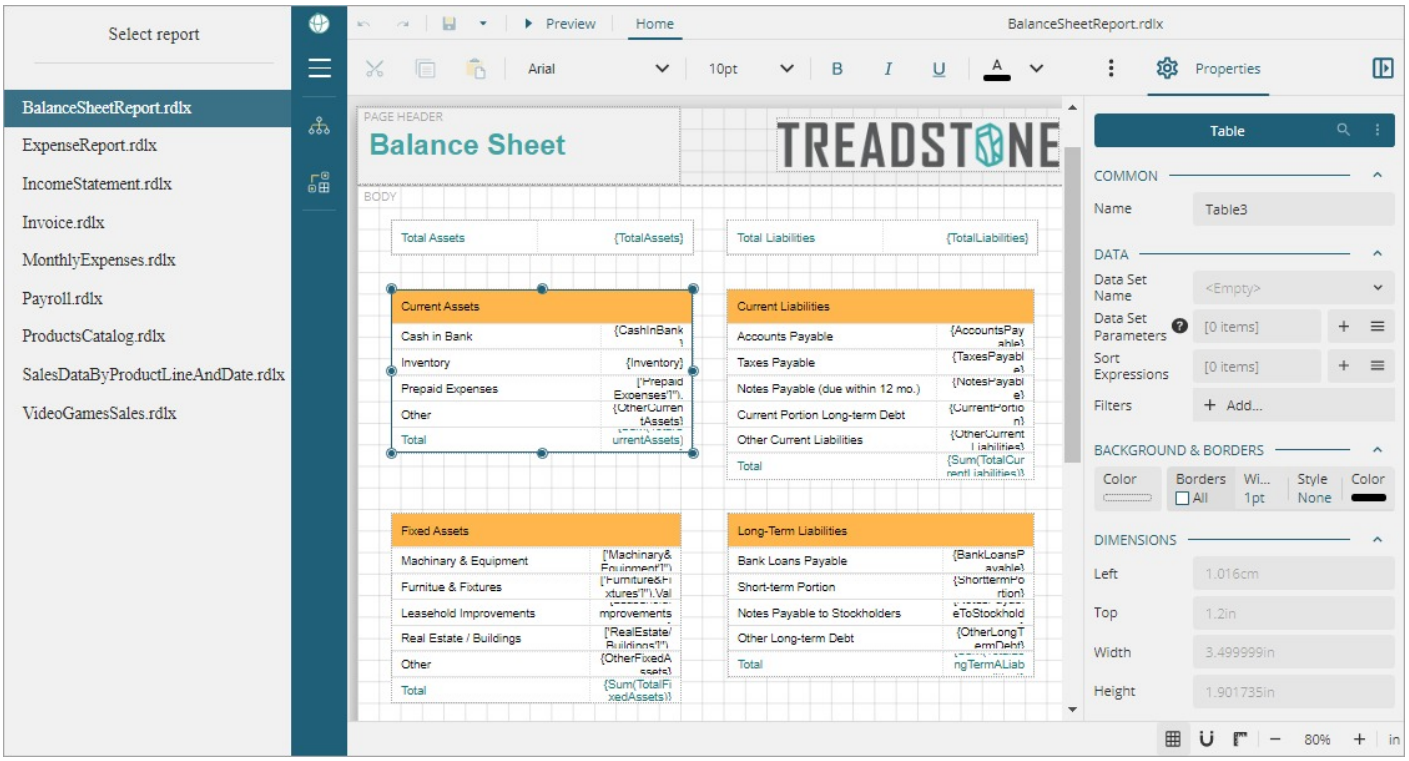
```
fonts : [
                        { header: 'Questionable Choice' },
                        { label: 'Pretty Font', value: 'Comic Sans MS' },
                        { header: '' },
                         'Arial',
                         'Courier New',
                         'Times New Roman'
        ]
```

- Property grid uses the **Basic** mode to show the properties of report elements

```
propertyGrid: { mode:'Basic'}
```

WebDesigner Blazor Custom

This sample shows how to use the Blazor WebDesigner API to create a customized UI for ActiveReports WebDesigner in Blazor.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

C#

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_Blazor_Custom

Details

This sample utilizes the WebDesigner API to create a customized UI for the WebDesigner. Some of the customizations demonstrated in the sample are shown below. You can check the **Index.razor** page of the sample for complete implementation.

- Use a **custom logo** (white labeling)

```
_menuSettings = new MenuSettings()
        {
            Logo = new Logo() { Custom = new MenuIcon() { Type = "css", Class =
"example-icon" } }
        };
```

- Hide **About** button

```
_appBarSettings = new AppBarSettings
        {
            AboutButton = new AboutButton() { Visible = false }
        };
```

- Hide **File** menu

```
_documentsSettings = new DocumentsSettings()
        {
            FileView = new FileView() { Visible = false }
        };
```

- **Lock the report layout**, that is, disable the operations that modify the report layout structure

```
<div id="ar-web-designer" class="ar-web-designer">
   <ReportDesigner @ref="_designer"
                   LockLayout="true"
</div>
```

- Hide **ToolBox**

```
_menuSettings = new MenuSettings()
        {
            ToolBox = new ToolBox() { Visible = false }
        };
```

- Hide **Parameters tab**, the design area for designing a custom parameter pane

```
_appBarSettings = new AppBarSettings
        {
            ParametersTab = new ParametersTab() { Visible = false }
        };
```

- Hide **Data tab**

```
_dataSettings = new DataSettings() { DataTab = new DataTab() { Visible = false } };
```

Hiding **Data tab** hides the options such as data sources, data sets, parameters, and common values.

- Hide **Properties Mode** button that helps switch between the Advanced and Basic properties of the report elements

```
_statusBarSettings = new StatusBarSettings()
    {
        PropertiesModeButton = new PropertiesModeButton() { Visible = false }
    };
```

- Limit font list to limited font families

```
_fonts = new object[]
   {
       new FontHeader() { Header = "Questionable Choice" },
       new Font() { Label = "Pretty Font", Value = "Comic Sans MS" },
```

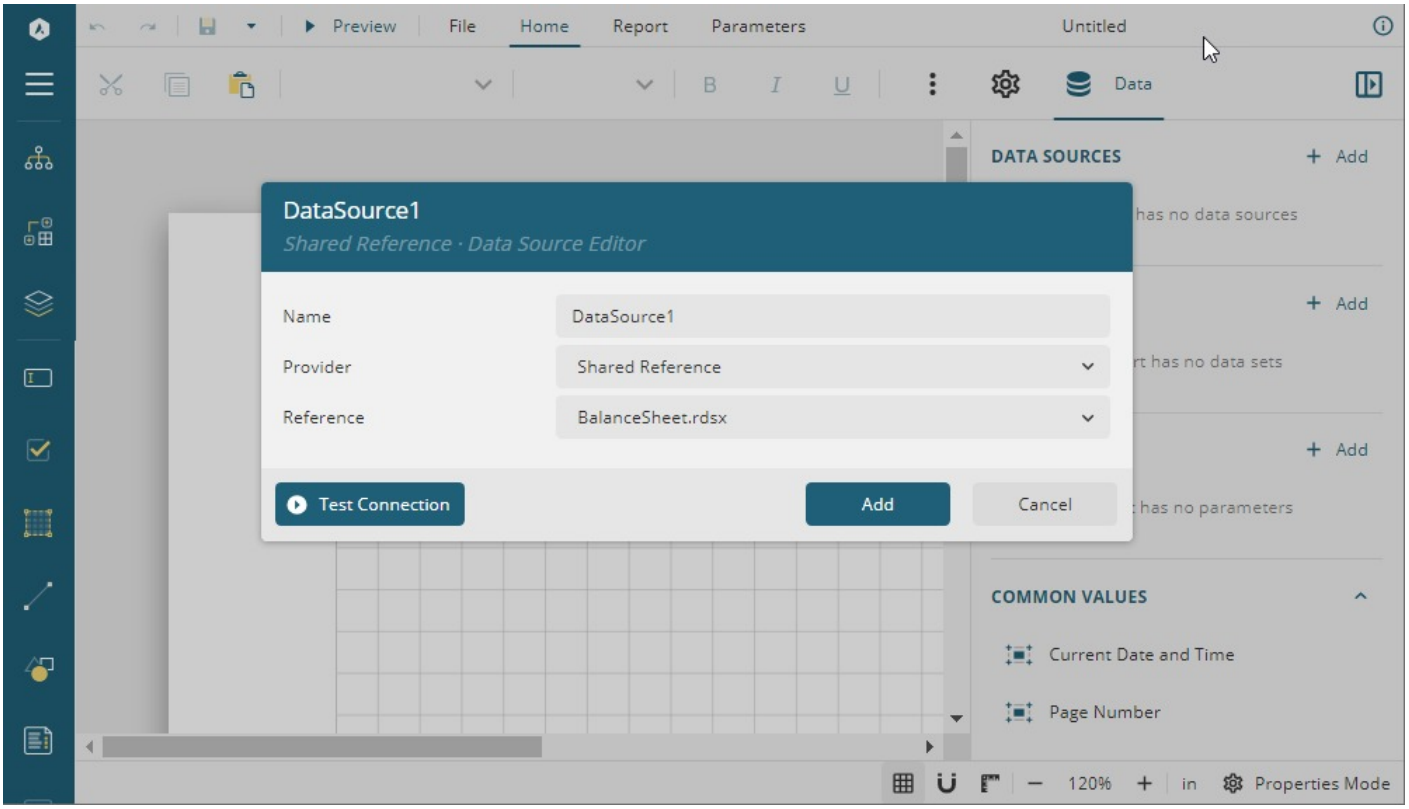
```
        new FontHeader() { Header = "" },
        "Arial",
        "Courier New",
        "Times New Roman"
    };
```

- Property grid uses the **Basic** mode to show the properties of report elements

```
_propertyGridSettings = new PropertyGridSettings() { Mode = Mode.Basic };
```

WebDesigner Custom Shared Data Sources

This sample demonstrates enabling shared data sources in ActiveReports WebDesigner with an ASP.NET Core back-end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

https://github.com/activereports/WebSamples18/tree/main/WebDesigner_CustomSharedDataSources

Details

When you run the sample, the WebDesigner opens in your browser wherein you can use shared data sources in your reports. The project consists of the following elements.

- **wwwroot**: Contains index.html and other static resources.
- **Implementation**: Contains implementation for 'IReportStore' and 'IResourceRepositoryProvider' .
- **resources**: Contains reports, data sources (csv, json, sqlite), and shared data sources (*.rdsx).
- **Startup.cs**: Does the following:
    1. configures the services and middleware used by the application
    2. registers the 'IReportStore' and 'IResourceRepositoryProvider' as singleton services
    3. adds reporting and designer services
    4. sets the path to the ActiveReports.config file, where the SQLite provider is added
    5. configures the reporting and designer middleware
    6. serves static files

JSViewer Samples

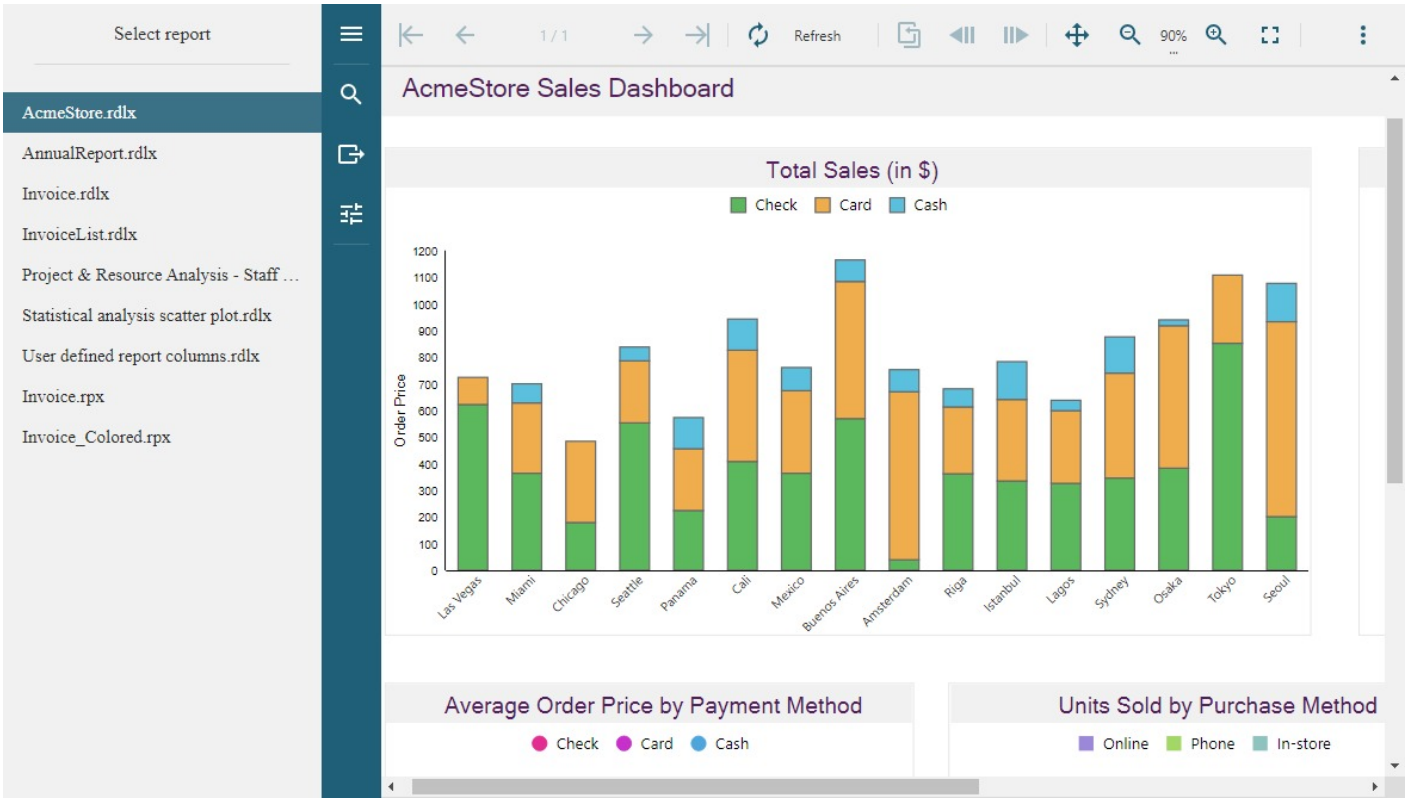
Blazor Viewer

Three samples are available for the Blazor Viewer demonstration.

> **Note**: To run these samples, you must have
>
> - Visual Studio 2022 (version 17.0 or later)
> - .NET 6.0 SDK or later
> - .NET Core Hosting Bundle (for deployment to IIS)

BlazorViewerServer

The sample demonstrates creating a Blazor Server Application with ActiveReports Blazor Viewer, using local report service and remote report service.



Sample Location

https://github.com/activereports/WebSamples18/tree/main/BlazorViewer/BlazorViewerServer

Details

When you run the sample, the Blazor Viewer opens in your browser. Clicking the report link in the left panel opens the report for preview.

The project uses the **MESCIUS.ActiveReports.Aspnetcore.Viewer** and the **MESCIUS.ActiveReports.Blazor.Viewer** NuGet packages.

The project consists of the following elements.

- wwwroot: Contains viewer CSS file for the Blazor application.
- Pages: This folder contains Razor pages and supporting files.
- _Imports.razor: The Razor template file to include the directives.
- Program.cs file: Create and run web host instance.
- Startup.cs: Contains necessary code to add services and configure the HTTP request.

ReportService

The sample demonstrates creating a remote report server for report processing to be used by ActiveReports Blazor Viewer.

Sample Location

https://github.com/activereports/WebSamples18/tree/main/BlazorViewer/ReportService

Details

When you run the sample, the browser displays the report service as running. The project uses the **MESCIUS.ActiveReports.Aspnetcore.Viewer** NuGet package.

The project consists of the following elements.

- Startup.cs: Contains necessary code to add services, configure the HTTP request and CORS.
- Reports: This folder contains all available reports.

BlazorViewerWebAssembly

The sample demonstrates how to make Blazor WebAssembly Application with ActiveReports Blazor Viewer using remote report service.

Sample Location

https://github.com/activereports/WebSamples18/tree/main/BlazorViewer/BlazorViewerWebAssembly

Details

You need to first build and run **ReportService** sample to start the report service. When you run the **BlazorViewerWebAssembly** sample, the Blazor Viewer opens in your browser. Clicking the report link in the left panel opens the report for preview.

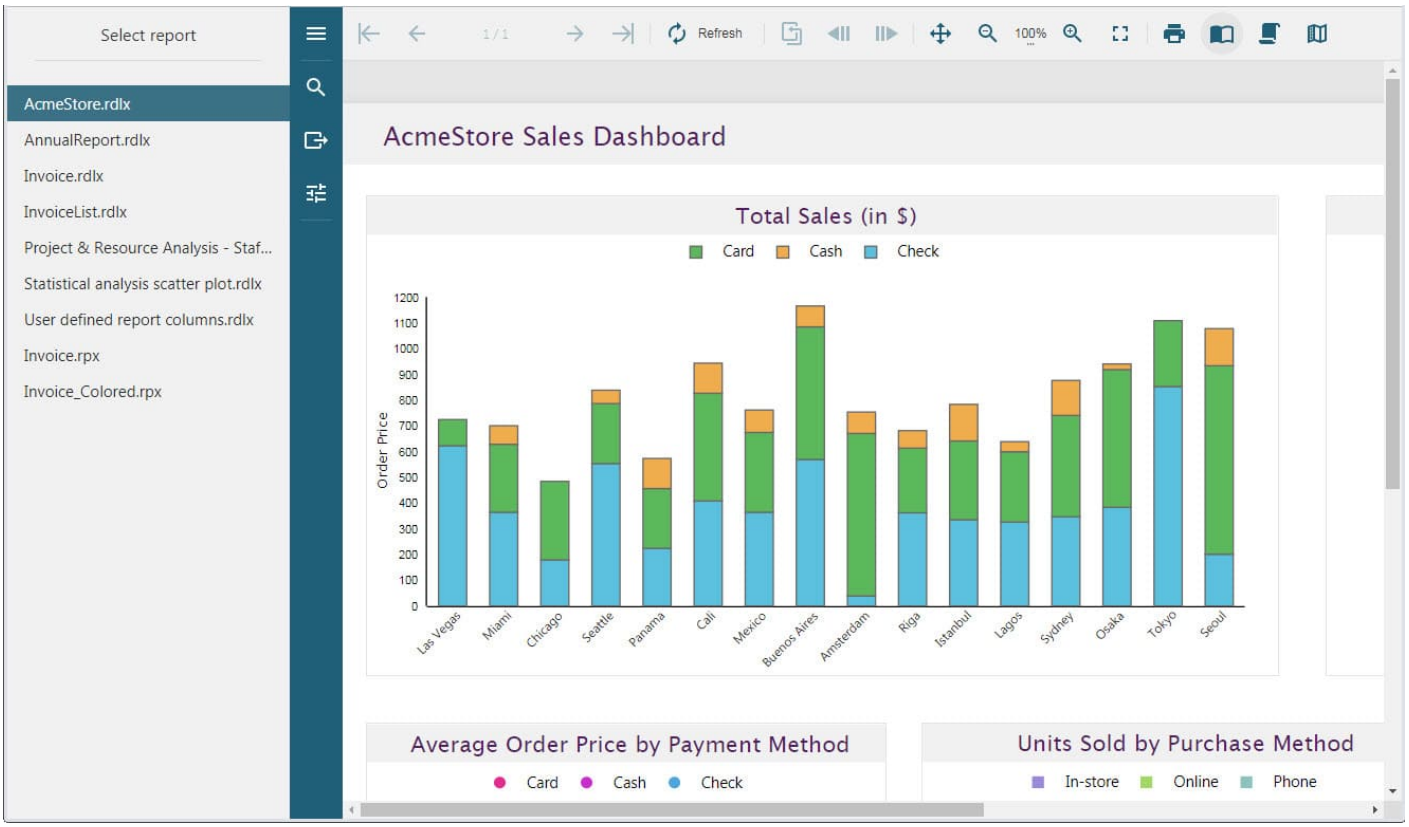
The project uses the **MESCIUS.ActiveReports.Blazor.Viewer** NuGet package.

The project consists of the following elements.

- wwwroot: Contains viewer CSS and index.html file for the Blazor application.
- Pages: This folder contains Razor pages and supporting files.
- _Imports.razor: The Razor template file to include the directives.
- Program.cs file: Create and run web host instance.

JSViewer Angular(Core)

The JSViewer_Angular(Core) sample demonstrates the use of the ActiveReports JSViewer with an Angular 8 app and ASP.NET Core back-end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)
- Angular 8 requires Node.js (version 10 or later)

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_Angular_Core

Details

When you run the sample, the default page appears in your browser. This page provides links to reports that demonstrate the use of the ActiveReports JSViewer with an Angular 7 app and ASP.NET Core back-end.

Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- InvoiceList.rdlx
- Invoice_Colored.rpx
- Project&ResourceAnalysis
- Statistical analysis scatter plot.rdlx - Staff Performance Analysis.rdlx
- User defined report columns.rdlx

> **Note**: The timeout error sometimes appears on running the JSViewer_Angular(Core) sample with default settings. In this case, you should increase the connection timeout period. See Troubleshooting for details on how to resolve this issue.

The project consists of the following elements.

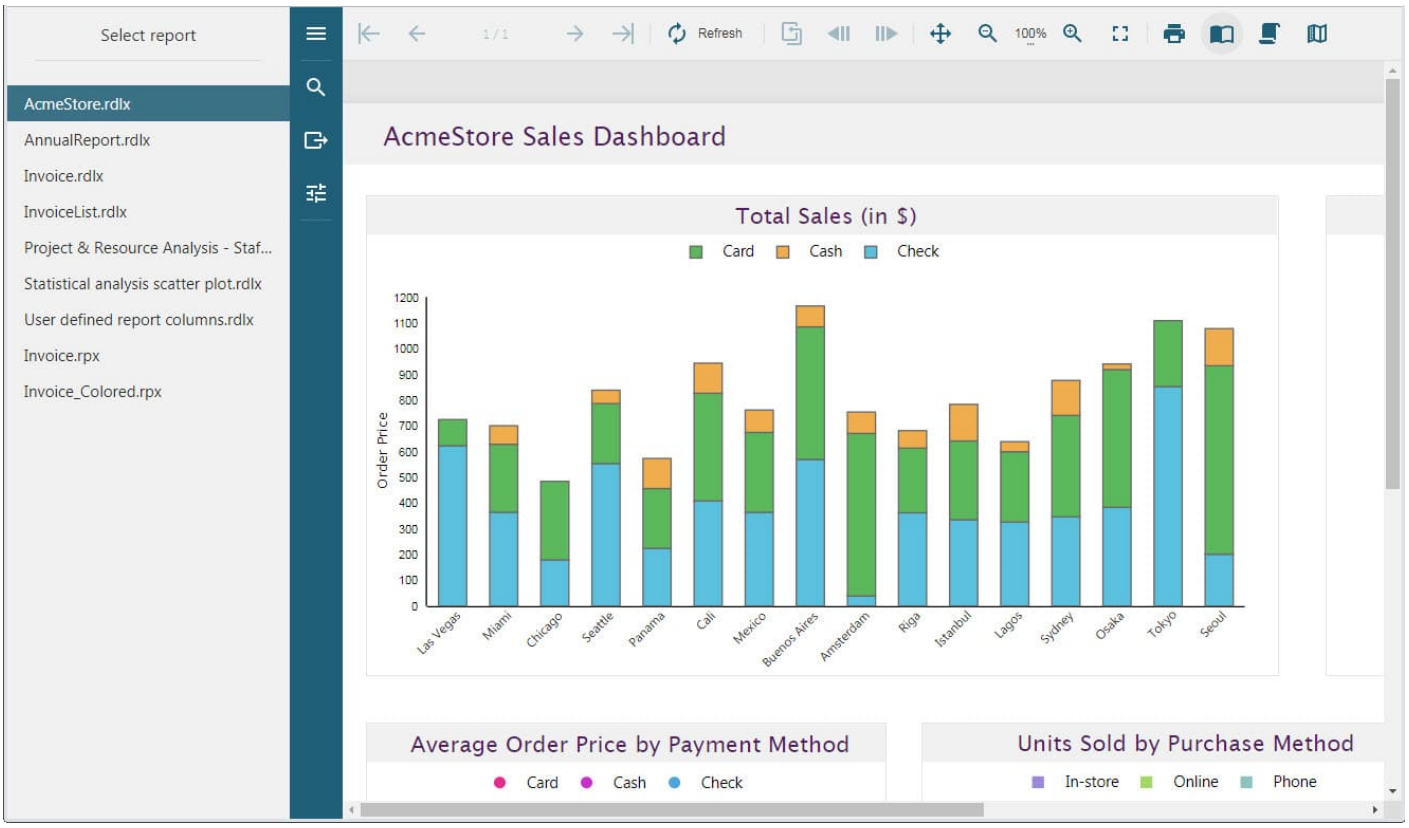
- ClientApp folder: This folder contains a standard Angular CLI app that is used for all UI concerns.
- Controllers folder: This folder contains the **ReportsController** files. The **ReportsController** handles the interaction with reports when a report is selected in the left panel.
- appsettings.json: The json configuration file.
- readme: This file contains the instructions on how to run the sample project.
- Startup.cs: This is the default startup file.
- Web.config: This configuration file contains the httpHandlers that allow ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.

> ⚠ Before publishing the sample, you must do the following.
>
> - In the **JSViewer_Angular(Core).csproj** file, set the **PublishToIIS** property to true as follows: `<PublishToIIS>true</PublishToIIS>`
> - Copy the sample **ViewerApp** folder to the **publish** folder.

JSViewer CORS

The JSViewer_CORS sample demonstrates using the ActiveReports JSViewer when the server is hosted elsewhere. The sample consists of two applications - client and server, to demonstrate using the CORS (Cross-origin resource sharing) where

- a client requests resources (reports) from one server and
- the server sends a response back to the client.



Note: To run each application in this sample, you must have:

- Visual Studio 2017 (version 17.0 or later)
- .NET Framework Dev Pack (version 4.6.2 or later)

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_CORS

Details

First, you need to build both applications and then run the **Server** application followed by the **Client** application.

Note: The client url is specified in the **web.config** as the value of the custom header "Access-Control-Allow-Origin".

When you run the applications, you will see that the client application displays the JSViewer on the browser with a list of reports. Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

- AcmeStore.rdlx
- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- Invoice_Colored.rpx

- InvoiceList.rdlx
- Project & Resource Analysis - Staff Performance Analysis.rdlx
- Statistical analysis scatter plot.rdlx
- User defined report columns.rdlx

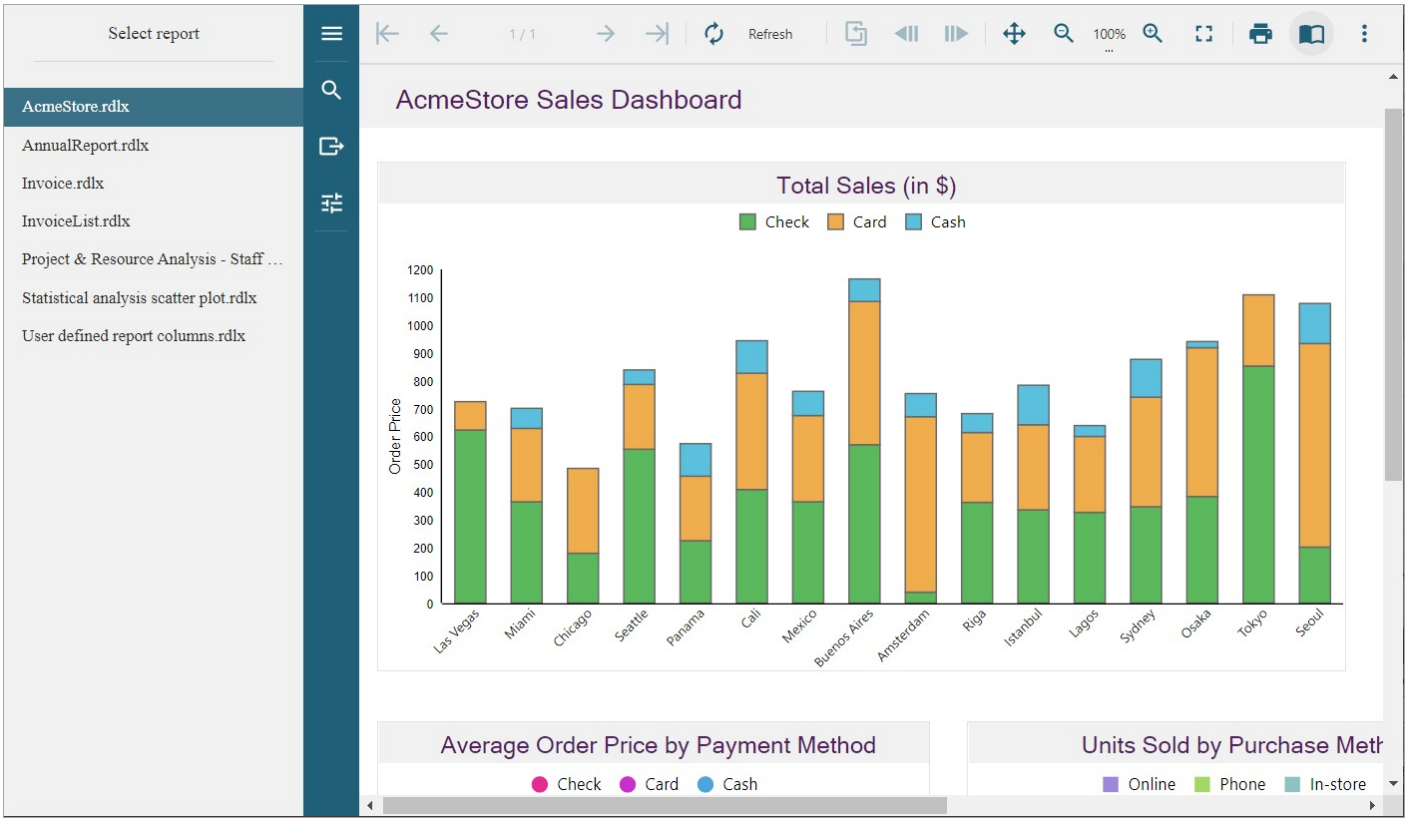
The Server application consists of the following elements.

- Controllers folder: This folder contains the **HomeController** that handles the user interaction and returns the main Index view.
- Global.asax: The default class that sets global URL routing values for this web application.
- packages.config
- Startup.cs: The startup file adds the UseReportViewer() middleware to configure the middleware for ActiveReports API and handlers.
- Web.config: This configuration file contains the necessary markup for CORS to work and the actual client url.

JSViewer CORS(Core)

The JSViewer_CORS_Core sample demonstrates using the ActiveReports JSViewer when the server is hosted elsewhere with an ASP.NET Core back-end. The sample consists of two applications - client and server, to demonstrate using the CORS (Cross-origin resource sharing) where

- a client requests resources (reports) from one server and
- the server sends a response back to the client.



Note: To run each application in this sample, you must have:

> - Visual Studio 2022 (version 17.0 or newer)
> - .NET Framework Dev Pack (version 4.6.2 or later)

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_CORS_Core

Details

First, you need to build both applications and then run the **Server** application followed by the **Client** application.

When you run the applications, you will see that the client application displays JSViewer on the browser with a list of reports. Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

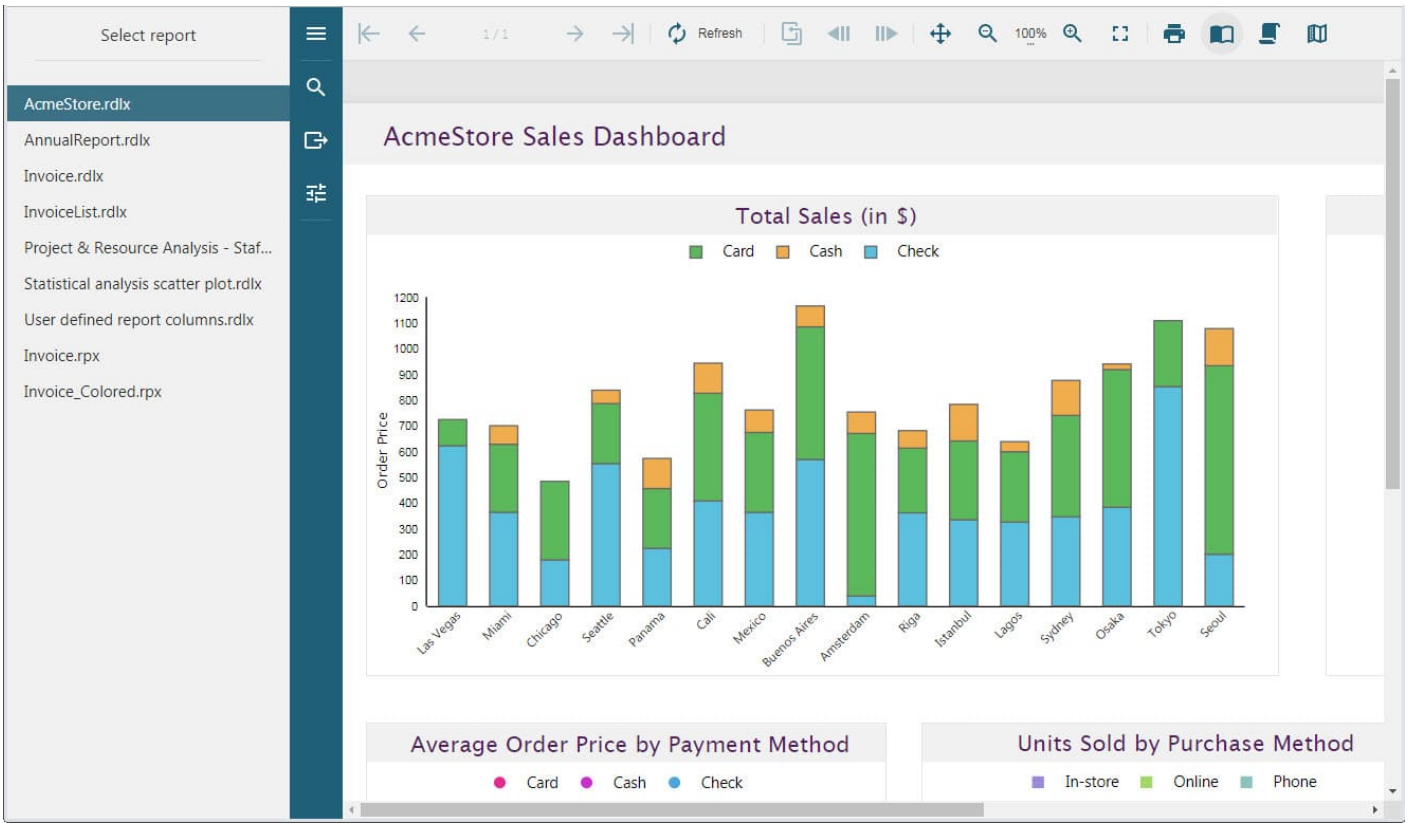
- AcmeStore.rdlx
- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- InvoiceList.rdlx
- Invoice_Colored.rpx
- Project & Resource Analysis - Staff Performance Analysis.rdlx
- Statistical analysis scatter plot.rdlx
- User defined report columns.rdlx

The Server application consists of the following elements.

- Controllers folder: This folder contains the **HomeController** that handles the user interaction and returns the main Index view.
- packages.config
- Startup.cs: The startup file adds the **UseReportViewer()** middleware to configure the middleware for ActiveReports API and handlers, and the **UseCors()** middleware to enable CORS.
- Web.config: This configuration file contains the httpHandlers that allow ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.

JSViewer MVC

The JSViewer_MVC sample demonstrates the use of the ActiveReports JSViewer with an ASP.NET MVC 5 back-end.



Note: To run this sample, you must have:

- Visual Studio 2017 (version 17.0 or later)
- .NET Framework Dev Pack version 4.6.2 or later

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_MVC

Details

When you run the sample, the default page appears in your browser. This page provides links to reports that demonstrate the use of the ActiveReports JSViewer with an ASP.NET MVC 5 back end.

Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- Invoice_Colored.rpx
- InvoiceList.rdlx
- Project & Resource Analysis - Staff Performance Analysis.rdlx
- User defined report columns.rdlx
- Statistical analysis scatter plot.rdlx

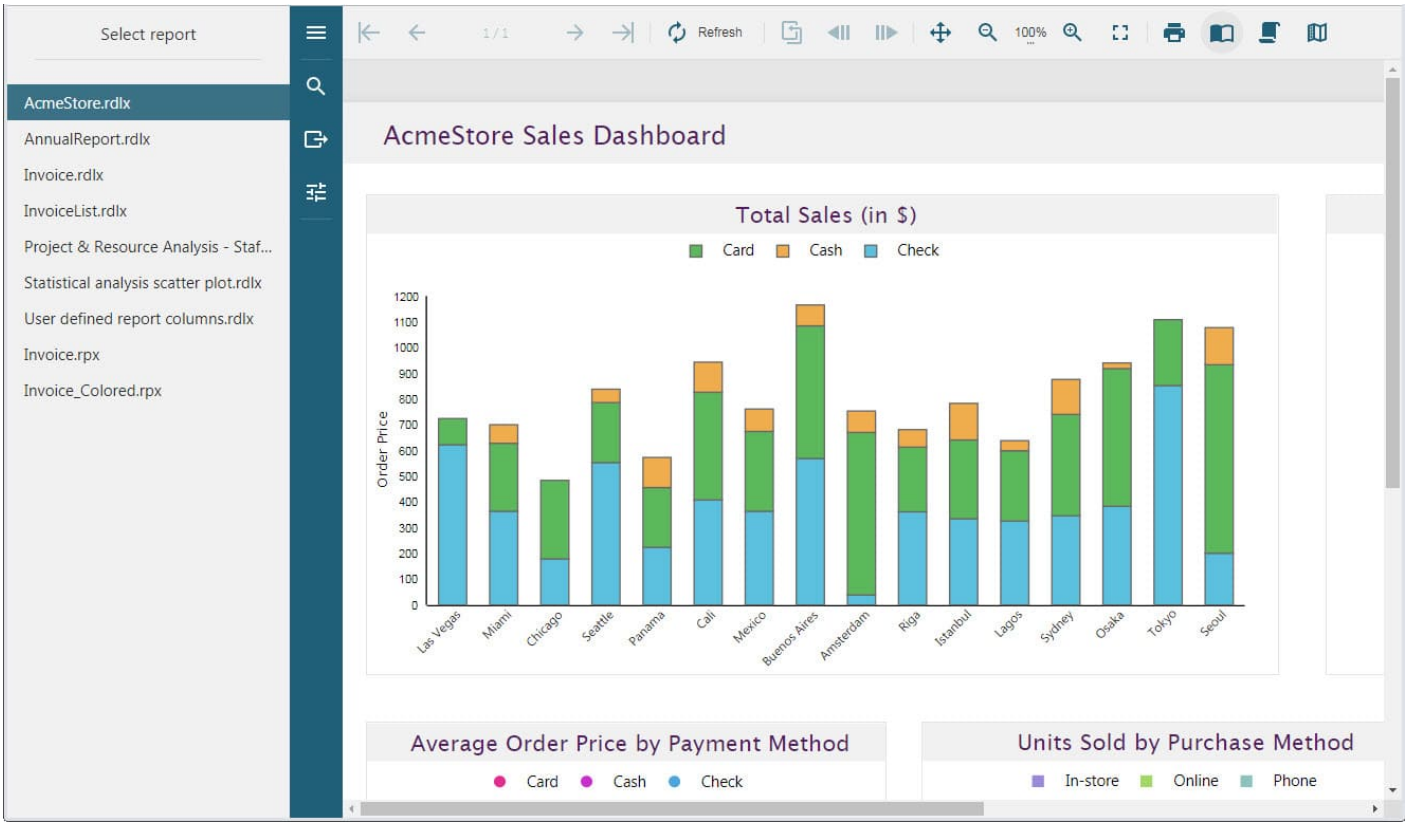
The project consists of the following elements.

- Controllers folder: This folder contains the **HomeController** that handles the user interaction and returns the main Index view.
- ViewerApp folder: Contains JSViewer CSS and JavaScript files.
- Global.asax: The default class that sets global URL routing values for this web application.
- packages.config
- Startup.cs: This is the default startup file.
- Web.config: This configuration file contains the httpHandlers that allow ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.

⚠ Before publishing the sample, you must copy the sample **ViewerApp** folder to the **publish** folder.

JSViewer MVC(Core)

The JSViewer_MVC(Core) sample demonstrates the use of the ActiveReports JSViewer with an ASP.NET Core back-end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK or later
- .NET Core Hosting Bundle (for deployment to IIS)

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_MVC_Core

Details

When you run the sample, the JSViewer opens in your browser. The viewer provides links to reports to demonstrate the ActiveReports JSViewer with an ASP.NET Core back-end.

Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- InvoiceList.rdlx
- Invoice_Colored.rpx
- Project & ResourceAnalysis - Staff Performance Analysis.rdlx
- Statistical analysis scatter plot.rdlx
- User defined report columns.rdlx

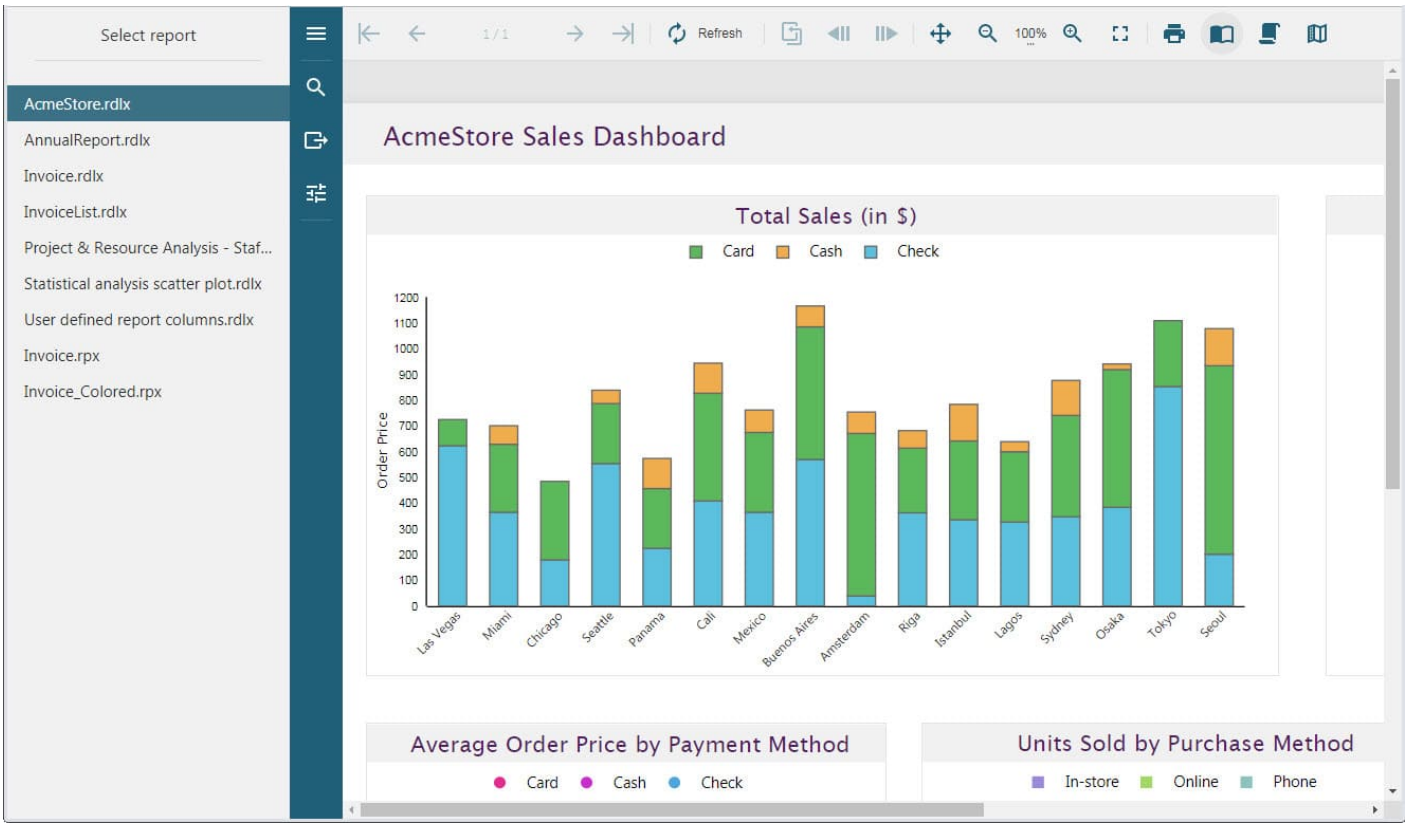
The project consists of the following elements.

- Controllers folder: This folder contains the **HomeController** that handles the user interaction and returns the main Index view.
- ViewerApp folder: Contains JSViewer CSS and JavaScript files.
- readme: This file contains the instructions on how to build and run the sample project.
- Startup.cs: This is the default startup file.
- Web.config: This configuration file contains the httpHandlers that allow ActiveReports to process reports on the Web. Note that you need to manually update version information here when you update your version of ActiveReports.

> ⚠ Before publishing the sample, you must do the following.
>
> - In the **index.html** file, uncomment the following line:
>   `<!--<base href="/JSViewer_MVC_Core/">-->`
> - Copy the sample **ViewerApp** folder to the **publish** folder.

JSViewer React(Core)

This sample demonstrates the use of the ActiveReports JSViewer with a ReactJS app and the ASP.NET Core back end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)
- Node.js 16.x or 18.x

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_React_Core

Details

When you run the sample, the JSViewer opens in your browser. The viewer provides links to reports to demonstrate the ActiveReports JSViewer with a ReactJS app and the ASP.NET Core back end.

Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

- AcmeStore.rdlx
- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- InvoiceList.rdlx
- Invoice_Colored.rpx
- Project & ResourceAnalysis - Staff Performance Analysis.rdlx
- Statistical analysis scatter plot.rdlx

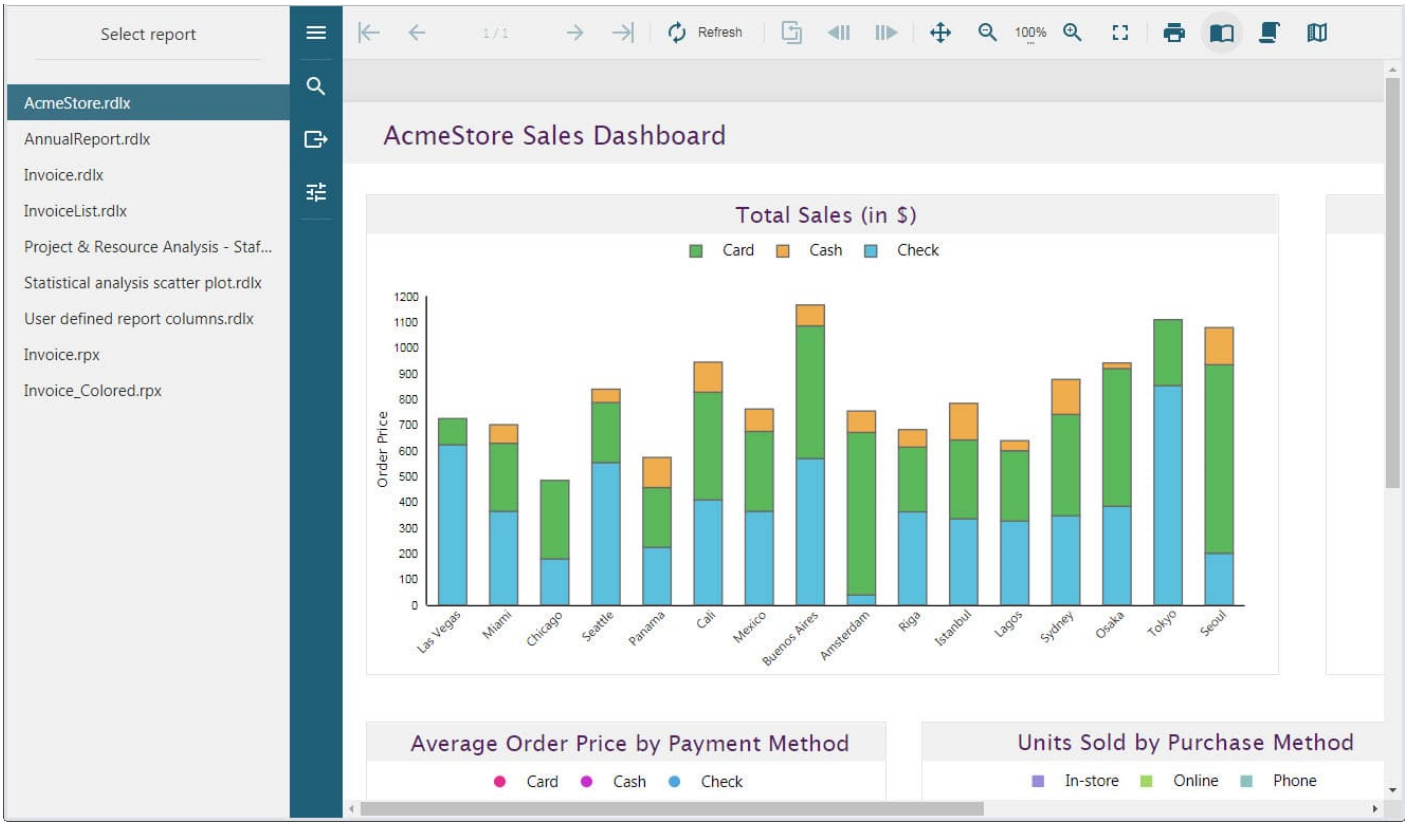
- User defined report columns.rdlx

The project consists of the following elements.

- **Controllers** folder: This folder contains the **HomeController** that handles the user interaction and returns the main Index view.
- **ClientApp** folder: This folder contains a standard Angular CLI app that is used for all UI concerns.
- **readme**: This file contains the instructions on how to build and run the sample project.
- **Startup.cs**: This is the default startup file.
- **wwwroot**: Contains designer CSS and JavaScript files.

JSViewer React(Hooks)

This sample demonstrates the use of the ActiveReports JSViewer with a ReactJS app using hooks and the ASP.NET Core back end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)
- Node.js 16.x or 18.x

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_React_Hooks

Details

When you run the sample, the JSViewer opens in your browser. The viewer provides links to reports to demonstrate the ActiveReports JSViewer using Hooks with a ReactJS app and the ASP.NET Core back end.

Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

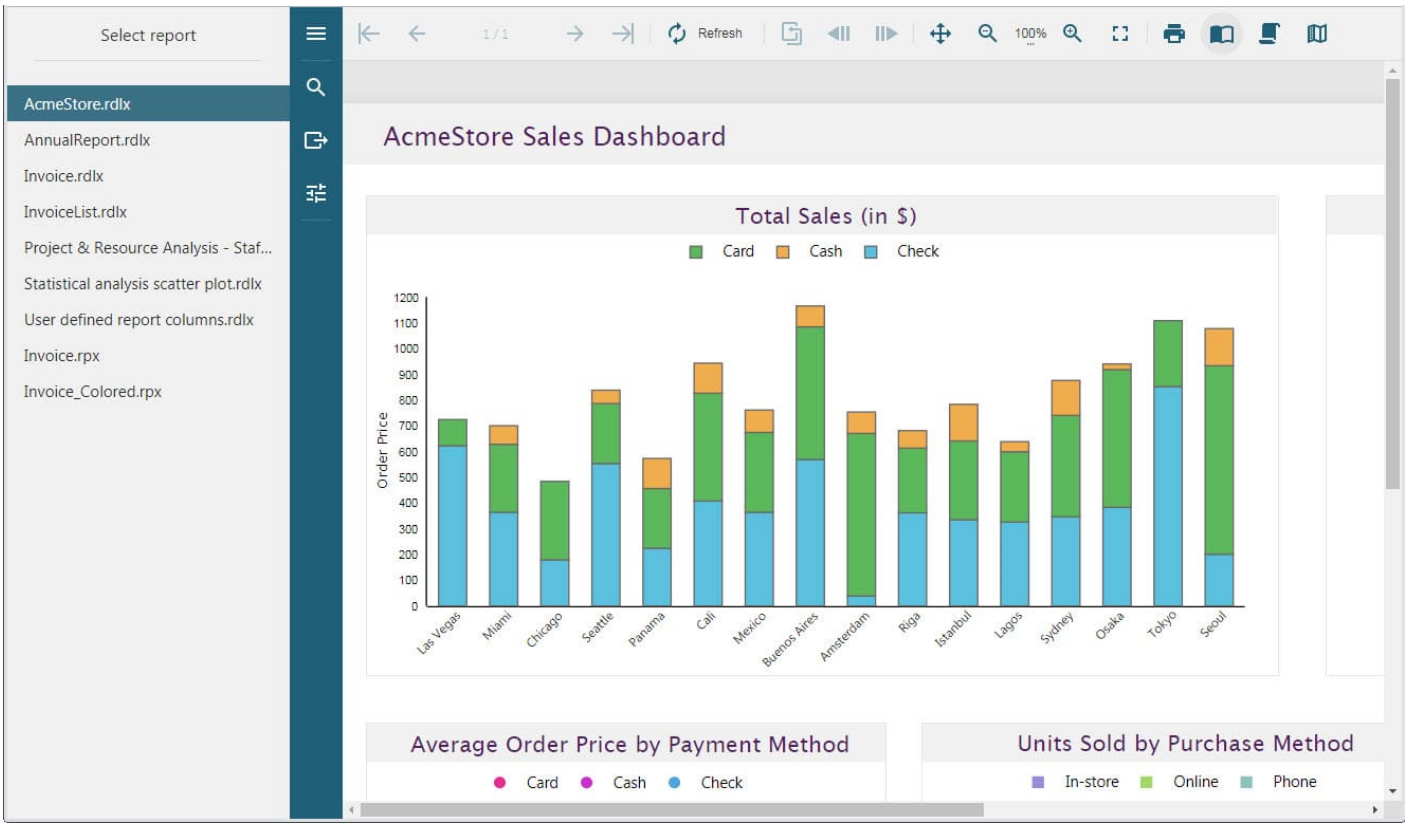
- AcmeStore.rdlx
- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- InvoiceList.rdlx
- Invoice_Colored.rpx
- Project & ResourceAnalysis - Staff Performance Analysis.rdlx
- Statistical analysis scatter plot.rdlx
- User defined report columns.rdlx

The project consists of the following elements.

- **Controllers** folder: This folder contains the **HomeController** that handles the user interaction and returns the main Index view.
- **ClientApp** folder: This folder contains a standard Angular CLI app that is used for all UI concerns.
- **readme**: This file contains the instructions on how to build and run the sample project.
- **Startup.cs**: The startup file adds the **UseReportViewer()** middleware to configure the middleware for ActiveReports API and handlers, and the **UseCors()** middleware to enable CORS. The **AddCors()** service call adds CORS services to the app's service container.
- **wwwroot**: Contains designer CSS and JavaScript files.

JSViewer Vue(Core)

The sample demonstrates the use of the ActiveReports JSViewer with an VueJS app and ASP.NET Core back end.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET 6.0 SDK
- .NET Core Hosting Bundle (for deployment to IIS)
- Node.js 16.x or 18.x

Sample Location

https://github.com/activereports/WebSamples18/tree/main/JSViewer_Vue_Core

Details

When you run the sample, the JSViewer opens in your browser. The viewer provides links to reports to demonstrate the ActiveReports JSViewer with a ReactJS app and the ASP.NET Core back end.

Clicking the report link in the left panel opens the report for preview. You can preview the following reports.

- AcmeStore.rdlx
- AnnualReport.rdlx
- Invoice.rdlx
- Invoice.rpx
- InvoiceList.rdlx
- Invoice_Colored.rpx
- Project & ResourceAnalysis - Staff Performance Analysis.rdlx
- Statistical analysis scatter plot.rdlx

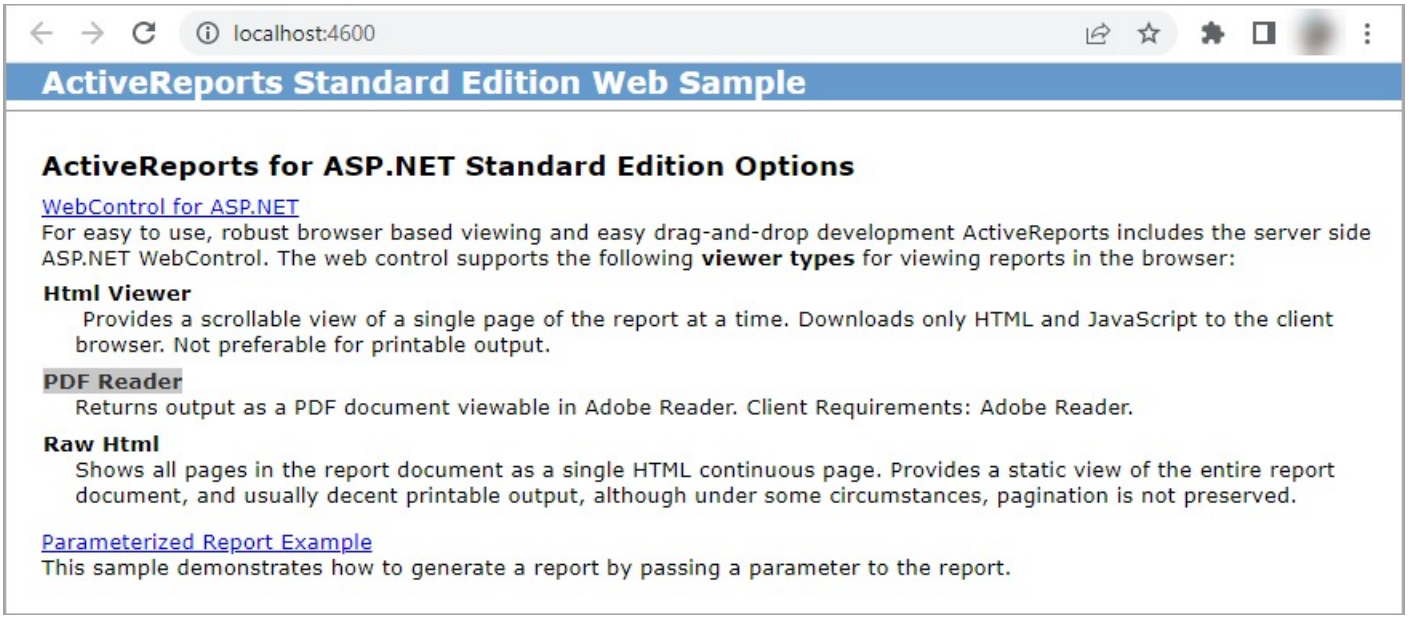
- User defined report columns.rdlx

The project consists of the following elements.

- **Controllers** folder: This folder contains the **HomeController** that handles the user interaction and returns the main Index view.
- **ClientApp** folder: This folder contains a standard Angular CLI app that is used for all UI concerns.
- **readme**: This file contains the instructions on how to build and run the sample project.
- **Startup.cs**: This is the default startup file.
- **wwwroot**: Contains designer CSS and JavaScript files.

WebViewer ASP.NET

ActiveReports WebViewer ASP.NET sample describes the standard ActiveReports web control feature and generating a parameterized report.



Note: To run this sample, you must have

- Visual Studio 2022 (version 17.0 or later)
- .NET Framework 4.7.2

Sample Location

Visual Basic.NET

https://github.com/activereports/WebSamples18/tree/main/WebViewer_ASP.NET_VB.NET

C#

https://github.com/activereports/WebSamples18/tree/main/WebViewer_ASP.NET_C#

Details

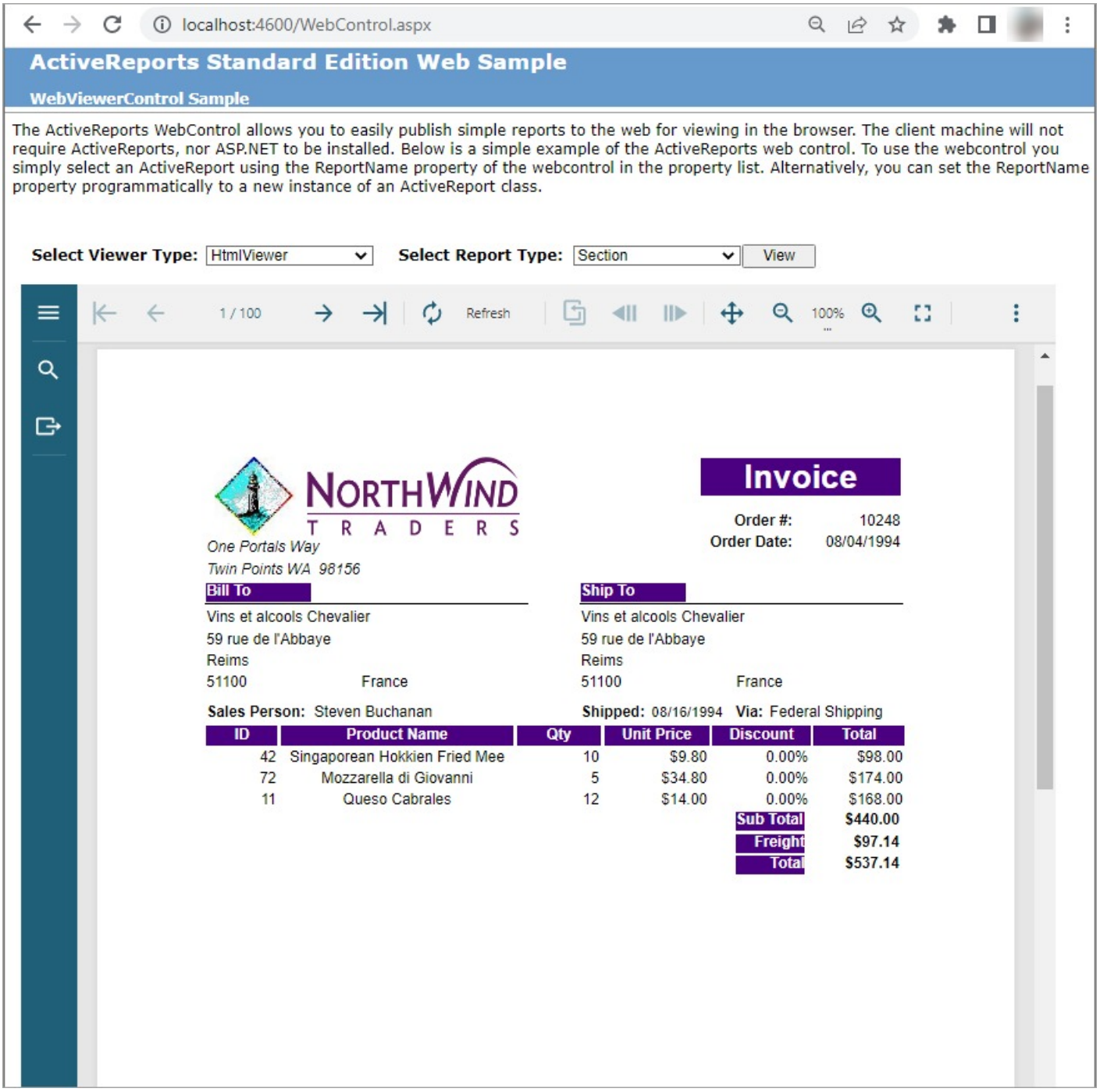
When you run the sample, the Default.aspx page appears in your browser. This page provides links to other sample

pages that demonstrate the following web features.

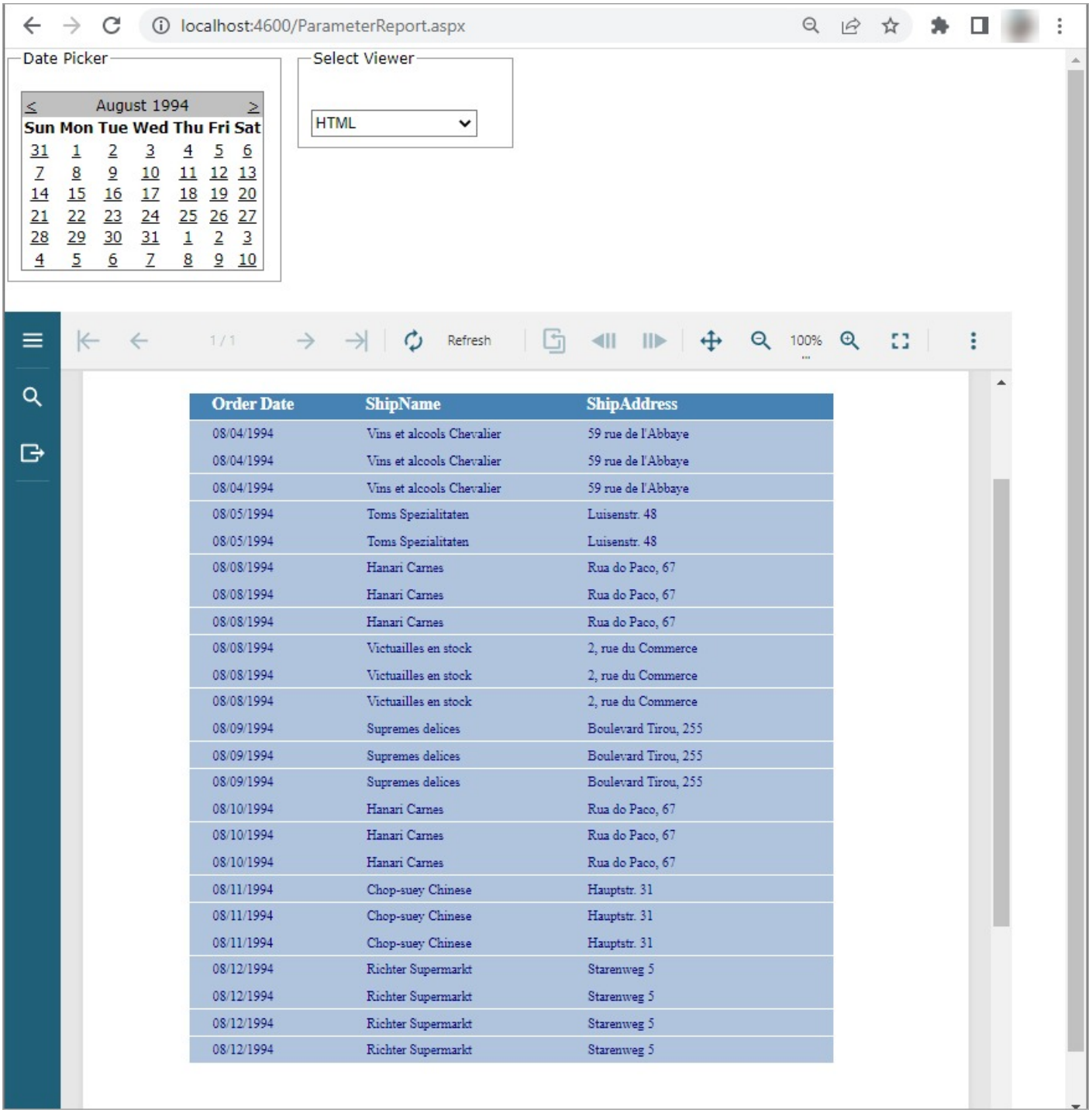
WebControl for ASP.NET: This link opens the WebControl.aspx page that allows you to select any of the three Viewer Types and it also allows you to select from Section, Page and RDLX report Type.

The three Viewer Types that are available are as follows:

- HtmlViewer
- AcrobatReader
- RawHtml



Parameterized Report Example: This link opens the page that demonstrates how to generate a report by passing a parameter to the report.



The project contains the following elements:

- PageReports: The PageReports folder contains the **Invoice_Grouped** and **PurchaseReport** report.
- RDLXReports: The RdlxReports folder contains the **SalesReceipt** report.

- RPXReports: The RpxReports folder contains the following reports - **Invoice**, **InvoiceFiltered**, **NwindLabels**, **NwindLabelsFiltered**, and **Params**.
  The Invoice.rpx report is used to demonstrate the WebViewer control options and is opened by clicking WebControl for ASP.NET on the Default.aspx page. For detailed information on the Invoice report, see the Cross Section Control Sample.

  The Params report is used by the ParameterReport.aspx page to demonstrate how to generate a report by passing a parameter to the report.

- Default.aspx: This is the main Web form of the sample that shows the introductory text and links to the following sample pages.
  - WebControl for ASP.NET (WebControl.aspx)
  - Parameterized Report Example(ParameterReport.aspx)

- ParameterReport.aspx: The web form that demonstrates how to generate a report by passing a parameter to the report. This sample uses the Params report from the RpxReports folder of this project. The date list is created by changing the SQL query of the report at run time. In this sample, when the date is selected from the Calendar control, the SQL query is updated and the report is generated. The report is generated dynamically in the SelectedIndexChanged event of the Calendar control. On this form, you can select the Viewer to display the report - **HTMLViewer**, **AcrobatReader**, or **RawHTML**.

  > **Note**: This sample requires write permissions to the ReportOutput folder that is located in the web samples directory.

- Web.config: The configuration file that allows ActiveReports to process reports on the Web.
- WebControl.aspx: This page is opened by clicking **WebControl for ASP.NET** on the Default.aspx page. By default, it displays the WebViewer control with the Invoice report. In this page, you can select from HTMLViewer, AcrobatReader, RawHtml viewer types and can also select from Page, Section and RDLX report types to be displayed.

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

The missing Microsoft.SqlServer.Types assembly may cause the query execution failure in VQD

Symptoms: The missing Microsoft.SqlServer.Types assembly may cause the invalid query error at executing a dataset in the Visual Query Designer.

Cause: The missing Microsoft.SqlServer.Types assembly is required to execute the query.

Solution: To resolve this problem, install the Microsoft.SqlServer.Types assembly manually.

Compilation Error related to conflicting assembly/package versions

Symptoms: Errors like - could not load file or assembly or one of its dependencies, or Found conflicts between different versions of the same dependent assembly appear on compiling.

Cause: Due to inability of Visual Studio to resolve assembly versions on its own.

Solution: To resolve this problem, add a binding redirect in your config file. You can quickly add a binding redirect by double-clicking on the error in Visual Studio.

Example to add bindingRedirect to config file

```
<dependentAssembly>
  <assemblyIdentity name="someAssembly"
    publicKeyToken="tokenName"
    culture="neutral" />
  <bindingRedirect oldVersion="7.0.0.0" newVersion="8.0.0.0" />
</dependentAssembly>
```

Visual Studio 2017 fails to build a VB.NET project sample

Symptoms: Visual Studio 2017 fails to build a VB.NET project sample with the following error:

Error BC30284 You cannot declare sub'OnCreateMainForm' as'Overrides' because you do not override sub in the base class.

Cause: Visual Studio 2017 does not fully support WinForms in VB.NET projects that use Microsoft.NET.Sdk.

Solution: To resolve this issue, you can do either of the following:

1. Use Visual Studio 2019 or above.
2. Use legacy .csproj format without Microsoft.NET.Sdk.

Error appears on exporting Page/RDLX reports from JSViewer run on ASP.NET Core MVC application

Symptoms: "Export report error" appears on exporting Page/RDLX Report from JSViewer when run through ASP.NET Core MVC applications.

Cause: ASP.NET Core MVC has undergone some changes that disable the synchronous server operations. See this for more information.

Solution: Add following content in Startup.cs to turn on the synchronous operations.

Startup.cs

```
services.Configure<IISServerOptions>(options =>
{
    options.AllowSynchronousIO = true;
});
```

Error appears on using parameterized queries with OLE DB provider in .NET core applications

Symptoms: Error is thrown on using parameterized queries with OLE DB provider in .NET Core applications.

Cause: The number of query parameters specified in the dataset do not match the parameters used in the SQL query for dataset. Note that this works perfectly well for OLE DB provider from Full .NET Framework.

Solution: The number of query parameters specified in the SQL query should be same as number of query parameters in dataset.

Report menu does not appear in the main menu of Visual Studio 2019 or Visual Studio 2022

Symptoms: When a report is opened in Visual Studio 2019 or Visual Studio 2022, Report menu does not appear in the main menu.

Cause: Due to new behavior relating to extensions, Report menu is removed as main menu.

Solution: Go to **Extensions** menu. You can see that the Report menu is available as submenu.

References missing from Visual Studio Add Reference dialog

Symptoms: When you try to add references to your project, only a few of the ActiveReports references are available.

Cause: The project's target framework is set to an old version of the .NET framework that does not support the new assemblies.

Solution:

1. In the Solution Explorer, right click the project and choose Properties.
2. On the Application tab in C# projects (or the Compile tab, then the Advanced Compile Options button in Visual Basic projects), drop down the Target framework box and select .NET Framework 4.6.2.

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 Panel.

An Exception occurs on previewing reports connecting Microsoft Access OLE DB provider in a 64-bit system

Symptoms: "System.InvalidOperationException: The 'Microsoft.Jet.OLEDB.4.0' provider is not registered on the local machine." occurs on previewing reports connecting to Microsoft.Jet.OLEDB.4.0 provider on a 64-bit operating system.

Cause: The Microsoft Access OLE DB provider, Microsoft.Jet.OLEDB.4.0, is not compatible with 64 bit, so it fails on a 64-bit operating systems.

Solution: To avoid this, you have two options.

1. (Preferred) Change the OLE DB Provider to **Microsoft.ACE.OLEDB.12.0**.

Note that both 32-bit and 64-bit version of Microsoft.ACE.OLEDB.12.0 should be available in your machine.

2. Change the project settings to use only **32 bit**.

1. With the project open in Visual Studio, from the **Project** menu, select Project Properties.
2. In the page that appears, select the **Compile** tab in a VB project, or the **Build** tab in a C# project.
3. Scroll to the bottom of the page and click the **Advanced Compile Options** button in VB, or skip this step in C#.
4. Drop down the **Target CPU** list in VB, or **Platform target** in C#, (set to use AnyCPU by default) and select **x86**.
5. Click **OK** to save the changes, or skip this step in C#.

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. This applies to all Page reports, RDLX reports and Section Reports.

1. In the project where you call the Print method, add a reference to the MESCIUS.ActiveReports.Viewer.Win assembly.
2. At the top of the code file where you call the Print method, add a using directive (Imports for VB) for **GrapeCity.ActiveReports**.
3. Call the Print method with the usePrintingThread parameter (the third parameter) set to false with code like the following.

C# code.

```
document.Print(false, false, false);
```

Visual Basic code.

```
document.Print(False, False, False)
```

Exception thrown when using Viewer.Print to print a report

Symptoms: An exception is thrown when the Viewer.Print method is used to print a report.

Cause: Print method was called before the page was loaded completely.

Solution: Use the Viewer.Print method in the **LoadCompleted ('LoadCompleted Event' in the on-line documentation)** event.

Code generation error appears with code-based report applications

Symptoms: The Code generation error appears when working with a code-based application in Visual Studio.

Cause: This is because of the cache problems in Visual Studio 2019 and Visual Studio 2022.

Solution: Use one of the solutions described below.

- Clean Visual Studio cache.
  - Remove the folder C:\Users\[user name]\AppData\Local\Microsoft\VisualStudio\[VS version dir]\ProjectAssemblies.
- Reset Visual Studio settings (see here for more details). For example, in Visual Studio,
  1. Open cmd.
  2. Switch to C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE folder.
  3. Run `devenv/resetsettings`.

> ⚠ The Visual Studio settings reset may result in the loss of important data.

Timeout error appears on running Angular(Core) samples for WebDesigner and JSViewer with default settings

Symptoms: The timeout error sometimes appears when running the WebDesigner_Angular(Core) and JSViewer_Angular(Core) samples with default settings. See Web Samples for more information.

Cause: The connection timeout period is not sufficient and must be increased.

Solution: To increase the solution timeout period, add the following code to the Startup.cs file.

```
if (env.IsDevelopment())
 {
 spa.UseAngularCliServer(npmScript: "start");
 spa.Options.StartupTimeout = TimeSpan.FromSeconds(200); //timeout
 }
```

System.NotSupportedException occurs on previewing reports with scripts in Windows Forms Viewer or WPF Viewer in the .NET Core desktop applications

Symptoms: When you preview reports using Windows Forms Viewer, WPF Viewer, and Windows Designer components in .NET Core applications, the exception "System.NotSupportedException. No data is available for encoding 1252" occurs. For information on defining a custom encoding, see the documentation for the Encoding.RegisterProvider method.

Cause: You need to register encodings before using .NET Core applications with Windows Forms Viewer, WPF Viewer, and Windows Designer components.

Solution: To avoid this situation, please do the following.

1. Add the System.Text.Encoding.CodePages package from NuGet .
2. Add **Encoding.RegisterProvider** to Program.cs.

```
static void Main()
{
System.Text.Encoding.RegisterProvider(System.Text.CodePagesEncodingProvider.Instance);Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new DesignerForm());
}
```

An exception appears when using a relative path in the connection string in Linux

Symptoms: When you run a project in Linux, an exception appears on rendering a report in Linux, but the same works correctly in Windows.

Cause: This may occur when the connection string uses slash other than forward '/' slash.

Solution: In the connection strings with a relative path to the reports, always use forward slash '/' for the report to render correctly across all platforms.

SqlException appears with the message like "The certificate chain was issued by an authority that is not trusted"

Symptoms: The SqlException with the message like "The certificate chain was issued by an authority that is not trusted" appears.

Cause: This error occurs because the default behavior of client drivers was changed. Now the new driver tries to validate the server's certificate and fails if the certificate is not installed. See The certificate chain was issued by an authority that isn't trusted for details.

Solution: Install the required certificate or use one of the following workarounds.

1. **Update the connection string**. Add the TrustServerCertificate=True setting to the data source connection string.
   Example: **“**`Data Source=20.186.17.78;Initial Catalog=Northwind;User ID=qatester;Password=qatesting;`**`TrustServerCertificate=True`****”**
2. **Rollback to the old System.Data.SqlClient**
   1. Configure a required MSSQL data provider in the ActiveReports.config file.

ActiveReports.config file

```
<Configuration>
    <Extensions>
        <Data>
            <Extension Name="SQL" Type="System.Data.SqlClient.SqlClientFactory, System.Data.SqlClient"
DisplayName="Microsoft SQL Server" />
        </Data>
    </Extensions>
</Configuration>
```

2. Reference the System.Data.SqlClient NuGet package.

Section Report Troubleshooting

Warning related to conflicting assemblies when working with code-based section reports

Symptoms: Visual Studio throws some warnings related to conflicting assemblies when creating projects from our code-based section report templates, or when adding code-based reports to existing projects.

- Found conflicts between different versions of 'System.Runtime.CompilerServices.Unsafe' that could not be resolved.
- Found conflicts between different versions of 'Microsoft.Bcl.AsyncInterfaces' that could not be resolved.

Cause: Due to dependency on different versions of VS SDK required by MESCIUS.ActiveReports.Serializer.VS2022.dll and VS2022. See about **MSBuild3277** in article and discussion.

Solution:

- In case of C# projects, add the following tag to project properties to suppress the warning:

Add following tag in .csproj

```
<Project ToolsVersion="4.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <PropertyGroup>
     ...
    <NoWarn>MSB3277</NoWarn>
     ...
  </PropertyGroup>
```

- Update System.Runtime.CompilerServices.Unsafe and Microsoft.Bcl.AsyncInterfaces to version requested by Visual Studio (6.0.0).

Blank pages printed between pages, or a red line appears 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 cannot 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: Adjust the PrintWidth in the report designer using either the property grid or by dragging the right edge of the report. Adjust page margins, height, and width 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.

No data appears in a report containing the OleObject control

Symptoms: No data appears in a report containing the OleObject control.

Cause: This issue occurs when the Microsoft .NET Framework 4.6.2 or above is used and the useLegacyV2RuntimeActivationPolicy attribute is not set to True.

Solution: 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="MyRunTimeVersion"/>
</startup>
</configuration>
```

An error message appears in the 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 for a parameter. If the field is a data type other than text, memo, or date/time in Access, the report still runs normally.

Solution: To display the fields in the Fields list in the Report Explorer, supply a default value for the parameter in the Properties Panel, or in the SQL query as below:

SQL Query

```
<%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:

SQL Query

```
<%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:

SQL Query

```
#<%InvoiceDate | Choose invoice date: | 11/2/04 | D | True%>#
```

or

SQL Query

```
"<%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.

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 ActiveReports 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.

An exception relating to System.Data.SqlClient occurs on previewing Section Reports in Windows Forms Viewer or WPF Viewer in the .NET Core 3.1 desktop application

Symptoms: The "Could not load file or assembly 'System.Data.SqlClient" exception occurs when you preview a Section Report in the Windows Forms Viewer or WPF Viewer in the .NET Core 3.1 desktop application. This exception also occurs when you design a Section Report in the Designer in the .NET Core 3.1 desktop application.

Cause: This is a Microsoft compatibility issue.

Solution: You need to manually add the Microsoft.Windows.Compatibility NuGet package. For more information, see this article.

Page/RDLX Report Troubleshooting

An expression containing a numeric field name does not display any data at run time.

Symptoms: An expression containing a numeric field name does not display any data at runtime.

Cause: Visual Basic syntax does not allow an identifier that begins with a number.

`i.e. =Fields!2004.Value`

Solution: Make the numeric field name a string.

`i.e. =Fields("2004").Value` or, `=Fields.Item("2004").Value`

DataSet field in PageHeader of an RDLX report

Symptoms: Cannot set a dataset field (bound field) in the PageHeader of an RDLX report.

Cause: ActiveReports is based on the RDLX 2005 specifications, therefore, referencing datasets in the PageHeader of an RDLX report is not supported.

Solution: There is no direct way to add a DataField in a PageHeader, however, as a workaround you can create a hidden report parameter that is bound to your dataset and has the default value set to your expression. For example, `="*" & First(Fields!name.Value)`. You can then use this parameter in the page header.
Alternatively, you can use a Page Report, which lets you place data fields anywhere on a page.

Exception thrown when using Viewer.Document property

Symptoms: An exception is raised when **Viewer.Document ('Document Property' in the on-line documentation)** is used with a Page Report or RDLX report.

Cause: Document property is available for Section Reports only.

WPF Viewer Troubleshooting

Report is not previewed properly in WPF Viewer when DPI is set to 125% or higher

Symptoms: A report is rendered on top of the WPF Viewer in the continuous mode on scrolling, when DPI is set to 125% or higher.

Cause: This is a WPF continuous mode limitation.

Solution: To resolve this problem, add the following code to the project.

Add to the Viewer.LoadCompleted event

```
Viewer.LoadCompleted += (_, __) =>
{
Viewer.ViewType = GrapeCity.Viewer.Common.Model.ViewType.Continuous;
};
```

Toolbox for WPF Viewer is missing from WPF Project

Symptoms: The Toolbox for WPF Viewer does not appear in a new WPF project in Visual Studio on adding WPF package, MESCIUS.ActiveReports.Viewer.Wpf.

Cause: It is a Visual Studio limitation where XAML Hot Reload does not work correctly.

Solution: The XAML Hot Reload should be enabled. To enable the XAML Hot Reload, follow these steps.

1. Go to **Debug** > **Options** > **General**.
2. Select options **Enable UI Debugging Tools for XAML** and **Enable XAML Hot Reload**.
3. Open MainWindow.xaml. The Toolbar should now display the **WPF Viewer** tab.

See Troubleshooting XAML Hot Reload for more information.

TargetInvocationException occurs when running the WPF browser application

Symptoms: When running the WPF browser application, the TargetInvocationException occurs.

Cause: The WPF browser application does not support Partial Trust.

Solution: Make sure that the WPF browser application uses Full Trust. To do that, in the Visual Studio Project menu, go to **YourProject Properties** and on the **Security** tab, under **EnableClickOnce** security settings, select the option **This is a full trust application**.

Design-time error appears on adding the WPF Viewer to the xaml page if the project is targeting .NET Core 3.1.

Symptoms: A design-time error appears on adding the WPF Viewer to the xaml page if the project is targeting .NET Core 3.1.

Cause: This is the .NET limitation.

Solution: To resolve this problem, add the **Microsoft.Windows.Compatibility** NuGet package to the project and then rebuild the project.

Memory Troubleshooting

> **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.

Symptoms: ActiveReports is consuming too much memory; 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 *in*correct syntax is as follows.

C# code.

```
//Incorrect!
rpt.Dispose();
rpt=null;
```

Visual Basic code.

```
'Incorrect!
rpt.Dispose()
rpt=Nothing
```

Solution: The correct syntax for disposing of a Section Report is as follows.

C# code.

```
//Correct!
rpt.Document.Dispose();
rpt.Dispose();
rpt=null;
```

Visual Basic code.

```
'Correct!
rpt.Document.Dispose()
rpt.Dispose()
rpt=Nothing
```

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 Optimize Section Reports 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.

Web Applications Troubleshooting

Some locale issues occur when generating reports on Linux Docker

Symptoms: When creating or using Docker images, locale-related problems such as missing fonts are discovered.

Cause: Due to missing configuration or incorrectly configured locale for Linux.

Solution: To avoid this problem, the invariant mode needs to be disabled, and the locale settings should be set explicitly in the Dockerfile from which the image is built when creating Docker images.

To do this, specify the following lines explicitly :

Add after the base layer in the Dockerfile

```
# Disable the invariant mode (set in base image)
ENV DOTNET_SYSTEM_GLOBALIZATION_INVARIANT=false\
    # Set the locale
    LC_ALL=en_US.UTF-8 \
    LANG=en_US.UTF-8
```

Chart and Image controls not showing in WebViewer in IIS

Symptoms: The charts and images are broken in the report preview in Web Viewer.

Cause: It is because IIS tries to search the physical files instead of handling the request in the back end.

Solution: To resolve this problem, add the following in web.config file:

Add to web.config file

```
<handlers>
      <add verb="*" path="/api/reporting/*" type="System.Web.Handlers.ScriptModule" name="nostaticfile"
resourceType="Unspecified" preCondition="integratedMode" />
</handlers>
```

This tells the IIS to redirect to our specific handler all requests which starts from "api/reporting". See WebViewer_ASP.NET_C# sample.

Error appears on adding MESCIUS.ActiveReports.Web.Design.VS2022 package in Visual Studio 2022

Symptoms: On installing MESCIUS.ActiveReports.Web.Design.VS2022 package, an error "Could not install this package. You are trying to install this package into a project that targets '.NETFramework=v4.7.1', but the package does not contain any assembly references or content files that are compatible with that framework."

Cause: In Visual Studio 2022, MESCIUS.ActiveReports.Web.Design.VS2022 package is not supported for .NET applications targeting .NET Framework 4.7.1 or below.

Solution: Change the target framework to .NET Framework 4.7.2, .NET Framework 4.8, or .NET Framework 4.8.1, and then try installing the package again. The package should be installed without any errors.

PDF opens in a new window when an application contains the WebViewer

Symptoms: When using Internet Explorer and Acrobat Reader to view a page containing a WebViewer in PDF mode, the resulting PDF always opens in a new window.

Cause: Acrobat Reader is only available in a 32-bit version. When the 64-bit version of Internet Explorer is used, it opens up an instance of the 32-bit version of Internet Explorer so that the plug-in and the PDF can load, rendering the resulting PDF in a new window.

Solution:

- Install a PDF reader plug-in that is 64-bit compatible.
  OR
- Use the 32-bit version of Internet Explorer.

PlatformNotSupportedException occurs on using WebViewer, JSViewer, and WebDesigner

Symptoms: PlatformNotSupportedException occurs in Web Applications using WebViewer, JSViewer, and WebDesigner in Classic Pipeline Mode.
Cause: The WebViewer, JSViewer, and WebDesigner are supported only in the Integrated pipeline mode.
Solution: Change the Application Pool mode as follows:

1. Open **IIS Manager**.
2. Go to **Application Pools**.
3. Select the application pool where your app runs.
4. Select **Basic Settings.**
5. In the **Edit Application Pool** dialog, change the **Managed Pipeline Mode** to **Integrated**.

The "Report not found" error occurs when a report name contains special symbols

Symptoms: When you specify a report name with special symbols, e.g. webViewer.ReportName="Folder\Report.rdlx", you may get a "Report not found" error.

Cause: Additional code needs to be added to the Web.config file to have the WebViewer use report names with the corresponding folders.

Solution: Add the following code to the Web.config file.

Paste inside the Web.config file

```
<system.web>
<httpRuntime requestPathInvalidCharacters="" requestValidationMode="2.0"/>
<pages validateRequest="false"/>
</system.web>

<system.webServer>
<security>
<requestFiltering allowDoubleEscaping="true"/>
</security>
</system.webServer>
```

The 'The type or namespace name 'Linq' does not exist in the namespace 'System'' error occurs when adding the WebViewer control in an ASP.NET Web Site project

Symptoms: When you add the WebViewer control in an ASP.NET Web Site project, the 'The type or namespace name 'Linq' does not exist in the namespace 'System' error occurs.

Cause: This is a known NuGet limitation.

Solution: You should install or upgrade the Microsoft.CodeDom.Providers.DotNetCompilerPlatform NuGet package.

"This application will be terminated because it was built without a license for PageReport" error occurs on deploying an application on Azure Functions Application.

Symptoms: When you deploy an application in an Azure Functions application, error "This application will be terminated because it was built without a license for PageReport" occurs.

Cause: This is because the application being deployed is not licensed properly.

Solution: You should follow the steps provided in the Licensing Compiled Code topic for correctly licensing the application before deploying it to the Azure Functions application.

The export file extension appears incorrect in Chrome while exporting using JSViewer in a reporting service hosted in an external application.

Symptoms: When using JSViewer in a reporting service hosted in an external application to export a report, the export file extension for download appears as .rdlx (in Chrome) instead of the corresponding format extension.

Cause: This issue occurs because of the restriction to access response headers when using Fetch API (which we use to download the exported file) over CORS.

Solution: You should add the following line in your CORS policy:

```
HttpContext.Current.Response.AddHeader("Access-Control-Expose-Headers", "Content-Disposition");
```