

Table of Contents

[Planning a Report](#)

[Report Design Tips](#)

[Report Authoring Concepts](#)

[Reports, Report Parts, and Report Definitions](#)

[Data Regions and Maps](#)

[Report Parameters Concepts](#)

[Tables, Matrices, and Lists](#)

[Tables](#)

[Create a Matrix](#)

[Create Invoices and Forms with Lists](#)

[Tablix Data Region](#)

[Tablix Data Region Areas](#)

[Tablix Data Region Cells, Rows, and Columns](#)

[Preparing Data for Display in a Tablix Data Region](#)

[Adding Data to a Tablix Data Region](#)

[Exploring the Flexibility of a Tablix Data Region](#)

[Controlling the Tablix Data Region Display on a Report Page](#)

[Controlling Row and Column Headings](#)

[Understanding Groups](#)

[Creating Recursive Hierarchy Groups](#)

[Add a Details Group](#)

[Add a Total to a Group or Tablix Data Region](#)

[Change an Item Within a Cell](#)

[Change Row Height or Column Width](#)

[Insert or Delete a Column](#)

[Insert or Delete a Row](#)

[Merge Cells in a Data Region](#)

[Create a Recursive Hierarchy Group](#)

[Add or Delete a Group in a Data Region](#)

[Display Headers and Footers with a Group](#)

[Create a Stepped Report](#)

[Add, Move, or Delete a Table, Matrix, or List](#)

[Report Parts](#)

[Publish and Republish Report Parts \(Report Builder and SSRS\)](#)

[Browse for Report Parts and Set a Default Folder \(Report Builder and SSRS\)](#)

[Charts](#)

[Add a Chart to a Report \(Report Builder and SSRS\)](#)

[Chart Types \(Report Builder and SSRS\)](#)

[Change a Chart Type \(Report Builder and SSRS\)](#)

[Area Charts \(Report Builder and SSRS\)](#)

[Bar Charts \(Report Builder and SSRS\)](#)

[Column Charts \(Report Builder and SSRS\)](#)

[Line Charts \(Report Builder and SSRS\)](#)

[Pie Charts \(Report Builder and SSRS\)](#)

[Polar Charts \(Report Builder and SSRS\)](#)

[Range Charts \(Report Builder and SSRS\)](#)

[Scatter Charts \(Report Builder and SSRS\)](#)

[Shape Charts \(Report Builder and SSRS\)](#)

[Stock Charts \(Report Builder and SSRS\)](#)

[Tree Map and Sunburst Charts in Reporting Services](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Chart Legend - Formatting](#)

[Chart Legend - Hide Items](#)

[Chart Legend - Change Item Text](#)

[Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)

[Chart Effects - 3D, Bevel, and Other](#)

[Chart Effects - Add 3D Effects](#)

[Chart Effects - Add Bevel, Emboss, or Texture](#)

[Add a Border Frame to a Chart \(Report Builder and SSRS\)](#)

[Start Pie Chart Values at the Top of the Pie \(Report Builder and SSRS\)](#)

[Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

[Add Empty Points to a Chart \(Report Builder and SSRS\)](#)

[Displaying a Series with Multiple Data Ranges on a Chart \(Report Builder and SSRS\)](#)

[Add Scale Breaks to a Chart \(Report Builder and SSRS\)](#)

[Multiple Series on a Chart \(Report Builder and SSRS\)](#)

[Specify a Chart Area for a Series \(Report Builder and SSRS\)](#)

[Plot Data on a Secondary Axis \(Report Builder and SSRS\)](#)

[Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

[Display the Same Data on a Matrix and a Chart](#)

[Add or Delete a Group in a Chart \(Report Builder and SSRS\)](#)

[Add a Moving Average to a Chart \(Report Builder and SSRS\)](#)

[Collect Small Slices on a Pie Chart \(Report Builder and SSRS\)](#)

[Troubleshoot Charts \(Report Builder and SSRS\)](#)

[Sparklines and Data Bars](#)

[Add Sparklines and Data Bars \(Report Builder and SSRS\)](#)

[Align the Data in a Chart in a Table or Matrix \(Report Builder and SSRS\)](#)

[Gauges](#)

[Formatting Scales on a Gauge](#)

[Formatting Pointers on a Gauge](#)

[Formatting Ranges on a Gauge](#)

[Add a Gauge to a Report](#)

[Set a Minimum or Maximum on a Gauge](#)

[Indicators](#)

[Add or Delete an Indicator](#)

[Change Indicator Icons and Indicator Sets](#)

[Set and Configure Measurement Units](#)

[Set Synchronization Scope](#)

[Specify the Size of an Indicator Using an Expression](#)

[Include Indicators and Gauges in a Gauge Pane](#)

[Maps](#)

[Plan a Map Report](#)

- [Map Wizard and Map Layer Wizard](#)
 - [Customize the Data and Display of a Map or Map Layer](#)
 - [Vary Polygon, Line, and Point Display by Rules and Analytical Data](#)
 - [Add, Change, or Delete a Map or Map Layer](#)
 - [Change Map Legends, Color Scale, and Associated Rules](#)
 - [Add Custom Locations to a Map](#)
 - [Troubleshoot Reports: Map Reports](#)
- [Images, Text Boxes, Rectangles, and Lines](#)
 - [Text Boxes \(Report Builder and SSRS\)](#)
 - [Add, Move, or Delete a Text Box \(Report Builder and SSRS\)](#)
 - [Set Text Box Orientation \(Report Builder and SSRS\)](#)
 - [Allow a Text Box to Grow or Shrink \(Report Builder and SSRS\)](#)
 - [Rectangles and Lines \(Report Builder and SSRS\)](#)
 - [Add a Rectangle or Container \(Report Builder and SSRS\)](#)
 - [Add and Modify a Line \(Report Builder and SSRS\)](#)
 - [Add a Border to a Report \(Report Builder and SSRS\)](#)
 - [Images \(Report Builder and SSRS\)](#)
 - [Add an External Image \(Report Builder and SSRS\)](#)
 - [Embed an Image in a Report \(Report Builder and SSRS\)](#)
 - [Add a Background Image \(Report Builder and SSRS\)](#)
 - [Add a Data-Bound Image \(Report Builder and SSRS\)](#)
- [Formatting Report Items](#)
 - [Formatting Text and Placeholders \(Report Builder and SSRS\)](#)
 - [Format Text in a Text Box \(Report Builder and SSRS\)](#)
 - [Importing HTML into a Report \(Report Builder and SSRS\)](#)
 - [Add HTML into a Report \(Report Builder and SSRS\)](#)
 - [Formatting Numbers and Dates \(Report Builder and SSRS\)](#)
 - [Formatting Lines, Colors, and Images \(Report Builder and SSRS\)](#)
 - [Set the Locale for a Report or Text Box](#)
- [Report Parameters](#)
 - [Add, Change, or Delete a Report Parameter](#)
 - [Add, Change, or Delete Available Values for a Report Parameter](#)

[Add, Change, or Delete Default Values for a Report Parameter](#)

[Change the Order of a Report Parameter](#)

[Add Cascading Parameters to a Report](#)

[Add a multi-value parameter to a Report](#)

[Set Parameters on a Published Report](#)

[Customize the Parameters Pane in a Report](#)

[Filter, Group, and Sort Data](#)

[Group Expression Examples](#)

[Filter Equation Examples](#)

[Add Dataset Filters, Data Region Filters, and Group Filters](#)

[Add a Filter](#)

[Sort Data in a Data Region](#)

[Commonly Used Filters](#)

[Grouping Pane](#)

[Drillthrough, Drilldown, Subreports, and Nested Data Regions](#)

[Drillthrough Reports \(Report Builder and SSRS\)](#)

[Add a Drillthrough Action on a Report \(Report Builder and SSRS\)](#)

[Subreports \(Report Builder and SSRS\)](#)

[Add a Subreport and Parameters \(Report Builder and SSRS\)](#)

[Drilldown Action \(Report Builder and SSRS\)](#)

[Add an Expand or Collapse Action to an Item \(Report Builder and SSRS\)](#)

[Nested Data Regions \(Report Builder and SSRS\)](#)

[Specifying Paths to External Items \(Report Builder and SSRS\)](#)

[Expressions](#)

[Add an Expression](#)

[Expression Examples](#)

[Expression Uses in Reports](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections](#)

[Custom Code and Assembly References in Expressions in Report Designer](#)

[Add an Assembly Reference to a Report](#)

[Add Code to a Report](#)

[Expression Reference](#)

[Report Builder Functions - Aggregate Functions Reference](#)

[Data Types in Expressions](#)

[Constants in Expressions](#)

[Operators in Expressions](#)

[Built-in Collections in Expressions](#)

[Formulas in Report Model Queries](#)

[Interactive Sort, Document Maps, and Links](#)

[Interactive Sort \(Report Builder and SSRS\)](#)

[Add Interactive Sort to a Table or Matrix \(Report Builder and SSRS\)](#)

[Add a Hyperlink to a URL \(Report Builder and SSRS\)](#)

[Add a Bookmark to a Report \(Report Builder and SSRS\)](#)

[Create a Document Map \(Report Builder and SSRS\)](#)

[Page Layout and Rendering](#)

[Rendering Behaviors](#)

[Rendering Data Regions](#)

[Rendering Report Items](#)

[Rendering Data](#)

[Pagination in Reporting Services](#)

[Page Headers and Footers](#)

[Controlling Page Breaks, Headings, Columns, and Rows](#)

[Add a Page Break](#)

[Display Row and Column Headers on Multiple Pages](#)

[Add or Remove a Page Header or Footer](#)

[Keep Headers Visible When Scrolling Through a Report](#)

[Display Page Numbers or Other Report Properties](#)

[Hide a Page Header or Footer on the First or Last Page](#)

[Report Parts in Report Designer](#)

[Managing Report Parts](#)

Planning a Report (Report Builder)

3/24/2017 • 3 min to read • [Edit Online](#)

Report Builder lets you create many kinds of paginated reports. For example, you can create reports that show summary or detailed sales data, marketing and sales trends, operational reports, or dashboards. You can also create reports that take advantage of richly formatted text, such as for sales orders, product catalogs, or form letters. All these reports are created by using different combinations of the same basic building blocks in Report Builder. To create a useful, easily understood report, it helps to plan first. Here are some things you might want to consider before you get started:

- **What format do you want the report to appear in?**

You can render reports online in a browser such as the Reporting Services web portal or export them to other formats such as Excel, Word, or PDF. The final form your report takes is an important consideration because not all features are available in all export formats. For more information, see [Export Reports \(Report Builder and SSRS\)](#).

- **What structure do you want to use to present the data in the report?**

You have a choice among tabular, matrix (similar to a cross-tab or PivotTable report), chart, free-form structures, or any combination of these to present data. For more information, see [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#) and [Charts \(Report Builder and SSRS\)](#).

- **What do you want your report to look like?**

Report Builder provides a lot of report items that you can add to your report to make it easier to read, highlight key information, help your audience navigate the report, and so on. Knowing how you want the report to appear can determine whether you need report items such as text boxes, rectangles, images, and lines. You might also want to show or hide items, add a document map, include drillthrough reports or subreports, or link to other reports. For more information, see [Images, Text Boxes, Rectangles, and Lines \(Report Builder and SSRS\)](#) and [Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#).

- **What data do you want your readers to see? Should the data or format be filtered for different audiences?**

You might want to narrow the scope of the report to specific users or locations, or to a particular time period. To filter the report data, use parameters to retrieve and display only the data you want. For more information, see [Report Parameters \(Report Builder and Report Designer\)](#).

- **Do you need to create your own calculations?**

Sometimes, your data source and datasets do not contain the exact fields that you need for your report. In that situation, you might have to create your own calculated fields. For example, you might want to multiply the price per unit times the quantity to get a line item sales amount. Expressions are also used to provide conditional formatting and other advanced features. For more information, see [Expressions \(Report Builder and SSRS\)](#).

- **Do you want to hide report items initially?**

Consider whether you want to hide report items, including data regions, groups and columns, when the report is first run. For example, you can initially present a summary table, and then drill down into the detailed data. For more information, see [Drilldown Action \(Report Builder and SSRS\)](#).

- **How are you going to deliver your report?**

You can save your report to your local computer and continue to work on it, or run it locally for your own information. However, to share your report with others, you need to save the report to a report server that is configured in native mode or a report server in SharePoint integrated mode. Saving it to a server lets others run it whenever they want to. Alternatively, the report server administrator can set up a subscription to the report or set up e-mail delivery of the report to other individuals. You can have the report delivered in a specific export format if you prefer. For more information, see [Finding, Viewing, and Managing Reports \(Report Builder and SSRS\)](#).

See Also

[Report Builder in SQL Server 2016](#)

[Report Authoring Concepts \(Report Builder and SSRS\)](#)

[Report Builder Tutorials](#)

Report Design Tips (Report Builder and SSRS)

3/24/2017 • 11 min to read • [Edit Online](#)

Use the following tips to help design your Reporting Services paginated reports.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Designing Reports

- A well-designed report conveys information that leads to action. Identify the questions that the report helps to answer. Keep those questions in mind as you design the report.
- To design effective data visualizations, think about how to display information that is easy for the report user to understand. Choose a data region that is a good match for the data that you want to visualize. For example, a chart effectively conveys summary and aggregated information better than a table that spans many pages of detailed information. You can visualize data from a dataset in any data region, which includes charts, maps, indicators, sparklines, databars, and tabular data in various grid layouts based on a tablix.
- If you plan to deliver the report in a specific export format, test the export format early in your design. Feature support varies based on the renderer that you choose.
- If you plan to deliver the report as a subscription, test the subscription early in your design. Parameter support varies based on the subscription that you create.
- When you build complex layouts, build the layout in stages. You can use rectangles as containers to organize report items. You can build data regions directly on the design surface to maximize your working area, and then, as you complete each one, drag it to the rectangle container. By using rectangles as containers, you can position all its contents in one step. Rectangles also help control the way report items render on each page.
- To reduce clutter in a report, consider using conditional visibility for specific report items and let the user choose whether to show the items. You can set visibility based on a parameter or a text box toggle. You can add conditionally hidden text boxes to show interim expression results. When a report displays unexpected data, you can show these interim results to help debug expressions.
- When you work with nested items in tablix cells or rectangles, you can set different background colors for the container and contained items. By default, the background color is **No color**. Items with a specific background color show through items with a background color set to **No color**. This technique can help you select the right item to set display properties, such as border visibility on tablix cells.

For more information about things to consider as you design your report, see [Planning a Report \(Report Builder\)](#).

Naming Conventions for Reports, Data Sources, and Datasets

- Use naming conventions for data sources and datasets that document the source of data.
 1. **Data sources.** If you do not want to use an actual server or database due to security reasons, use an alias that indicates to the user what the source of data is.

2. **Datasets.** Use a name that indicates which data source it is based on.
 3. **Data regions.** Indicate the type of data region and what data it displays. Data region names are useful in the following scenarios:
 - a. **Data region as a report part.** When report authors browse the Report Part Gallery, a descriptive name helps them find the report parts they are looking for.
 - b. **Data region as a data feed.** With appropriate permissions, a report reader can create an ATOM data feed from a data region.
- Use underscores instead of spaces in report names. If you download a report from a Reporting Services web portal, spaces are replaced by underscores. If you use the download feature to save reports locally, and then include them in SQL Server Data Tools (SSDT), using underscores helps to keep report dependencies for subreports and drillthrough links accurate.

Working with Data

- As a first step, get all the data that you want to work with to appear in the Report Data pane. As you refine the questions that the report is designed to answer, think about how to limit the data in the report datasets to just what is needed.
- In general, only include the data that you will display in a report. Use query variables in your dataset queries to enable the user to choose which data they want to see in the report. If you are creating shared datasets, provide filters based on report parameters to provide the same functionality.
- If you are an experienced query writer, understand that for intermediate amounts of data, you might want to group data in the report, and not in the query. If you do all your grouping in the query, then the report tends to be a presentation of the query result set. On the other hand, to display aggregated values for large amounts of data on a chart or matrix, there is no need to include detail data.
- Depending on your requirements, you can display names and locations of report data sources, dataset query command text, and parameter values in the report. The first question many new users have is about where the data comes from. To reduce clutter in the report, you can conditionally hide text boxes with this type of information and let users choose whether to see it. Try adding this information on the last page of report. Set the text box visibility based on a parameter that the user can change.

Interacting with the Report Design Surface

The report design surface is not WYSIWIG. When you place report items on the design surface, their relative location affects the way that the items appear on the rendered report page. White space is preserved.

- Use snaplines and layout buttons to align and arrange items on the report design surface. For example, you can align the tops or edges of selected items, expand an item to match the size of another item, or adjust the spacing between items.
- Use arrow keys to adjust the position and size of selected items on the design surface. For example, the following key combinations are very useful:
 - **Arrow keys** Move the selected report item.
 - **CTRL+Arrow keys** Nudge the selected report item.
 - **CTRL+SHIFT+Arrow keys** Increase or decrease the size of the selected report item.
- To add an item to a rectangle, use the upper left tip of the mouse to point to the initial location of the item in the rectangle container. Use keyboard shortcuts to help position selected objects. The rectangle automatically expands to accommodate the size of the contained items.

- To add multiple items to a tablix cell, first add a rectangle, and then add the items.

By default, each tablix cell contains a text box. When you add a rectangle to a cell, the rectangle replaces the text box. For example, place nested indicators in a rectangle in a tablix cell to help control how the size of a chart or indicator expands as you change the height of the row that the cell is in.

- Use the **Zoom** control to adjust your view of the design surface. You can work with the whole page or smaller sections of the page.
- To drag fields from the Report Data pane to the Grouping pane, avoid dragging the field across other report items on the design surface because this selects the other items and unselects the tablix data region. Drag the field down the Report Data pane and then across to the Grouping pane.

Selecting Items

To help select the object that you want on the report design surface, use the ESC key, the right-click context menu, the Properties pane, and the Grouping pane.

- - Press ESC to cycle through the stack of report items that occupy the same space on the design surface.
 - On some report items, try using the right-click context menu to select the report item or the part of the report item that you want.
 - The Properties pane displays properties for the current selection.
 - To work with row groups and column groups in a tablix data region, select the group from the Grouping pane.

In Report Designer in SQL Server Data Tools, you can select from the drop-down list of objects in the Properties pane toolbar or from the hierarchical view of report items in the Document Outline window. You can select items in this pane and see which item is selected on the design surface. To open the Document Outline window, from the **View** menu, point to **Other Windows**, and then click **Document Outline**.

Working with Specific Types of Report Items

Working with Parameters

- The primary purpose of report parameters is to filter data at the data source, and retrieve just what is needed for the purpose of the report.
- For report parameters, find a balance between enabling interactivity and helping a user get the results they want. For example, you can set default values for a parameter to values that you know are popular.

Working with Text

- When you paste multiline into a text box, the text is added as one text run. Each text run can only be formatted as a unit. To format each line independently, insert a new line by pressing RETURN in the text run as needed. You can then apply formatting and styles to each independent line of text in the text box.
- You can set format properties and actions on a text box or on placeholder text in the text box. If there is only one line of text, it is more efficient to set properties on the text box than on the text.

Working with Expressions

- Understand simple and complex expression formats. You can type simple expression format directly into text boxes, properties in the Property pane, or in locations in dialog boxes that accept an expression. For more information, see [Expressions \(Report Builder and SSRS\)](#).
- When you create an expression, it helps to create each part independently and verify its value. You can then combine all the parts into a final expression. A useful technique is to add a text box in a matrix cell, display

each part of the expression, and set conditional visibility on the text box. To control the border style and color when the text box is hidden, first place the text box in a rectangle, and then set the border style and color of the rectangle to match the matrix.

Working with Indicators

- By default, an indicator shows at least three states. After you add an indicator to a report, you can configure it by adding or removing states. For easier viewing by your users, choose an indicator that varies by both color and shape.

Controlling the Rendering of Report Items on the Report Page

- On the report design surface, report items grow to accommodate the contents from the associated dataset, expression, subreport, or text.
 - When you position an item on the report page, the distance between the item and all items that begin to the right of it becomes the minimum distance that must be maintained as a report item grows horizontally. Similarly, the distance between an item and the item above it becomes a minimum distance that must be maintained as the top item grows vertically.
 - An item in a report grows to accommodate its data and pushes peer items (items within the same parent container) out of the way using the following rules:
 - Each item moves down to maintain the minimum space between itself and the items that end above it.
 - Each item moves right to maintain the minimum space between itself and the items that end to the left of it. For systems that right-to-left layouts, each item moves left to maintain the minimum space between itself and the items that end to the right of it.
 - Containers expand to accommodate the growth of child items. For a selected item, in the Properties pane, the Parent property identifies the container for the item. You can also use the Document Outline pane to see the containment hierarchy of report items.
 - The **Layout** toolbar provides multiple buttons to help align edges, centers, and spacing for report items. To enable the **Layout** toolbar, from the **View** menu, point to **Toolbars**, and then click **Layout**.
- If you plan to save the report as a .pdf file, the report width must be explicitly set to a value that gives you the results that you want in the export file format. For example, set the report page width to exactly 7.9375 inches and the left and right margins to .5 inches.
- Use **Print Layout** and **Page Setup** on the report viewer toolbar to render a report in a print-compatible view. To help remove unwanted horizontal pages, do the following:
 1. Remove all extra white space between data regions and on the edges of the report.
 2. Reduce page margins in the **Report Properties** dialog box.
 3. Use **Rectangles** as containers to help control the way report items render.
 4. In column headers, change the text box property **WritingMode** to use vertical text.

The combination of this behavior, the width and height properties of report items, the size of the report body, the page height and page width definition, the margin settings of the parent report, and the renderer-specific support for paging all combine to determine what report items fit together on a rendered page. For more information, see [Pagination in Reporting Services \(Report Builder and SSRS\)](#).

See Also

[Report Builder in SQL Server 2016](#)

[Reporting Services Tutorials \(SSRS\)](#)

[Report Builder Tutorials](#)

Report Authoring Concepts (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

This section briefly defines some key concepts used to describe paginated reports in the Report Builder and Report Designer documentation. For definitions of specific words or terms, see the [Glossary \(Report Builder\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

In This Section

[Reports, Report Parts, and Report Definitions \(Report Builder and SSRS\)](#)

Describes the variety of terms used to describe a report in different states, including the initial definition, the published report, and the viewed report as it appears to the user.

[Embedded and Shared Datasets \(Report Builder and SSRS\)](#)

Explains the differences in creating, storing, and managing embedded and shared datasets.

[Data Regions and Maps \(Report Builder and SSRS\)](#)

Describes the types of data regions that can be added to a report layout. Data regions determine the appearance of a report: tabular, matrix, list, or chart.

[Report Parameters Concept \(Report Builder and SSRS\)](#)

Describes the ways to define and use report parameters, and how they are independently managed from the report definition on the report server.

See Also

[Report Builder in SQL Server 2016](#)

Reports, Report Parts, and Report Definitions (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

Reporting Services uses a variety of terms to describe a paginated report in different states, including the initial definition, the published report, and the viewed report as it appears to the user.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Report Definition (.rdl) Files

A report definition is a file that you create in Report Builder or Report Designer. It provides a complete description of data source connections, queries used to retrieve data, expressions, parameters, images, text boxes, tables, and any other design-time elements that you might include in a report. Although report definitions can be complex, at a minimum they specify a query and other report content, report properties, and a report layout.

Report definitions are rendered at run time as a processed report. At that time, the data is retrieved from the data source and formatted according to the instructions in the report definition. A report definition can be run directly from your computer and saved locally, or it can be published to a report server for others to run as well.

Report definitions are written in XML that conforms to an XML grammar called Report Definition Language (RDL). RDL describes the XML elements, encompassing all possible variations that a report can assume.

Client Report Definition (.rdlc) Files

The Visual Studio Report Designer produces client report definition (.rdlc) files for use with the ReportViewer control. The .rdlc files can be converted to .rdl files for use with Reporting Services Report Designer.

Report Part (.rsc) Files

Report parts are self-contained report items that are stored on the report server and can be included in other reports. Use Report Builder to browse and select parts from the Report Part Gallery to add to your reports. Use Report Designer or Report Builder to save report parts for use in the Report Part Gallery.

A report part definition is an XML fragment of a report definition file. You create report parts by creating a report definition, and then selecting report items in the report to publish separately as report parts. Report parts include data regions, rectangles and their contained items, and images. You can save a report part with its dependent datasets and shared data source references so it can be reused in other reports.

For more information, see [Report Parts \(Report Builder and SSRS\)](#) and [Report Parts in Report Designer \(SSRS\)](#).

Published Reports

After you create an .rdl file, you can save it locally, or save it to a personal folder (such as the My Reports folder) on the report server. When the report is ready for others to see, you publish it by saving it from Report Builder to a public folder on the report server, uploading it through the Reporting Services web portal, or deploying a report project solution from Report Designer. A published report is an item that is stored in a report server database and

managed on a report server or SharePoint site.

A published report is secured through role assignments using the Reporting Services role-based security model. Published reports are accessed through URLs, SharePoint Web parts, or the Reporting Services web portal, or you can navigate to and open them in Report Builder.

Report Snapshots

A report can also be published as a snapshot that contains both layout information and data as of the time the report was initially run. Report snapshots are not saved in a particular rendering format. Instead, report snapshots are rendered in a final viewing format (such as HTML) only when a user or an application requests it. For more information, see [Finding and Viewing Reports in Report Manager \(Report Builder and SSRS\)](#).

Rendered Reports

A rendered report is a fully processed report that contains both data and layout information in a format suitable for viewing (such as HTML). Until a report is rendered into an output format, it cannot be viewed. You can render a report by doing one of the following:

- Create or open a report in Report Builder or Report Designer and run it.
- Find and run a report in the Reporting Services web portal.
- Find and run a report on a SharePoint site that is integrated with a Reporting Services report server.
- Subscribe to a report, which is delivered to an e-mail inbox or a file share in an output format that you specify.

Subscribe to a report, which is delivered to an e-mail inbox or a file share in an output format that you specify. The default rendering format for a report is HTML 4.0. In addition to HTML, reports can be rendered in a variety of output formats, including Excel, Word, XML, PDF, TIFF, and CSV. As with published reports, rendered reports cannot be edited or saved back to a report server. For more information, see [Export Reports \(Report Builder and SSRS\)](#).

See Also

[Report Authoring Concepts \(Report Builder and SSRS\)](#)

[Report Builder in SQL Server 2016](#)

[Finding, Viewing, and Managing Reports \(Report Builder and SSRS \)](#)

[Export Reports \(Report Builder and SSRS\)](#)

Data Regions and Maps (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

A data region is an object in a report that displays data from a report dataset. Report data can be displayed as numbers and text in a table, matrix, or list; graphically in a chart or gauge; and against a geographic background in a map. Tables, matrices, and lists are all based on the *tablix* data region, which expands as needed to display all the data from the dataset. A tablix data region supports multiple row and column groups and both static and dynamic rows and columns. A chart displays multiple series and category groups in a variety of chart formats. A gauge displays a single value or an aggregated value for a dataset. A map displays spatial data as map elements that can vary in appearance based on aggregated data from a dataset.

You can save a data region or map as a *report part*. Read more about [Report Parts](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Table

A table is a data region that presents data row by row. Table columns are static: you determine the number of columns when you design your report. Table rows are dynamic: they expand downwards to accommodate the data. You can add groups to tables, which organize data by selected fields or expressions. For information about adding a table to a report, see [Tables \(Report Builder and SSRS\)](#).

Matrix

A matrix is also known as a crosstab. A matrix data region contains both dynamic columns and rows: they expand to accommodate the data. A matrix can have dynamic columns and rows and static columns and rows. Columns or rows can contain other columns or rows, and can be used to group data. Read more about [adding a matrix to a report](#).

List

A list is a data region that presents data arranged in a freeform fashion. You can arrange report items to create a form with text boxes, images, and other data regions placed anywhere within the list. Read more about [adding a list to a report](#).

Chart

A chart presents data graphically. Examples of charts include bar, pie, and line charts, but many more styles are supported. Read more about [adding a chart to a report](#).

Gauge

A gauge presents data as a range with an indicator pointing to a specific value within the range. Gauges are used to display key performance indicators (KPIs) and other metrics. Examples of gauges include linear and circular. Read more about [adding a gauge to a report](#).

Map

A map enables you to present data against a geographical background. Map data can be spatial data from a SQL Server query, an ESRI shapefile, or Microsoft Bing map tiles. Spatial data consists of sets of coordinates that define polygons that represent shapes or areas, lines that represent routes or paths, and points represented by markers. You can associate aggregate data with map elements to automatically vary their color and size. For example, you can vary the marker type for a store based on sales amount or the color for a road based on speed limit. For more information, see [Maps \(Report Builder and SSRS\)](#).

Data Regions in the Report Layout

You can add multiple data regions to a report. Data regions grow to accommodate the data from the report dataset that they are linked to. For example, a matrix that displays sales for each product by year has a row group based on product names and a column group based on years. When you run the report, the matrix expands down the page for each product and across the page for each year. A chart that is placed next to the matrix on the report design surface displays next to the expanded matrix in the rendered report. The way data regions render on a page follows a set of rules based on the output format of a report. For example, to help control how a chart and matrix render on a page, you might use a rectangle as a container or nest both data regions in a list. For more information, see [Page Layout and Rendering \(Report Builder and SSRS\)](#).

Nested Data Regions

You can nest data regions within other data regions. For example, if you want to create a sales record for each sales person in a database, you can create a list with text boxes and an image to display information about the employee, and then add table and chart data regions to the list to show the employee's sales record. For more information, see [Nested Data Regions \(Report Builder and SSRS\)](#).

Multiple Data Regions Linked to the Same Dataset

You can link more than one data region to the same dataset to provide different views of the same data. For example, you can show the same data in a table and in a chart. You can author the report to provide interactive sort buttons on the table, so that when you sort the table, the chart is also automatically sorted. For more information, see [Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#).

Data for a Data Region

Each tablix, chart, and gauge is designed to display data from a single dataset. A map displays spatial data and analytical data from the same or different datasets. You can also include values from datasets that are not linked to the data region in the following ways:

- Aggregate values that do not depend on sort order or grouping that are scoped to a different dataset.
- Lookup values from name/value pairs in a different dataset.

For more information, see [Expressions \(Report Builder and SSRS\)](#).

See Also

- [Report Authoring Concepts \(Report Builder and SSRS\)](#)
- [Reports, Report Parts, and Report Definitions \(Report Builder and SSRS\)](#)
- [Page Layout and Rendering \(Report Builder and SSRS\)](#)
- [Report Builder Tutorials](#)
- [Reporting Services Tutorials \(SSRS\)](#)

Report Parameters Concepts (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

You can add parameters to a report to link related reports, to control the report appearance, to filter report data, or to narrow the scope of a report to specific users or locations.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Report parameters are created in the following ways:

- Automatically, when you define dataset query that contains query variables. For each query variable, a corresponding dataset query parameter and report parameter with the same names are created. A query parameter can be a reference to a query variable or to an input parameter for a stored procedure.
- Automatically, when you add a reference to a shared dataset that contains query parameters.
- Manually, when you create report parameters in the Report Data pane. Parameters are one of the built-in collections that you can include in an expression in a report. Because expressions are used to define values throughout a report definition, you can use parameters to control report appearance or to pass values to related subreports or reports that also use parameters.

For more information, see [Report Parameters \(Report Builder and Report Designer\)](#).

Parameters are frequently used to filter report data both before and after the data is returned to the report. For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).

When you design a report, report parameters are saved in the report definition. When you publish a report, report parameters are saved and managed separately from the report definition. After you save the report to the report server, you can do the following:

- Change report parameter values directly on the report server independently from the report definition.
- Create multiple linked reports in which each linked report is a link to the report definition with a separate set of parameter values that can be managed independently on the report server.

If you plan to create report snapshots, histories, or subscriptions to a published report, you must understand how report parameters affect the design requirements for the report.

See Also

[Report Authoring Concepts \(Report Builder and SSRS\)](#)

[Report Embedded Datasets and Shared Datasets \(Report Builder and SSRS\)](#)

[Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)

Tables, Matrices, and Lists (Report Builder and SSRS)

3/24/2017 • 9 min to read • [Edit Online](#)

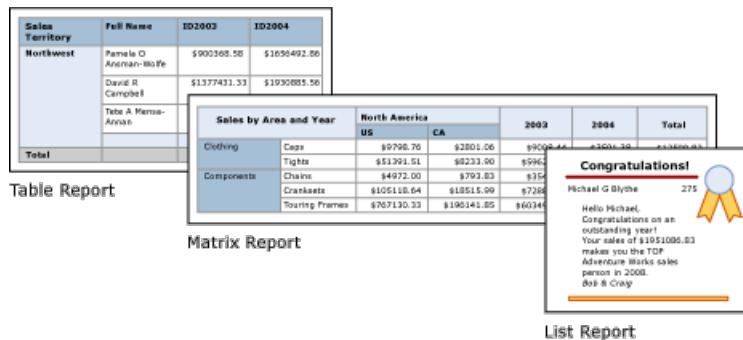
In Reporting Services, tables, matrices, and lists are *data regions* that display paginated report data in cells that are organized into rows and columns. The cells typically contain text data such as text, dates, and numbers but they can also contain gauges, charts, or report items such as images. Collectively, tables, matrices, and lists are frequently referred to as *tablix* data regions.

The table, matrix, and list templates are built on the tablix data region, which is a flexible grid that can display data in cells. In the table and matrix templates, cells are organized into rows and columns. Because templates are variations of the underlying generic tablix data region, you can display data in combination of template formats and change the table, matrix, or list on to include the features of another data region as you develop your report. For example, if you add a table and find it does not serve your needs, you can add column groups to make the table a matrix.

The table and matrix data regions can display complex data relationships by including nested tables, matrices, lists, charts and gauges. Tables and matrices have a tabular layout and their data comes from a single dataset, built on a single data source. The key difference between tables and matrices is that tables can include only row groups, whereas matrices have row groups and column groups.

Lists are a little different. They support a free-layout that and can include multiple peer tables or matrices, each using data from a different dataset. Lists can also be used for forms, such as invoices.

The following pictures show simple reports with a table, matrix, or list.



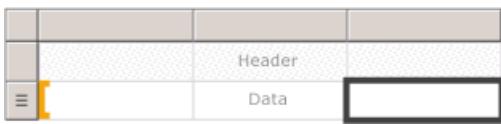
To quickly get started with tables, matrices, and lists, see [Tutorial: Creating a Basic Table Report \(Report Builder\)](#), [Tutorial: Creating a Matrix Report \(Report Builder\)](#), and [Tutorial: Creating a Free Form Report \(Report Builder\)](#).

NOTE

You can publish tables, matrices, and lists separately from a report as report parts. Read more about [Report Parts](#).

Table

Use a table to display detail data, organize the data in row groups, or both. The Table template contains three columns with a table header row and a details row for data. The following figure shows the initial table template, selected on the design surface:



You can group data by a single field, by multiple fields, or by writing your own expression. You can create nested groups or independent, adjacent groups and display aggregated values for grouped data, or add totals to groups. For example, if your table has a row group called [Category], you can add a subtotal for each group as well as a grand total for the report. To improve the appearance of the table and highlight data you want to emphasize, you can merge cells and apply formatting to data and table headings.

You can initially hide detail or grouped data, and include drilldown toggles to enable a user to interactively choose how much data to show.

For more information, see [Tables \(Report Builder and SSRS\)](#).

Matrix

Use a matrix to display aggregated data summaries, grouped in rows and columns, similar to a PivotTable or crosstab. The number of rows and columns for groups is determined by the number of unique values for each row and column groups. The following figure shows the initial matrix template, selected on the design surface:



You can group data by multiple fields or expressions in row and column groups. At run time, when the report data and data regions are combined, a matrix grows horizontally and vertically on the page as columns for column groups and rows for row groups are added. The matrix cells display aggregate values that are scoped to the intersection of the row and column groups to which the cell belongs. For example, if your matrix has a row group (Category) and two column groups (Territory and Year) that display the sum of sales, the report displays two cells with sums of sales for each value in the Category group. The scope of the cells are the two intersections are: Category and Territory and Category and Year. The matrix can include nested and adjacent groups. Nested groups have a parent-child relationship and adjacent groups a peer relationship. You can add subtotals for any and all levels of nested row and column groups within the matrix.

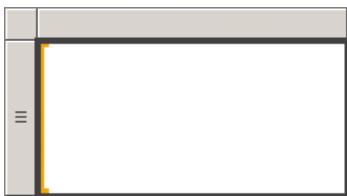
To make the matrix data more readable and highlight the data you want to emphasize, you can merge cells or split horizontally and vertically and apply formatting to data and group headings.

You can also include drilldown toggles that initially hide detail data; the user can then click the toggles to display more or less detail as needed.

For more information, see [Create a Matrix](#).

List

Use a list to create a free-form layout. You are not limited to a grid layout, but can place fields freely inside the list. You can use a list to design a form for displaying many dataset fields or as a container to display multiple data regions side by side for grouped data. For example, you can define a group for a list; add a table, chart, and image; and display values in table and graphic form for each group value, as you might for an employee or patient record.



For more information, see [\[Create Invoices and Forms with Lists\]](#).

Preparing Data

A table, matrix, and list data regions display data from a dataset. You can prepare the data in the query that retrieves the data for the dataset or by setting properties in the table, matrix, or list.

The query languages such as Transact-SQL, that you use to retrieve the data for the report datasets can prepare the data by applying filters to include only a subset of the data, replacing null values or blanks with constants that make the report more readable, and sorting and grouping data.

If you choose to prepare the data in the table, matrix, or list data region of a report, you set properties on the data region or cells within the data region. If you want to filter or sort the data, set the properties on the data region. For example, to sort the data you specify the columns to sort on and the sort direction. If you want to provide an alternative value for a field, you set the values of the cell text that displays the field. For example, to display Blank when a field is empty or null, you use an expression to set the value.

For more information, see [Preparing Data for Display in a Tablix Data Region \(Report Builder and SSRS\)](#).

Building and Configuring a Table, Matrix, or List

When you add tables or matrices to your report, you can use the Table and Matrix Wizard or build them manually from the templates that Report Builder and Report Designer provide. Lists are built manually from the list template.

The wizard guides you through the steps to quickly build and configure a table or matrix. After you complete the wizard or if you build the tablix data regions from scratch, you can further configure and refine them. The dialog boxes, available from the right-click menus on the data regions, make it easy to set the most commonly used properties for page breaks, repeatability and visibility of headers and footers, display options, filters, and sorting. But the tablix data region provides a wealth of additional properties, which you can set only in the Properties pane of Report Builder. For example, if you want to display a message when the dataset for a table, matrix, or list is empty, you specify the message text in the NoRowsMessage tablix property in the Properties pane.

Changing Between Tablix Templates

You are not limited by your initial tablix template choice. As you add groups, totals, and labels, you might want to modify your tablix design. For example, you might start with a table and then delete the details row and add column groups. For more information, see [Exploring the Flexibility of a Tablix Data Region \(Report Builder and SSRS\)](#).

You can continue to develop a table, matrix, or list by adding any tablix feature. Tablix features include displaying detail data or aggregates for grouped data on rows and columns. You can create nested groups, independent adjacent groups, or recursive groups. You can filter and sort grouped data, and easily combine groups by including multiple group expressions in a group definition.

You can also add totals for a group or grand totals for the data region. You can hide rows or columns to simplify a report and enable the user to toggle the display of the hidden data, as in a drilldown report. For more information, see [Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#).

How-To Topics

This section lists procedures that show you, step by step, how to work with tables, matrices and lists in your reports; how to display data in rows and columns, add and delete columns, merge cells, and include subtotals for row and column groups.

- [Add a Details Group \(Report Builder and SSRS\)](#)
- [Add a Total to a Group or Tablix Data Region \(Report Builder and SSRS\)](#)
- [Change an Item Within a Cell \(Report Builder and SSRS\)](#)
- [Change Row Height or Column Width \(Report Builder and SSRS\)](#)
- [Insert or Delete a Column \(Report Builder and SSRS\)](#)
- [Insert or Delete a Row \(Report Builder and SSRS\)](#)
- [Merge Cells in a Data Region \(Report Builder and SSRS\)](#)
- [Create a Recursive Hierarchy Group \(Report Builder and SSRS\)](#)
- [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#)
- [Display Headers and Footers with a Group \(Report Builder and SSRS\)](#)
- [Create a Stepped Report \(Report Builder and SSRS\)](#)
- [Add, Move, or Delete a Table, Matrix, or List \(Report Builder and SSRS\)](#)

In This Section

The following topics provide additional information about working with the tablix data region.

[Tablix Data Region \(Report Builder and SSRS\)](#)

Explains key concepts related to the tablix data region such as areas of the tablix, detail and grouped data, column and row groups, and static and dynamic rows and columns.

[Adding Data to a Tablix Data Region \(Report Builder and SSRS\)](#)

Provides detailed information about adding detail and grouped data, subtotals and totals, and labels to a tablix data region.

[Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#)

Describes properties for a tablix data region that you can modify to change the way a tablix data region appears when you view it in a report.

[Controlling Row and Column Headings \(Report Builder and SSRS\)](#)

Describes how to control row and column headings when a table, matrix, or list data region can span multiple pages horizontally or vertically.

[Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#)

Describes how to display recursive data where the relationship between parent and child is represented by fields in the dataset.

[Understanding Groups \(Report Builder and SSRS\)](#)

Explains what groups are and when you use them and describes the groups available for the different tablix data regions.

See Also

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Nested Data Regions \(Report Builder and SSRS\)](#)

[Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Report Parameters \(Report Builder and Report Designer\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Gauges \(Report Builder and SSRS\)](#)

Tables (Report Builder and SSRS)

3/24/2017 • 7 min to read • [Edit Online](#)

In Reporting Services, you can use a table to display detail data or grouped data, or a combination of both in a paginated report.

You can group data by a single field, by multiple fields, or by writing your own expression. You can create nested groups or independent, adjacent groups. To display aggregated values for grouped data, add totals to groups. Format the rows and columns to highlight the data you want to emphasize. You can initially hide detail or grouped data, and include drilldown toggles to enable a user to interactively choose how much data to show.

To quickly get started with tables, see [Tutorial: Creating a Basic Table Report \(Report Builder\)](#) or [Create a Basic Table Report \(SSRS Tutorial\)](#).

NOTE

You can publish tables separately from a report as report parts. Read more about [Report Parts](#).

Adding a Table to Display Detail Data

Add a table to the design surface from the Insert tab on the ribbon. You can add a table by using the Table or Matrix Wizard, which includes creating a data source connection and dataset and configuring the table, or a table based on the table template, which you configure manually.

NOTE

The wizard is available only in Report Builder.

To describe how to configure a table from beginning to end, this topic uses the table template.

By default, a new table has a fixed number of columns with a header row for labels and a data row for detail data. The following figure shows a new table added to the design surface.

Header		
Data		

When you select the table, row and column handles appear on the outside of the table and brackets appear inside cells. Row handles display graphics that help you understand the purpose of each row. Brackets indicate group membership for a selected cell. The following figure shows a selected empty cell in a default table.

Header		
Data		

The row handle for the Data row shows the details symbol (≡). To display data on these rows, drag fields from the Report Data pane to the table cells in either the header or the details row. Both rows are filled in simultaneously. To add additional columns, drag the field to the table until you see an insertion point. After you add dataset fields to the table, you can change the default format for dates and currency to control the way they display in the report. The following diagram shows a table data region with these fields: Date, Order, Product, Qty, and Line Total.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]

Check your design by viewing the report in Preview. The table expands down the page as needed. The label row and the details row each display once for every row in the dataset query result set. Each product sold in the order is listed on a separate row, along with the quantity and the line total for the item, as shown in the following figure:

Date	Order	Product	Qty	Line Total
August 31, 2011	SO44285	Mountain Bike Socks, M	12	\$64.80
October 31, 2011	SO44792	Mountain Bike Socks, M	12	\$64.80
May 30, 2012	SO46604	Men's Bib-Shorts, S	2	\$107.99
June 30, 2012	SO47004	Men's Bib-Shorts, S	5	\$269.97

The table that you start with is a template based on the tablix data region. You can enhance the design of your table by adding features that are supported by the underlying tablix data region. For more information, see [Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#). You can also continue to develop your table by adding row groups, column groups, and by adding or removing detail groups. For more information, see [Exploring the Flexibility of a Tablix Data Region \(Report Builder and SSRS\)](#).

Adding Totals for Detail Data

To add totals, select cells with numeric data, and then use the shortcut menu to automatically add labels and totals for detail data for numeric fields. You can also specify other labels and totals manually. The following figure shows a typical totals row that includes both automatic and manually specified totals:

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]
Total Orders:	[CountDistinct(Order)]	Total:	[Sum(Qty)]	[Sum(LineTotal)]

In Preview, the report displays the header row and the details row once for every row in the dataset query result set, and it displays the totals row. The follow figure shows the last few rows of the table including the total row.

June 01, 2004	SO71952	Women's Mountain Shorts, M	3	\$125.98
June 01, 2004	SO71952	Women's Mountain Shorts, S	3	\$125.98
Total Orders:	2416		Total Sales:	\$1,780,769.91

For more information, see [Add a Total to a Group or Tablix Data Region \(Report Builder and SSRS\)](#).

Adding Row Groups to a Table

Just as you can drag a field from the Report Data pane to a cell to display detail data, you can drag a field to the Grouping pane to add a group. For a table, drag the field to the Row Groups pane. After you add a group, the table automatically adds cells in new columns in the row group area in which to display the group values. For more information about areas, see [Tablix Data Region Areas \(Report Builder and SSRS\)](#).

The following figure shows a table with two nested row groups in Design view. The row groups were created by dragging the Order field and then the Date field to the Row Groups pane and inserting each group as a parent of the existing groups. The figure shows a parent group based on date and a child group based on order number, as well as the details group that was defined by default.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]

In Preview, the report displays the order data grouped first by date, and then by order, as shown in the follow figure.

Date	Order	Product	Qty	Line Total
May 31, 2011	SO43659	Long-Sleeve Logo Jersey, M	3	\$86.52
		Long-Sleeve Logo Jersey, XL	1	\$28.84
		Mountain Bike Socks, M	6	\$34.20
		AWC Logo Cap	2	\$10.37
	SO43661	Long-Sleeve Logo Jersey, L	4	\$115.36
		Long-Sleeve Logo Jersey, XL	2	\$57.68
		AWC Logo Cap	4	\$20.75
	SO43664	Long-Sleeve Logo Jersey, XL	1	\$28.84
		Long-Sleeve Logo Jersey, M	1	\$28.84
	SO43665	Long-Sleeve Logo Jersey, L	2	\$57.68

An alternative way of displaying grouped data is to indent the group hierarchy to display the nested relationship of groups instead of presenting each value in its own column. This style of formatting is called a stepped report. For more information about how to format group information as a stepped report, see [Create a Stepped Report \(Report Builder and SSRS\)](#).

Adding Totals to Row Groups

To show totals for a group, you can use the context-sensitive **Add Total** command. For a row group, the Add Total command adds a row outside the group so that it repeats only once in relation to the group. For nested groups, the total row for the child group is outside the child group but inside the parent group. In such a case, it is useful to set the background color of the total row for the child group to distinguish it from the detail rows. You can also use a different background color to distinguish the table header and footer rows. The following figure shows the table with a total row added for the group based on order numbers.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]
	[CountDistinct(Order)]	Subtotal:	[Sum(Qty)]	[Sum(LineTotal)]
Total Orders	[CountDistinct(Order)]	Total:	[Sum(Qty)]	[Sum(LineTotal)]

When you view the report, the row displaying the order subtotals repeats once for every order number. The table footer displays totals for all dates. In the following figure, the last few rows show the last three detail rows, the subtotal for the last order number SO71952, and the totals for all dates in the table.

	SO71952	Women's Mountain Shorts, L	15	\$548.55
		Women's Mountain Shorts, M	3	\$125.98
		Women's Mountain Shorts, S	3	\$125.98
	111	Subtotal:	2999	\$83,642.49
Total Orders:	2416	Total:	64569	\$1,780,769.91

For more information, see [Add a Total to a Group or Tablix Data Region \(Report Builder and SSRS\)](#).

Removing or Hiding Detail Rows

After you preview a table in a report, you may decide to remove existing detail rows. Or you might decide to hide

them by default and allow the user to toggle between viewing more or less detail, as in a drilldown report.

To remove detail rows from a table, use the Grouping pane. Select the detail group, and use the shortcut menu to delete the group and the rows that display the detail data. The following figure shows the design view for a table grouped by date and order number, but with no detail rows. No total rows have been added to this table.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Sum(Qty)]	[Sum(LineTotal)]

After you delete the details row, values are scoped to the row groups. The detail data no longer displays.

NOTE

Verify that after you remove a details row, the expression in each cell specifies an aggregate expression where appropriate. If necessary, edit the expression to specify aggregate functions as needed.

The following figure shows this report in Preview.

Date	Order	Product	Qty	Line Total
July 01, 2001	S043659	AWC Logo Cap	12	\$159.93
	S043659	AWC Logo Cap	10	\$193.79
	S043659	Long-Sleeve Logo Jersey M	2	\$57.68
	S043659	AWC Logo Cap	10	\$102.25

To add or remove rows from the table, see [Insert or Delete a Row \(Report Builder and SSRS\)](#).

You can also hide the detail rows when the report is initially viewed. To do so, you can create a drilldown report, in which only the parent group data is displayed. For each inner group (including the details group), add a visibility toggle to the grouping cell of the containing group. For example, for the details group, add a toggle to the text box that displays the order number group value. For the order number group, add a toggle to the text box that displays the date group value. The following figure shows the row for September 01, 2001, expanded to display the first few orders.

Date	Order	Product	Qty	Line Total
July 01, 2001		Mountain Bike Socks, M	167	\$2,875.15
August 01, 2001		AWC Logo Cap	449	\$7,038.58
September 01, 2001	S044074	AWC Logo Cap	1	\$5.19
	S044075	Mountain Bike Socks, L	1	\$5.70
		Mountain Bike Socks, M	2	\$11.40

For more information, see [Add an Expand or Collapse Action to an Item \(Report Builder and SSRS\)](#).

See Also

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Tables, Matrices, and Lists (Report Builder and SSRS)

Create a Matrix (Report Builder and SSRS)

3/24/2017 • 5 min to read • [Edit Online](#)

Use a matrix to display grouped data and summary information. You can group data by multiple fields or expressions in row and column groups. Matrices provide functionality similar to crosstabs and pivot tables. At run time, as the report data and data regions are combined, a matrix grows horizontally and vertically on the page. Values in matrix cells display aggregate values scoped to the intersection of the row and column groups to which the cell belongs. You can format the rows and columns to highlight the data you want to emphasize. You can also include drilldown toggles that initially hide detail data; the user can then click the toggles to display more or less detail as needed.

After your initial design, you can continue to develop a matrix to improve the viewing experience for the user. For more information, see [Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#).

To quickly get started with matrices, see [Tutorial: Creating a Matrix Report \(Report Builder\)](#).

NOTE

You can publish lists separately from a report as report parts. Read more about [Report Parts \(Report Builder and SSRS\)](#).

Adding a Matrix to Your Report

Add a matrix to the design surface from the Insert tab on the ribbon. You have the option to add a matrix by using the Table or Matrix Wizard, which includes creating a data source connection and dataset, and configuring the matrix or adding a matrix based on the matrix template.

NOTE

The wizard is available only in SQL Server Report Builder for SQL Server 2012.

To describe how to configure a table from beginning to end, this topic uses the matrix template. The matrix initially has a row group, a column group, a corner cell, and a data cell, as shown in the following figure.



When you select a matrix on the design surface, row and column handles appear, as shown in the following figure.



Add groups by dragging dataset fields to the Row Groups and Column Groups areas of the Grouping pane. The first field that you drag to the row groups or column groups pane replaces the initial empty default group. You can then apply formatting for each cell, depending on the data.

		[Geography]
	[Category]	[Sum(LineTotal)]

In Preview, the matrix expands to show the row group and column group values. The cells display summary values, as shown in the following figure.

	North America	Europe	Pacific
Accessories	\$384,057	\$148,057	\$18,599
Bikes	\$51,990,891	\$11,503,109	\$1,186,388
Clothing	\$1,251,999	\$443,035	\$33,683
Components	\$8,601,244	\$2,746,093	\$183,142

The matrix you start with is a template based on the tablix data region. You can continue to develop your matrix design by adding nested or adjacent row groups or column groups, or even adding detail rows. For more information, see [Exploring the Flexibility of a Tablix Data Region \(Report Builder and SSRS\)](#).

Adding a Parent Group or Child Group to a Matrix

To add a group based on a single dataset field, drag the field from the Report Data pane to the appropriate Row Groups or Column Groups area of the Grouping pane. Drop the field in the group hierarchy to set its relationship to existing groups. Drop it above an existing group to create a parent group, or drop it below an existing group to create a child group.

Several things happen when you drop a field in the **Grouping** pane:

- A new group with a unique name based on the field name is automatically created. The group expression is set to the simple field name reference, for example [Category].
- A new row or column appears in the corresponding row group or column group area.
- In the new column, a row group cell appears for the default data rows from the report dataset. Cells in the tablix body for this row are now members of the row group. If there are any column groups defined, cells that are in the columns are members of those column groups. Group indicators provide visual cues for the group membership of each cell.

To customize the group after it is created, use the **Tablix Group** dialog box. You can change the group name, and edit or add additional expressions to the group definition. To add or remove rows from the table, see [Insert or Delete a Row \(Report Builder and SSRS\)](#).

When the report runs, dynamic column headers expand right (or left, if the Direction property of the matrix is set to RTL) for as many columns as there are unique group values. Dynamic rows expand down the page. The data that appears in the tablix body cells are aggregates based on the intersections of row and column groups, as shown in the following figure.

		[Geography]	Total
		[CountryRegion]	
	[Category]	[Subcat]	[Sum(LineTotal)] [Sum(LineTotal)]
	Total		[Sum(LineTotal)] [Sum(LineTotal)]
			[Sum(LineTotal)] [Sum(LineTotal)]

In preview, the report displays as in the following figure.

Sales by Area and Year		North America		2003	2004	Total
		CA	US			
Clothing	Caps	\$2,801.06	\$9,798.76	\$9,008.44	\$3,591.38	\$12,599.82
	Tights	\$8,233.90	\$51,391.51	\$59,625.41		\$59,625.41
	Total	\$11,034.96	\$61,190.27	\$68,633.85	\$3,591.38	\$72,225.23
Components	Chains	\$793.83	\$4,972.00	\$3,543.48	\$2,222.35	\$5,765.83
	Cranksets	\$18,515.99	\$105,118.64	\$72,889.77	\$50,744.86	\$123,634.62
	Touring Frames	\$196,141.85	\$767,130.33	\$603,497.08	\$359,775.10	\$963,272.17
	Total	\$215,451.67	\$877,220.97	\$679,930.33	\$412,742.30	\$1,092,672.63
Grand Total		\$226,486.63	\$938,411.23	\$748,564.18	\$416,333.68	\$1,164,897.86

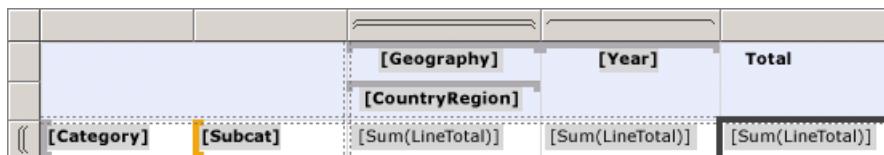
To write expressions that specify a scope other than the default scope, you must specify the name of a dataset, data region, or group in the aggregate function all. To calculate the percentage each subcategory contributes to the Clothing category group values, add a column inside the Category group next to the Total column, format the text box to show percentage, and add an expression that uses the default scope in the numerator, and the Category group scope in the denominator, as shown in the following example.

```
=SUM(Fields!Linetotal.Value)/SUM(Fields! Linetotal.Value,"Category")
```

For more information, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Adding an Adjacent Group to a Matrix

To add an adjacent group based on a single dataset field, use the shortcut menu in the Grouping pane. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#). The following figure shows a group based on geography and an adjacent group based on year.



In this example, the query has filtered data values to only include those values for Europe and for the years 2003 and 2004. However, you can set filters on each group independently. In preview, the report displays as in the following figure.

		Europe			2003	2004	Total
		DE	FR	GB			
Accessories	Bike Racks	\$14,223	\$20,563	\$22,843	\$32,637	\$24,991	\$57,629
	Bottles and Cages	\$758	\$636	\$754	\$1,300	\$848	\$2,148
	Cleaners	\$860	\$1,022	\$1,323	\$1,936	\$1,269	\$3,206
	Helmets	\$10,661	\$12,587	\$20,495	\$28,389	\$15,804	\$44,193
	Hydration Packs	\$6,787	\$5,750	\$7,209	\$11,257	\$8,488	\$19,745

To add a total column for an adjacent column group, click in the column group definition cell and use the **Add Total** command. A new static column is added next to the column group, with a default aggregate sum for every numeric field in the existing rows. To change the expression, manually edit the default aggregate, for example, `Avg([Sales])`. For more information, see [Add a Total to a Group or Tablix Data Region \(Report Builder and SSRS\)](#).

See Also

[Aggregate Functions Reference \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Create Invoices and Forms with Lists (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

A list data region repeats with each group or row in the Reporting Services paginated report dataset. A list can be used to create free-form reports or forms, such as invoices, or in conjunction with other data regions. You can define lists that contain any number of report items. A list can be nested with

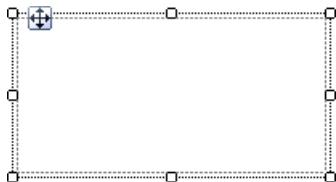
To quickly get started with lists, see [Tutorial: Creating a Free Form Report \(Report Builder\)](#).

NOTE

You can publish lists separately from a report as report parts. Read more about [Report Parts \(Report Builder and SSRS\)](#).

Adding a List to Your Report

Add a list to the design surface from the Insert tab on ribbon. By default, the list initially has a single cell in a row associated with the detail group.



When you select a list on the design surface, row and column handles appear, as shown in the following figure.



The list you start with is a template based on the tablix data region. After you add a list, you can continue to enhance the design by changing the content or appearance of the list by specifying filter, sort, or group expressions, or changing the way the list displays across report pages. For more information, see [Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#). Although the list starts with a single column and row, you can further continue to develop your list design by adding nested or adjacent row groups or column groups, or adding additional detail rows. For more information, see [Exploring the Flexibility of a Tablix Data Region \(Report Builder and SSRS\)](#).

Displaying Data in a Free-form Layout

To organize report data in a free-form layout instead of a grid, you can add a list to the design surface. Drag fields from the Report Data pane to the cell. By default, the cell contains a rectangle that acts as a container. Move each field in the container until you have the design you want. Use the snaplines that appear when you drag text boxes in the rectangle container to help you align edges vertically and horizontally. Remove unwanted white space by adjusting the size of the cell. For more information, see [Change Row Height or Column Width \(Report Builder and SSRS\)](#).

The following figure shows a list that displays information about an order, including these fields: Date, Order, Qty,

Product, LineTotal, and an image.



In Preview, the list repeats to display the field data in the free-form format, as shown in the following figure:

July 01, 2001	3 Long-Sleeve Logo Jersey, L	S043677	
July 01, 2001	1 Road-150 Red, 56	S043677	

NOTE

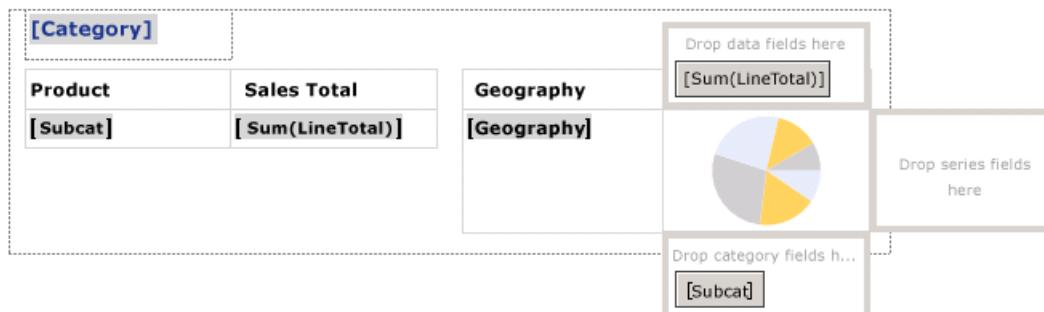
The dotted lines displays in these figures are included to show the free-form layout for each field value. Typically, you would not use dotted lines in a production report.

Displaying Data with One Level of Grouping

Because a list automatically provides a container, you can use a list to display grouped data with multiple views.

To change the default list to specify a group, edit the Details group, specify a new name, and specify a group expression.

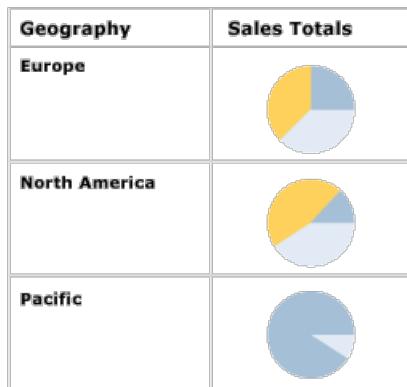
For example, you can embed a table and a chart that show different views of the same dataset. You can add a group to the list so that the nested report items will repeat once for every group value. The following figure shows a list grouped by product category. Notice that there is no detail row. Two tables are nested side by side in the list. The first table displays the subcategories with total sales. The second table displays the category grouped by geographical area, with a chart that shows the distribution of subcategories.



In Preview, the table displays total sales for all subcategories of bicycles, and the table beside it displays the breakdown of sales per geographical area. By using an expression to specify the background color for the table and a custom palette for the chart, the first table also provides the legend for the chart colors.

Bikes

Product	Sales Total
Mountain Bikes	\$25,998,934.56
Road Bikes	\$28,760,967.50
Touring Bikes	\$9,920,485.64



See Also

[Aggregate Functions Reference \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Tablix Data Region (Report Builder and SSRS)

3/24/2017 • 6 min to read • [Edit Online](#)

In Reporting Services, the tablix data region is a generalized layout report item that displays paginated report data in cells that are organized into rows and columns. Report data can be detail data as it is retrieved from the data source, or aggregated detail data organized into groups that you specify. Each tablix cell can contain any report item, such as a text box or an image, or another data region, such as a tablix region, chart, or gauge. To add multiple report items to a cell, first add a rectangle to act as a container. Then, add the report items to the rectangle.

The table, matrix, and list data regions are represented on the ribbon by templates for the underlying tablix data region. When you add one of these templates to a report, you are actually adding a tablix data region that is optimized for a specific data layout. By default, a table template displays detail data in a grid layout, a matrix displays group data in a grid layout, and a list displays detail data in a free-form layout.

By default, each tablix cell in a table or matrix contains a text box. The cell in a list contains a rectangle. You can replace a default report item with a different report item, such as an image.

When you define groups for a table, matrix, or list, Report Builder and Report Designer add rows and columns to the tablix data region on which to display grouped data.

To understand the tablix data region, it helps to understand the following:

- The difference between detail data and grouped data.
- Groups, which are organized as members of group hierarchies on the horizontal axis as row groups and on the vertical axis as column groups.
- The purpose of tablix cells in the four areas of a tablix data region: the body, the row group headers, the column group headers, and the corner.
- Static and dynamic rows and columns, and how they relate to groups.

This article spells out these concepts to explain the structure that Report Builder and Report Designer add for you when you add templates and create groups, so you can modify the structure to suit your own needs. Report Builder and Report Designer provide multiple visual indicators to help you recognize tablix data region structure. For more information, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Detail and Grouped Data

Detail data is all the data from a report dataset as it comes back from the data source. Detail data is essentially what you see in the query designer results pane when you run a dataset query. The actual detail data includes calculated fields that you create, and is restricted by filters set on the dataset, data region, and details group. You display detail data on a detail row by using a simple expression such as [Quantity]. When the report runs, the detail row repeats once for each row in the query results at run time.

Grouped data is detail data that is organized by a value that you specify in the group definition, such as

[SalesOrder]. You display grouped data on group rows and columns by using simple expressions that aggregate the grouped data, such as [Sum(Quantity)]. For more information, see [Understanding Groups \(Report Builder and SSRS\)](#).

Understanding Group Hierarchies

Groups are organized as members of group hierarchies. Row group and column group hierarchies are identical structures on different axes. Think of row groups as expanding down the page and column groups as expanding across the page.

A tree structure represents nested row and column groups that have a parent/child relationship, such as a category with subcategories. The parent group is the root of the tree and child groups are its branches. Groups can also have an independent, adjacent relationship, such as sales by territory and sales by year. Multiple unrelated tree hierarchies are called a forest. In a tablix data region, row groups and columns groups are each represented as an independent forest. For more information, see [Understanding Groups \(Report Builder and SSRS\)](#).

Understanding Tablix Data Region Areas

A tablix data region has four possible areas for cells: the tablix corner, the tablix row group hierarchy, the tablix column group hierarchy, or the tablix body. The tablix body always exists. The other areas are optional.

Cells in the tablix body area display detail and group data.

Cells in the Row Groups area are created automatically when you create a row group. These are row group header cells and display row group instance values by default. For example, when you group by [SalesOrder], group instance values are the individual sales orders that you are grouping by.

Cells in the Column Groups area are created automatically when you create a column group. These are column group header cells, and they display column group instance values by default. For example, when you group by [Year], group instance values are the individual years that you are grouping by.

Cells in the tablix corner area are created automatically when you have both row groups and column groups defined. Cells in this area can display labels, or you can merge the cells and create a title.

For more information, see [Tablix Data Region Areas \(Report Builder and SSRS\)](#).

Understanding Static and Dynamic Rows and Columns

A tablix data region organizes cells in rows and columns that are associated with groups. Group structures for row groups and columns are identical. This example uses row groups, but you can apply the same concepts to column groups.

A row is either a static or dynamic. A static row is not associated with a group. When the report runs, a static row renders once. Table headers and footers are static rows. Static rows display labels and totals. Cells in a static row are scoped to the data region.

A dynamic row is associated with one or more groups. A dynamic row renders once for every unique group value for the innermost group. Cells in a dynamic row are scoped to the innermost row group and column group to which the cell belongs.

Dynamic detail rows are associated with the Details group that is automatically created when you add a table or list to the design surface. By definition, the Details group is the innermost group for a tablix data region. Cells in detail rows display detail data.

Dynamic group rows are created when you add a row group or column group to an existing tablix data region. Cells in dynamic group rows display aggregated values for the default scope.

The Add Total feature automatically creates a row outside the current group on which to display values that are scoped to the group. You can also add static and dynamic rows manually. Visual indicators help you understand which rows are static and which rows are dynamic. For more information, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder\)](#) and [SSRS](#).

See Also

[Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

[Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#)

[Exploring the Flexibility of a Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Tablix Data Region Areas (Report Builder and SSRS)

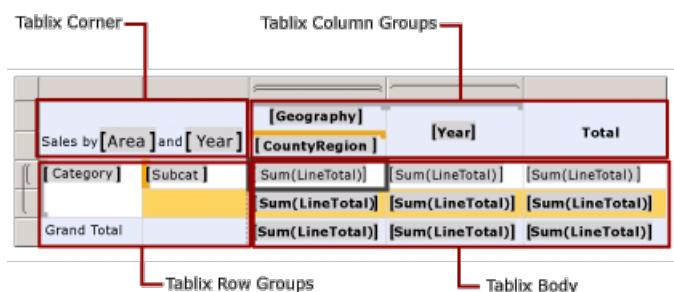
3/24/2017 • 3 min to read • [Edit Online](#)

In a Reporting Services paginated report, a tablix data region has four areas that contain tablix cells:

- The corner
- The row group area
- The column group area
- The body

Cells in each area have a distinct function. You add cells to the tablix body area to display detail and grouped data. Report Builder and Report Designer add cells to the row group or column group area when you create a group in order to display group instance values. Report Builder and Report Designer create tablix corner cells when both row groups and column groups exist.

On the design surface, dotted lines denote the four areas of a selected tablix data region. The following figure shows the areas for a tablix region with nested row groups based on category and subcategory, nested column groups based on geography and country/region, and an adjacent column group based on year.



The following list describes each area:

- **Tablix corner area.** (Optional) A tablix corner is located in the upper-left corner, or upper-right corner for right-to-left (RTL) layouts. This area is automatically created when you add both row groups and column groups to a tablix data region. In this area, you can merge cells and add a label or embed another report item. In the figure, merged cells in the corner display the label Sales by Area and Year.
- **Tablix column groups area.** (Optional) Tablix column groups are located in the upper-right corner (upper-left corner for RTL layout). This area is automatically created when you add a column group. Cells in this area represent members of the column groups hierarchy, and display the column group instance values. In the figure, the cells that display [Geography] and [CountryRegion] are nested column groups, and the cell that displays [Year] is an adjacent column group. The column [Total] displays the aggregated totals across each row.
- **Tablix row groups area.** (Optional) Tablix row groups are located on the lower-left corner (lower right for RTL layout). This area is automatically created when you add a row group. Cells in this area represent members of the row groups hierarchy, and display row group instance values. In the figure, the cells that display [Category] and [Subcat] are nested row groups. The Total row below Subcat repeats with each category group to show the aggregated subtotals for each column. The grand total row shows the totals for all categories.
- **Tablix body area.** The tablix body is located in the lower right corner (lower left for RTL layout). The tablix body displays detail and grouped data. In this example, only aggregated data is used. The scope for the expression is determined by the innermost groups to which the text box belongs. Cells in the tablix body display detail data when they are members of a detail row and they represent aggregate data when they

are members of a row or column associated with a group. By default, cells in a group row or column that contain simple expressions that do not include an aggregate function, evaluate to the first value in the group. In the figure, the cells display the aggregate totals for line totals for all sales order.

When the report runs, column groups expand right (or left, if the Direction property of the tablix data region is set to RTL) for as many columns as there are unique values for a group expression. Row groups expand down the page. For more information, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder and SSRS\)](#).

The following figure shows the tablix data region in Preview.

The diagram illustrates the structure of a Tablix data region. It features a header row with three columns: 'Sales by Area and Year', 'North America' (which further branches into 'CA' and 'US'), and three year columns ('2003', '2004', 'Total'). Below this is a body section containing several rows. Some rows are grouped under 'Clothing' (with subrows for 'Caps', 'Tights', and a total row), while others are grouped under 'Components' (with subrows for 'Chains', 'Cranksets', 'Touring Frames', and a total row). A 'Grand Total' row spans the entire width of the table. Red boxes and arrows highlight these groupings: one arrow points from the 'Tablix Corner' to the 'Sales by Area and Year' header; another points from 'Tablix Column Groups' to the 'North America' header; a third points from 'Tablix Row Groups' to the 'Clothing' and 'Components' sections; and a fourth points from 'Tablix Body' to the 'Grand Total' row.

Sales by Area and Year		North America		2003	2004	Total
		CA	US			
Clothing	Caps	\$2,801.06	\$9,798.76	\$9,008.44	\$3,591.38	\$12,599.82
	Tights	\$8,233.90	\$51,391.51	\$59,625.41		\$59,625.41
	Total	\$11,034.96	\$61,190.27	\$68,633.85	\$3,591.38	\$72,225.23
Components	Chains	\$793.83	\$4,972.00	\$3,543.48	\$2,222.35	\$5,765.83
	Cranksets	\$18,515.99	\$105,118.64	\$72,889.77	\$50,744.86	\$123,634.62
	Touring Frames	\$196,141.85	\$767,130.33	\$603,497.08	\$359,775.10	\$963,272.17
	Total	\$215,451.67	\$877,220.97	\$679,930.33	\$412,742.30	\$1,092,672.63
Grand Total		\$226,486.63	\$938,411.23	\$748,564.18	\$416,333.68	\$1,164,897.86

The row group area displays two category group instances for Clothing and Components. The column group are displays a geography group instance for North America, with two nested country/region group instances for Canada (CA) and the United States (US). In addition, the adjacent column displays two year group instances for 2003 and 2004. The Total column row displays the row totals; the totals row that repeats with the category group shows subcategory totals, and the grand total row displays the category totals once for the data region.

See Also

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Report Builder Tutorials](#)

[Tables \(Report Builder and SSRS\)](#)

[Create a Matrix](#)

[Create Invoices and Forms with Lists](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

Tablix Data Region Cells, Rows, and Columns (Report Builder) and SSRS

3/24/2017 • 8 min to read • [Edit Online](#)

To control how the rows and columns of a tablix data region display data in a Reporting Services paginated report, you must understand how to specify rows and columns for detail data, for group data, and for labels and totals. In many cases, you can use the default structures for a table, matrix, or list to display your data. For more information, see [Tables \(Report Builder and SSRS\)](#), [Matrixes](#), or [Lists](#).

A tablix data region displays detail data on detail rows and detail columns and grouped data on group rows and group columns. When you add row groups and column groups to a tablix data region, rows and columns on which to display the data are automatically added. You can manually add and remove rows and columns to customize a tablix data region and control the way your data displays in the report.

To understand how to customize a tablix data region, you should first understand how to interpret the visual cues you see when you select a tablix data region on the design surface.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Tablix Visual Cues

Visual cues on a tablix data region help you work with a tablix data region to display the data you want.

Row and Column Handles

When you select a tablix data region, the row and column handle graphics indicate the purpose of each row and column. Handles indicate rows and columns that are inside a group or outside a group. The following table shows a variety of handle displays.

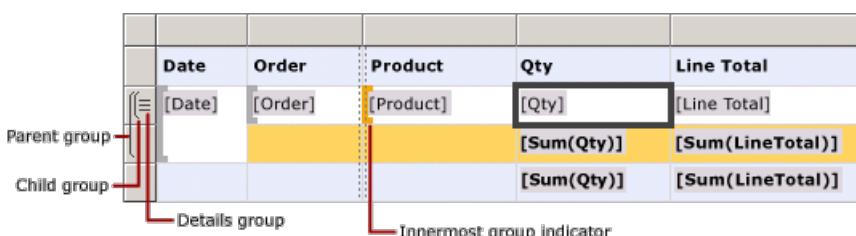
ICON	DESCRIPTION
	Only the details group on the row group hierarchy
	One outer group and the child details group
	One outer group, one inner group; no details group
	One outer group, one inner group, and the child details group
	One outer group with a footer row for totals and one inner group
	One outer group with a footer row for totals, one inner group with a footer row for totals, and one details group

ICON	DESCRIPTION
	One outer group with a header for labels and a footer for totals, and an inner group; no details group

Group Rows

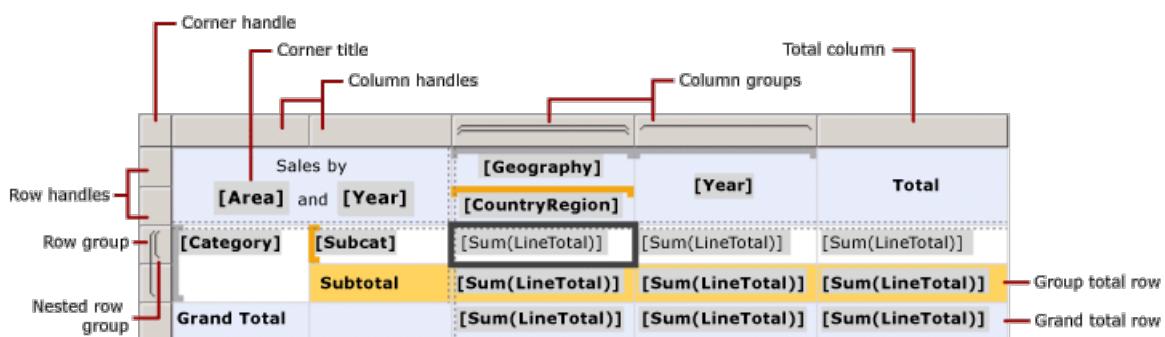
Rows inside a group repeat once per unique group value and are typically used for aggregate summaries. Rows outside a group repeat once with respect to the group and are used for labels or subtotals. When you select a tablix cell, row and column handles and brackets inside the tablix data region show the groups to which a cell belongs. This figure displays the following visual cues:

- Row and column handles that indicate group associations.
- Highlighted group indicators that show the innermost group membership for a selected cell.
- Group indicators that show all group memberships for a selected cell.



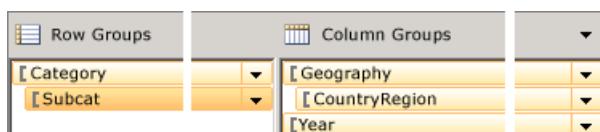
Total Rows

After you add row and column groups, you can add a row to display totals for columns and a column to display totals for rows. The following figure shows a matrix with both row and column groups, and a total row and a total column.



Grouping Pane

The Grouping pane displays the row and column groups for the currently selected tablix data region on the design surface. The following figure shows the Grouping pane for this tablix data region.



The Row Groups pane shows the parent group Category and child group Subcat. The Column groups pane shows the parent group Geography and child group CountryRegion, and also the Year group, which is an adjacent group to the Geography group. When you select the Subcat group in the Row Groups pane, the group bar turns a darker shade of orange, and the corresponding row group member cell is selected on the design surface.

Displaying Data on Rows and Columns

Rows and row groups and columns and column groups have identical relationships. The following discussion describes how to add rows to display detail and group data on rows in a tablix data region, but the same principles apply to adding columns to display detail and grouped data.

For each row in a tablix data region, a row is either inside or outside each row group. If the row is inside a row group, it repeats once for every unique value of the group, known as a *group instance*. If the row is outside a row group, it repeats only once in relation to that group. Rows outside all row groups are static and repeat only once for the data region. For example, a table header or footer row is a static row. Rows that repeat with at least one group are dynamic.

When you have nested groups, a row can be inside a parent group but outside a child group. The row repeats for every group value in the parent group, but displays only once in relation to the child group. To display labels or totals for a group, add a row outside the group. To display data that changes for every group instance, add a row inside the group.

When you have detail groups, each detail row is inside the detail group. The row repeats for every value in the dataset query result set.

For more information about group hierarchies, see [Understanding Groups \(Report Builder and SSRS\)](#).

The following figure shows a tablix data region with nested row groups and a details group.

Date	Order	Product	Qty	Line Total
[Date]	[Order]	[Product]	[Qty]	[Line Total]
	[CountDistinct(Order)]	Subtotal:	[Sum(Qty)]	[Sum(LineTotal)]
Total Orders	[CountDistinct(Order)]	Total:	[Sum(Qty)]	[Sum(LineTotal)]

For a tablix data region that displays detail data, the details group is the innermost child group. Rows that you add to a details group repeat once per row in the result set for the query for the dataset linked to this tablix data region. The following figure shows the last page of the rendered report. In this figure, you can see the last detail rows and the subtotal row for the last order.

	S071952	Women's Mountain Shorts, L	15	\$548.55
		Women's Mountain Shorts, M	3	\$125.98
		Women's Mountain Shorts, S	3	\$125.98
	111	Subtotal:	2999	\$83,642.49
Total Orders:	2416	Total:	64569	\$1,780,769.91

For each column in a tablix data region, the same principles apply. For example, a column is either inside or outside each column group; to display totals, add a column outside the group.

To remove rows and columns associated to a group, you can delete the group. When you delete a group, you have the choice between deleting the group definition only or deleting the group and all its associated rows and columns. By deleting just the group, you preserve the row and column layout on the data region. When you delete the group and its related rows and columns, you are deleting all static rows and columns (including group headers and footers) and dynamic rows and columns (including group instances) that are associated with that group.

For step-by-step instructions about adding or deleting rows and columns, see [Insert or Delete a Row \(Report Builder and SSRS\)](#) and [Insert or Delete a Column \(Report Builder and SSRS\)](#).

Understanding Tablix Cells

Tablix cells belong to one of four tablix areas: the tablix body, tablix row or tablix column group areas, or the tablix corner. Although each cell can potentially display any value in the dataset, the default function for each cell is determined by its location. For detailed information about tablix areas, see [Tablix Data Region Areas \(Report Builder and SSRS\)](#).

By default, cells in tablix row and column group areas represent group members. Group members are organized into multiple tree structures in the report definition. The row group hierarchy expands horizontally. The column group hierarchy expands vertically. These cells are added automatically when you create a group, and display the unique values for a group at run time.

Cells in the tablix corner are created when there are both row and column group areas. You can merge cells in this area to create a label or embed another report item.

Cells in the tablix body area can display detail data when the cell is in a detail row or column and aggregated group data when the cell is in a group row or column. The scope for the data in a cell is the intersection of the innermost row group and innermost column group to which the cell belongs.

NOTE

The actual data that is displayed for each cell is the evaluated expression for the report item that the cell contains, which is typically a text box. In a cell that belongs to a detail row or column, the expression defaults to the detail data (for example, **[LineTotal]**). In a cell that does not belong to a detail row or column, the expression defaults to an aggregate function (for example, **Sum[LineTotal]**). If an expression does not specify an aggregate function even though the cell belongs to a group row or column, the first value in the group is displayed. For more information about aggregates, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Merging and Splitting Cells

Inside a tablix area, you can merge multiple adjacent cells together. For example, you can create cells for labels that span multiple columns or rows.

In the tablix corner area, cells can be combined in only one direction at a time: horizontally across columns or vertically down rows. To merge a block of cells, merge the cells horizontally first. After all cells have been merged into a single cell in each row, select adjacent cells (you can select all adjacent cells in a column) and merge them.

In the tablix body area, cells can only be merged horizontally. Merging cells vertically is not supported.

For more information, see [Merge Cells in a Data Region \(Report Builder and SSRS\)](#).

You can split a cell that was previously merged. You can split cells horizontally across columns or vertically down rows. To split a cell into a block of cells, split the cell horizontally first, and then split vertically as many times as necessary.

See Also

[Tablix Data Region \(Report Builder and SSRS\)](#)

Preparing Data for Display in a Tablix Data Region (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

A tablix data region displays data from a dataset. You can view all the data retrieved for the dataset or you can create filters so that you see only a subset of the data. You can also add conditional expressions to fill in null values or modify the query for a dataset to include columns that define the sort order for an existing column.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Working with Nulls and Blanks in Field Values

Data for the field collection in a dataset includes all values retrieved from the data source at run time, including null values and blanks. Normally null values and blanks are indistinguishable. In most cases, this is the desired behavior. For example, Numeric aggregate functions like [Sum](#) and [Avg](#) ignore null values. For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

If you do want to handle null values differently, you can use conditional expressions or custom code to substitute a custom value for the null value. For example, the following expression substitutes the text `Null` wherever a null value occurs in the field `[Size]`.

```
=IIF(Fields!Size.Value IS NOTHING,"Null",Fields!Size.Value)
```

For more information about eliminating nulls in your data before retrieving the data from a SQL Server data source using Transact-SQL queries, see "Null Values" and "Null Values and Joins" in the SQL Server documentation in [SQL Server Books Online](#).

Handling Null Field Names

Testing for null values in an expression is fine as long as the field itself exists in the query result set. From custom code, you can test whether the field itself is present in the collection fields returned from the data source at run time. For more information, see [Dataset Fields Collection References \(Report Builder and SSRS\)](#).

Adding a Sort Order Column

By default, you can alphabetically sort values in a dataset field. To sort in a different order, you can add a new column to your dataset that defines the sort order you want in a data region. For example, to sort on the field `[Color]` and sort white and black items first, you can add a column `[ColorSortOrder]`, shown in the following query:

```
SELECT ProductID, p.Name, Color,
CASE
    WHEN p.Color = 'White' THEN 1
    WHEN p.Color = 'Black' THEN 2
    WHEN p.Color = 'Blue' THEN 3
    WHEN p.Color = 'Yellow' THEN 4
    ELSE 5
END As ColorSortOrder
FROM Production.Product p
```

To sort a table data region according to this sort order, set the sort expression on the detail group to `=Fields!ColorSortOrder.Value`. For more information, see [Sort Data in a Data Region \(Report Builder and SSRS\)](#).

See Also

[Dataset Fields Collection \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

Adding Data to a Tablix Data Region (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

In Reporting Services paginated reports, to display data from a report dataset in a table or matrix, in each data cell, specify the name of a dataset field to display. You can display detail data or grouped data. If you add groups to a table or matrix, rows and columns for group values and group data are added automatically. You can then add subtotals and totals for your data.

All data in a data region belongs to at least one group. Detail data is a member of the details group. For more information about detail and grouped data, see [Understanding Groups \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Adding Detail Data

Detail data is all the data from a report dataset after filters are applied to the dataset, data region, and details group. All detail data displayed in a single tablix data region must come from the same report dataset.

To add detail data from a report dataset to a tablix data region, drag a dataset field from the Report Data pane to each cell in the detail row. For existing cells in a tablix data region, you can add or edit a dataset field expression by using the field selector in each cell or by dragging a field from the Report Data pane to the cell. To create additional columns, you can drag the field from the Report Data pane and insert it into an existing tablix data region.

By default, at run-time, a cell in the details row displays detail data and a cell in a group row displays an aggregate value. For more information about tablix rows and columns, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#).

A table template and a list template provide a details row. A matrix template has no details row. If your tablix data region has no details row, you can add one by defining a details group. For more information, see [Add a Details Group \(Report Builder and SSRS\)](#).

Adding Grouped Data

Grouped data is all the detail data specified by a group expression after filters are applied to the dataset, data region, and the group. To organize detail data in groups, drag fields from the Report Data pane to the Grouping pane. When you add a group, Reporting Services automatically adds related rows or columns to the tablix data region on which to display grouped data. Cells in these rows or columns are associated with grouped data. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#).

By default, when you add a dataset field that represents numeric data to a cell in a group row or column, the value of the cell is the sum of the grouped data scoped to the innermost row and column group memberships for the cell. You can change the default aggregate function Sum to any other aggregate function, such as Avg or Count. You can also change the default scope for an aggregate calculation, for example, to calculate the percentage a value contributes to a row group. For more information, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

By default, all grouped data comes from the same report dataset. In a tablix data region, you can include aggregate values from another dataset by specifying the dataset name as a scope. You can specify multiple aggregate values from multiple datasets within a single tablix data region. For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

Adding Subtotals and Totals

To add subtotals for a group and grand totals for the data region, use the Add Total feature on the shortcut menu in a cell or in the Grouping pane. The rows and columns on which to display the totals are automatically added for you. Subtotal and total expressions default to using the [Sum](#) aggregate function. After you add the expression, you can change the default function. For more information, see [Add a Total to a Group or Tablix Data Region \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Adding Labels

To add labels for a group or for the data region, add a row or column outside the group that you want to label. Label rows and columns are similar to rows and columns that you add to display totals. For more information, see [Insert or Delete a Row \(Report Builder and SSRS\)](#) or [Insert or Delete a Column \(Report Builder and SSRS\)](#).

Adding an Existing Tablix Data Region from Another Report

You can copy a data region from another report and paste it into a new or existing report. After you paste the data region, you must ensure that the dataset the data region uses is defined, and that the dataset fields have identical names and data types as in the original report. You cannot copy datasets from one report to another, but if your reports use shared data sources, you can quickly duplicate the dataset in the another report. Also you can import the query text for the queries that retrieve the data for the dataset, which makes it simple to duplicate the queries in reports. For more information, see [Report Embedded Datasets and Shared Datasets \(Report Builder and SSRS\)](#).

See Also

[Expressions \(Report Builder and SSRS\)](#)

[Report Parameters \(Report Builder and Report Designer\)](#)

[Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#)

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Add, Edit, Refresh Fields in the Report Data Pane \(Report Builder and SSRS\)](#)

[Add an Expression \(Report Builder and SSRS\)](#)

Exploring the Flexibility of a Tablix Data Region (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

In a Reporting Services paginated report, when you add a table, matrix, or list data region from the Insert tab on the ribbon, you start with an initial template for a tablix data region. But you are not limited by that template. You can continue to develop how your data displays by adding or removing any tablix data region feature such as groups, rows, and columns.

When you delete a row or column group, you have the option of deleting the rows and columns that are used to display group values. You can also add or remove rows and columns manually. To understand how rows and columns are used to display detail and group data, see [Tablix Data Region \(Report Builder and SSRS\)](#).

After you change the structure of the tablix data region, you can set properties to help control the way the report renders the data region; for example, you can repeat column headers at the top of every page, or keep a group header with the group. For more information, see [Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Changing a Table to a Matrix

By default, a table has detail rows that display the values from the report dataset. Typically, a table includes row groups that organize the detail data by group, and then includes aggregated values based on each group. To change the table to a matrix, add column groups. Typically, you would remove the details group when the data region has both row and column groups so that you can display only the summary values for the groups. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#).

By definition, when you create a matrix, you add a tablix corner cell. You can merge cells in this area and add a label. For more information, see [Merge Cells in a Data Region \(Report Builder and SSRS\)](#).

Changing a Matrix to a Table

By default, a matrix has row groups and column groups and no detail group. To change a matrix to a table, remove column groups and add a details group to display on the details row. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#) and [Add a Details Group \(Report Builder and SSRS\)](#).

Changing a Default List to a Grouped List

By default, a list has detail rows and no groups. To change the list to use a group row, rename the details group and specify a group expression. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#)

Creating Stepped Displays

By default, when you add groups to a tablix data region, cells in the row group header area display group values

in column. When you have nested groups, each group displays in a separate column. To create a stepped display, remove all group columns except one, and format the remaining column to display the group hierarchy as an indented text display. For more information, see [Create a Stepped Report \(Report Builder and SSRS\)](#).

Adding an Adjacent Details Group

By default, the details group is the innermost child group in a group hierarchy. You cannot nest a group under the details group. You can create additional adjacent details groups, to display the top 5 products and the bottom 5 products by sales, for example. Because you can add filter and sort expressions on each group, you can show two views of detail data from the same dataset in one tablix data region. For more information, see [Understanding Groups \(Report Builder and SSRS\)](#), [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#), and [Add a Filter to a Dataset \(Report Builder and SSRS\)](#).

See Also

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Tables \(Report Builder and SSRS\)](#)

[Matrices \(Report Builder and SSRS\)](#)

[Lists \(Report Builder and SSRS\)](#)

Controlling the Tablix Data Region Display on a Report Page

3/29/2017 • 5 min to read • [Edit Online](#)

Read about properties you can set in a Reporting Services paginated report for a table, matrix, or list data region, to change how it appears when you view the report.

Controlling the Appearance of Data

Table, matrix, and list data regions are all examples of *tablix* data regions. The following features help control the appearance of a tablix data region:

- **Formatting data.** To format data in a table, matrix, or list, set the format properties of the text box in the cell. You can set properties for multiple cells at the same time. To format data in a chart, set formatting properties on the series. For more information, see [Formatting Report Items \(Report Builder and SSRS\)](#) and [Formatting a Chart \(Report Builder and SSRS\)](#).
- **Writing expressions.** For more information, see [Expression Uses in Reports \(Report Builder and SSRS\)](#), and [Expression Examples \(Report Builder and SSRS\)](#).
- **Controlling sort order.** To control the sort order, you define sort expressions on the data region. To control sort order for rows and columns associated with a group, you define sort expressions on the group, including the details groups. You can also add interactive sort buttons to enable the user to sort a tablix data region or its groups. For more information, see [Sort Data in a Data Region \(Report Builder and SSRS\)](#).
- **Displaying a message when there is no data.** When no data exists for a report dataset at run time, you can write your own message to display in place of the data region. For more information, see [Set a No Data Message for a Data Region \(Report Builder and SSRS\)](#).
- **Conditionally hiding data.** To conditionally control whether to show or hide a data region or parts of a data region, you can set the Hidden property to **True** or to an expression. Expressions can include references to report parameters. You can also specify a toggle item, so that user can decide to display detail data. For more information, see [Drilldown Action \(Report Builder and SSRS\)](#).
- **Merging cells.** Multiple contiguous cells within a table can be combined into a single cell. This is known as a column span or a cell merge. Cells can only be combined horizontally or vertically. When you merge cells, only the data in the first cell is preserved. Data in other cells is removed. Merged cells can be split into their original columns. For more information, see [Merge Cells in a Data Region \(Report Builder and SSRS\)](#).

Controlling Tablix Data Region Position and Expansion on a Page

The following features help control the way a tablix data region displays in a rendered report:

- **Controlling the position of a tablix data region in relation to other report items.** A tablix data region can be positioned above, next to, or below other report items on the report design surface. At run time, Reporting Services expands the tablix data region as needed for the data retrieved for the linked dataset, moving peer report items aside as needed. To anchor a tablix next to another report item, make the report items peers and adjust their relative positions. For more information, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

- **Changing the Expansion Direction.** To control whether a tablix data region expands across the page from left-to-right (LTR) or from right-to-left (RTL), use the Direction property, which can be accessed through the Properties window. For more information, see [Rendering Data Regions \(Report Builder and SSRS\)](#).

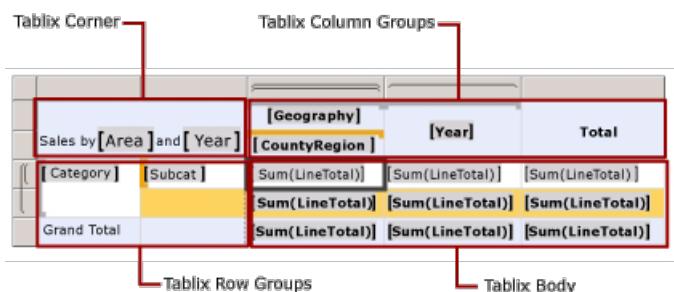
Controlling How a Tablix Data Region Renders on a Page

The following list describes ways that you can help control how a tablix data region appears in a report:

- **Controlling paging.** To control the amount of data that displays on each report page, you can set page breaks on data regions. You can also set page breaks on groups. Page breaks can affect the on-demand rendering performance by reducing the amount of data that needs to be processed on each page. For more information, see [Pagination in Reporting Services \(Report Builder and SSRS\)](#) and [Add a Page Break \(Report Builder and SSRS\)](#).
- **Displaying data on either side of row headers.** You are not limited to displaying row headers on the side of a tablix data region. You can move the row headers between columns, so that columns of data appear before the row headers. To do this, modify the GroupsBeforeRowHeaders property for the matrix. You can access this property through the Properties window. The value for this property is an integer; for example, a value of 2 will display two groups instances of data region column data before displaying the column containing the row headers.

Controlling How Tablix Row and Column Groups Render

To control how a tablix data region groups render depends on the group structure. A tablix data region can have four areas, as shown in the following figure:



The row group area and column group area contain group headers. When a tablix data region has group headers, you control how rows and columns repeat by setting properties on the **General** page of the **Tablix Properties** dialog Box.

If a tablix data region has only a tablix body area, there are no group headers. There are only static and dynamic tablix members. A static member displays once in relation to a tablix row or column group. A dynamic member repeats once for every unique group value. For example, in a tablix data region that displays a sales order, the column names in the sales order can be displayed on a static row member. Each line in the sales order is displayed on a dynamic row member.

You can help control how a tablix member renders by setting properties in the Properties pane. For more information, see "Advanced mode" in [Grouping Pane \(Report Builder\)](#).

The following list describes ways that you can help control how a tablix data region appears in a report:

- **Repeating row and column headers on multiple pages.** You can display row and column headers on each page that a tablix data region spans. For more information, see [Display Row and Column Headers on Multiple Pages \(Report Builder and SSRS\)](#).
- **Keeping row and column headers in view when scrolling.** You can control whether to keep the row and column headers in view when you scroll a report using a browser. For more information, see [Keep](#)

[Headers Visible When Scrolling Through a Report \(Report Builder and SSRS\)](#).

For more information about how exporting a report to different formats affects the way a tablix data region renders on a page, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

See Also

[Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

[Nested Data Regions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

[Controlling Page Breaks, Headings, Columns, and Rows \(Report Builder and SSRS\)](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables \(Report Builder and SSRS\)](#)

[Create a Matrix](#)

[Create Invoices and Forms with Lists](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Controlling Row and Column Headings (Report Builder and SSRS)

3/24/2017 • 8 min to read • [Edit Online](#)

A table, matrix, or list data region can span multiple pages horizontally or vertically. You can specify whether to repeat row or column headings on each page. In an interactive renderer such as Report Manager or report preview, you can also specify whether to freeze row or column headings to keep them in view when you scroll across or down a report. In a table or matrix, the first row usually contains column headings that label data in each column; the first column usually contains row headings that label the data in each row. For nested groups, you might want to repeat the initial set of row and column headings that contain group labels. By default, a list data region does not include headings.

How you control whether headings repeat or freeze depends on the following:

- For column headings that repeat at the top of each page:
 - Whether the table or matrix has a column group area that expands horizontally.
 - Whether you want to control all rows that are associated with column groups as a unit.
- For row headings that repeat along the side of each page:
 - Whether the table or matrix has a row group area that expands vertically. Row headings are supported only for row groups with a row group header.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Rows and Columns in a Tablix Data Region

A table or matrix is a template for the underlying tablix data region. A tablix data region has four possible areas: the row group area that controls rows that expand down a report, the column group area that controls columns that expand across a report, the body that displays data, and the corner. To understand where to set properties to control repeating or freezing headers, it helps to understand that there are two representations for a tablix data region:

- **In the report definition** Each row or column in a tablix data region definition is a tablix member of a specific row or column group. A tablix member is static or dynamic. A static tablix member contains labels or subtotals and repeats once per group. A dynamic tablix member contains group values and repeats once per unique value of a group, also known as a group instance.
- **On the design surface** On the design surface, dotted lines divide a tablix data region into the four areas. Each cell in a tablix data region area is organized into rows and columns. Rows and columns are associated with groups, including the details group. For a selected tablix data region, row and column handles and highlight bars indicate group membership. Cells in the row group or column group area represent group headers for tablix members. A single row or column can be associated with multiple groups.

For more information, see [Tablix Data Region \(Report Builder and SSRS\)](#) and [Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#).

For tablix data regions with row group or column group areas, control the associated rows and columns by setting properties on tablix data region. For all other cases, control the rows and columns by setting properties in the Properties pane for the selected tablix member. For step-by-step instructions, see [Display Row and Column Headers on Multiple Pages \(Report Builder and SSRS\)](#) and [Keep Headers Visible When Scrolling Through a Report \(Report Builder and SSRS\)](#).

Examples

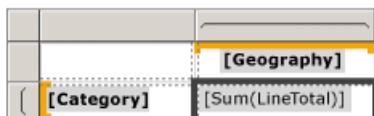
The most common examples of tablix data regions are for a matrix, a table with no groups, and a table with a row group and a row group header, and a table with a row group but no row group header. To control how to repeat or freeze headers, you must determine if the rows or columns that you want to control are associated with a group header in the row groups or column groups area.

The following sections provide examples for common layouts for a tablix data region:

- [Matrix](#)
- [Table with no groups](#)
- [Table with row groups and row group area](#)
- [Table with row groups but no row group area](#)

Matrix

By default, a simple matrix has one row group and one column group. The following figure shows a matrix with a row group that is based on Category and a column group that is based on Geography:



The dotted lines show the four tablix areas. The row group area has a row group header that controls the category labels in the first column. Similarly, the column group area has a column group header that controls the geography labels in the first row. In preview, as the matrix expands across the page, the first row displays the column headings, as shown in the following figure:

	North America	Europe	Pacific
Accessories	\$384,057	\$148,057	\$18,599
Bikes	\$51,990,891	\$11,503,109	\$1,186,388
Clothing	\$1,251,999	\$443,035	\$33,683
Components	\$8,601,244	\$2,746,093	\$183,142

To repeat or freeze column headings in the first row, set properties for column headers on the tablix data region. Column headers for nested column groups are automatically included.

To repeat or freeze row headings in the first column, set properties for row headers on the tablix data region. Row headers for nested row groups are automatically included.

[Return to top](#)

Table with no row groups

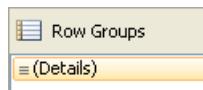
By default, a simple table with no groups does include the details group. The following figure shows a table that displays category, order number, and sales data:

	Category	Order	Sales
=	[Category]	[Order]	[Sales]

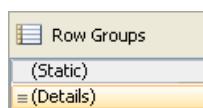
There are no dotted lines because the table consists only of the tablix body area. The first row displays column headers, and represents a static tablix member that is not associated with a group. The second row displays detail data, and represents a dynamic tablix member that is associated with the details group. The following figure shows the table in preview:

Category	Order	Sales
Accessories	SO46604	\$245.01
Accessories	SO46605	\$84.11
Accessories	SO46608	\$589.83

To repeat or freeze column headings, set properties on the tablix member for static row that is part of the tablix data region definition. To select the static row, you must use the Advanced mode of the Grouping pane. The following figure shows the Row Groups pane:



In Advanced mode, the following figure shows the static and dynamic tablix members for the row groups in the table:



To repeat or freeze column headings for the tablix member, select the static row that is labeled **(Static)**. The properties pane displays the properties for the selected tablix member. By setting properties for this tablix member, you can control how the first row repeats or stays in view.

[Return to top](#)

Table with row groups and a row group area

If you add a row group to a simple table, a row group area is added to the table on the design surface. The following figure shows a table with a row group that is based on Category:

Category	Order	Sales	
[Category]	[Order]	[Sales]	

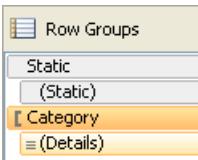
The dotted lines show the tablix row groups area and the tablix body area. The row group area has a row group header but no column group header. The following figure shows this table in preview:

Category	Order	Sales
Accessories	SO46604	\$245.01
	SO46605	\$84.11
	SO46608	\$589.83

To repeat or freeze column headings, use the same approach as the previous example. The following figure shows the default view of the Row Groups pane:



Use the **Advanced** mode of the Row Groups pane to display the tablix members, as shown in the following figure:



For tablix members are listed: **Static**, **(Static)**, Category, and **(Details)**. A tablix member that includes parentheses () indicates that there is no corresponding group header. To repeat or freeze column headings, select the top Static tablix member, and set properties in the Properties pane.

[Return to top](#)

Table with row groups and no row group area

A table can have row groups but no row groups area in several ways. Two possible ways for this to happen include:

- Start with a table with row groups and a row group area and delete the columns for the row group area. Delete the columns only and not the groups. For example, you might want to control the table format to be a simple grid.
- Upgrade a report that was created for a previous RDL version, before tablix data regions were introduced.

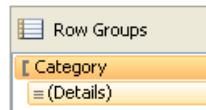
The following figure shows a table with a row group but no row group area on the design surface:

	Category	Order	Sales
	[Category]		[Sum(Sales)]

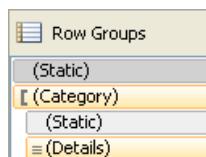
The table has three rows. The first row contains column headers. The second row contains the group value and subtotals. The third row contains the detail data. There are no dotted lines because there is only a tablix body area. The following figure shows this table in preview:

Category	Order	Sales
Accessories		\$1,208,192.06
	SO46604	\$245.01
	SO46605	\$84.11
	SO46608	\$689.83

To control how the rows repeat or stay in view, you must set properties on the tablix member for each row. In default mode, there is no difference between this example and the previous example for a table with a row group and a group header. The following figure shows the Grouping pane in default mode for this table:



However, in advanced mode, this layout structure shows a different set of tablix members. The following figure shows the Grouping pane in advanced mode for this table:



In the Row Groups pane, the following tablix members are listed: **Static**, **(Category)**, **(Static)**, and **(Details)**. To repeat or freeze column headings, select the top **(Static)** tablix member, and set properties in the Properties pane.

[Return to top](#)

Renderer Support for Repeating or Freezing Headers

Renderers vary in support for repeating or freezing headers.

Renderers that use physical pages (PDF, Image, Print) support the following features:

- Repeat row headers when a tablix data region expands horizontally across multiple pages.
- Repeat column headers when a tablix data region expands vertically down multiple pages.

In addition, renderers that use soft page breaks (Report Manager, report preview, or the report viewer control) support the following features:

- Keep row headers in view when you scroll horizontally across a report.
- Keep column headers in view when you scroll vertically down a report.

For more information, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

See Also

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Pagination in Reporting Services \(Report Builder and SSRS\)](#)

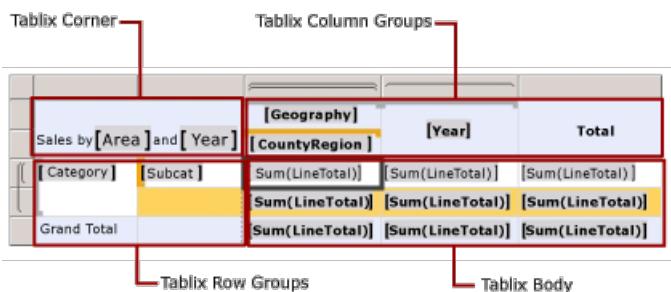
[Export Reports \(Report Builder and SSRS\)](#)

Understanding Groups (Report Builder and SSRS)

3/24/2017 • 8 min to read • [Edit Online](#)

In a Reporting Services paginated report, a group is a named set of data from the report dataset that is bound to a data region. Basically, a group organizes a view of a report dataset. All groups in a data region specify different views of the same report dataset.

To help visualize what a group is, refer to the following figure that shows the tablix data region in Preview. In this figure, the row groups categorize the dataset by product type and the column groups categorize the dataset by geographic region and year.



The following sections help describe the various aspects of groups.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

What Makes a Group?

A group has a name and a set of group expressions that you specify. The set of group expressions can be a single dataset field reference or a combination of multiple expressions. At runtime, group expressions are combined, if the group has multiple expressions, and applied to data in a group. For example, you have a group that uses a date field to organize the data in the data region. At run time, data is organized by date, and then displayed with totals other dataset values for each date.

When Do I Create Groups?

In most cases, Report Builder and Report Designer automatically create a group for you when you design a data region. For a table, matrix, or list, groups are created when you drop fields on the Grouping pane. For a chart, groups are created when you drop fields on the chart drop-zones. For a gauge, you must use the gauge properties dialog box. For a table, matrix, or list, you can also create a group manually. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#). For an example of how to add groups when you create a report, see [Tutorial: Creating a Basic Table Report \(Report Builder\)](#) or [Create a Basic Table Report \(SSRS Tutorial\)](#).

How Can I Modify a Group?

After you create a group, you can set data region-specific properties, such as filter and sort expressions, page breaks, and group variables to hold scope-specific data. For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).

To modify an existing group, open the appropriate group properties dialog box. You can change the name of the group. Also, you can specify group expressions based on a single field or multiple fields, or on a report parameter that specifies a value at run time. You can also base a group on a set of expressions, such as the set of expressions that specify age ranges for demographic data. For more information, see [Group Expression Examples \(Report Builder and SSRS\)](#).

NOTE

If you change the name of a group, you must manually update any group expressions that refer to the previous name of the group.

How are Groups Organized?

Understanding group organization can help you design data regions that display different views of the same data by specifying identical group expressions.

Groups are internally organized as members of one or more hierarchies for each data region. A group hierarchy has parent/child groups that are nested and can have adjacent groups.

If you think of the parent/child groups as a tree structure, each group hierarchy is forest of tree structures. A tablix data region includes a row group hierarchy and a column group hierarchy. Data associated with row group members expands horizontally across the page and data associated with column group members expands vertically down the page. The Grouping pane displays row group and column group members for the currently selected tablix data region on the design surface. For more information, see [Grouping Pane \(Report Builder\)](#).

A chart data region includes a category group hierarchy and a series group hierarchy. Category group members are displayed on the category axis and series group members are displayed on the series axis.

Although typically not needed for gauge data regions, groups do let you specify how to group data to aggregate on the gauge.

What Types of Groups are Available per Data Region?

Data regions that expand as a grid support different groups than data regions that display summary data visually. Thus, a tablix data region, and the tables, lists, and matrices that are based on the tablix data region, support different groups than a chart or gauge. The following sections discuss the type of and purpose for grouping in each type of data region.

NOTE

Although groups have different names in different data regions, the principles behind how you create and use groups are the same. When you create a group for a data region, you specify a way to organize the detail data from the dataset that is linked to the data region. Each data region supports a group structure on which to display grouped data.

Groups in a Tablix Data Region: Details, Row, and Column Groups

As shown earlier in this topic, a tablix data region enables you to organize data into groups by rows or columns. However, row and column groups are not the only groups available in a tablix data region. This data region can have the following types of groups:

- **Details Group** The Details group consists of all data from a report dataset after Report Builder or Report Designer apply dataset and data region filters. Thus, the Details group is the only group that has no group expression.

Basically, the details group specifies the data that you would see when you run a dataset query in a query

designer. For example, you have a query that retrieves all columns from a sales order table. Thus, the data in this detail group includes all the values for every row for all the columns in the table. The data in this detail group also includes values for any calculated dataset fields that you have created.

NOTE

The data in a Detail group can also include server aggregates, which are aggregates that are calculated on the data source and retrieved in your query. By default, Report Builder and Report Designer treat server aggregates as detail data unless your report includes an expression that uses the Aggregate function. For more information, see [Aggregate](#).

By default, when you add a table or list to your report, Report Builder and Report Designer automatically create the Details group for you, and adds a row to display the detail data. By default, when you add dataset fields to cells in this row, you see simple expressions for the fields, for example, [Sales]. When you view the data region, the details row repeats once for every value in the result set.

- **Row groups and column groups** You can organize data into groups by rows or columns. Row groups expand vertically on a page. Column groups expand horizontally on a page. Groups can be nested, for example, group first by [Year], then by [Quarter], then by [Month]. Groups can also be adjacent, for example, group on [Territory] and independently on [ProductCategory].

When you create a group for a data region, Report Builder and Report Designer automatically add rows or columns to the data region and use these rows or columns to display group data.

- **Recursive hierarchy groups** A recursive hierarchy group organizes data from a single report dataset that includes multiple levels. For example, a recursive hierarchy group could display an organization hierarchy, for example, [Employee] that reports to [Employee]. Reporting Services provides group properties and built-in functions to enable you to create groups for this kind of report data. For more information, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

The following list summarizes the way you work with groups for each data region:

- **Table** Define nested row groups, adjacent row groups, and recursive hierarchy row groups (such as for an organizational chart). By default, a table includes a details group. Add groups by dragging dataset fields to the Grouping pane for a selected table.
- **Matrix** Define nested row and column groups, and adjacent row and column groups. Add groups by dragging dataset fields to the Grouping pane for a selected matrix.
- **List** By default, supports the details group. Typical use is to support one level of grouping. Add groups by dragging dataset fields to the Grouping pane for a selected list.

After you add a group, the row and column handles of the data region change to reflect group membership. When you delete a group, you have the choice between deleting the group definition only or deleting the group and all its associated rows and columns. For more information, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder and SSRS\)](#).

To limit the data to display or use in calculations for detail or group data, set filters on the group. For more information, see [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#).

By default, when you create a group, the sort expression for the group is the same as the group expression. To change the sort order, change the sort expression. For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).

Understanding Group Membership for Tablix Cells

Cells in a row or column of a tablix data region can belong to multiple row and column groups. When you define

an expression in the text box of a cell that uses an aggregate function (for example, `=Sum(Fields!FieldName.Value)`), the default group scope for a cell is the inner most child group to which it belongs. When a cell belongs to both row and column groups, the scope is both innermost groups. You can also write expressions that calculate aggregate subtotals scoped to a group relative to another set of data. For example, you can calculate the percent of a group relative to the column group or to all data for the data region (such as

`=Sum(Fields!FieldName.Value)/Sum(Fields!FieldName.Value,"ColumnGroup")`). For more information, see [Tablix Data Region \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

See Also

[Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#)

[Add a Total to a Group or Tablix Data Region \(Report Builder and SSRS\)](#)

[Sort Data in a Data Region \(Report Builder and SSRS\)](#)

[Drilldown Action \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Creating Recursive Hierarchy Groups (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

To display recursive data in Reporting Services paginated reports (where the relationship between parent and child is represented by fields in the dataset), set the data region group expression based on the child field and set the Parent property based on the parent field.

Displaying hierarchical data is a common use for recursive hierarchy groups, for example, employees in an organizational chart. The dataset includes a list of employees and the managers, where the manager names also appear in the list of employees.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Creating Recursive Hierarchies

To build a recursive hierarchy in a tablix data region, you must set the group expression to the field that specifies the child data and the Parent property of the group to the field that specifies the parent data. For example, for a dataset that includes fields for employee ID and manager ID where employees report to managers, set the group expression to employee ID and the Parent property to manager ID.

A group that is defined as a recursive hierarchy (that is, a group that uses the Parent property) can have only one group expression. You can use the **Level** function in text box padding to indent employee names based on their level in the hierarchy.

For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#) and [Create a Recursive Hierarchy Group \(Report Builder and SSRS\)](#).

Aggregate Functions that support Recursion

You can use Reporting Services aggregate functions that accept the parameter *Recursive* to calculate summary data for a recursive hierarchy. The following functions accept **Recursive** as a parameter: [Sum](#), [Avg](#), [Count](#), [CountDistinct](#), [CountRows](#), [Max](#), [Min](#), [StDev](#), [StDevP](#), [Sum](#), [Var](#), and [VarP](#). For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

See Also

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Aggregate Functions Reference \(Report Builder and SSRS\)](#)

[Tables \(Report Builder and SSRS\)](#)

[Matrices \(Report Builder and SSRS\)](#)

[Lists \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Add a Details Group (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a Reporting Services paginated report, the detail data from a report dataset is specified as a group with no group expression. Add a detail group to an existing tablix data region when you want to display the detail data for a matrix, add back detail data that you have deleted from a table or list, or to add additional detail groups. For more information about groups, see [Understanding Groups \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a details group to a tablix data region

1. On the design surface, click anywhere in a tablix data region to select it. The Grouping pane displays the row and column groups for the selected data region.
2. In the Grouping pane, right-click a group that is an innermost child group. Click **Add Group**, and then click **Child Group**. The **Tablix Group** dialog box opens.
3. In **Group expression**, leave the expression blank. A details group has no expression.
4. Select **Show detail data**.
5. Click **OK**.

A new details group is added as a child group in the Grouping pane, and the row handle for the group you selected in step 1 displays the details group icon. For more information about handles, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#).

See Also

[Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#)

[Understanding Groups \(Report Builder and SSRS\)](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables \(Report Builder and SSRS\) Matrices \(Report Builder and SSRS\)](#)

[Lists \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Add a Total to a Group or Tablix Data Region (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

In a Reporting Services paginated report, you can add totals in a tablix data region for a group or for the entire data region. By default, a total is the sum of the numeric, non-null data in a group or in the data region, after filters are applied. To add totals for a group, click **Add Total** on the shortcut menu for the group in the Grouping pane. To add totals for an individual cell in the tablix body area, click **Add Total** on the shortcut menu for the cell. The **Add Total** command is context-sensitive and enabled only for numeric fields. Depending on the tablix cell that you select, you can add a total for a single cell by selecting a cell in the tablix body area or for the entire group by selecting a cell in the tablix row group area or the tablix column group area. For more information about tablix areas, see [Tablix Data Region \(Report Builder and SSRS\)](#).

After you add a total, you can change the default function Sum to a different aggregate function from the list of built-in report functions. For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#). You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a total for an individual value in the tablix body area

- In the tablix data region body area, right-click the cell where you want to add the total. The cell must contain a numeric field. Point to **Add Total**, and then click **Row** or **Column**.

A new row or column outside the current group is added to the data region, with a default total for the field in the cell you clicked.

If the tablix data region is a table, a row is automatically added.

To add totals for a row group

- In the tablix data region row group area, right-click a cell in the row group area for which you want totals, point to **Add Total**, and then click **Before** or **After**.

A new row outside the current group is added to the data region, and then a default total is added for each numeric field in the row.

To add totals for a column group

- In the tablix data region row group area, right-click a cell in the column group area for which you want totals, then point to **Add Total**, and click **Before** or **After**.

A new column outside the current group is added to the data region, and then a default total is added for each numeric field in the column.

See Also

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Change an Item Within a Cell (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In Reporting Services paginated reports, only a non-container item, such as a text box, line, or image, can be replaced by a new report item. For example, you can drag a table into a text box to replace the text box with a table.

If the cell contains a container item such as a rectangle, list, table, or matrix, the new item is added to the containing item instead of replacing it. To replace a container item with a new item, delete the container. Deleting the container item replaces it with a text box, which you can then replace with another item.

By default, all cells in a table, matrix, or list data region contain a text box.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To change an item within a cell

- On the **Insert** tab, in the **Data Regions** or **Report Items** group, click the item that you want to add to the report, and then click the report. The item is added to the report.

NOTE

The **Image Properties** dialog box opens when you drag an image report item to a cell, where you can set properties such as the source of the image before the image is added to the cell.

See Also

[Images, Text Boxes, Rectangles, and Lines \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Change Row Height or Column Width (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

When you set a row height, you are specifying the maximum height for the row in the rendered report. However, by default, text boxes in the row are set to grow vertically to accommodate their data at run-time, and this can cause a row to expand beyond the height that you specify. To set a fixed row height, you must change the text box properties so they do not automatically expand.

When you set a column width, you are specifying the maximum width for the column in the rendered report. Columns do not automatically adjust horizontally to accommodate text.

If a cell in a row or column contains a rectangle or data region, the minimum height and width of the cell is determined by the height and width of the contained item. For more information, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To change row height by moving row handles

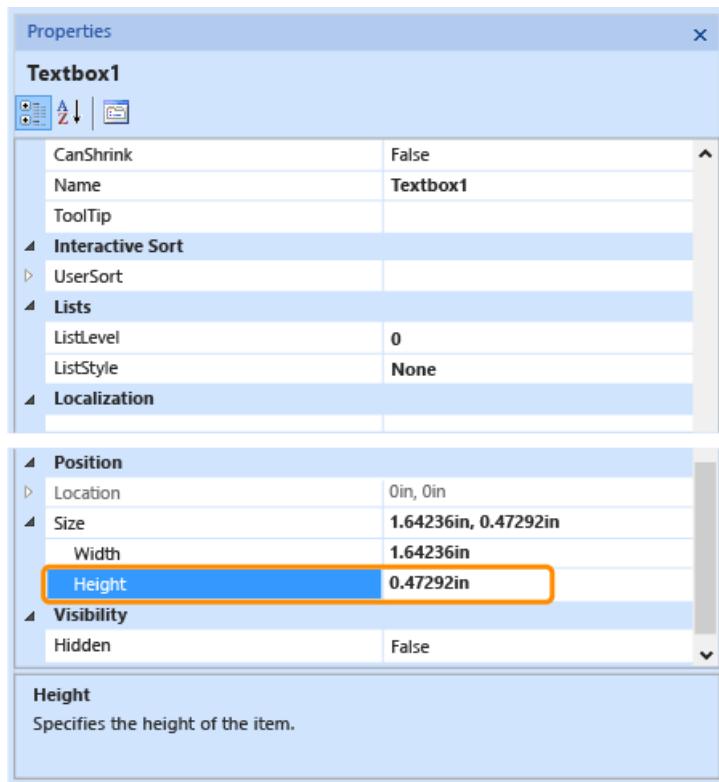
1. In Design view, click anywhere in the tablix data region to select it. Gray row handles appear on the outside border of the tablix data region.
2. Hover over the row handle edge that you want to expand. A double-headed arrow appears.
3. Click to grab the edge of the row and move it higher or lower to adjust the row height.

To change row height by setting cell properties

1. In Design view, click a cell in the table row.

	Product ID	Name	Color
≡	[Product ID]	[Name]	[Color]

2. In the **Properties** pane that displays, modify the **Height** property, and then click anywhere outside the **Properties** pane.



To prevent a row from automatically expanding vertically

1. In Design view, click anywhere in the tablix data region to select it. Gray row handles appear on the outside border of the tablix data region.
2. Click the row handle to select the row.
3. In the Properties pane, set CanGrow to **False**.

NOTE

If you cannot see the Properties pane, from the **View** menu, click **Properties**.

To change column width

1. In Design view, click anywhere in the tablix data region to select it. Gray column handles appear on the outside border of the tablix data region.
2. Hover over the column handle edge that you want to expand. A double-headed arrow appears.
3. Click to grab the edge of the column and move it left or right to adjust the column width.

See Also

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#)

[Tables \(Report Builder and SSRS\)](#)

[Matrices \(Report Builder and SSRS\)](#)

[Lists \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Insert or Delete a Column (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can add or delete columns in a tablix data region in a Reporting Services paginated report. The tablix data region can be a table, a matrix, or a list. The following procedures do not apply to the chart and gauge data regions.

In a tablix data region, you can add columns that are associated with a group (inside a group) or that are not associated with a group (outside a group). A column that is inside a group repeats once per unique group value. For example, a column inside a group based the value in a data column that contains color names, repeats once per distinct color name. For nested groups, a column can be outside the child group but inside the parent group. In this case, the row repeats once for each unique value in the parent group.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To select a data region so that the row and column handles appear

- In Design view, click the upper left corner of the tablix data region, so that column and row handles appear above and next to it.

For more information about data region areas, see [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#).

To insert a column in a selected data region

- Right-click a column handle where you want to insert a column, click **Insert Column**, and then click **Left** or **Right**.
-- or --
- Right-click a cell in the data region where you want to insert a row, click **Insert Column**, and then click **Left** or **Right**.

To delete a column from a selected data region

- Select the column or columns that you want to delete, right-click the handle for one of the columns you selected, and then click **Delete Columns**.
-- or --
- Right-click a cell in the data region where you want to delete a column, and then click **Delete Columns**.

To insert a column in a group in a selected data region

- Right-click a column group cell in the column group area of a tablix data region where you want to insert a column, click **Insert Column**, and then click **Left - Outside Group**, **Left - Inside Group**, **Right - Inside Group**, or **Right - Outside Group**.

A column is added either inside or outside the group represented by the column group cell you clicked on.

To delete a column from a group in a selected data region

- Right-click a column group cell in the column group area of a tablix data region where you want to delete a column, and then click **Delete Columns**.

See Also

[Understanding Groups \(Report Builder and SSRS\)](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables \(Report Builder and SSRS\)](#)

[Matrices \(Report Builder and SSRS\)](#)

[Lists \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Insert or Delete a Row (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can add or delete rows in a tablix data region in a Reporting Services paginated report. The tablix data region can be a table, a matrix, or a list. The following procedures do not apply to the chart and gauge data regions.

In a tablix data region, you can add rows that are associated with a group (inside a group) or that are not associated with a group (outside a group). A row that is inside a group repeats once per unique group value. For example, a row inside a group based on the value in a data column that contains color names, repeats once per distinct color name. For nested groups, a row can be outside the child group but inside the parent group. In this case, the row repeats once for each unique value in the parent group.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To select a data region so the row and column handles appear

- In Design view, click the upper left corner of the tablix data region so that column and row handles appear above and next to it.

For more information about data region areas, see [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#).

To insert a row in a selected data region

- Right-click a row handle where you want to insert a row, click **Insert Row**, and then click **Above** or **Below**.
 - or -
- Right-click a cell in the data region where you want to insert a row, click **Insert Row**, and then click **Above** or **Below**.

To delete a row from a selected data region

- Select the row or rows that you want to delete, right-click the handle for one of the rows you selected, and then click **Delete Rows**.
 - or -
- Right-click a cell in the data region where you want to delete a row, and then click **Delete Rows**.

To insert a row in a group in a selected data region

- Right-click a row group cell in the row group area of a tablix data region where you want to insert a row, click **Insert Row**, and then click **Above - Outside Group**, **Above - Inside Group**, **Below - Inside Group**, or **Below - Outside Group**.

A row is added either inside or outside the group represented by the row group cell you clicked on.

To delete a row from a group in a selected data region

- Right-click a row group cell in the row group area of a tablix data region where you want to delete a row, and then click **Delete Rows**.

See Also

- [Tablix Data Region \(Report Builder and SSRS\)](#)
- [Understanding Groups \(Report Builder and SSRS\)](#)
- [Tables \(Report Builder and SSRS\)](#)
- [Matrices \(Report Builder and SSRS\)](#)
- [Lists \(Report Builder and SSRS\)](#)
- [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Merge Cells in a Data Region (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a Reporting Services paginated report, you can merge cells in a data region to combine cells, improve data region appearance, or provide spanning labels for column groups and row groups.

You can only merge cells within each area of a data region: corner, column headers, group definition (or row headers), and body. You can't merge cells that cross area boundaries. For example, you can't merge a cell in the data region corner area with a cell in the row group area. Read more about [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To merge cells in a data region

1. In the data region on the report design surface, click the first cell to merge. Holding the left mouse button down, drag vertically or horizontally to select adjacent cells. The selected cells are highlighted.
2. Right-click the selected cells and select **Merge Cells**. The selected cells are combined into a single cell.
3. Repeat steps 1 and 2 to merge other adjacent cells in a data region.

See Also

[Tablix Data Region](#)

[Tables \(Report Builder and SSRS\)](#)

[Create a Matrix](#)

[Create Invoices and Forms with Lists](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Create a Recursive Hierarchy Group (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

In Reporting Services paginated reports, a recursive hierarchy group organizes data from a single report dataset that includes multiple hierarchical levels, such as the report-to structure for manager-employee relationships in an organizational hierarchy.

Before you can organize data in a table as a recursive hierarchy group, you must have a single dataset that contains all the hierarchical data. You must have separate fields for the item to group and for the item to group by. For example, a dataset where you want to group employees recursively under their manager might contain a name, an employee name, an employee ID, and a manager ID.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To create a recursive hierarchy group

1. In Design view, add a table, and drag the dataset fields to display. Typically, the field that you want to show as a hierarchy is in the first column.
2. Right-click anywhere in the table to select it. The Grouping pane displays the details group for the selected table. In the Row Groups pane, right-click **Details**, and then click **Edit Group**. The **Group Properties** dialog box opens.
3. In **Group expressions**, click **Add**. A new row appears in the grid.
4. In the **Group on** list, type or select the field to group.
5. Click **Advanced**.
6. In the **Recursive Parent** list, enter or select the field to group on.
7. Click **OK**.

Run the report. The report displays the recursive hierarchy group, although there is no indent to show the hierarchy.

To format a recursive hierarchy group with indent levels

1. Click the text box that contains the field to which you want to add indent levels to display a hierarchy format. The properties for the text box appear in the Properties pane.

NOTE

If you do not see the Properties pane, click **Properties** on the **View** tab.

2. In the Properties pane, expand the **Padding** node, click **Left**, and from the drop-down list, select **<Expression...>**.

3. In the Expression pane, type the following expression:

```
=CStr(2 + (Level()*10)) + "pt"
```

The Padding properties all require a string in the format *nnyy*, where *nn* is a number and *yy* is the unit of measure. The example expression builds a string that uses the **Level** function to increase the size of the padding based on recursion level. For example, a row that has a level of 1 would result in a padding of $(2 + (1*10))=12\text{pt}$, and a row that has a level of 3 would result in a padding of $(2 + (3*10))=32\text{pt}$. For information about the **Level** function, see [Level](#).

4. Click **OK**.

Run the report. The report displays a hierarchical view of the grouped data.

See Also

- [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#)
- [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)
- [Aggregate Functions Reference \(Report Builder and SSRS\)](#)
- [Tables \(Report Builder and SSRS\)](#)
- [Matrices \(Report Builder and SSRS\)](#)
- [Lists \(Report Builder and SSRS\)](#)
- [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Add or Delete a Group in a Data Region (Report Builder and SSRS)

3/24/2017 • 5 min to read • [Edit Online](#)

In Reporting Services paginated reports, add a group to a data region when you want to organize data by a specific value or set of expressions, for display and calculations. A group has a name and an expression that identifies which data from a dataset belongs to the group. For more information about groups, see [Understanding Groups \(Report Builder and SSRS\)](#).

In a tablix data region, click in the table, matrix, or list to display the Grouping pane. Drag dataset fields to the Row Group and Column Group pane to create parent or child groups. Right-click an existing group to add an adjacent group. By definition, the details group is the innermost group and can only be added as a child group. Right-click an existing group to delete it. Rows and columns on which to display group values are automatically added for you. For more information, see [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#).

In a Chart data region, click in the chart to display the drop-zones. Create groups by dragging dataset fields to the category and series drop zones. To add nested groups, add multiple fields to the drop-zone.

Groups are not defined in a gauge by default. The default behavior for the gauge is to aggregate all values in the specified field into one value that is displayed on the gauge. For more information, see [Gauges \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a parent or child row or column group to a tablix data region

1. Drag a field from the **Report Data** pane to the **Row Groups** pane or the **Column Groups** pane.

NOTE

If you do not see the Grouping pane, on the View tab, click **Grouping**.

2. Drop the field above or below the group hierarchy using the guide bar to place the group as a parent group or a child group to an existing group.

The group is added with a default name, group expression, and sort expression that is based on the field name.

To add an adjacent row or column group to a tablix data region

1. In the Grouping pane, right-click a group that is a peer to the group that you want to add. Click **Add Group**, and then click **Adjacent Before** or **Adjacent After** to specify where to add the group. The **Tablix Group** dialog box opens.
2. In **Name**, type a name for the group.
3. In **Group expression**, type an expression or click the expression button (**fx**) to create an expression.

4. Click **OK**.

A new group is added to the Grouping pane and a row or column on which to display group values is added to the tablix data region on the design surface.

To add a details group to a tablix data region

1. In the Grouping pane, right-click a group that is the innermost child group, but not the **Details** group. Click **Add Group**, and then click **Child Group**. The **Tablix Group** dialog box opens.
2. In **Group expression**, leave the expression blank. A details group has no expression.
3. Select **Show detail data**.
4. Click **OK**.

A new details group is added as a child group in the Grouping pane, and the row handle for the group you selected in step 1 displays the details group icon. For more information about handles, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#).

To edit a row or column group in a tablix data region

1. On the report design surface, click anywhere in the tablix data region to select it. The Grouping pane displays the row and column groups.
2. Right-click the group, and then click **Group Properties**.
3. In **Name**, type the name of the group.
4. In **Group expressions**, type or select a simple expression, or click the Expression (**fx**) button to create a group expression.
5. Click **Add** to create additional expressions. All expressions you specify are combined using a logical AND to specify data for this group.
6. (Optional) Click **Page Breaks** to set page break options.
7. (Optional) Click **Sorting** to select or type expressions that specify the sort order for values in the group.
8. (Optional) Click **Visibility** to select the visibility options for the item.
9. (Optional) Click **Filters** to set filters for this group.
10. (Optional) Click **Variables** to define variables scoped to this group and accessible from any child groups.
11. Click **OK**.

To delete a group from a tablix data region

1. In the Grouping pane, right-click the group, and then click **Delete Group**.
2. In the **Delete Group** dialog box, select one of the following options:
 - **Delete group and related rows and columns** Choose this option to delete the group definition and all related rows that display group data. For the details group, if the same row belongs to both detail and group data, only the detail data rows are deleted.
 - **Delete group only** Choose this option to keep the structure of the tablix data region the same and delete only the group definition.
3. Click **OK**.

To delete a details group from a tablix data region

1. In the Grouping pane, right-click the details group, and then click **Delete Group**.
2. In the **Delete Group** dialog box, select one of the following options:
 - **Delete group and related rows and columns** Choose this option to delete the group definition and all related rows that display group data. For the details group, if the same row belongs to both detail and group data, only the detail data rows are deleted.
 - **Delete group only** Choose this option to keep the structure of the tablix data region the same and delete only the group definition.
3. Click **OK**.

The details group is deleted.

NOTE

Verify that after you remove a details row, the expression in each cell specifies an aggregate expression where appropriate. If necessary, edit the expression to specify aggregate functions as needed.

See Also

[Report and Group Variables Collections References \(Report Builder and SSRS\)](#)

[Group Expression Examples \(Report Builder and SSRS\)](#)

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables \(Report Builder and SSRS\)](#)

[Matrices \(Report Builder and SSRS\)](#)

[Lists \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Display Headers and Footers with a Group (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

You can help control whether a static row, such as a group header or footer, renders with dynamic rows that are associated with a group in a tablix data region.

To repeat all the column headings or row headings on multiple pages, you can set properties for the tablix data region. For more information, see [Display Row and Column Headers on Multiple Pages \(Report Builder and SSRS\)](#).

To control the rendering behavior for dynamic rows and columns that are associated with nested groups, or for static rows and columns that are associated with labels or subtotals, you must set properties for the tablix member. A tablix member represents a static or dynamic row or column. A static member repeats once. For example, a grand total row is a static row. A dynamic member repeats once for each group instance. For example, a row that is associated with a group that has the group expression [Territory] repeats once for each unique value for territory. For more information about tablix members, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#).

You can select a tablix member in the Grouping pane and set the properties **KeepWithGroup**, **KeepTogether**, and **RepeatOnNewPage** in the Properties pane. Use **KeepWithGroup** to help display group headers and footers on the same page as the group. Use **KeepTogether** to help display static members with the rows or columns of a group. Use **RepeatOnNewPage** to repeat the group header or footer on every page that displays at least one complete instance of the row group member designated by the **KeepWithGroup** value. **RepeatOnNewPage** is not supported for column group members.

NOTE

KeepWithGroup, **KeepTogether**, and **RepeatOnNewPage** are group member properties that you can set by using the **Advanced Mode** of the Grouping pane. For more information, see [Grouping Pane \(Report Builder\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To keep a static row with a set of dynamic rows associated with a row group

1. On the design surface, click anywhere in the tablix data region to select it. The Grouping pane displays the row and column groups for the data region.
2. On the right side of the Grouping pane, click the down arrow, and then click **Advanced Mode**. The Row Groups pane displays the hierarchical static and dynamic members for the row groups hierarchy.
3. Click the static member that corresponds to the row header or footer that you want to keep with the group rows. The Properties pane displays the **Tablix Member** properties.
4. In the Properties pane, click **KeepWithGroup**, and then choose one of the following values from the drop-down list:
 - **None** Select this option to indicate no preference for keeping this member with the members of the selected row group.

- **Before** Select this option to keep this member with the members of the previous group. You might use this for group footer rows.
 - **After** Select this option to keep this member with the members of the following group. You might use this for group header rows.
5. (Optional) Preview the report. Where possible, the report renderer keeps this member with the specified row group members.
- To keep a static column with a set of dynamic columns associated with a column group**
1. On the design surface, click anywhere in the tablix data region to select it. The Grouping pane displays the row and column groups for the data region.
 2. On the right side of the Grouping pane, click the down arrow, and then click **Advanced Mode**. The Column Groups pane displays the hierarchical static and dynamic members for the column group hierarchy.
 3. Click the static member that corresponds to the static column that you want to keep with the group columns. The Properties pane displays the **Tablix Member** properties.
 4. In the Properties pane, click **KeepWithGroup**, and then choose one of the following values from the drop-down list:
 - **None** Select this option to indicate no preference for keeping this member with the members of the selected column group.
 - **Before** Select this option to keep this member with the members of the previous group. You might use this for column labels that display before the specified column group members.
 - **After** Select this option to keep this member with the members of the following group. You might use this for column totals that display after the specified column group members.
 5. (Optional) Preview the report. Where possible, the report renderer keeps this member with the specified column group members.

See Also

[Tablix Data Region Cells, Rows, and Columns \(Report Builder\) and SSRS](#)

Create a Stepped Report (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

A stepped report is a type of Reporting Services paginated report that shows detail rows or child groups indented under a parent group in the same column, as shown in the example below:

Model	Size	Quantity
Socks		
Mountain Bike Socks	M	180
Mountain Bike Socks	L	216
Racing Socks	M	252
Racing Socks	L	288
Caps		
Cycling Cap		288
Jerseys		
Long-Sleeve Logo Jersey	S	144

Traditional table reports place the parent group in an adjacent column on the report. The new tablix data region enables you to add a group and detail rows or child groups to the same column. To differentiate the group rows from the detail or child group rows, you can apply formatting such as font color, or you can indent the detail rows.

The procedures in this topic show you how to manually create a stepped report, but you can also use the New Table and Matrix Wizard. It provides the layout for stepped reports, making it easy to create them. After you complete the wizard, you can further enhance the report.

NOTE

The wizard is available only in Report Builder.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To create a stepped report

1. Create a table report. For example, insert a tablix data region and add fields to the Data row.
2. Add a parent group to your report.
 - a. Click anywhere in the table to select it. The Grouping pane displays the Details group in the Row Groups pane.
 - b. In the Grouping Pane, right-click the Details Group, point to **Add Group**, and then click **Parent Group**.
 - c. In the **Tablix Group** dialog box, provide a name for the group and type or select a group expression from the drop-down list. The drop-down list displays the simple field expressions that are available in the Report Data pane. For example, [PostalCode] is a simple field expression for the PostalCode field in a dataset.

- d. Select **Add group header**. This option adds a static row above the group for the group label and group totals. Likewise, you can select **Add group footer** to add a static row below the group. Click **OK**.

You now have a basic tabular report. When it is rendered, you see one column with the group instance value, and one or more columns with grouped detail data. The following figure shows what the data region might look like on the design surface.

	Model	Size	Quantity
[Subcategory]	[Model]	[Size]	[Quantity]

The following figure shows how the rendered data region might look when you view the report.

	Model	Size	Quantity
Socks	Mountain Bike Socks	M	180
	Mountain Bike Socks	L	216
	Racing Socks	M	252
	Racing Socks	L	288
Caps	Cycling Cap		288
Jerseys	Long-Sleeve Logo Jersey	S	144

3. For a stepped report, you do not need the first column that shows the group instance. Instead, copy the value in the group header cell, delete the group column, and paste in the first text box in the group header row. To remove the group column, right-click the group column or cell, and click **Delete Columns**. The following figure shows what the data region might look like on the design surface.

	Model	Size	Quantity
[Subcategory]	[Subcategory]		
	[Model]	[Size]	[Quantity]

4. To indent the detail rows under the group header row in the same column, change the padding of the detail data cell.

- Select the cell with the detail field that you want to indent. The text box properties for that cell appear in the Properties pane.
- In the Properties pane, under **Alignment**, expand the properties for **Padding**.
- For **Left**, type a new padding value, such as **.5in**. Padding indents the text in the cell by the value you specify. The default padding is 2 points. Valid values for the Padding properties are zero or a positive number, followed by a size designator.

Size designators are:

in	Inches (1 inch = 2.54 centimeters)
cm	Centimeters
mm	Millimeters
pt	Points (1 point = 1/72 inch)

Your data region will look similar to the following example.

Model	Size	Quantity
[Subcategory]		
= [Model]	[Size]	[Quantity]

Data Region for Stepped Report Layout

On the **Home** tab Click **Run**. The report displays the group with indented levels for the child group values.

To create a stepped report with multiple groups

1. Create a report as described in the previous procedure.
 2. Add additional groups to your report.
 - a. In the Row Groups pane, right-click the group, click **Add Group**, and then choose the type of group you want to add.
- NOTE**
- There are several ways to add groups to a data region. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#).
- b. In the **Tablix Group** dialog box, type a name.
 - c. In **Group expression**, type an expression or select a dataset field to group on. To create an expression, click the expression (**fx**) button to open the **Expression** dialog box.
 - d. Click **OK**.
3. Change the padding for the cell that displays the group data.

See Also

[Page Headers and Footers \(Report Builder and SSRS\)](#)

[Formatting Report Items \(Report Builder and SSRS\)](#)

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Tables \(Report Builder and SSRS\)](#)

[Matrices \(Report Builder and SSRS\)](#)

[Lists \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Add, Move, or Delete a Table, Matrix, or List (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

In a Reporting Services paginated report, a data region displays data from a report dataset. Data regions include table, matrix, list, chart, and gauge. To nest one data region inside another, add each data region separately, and then drag the child data region onto the parent data region.

The simplest way to add a table or matrix data region to your report is to run the New Table or New Matrix Wizard. These wizards will guide you through the process of choosing a connection to a data source, arranging fields, and choosing the layout and style. The wizards are only available in Report Builder.

To add a table or matrix to a report by using the New Table or New Matrix Wizard

1. On the **Insert** tab, click **Table** or **Matrix**, and then click **Table Wizard** or **Matrix Wizard**.
2. Follow the steps in the **New Table** or **New Matrix** wizard.
3. On the **Home** tab, click **Run** to see the rendered report.
4. On the **Run** tab, click **Design** to continue working on the report.

To add a data region

1. On the **Ribbon**, in the **Data Regions** group, click the data region to add.
2. Click the design surface, and then drag to create a box that is the desired size of the data region.
3. Drag a report dataset field from the Report Data pane onto a data region cell. The data region is now bound to data from the report dataset.

To select a data region

- For a tablix data region, right-click the corner handle. For a chart or gauge data region, click in the data region.

A selection handle and eight resizing handles appear.

For nested data regions, right-click in the nested data region, click **Select**, and then select the report item you want. To verify which report item is selected, use the Properties pane. The name of the selected item on the design surface appears in the toolbar of the Properties pane.

To move a data region

- To move a data region, click the selection handle of the data region and drag it. Use snaplines to align it to existing report items.

If the ruler is not visible, click the View tab and select the **Ruler** option.

Alternatively, use the arrow keys to move the selected data region on the design surface.

To delete a data region

- Select the data region, right-click in the data region, and then click **Delete**.

See Also

[Tablix Data Region \(Report Builder and SSRS\)](#)
[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Report Parts (Report Builder and SSRS)

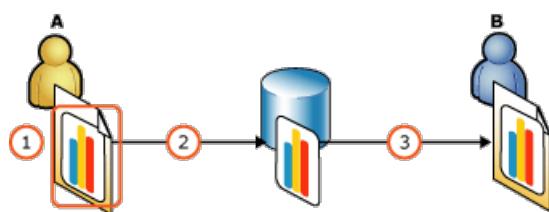
3/24/2017 • 10 min to read • [Edit Online](#)

Report items such as tables, matrices, charts, and images can be published as *report parts*. Report parts are paginated report items that have been published separately to a report server and that can be reused in other paginated reports. Report parts have an .rsc file extension.

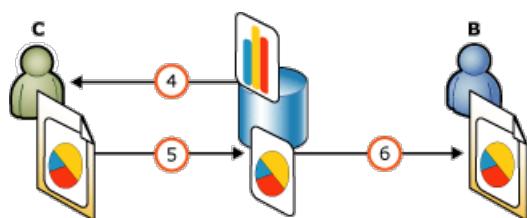
With report parts, work groups can now take advantage of the different strengths and roles of their team members. For example, if you are responsible for creating charts, you can save your charts as separate parts that you and your coworkers can reuse in other reports. You can publish report parts on a report server or SharePoint site integrated with a report server. You can reuse them in multiple reports, and you can update them on the server.

The report part that you add to your report maintains a relationship to the instance of the report part on the site or server by means of a unique ID. After you add report parts from a site or server to a report, you can modify them, independent of the original report part on the site or server. You can accept updates that others have made to the report part on the site or server, and you can save the modified report part back to the site or server, either adding a new report part or writing over the original, if you have sufficient permissions.

Life Cycle of a Report Part



1. Person A creates a report with a chart that depends on an embedded dataset.
2. Person A chooses to publish the chart to the report server. Report Builder assigns a unique ID to the published chart. Person A does not choose to share the dataset, so the dataset remains embedded in the chart.
3. Person B creates a blank report, searches the Report Part Gallery, finds the chart, and adds it to the report. The chart is now part of Person B's report, along with the embedded dataset. Person B can modify the instances of the chart and dataset that are in the report. This will have no effect on the instances of the chart and dataset on the report server, nor will it break the relationship between the instances in the report and on the report server.



4. Person C adds the chart to a report and changes this chart in the report from a bar to a pie chart.
5. Person C has permissions to overwrite the chart on the server and does so, republishing it to the server. This updates the published copy of the chart on the server. Person C does not choose to share the dataset either, so it remains embedded in the chart.
6. Person B accepts the updated chart from the server. This overwrites the changes that Person B had made.

to the chart in Person B's report.

Publishing Report Parts

When you publish a report part, Report Builder assigns it a unique ID, which is distinct from the report part name. Report Builder maintains that ID, no matter what else you change about the report part. The ID links the original report item in your report to the report part. When other report authors reuse the report part, the ID also links the report part in their report to the report part on the report server.

These are the report items you can publish as report parts:

- Charts
- Gauges
- Images
- Maps
- Parameters
- Rectangles
- Tables
- Matrices
- Lists

When you publish a report item that displays data, such as a table, matrix, or chart, the dataset that the report item depends on is saved with it, as a dataset embedded in it. You can also save the dataset separately, as a shared dataset that you and others can use as a basis for other report parts. For more information, see [Report Parts and Datasets in Report Builder](#).

Some report parts can contain other report items. For example, a table can contain a chart, and a rectangle can contain a matrix and a chart. When you publish a report item that contains other report items, they are saved as a unit. The other report items are saved embedded in the container report part. You cannot update them separately, and you cannot save the items in the container as separate report parts.

For more information on publishing report parts, see [Publish and Republish Report Parts \(Report Builder and SSRS\)](#).

Modifying Report Part Metadata

You can publish report parts with default settings to a default location, or you can save each report part to a different location, and modify the metadata, such as the title and description.

It is a good idea to give the report part a clear name and description when you publish it to help people identify it when searching. You could end up with a lot of report parts with similar names on your site or server.

Consider using naming conventions to illustrate relationships between report parts and their dependent items.

Also, consider saving shared data sources, shared datasets, and the report parts that depend on them in the same folder.

You can also edit the description in the Properties pane.

Reusing Report Parts

The easiest way to create a report is to add an existing report part, like a table or chart, to your report from the Report Part Gallery. After you add it to your report, you can modify it as much as you need, or accept updates

from the server. Changing the report item in your report will not affect the instance of the report part published on the site or server, nor will it break the relationship between the instance in the report and on the site or server. If you have sufficient permissions, you can save the updated copy back to the site or server. If someone else modifies the copy on the site or server, you can decide to keep your copy as it is, or you can update it to be like the copy on the site or server.

Searching for Report Parts

You look for report parts to add to your report in the Report Part Gallery. You can filter the report parts by all or part of the name of the part, who created it, who last modified it, when it was last modified, where it's stored, or what type of report part it is. For example, you could search for all charts created last week by one of your coworkers.

You can view the search results either as thumbnails or as a list, and sort the search results by name, created and modified dates, and creator. For more information, see [Browse for Report Parts and Set a Default Folder \(Report Builder and SSRS\)](#).

What Comes with a Report Part

When you add a report part to your report, you are also adding everything it must have to work. For example, any object that displays data is dependent on a dataset – a query and a connection to a data source. It may also have one or more parameters. All of the items it is dependent on are its *dependencies*, and all of them, or pointers to them, are included with the report part when you add it to your report. The dataset and parameters are listed in the Report Data pane of your report.

The dataset for the report part may be embedded in the report part, or it may be a separate, shared dataset that the report part points to. If it is embedded in the report part, you may be able to modify it. If it is a shared dataset, it is a separate object that you would need permissions for. For more information about shared and embedded datasets, see [Report Datasets \(SSRS\)](#).

Resolving Naming Conflicts

When you add a report part, Report Builder fixes any name conflicts. For example, if you have a Chart1 in your report already and you add a report part called Chart1, Report Builder automatically renames the new report part Chart2. If you have a Dataset1 in your report already, and you add a report part that refers to a different dataset that is also called Dataset1, Report Builder renames the new dataset Dataset2 and updates the references.

Adding More Than One Report Part

You can add an unlimited number of report parts to your report. However, you can only add one report part at a time. You can even add multiple instances of one report part to the same report. They will all have unique names, but will all be instances of the same report part on the server and have the same unique ID.

When you add another report part that uses a dataset identical to a dataset already in your report, the wizard does not add another version of that dataset to your report; it redirects the references in the report part to go to the existing dataset. For more information, see [Report Parts and Datasets in Report Builder](#).

Updating Report Parts with Changes from the Server

Every time you open a report, Report Builder checks to see if the server instances of report parts in that report have been updated on the server. It also checks for changes in the report parts' dependent items, such as the dataset and parameters. If any published report parts or their dependencies have been updated on the server, an information bar in your report displays the number that have been updated. You can choose to view and accept or reject the updates, or dismiss the information bar. If you choose to view the updates, you see a thumbnail of the report part, who last modified it, and when. Then you can accept any or all of the updated items.

NOTE

You can disable the information bar and not be informed if a report part has changed. You set this option when you add the report part to your report. Even if you have disabled the information bar, you can still check for updates. For more information, see [Check for Updates or Turn Updates Off \(Report Builder and SSRS\)](#).

Report Builder checks for differences between the date the report part was last updated on the server and the date when you last synchronized the report part with the server. It does not check the date that you modified the report part in your report. Thus, the report part in your report and the report part on the server could be quite different, but when Report Builder checks for updates, it will not find any.

Accepting Updates

When you accept an update for a report part, it completely replaces the copy of the report part already in your report. You cannot combine features of the report part in the report with features of the published report part on the server. However, if you have changed one of the report part's dependencies, such as an embedded dataset, Report Builder does not copy over the dependency already in your report. It downloads a new copy of the dependency, and updates the report part to point to the new copy.

Reverting to a Previous Version of a Report Part

If you have changed a version of a report part in your report and decide you want to replace it with the version that is on the server, you cannot use the **Update** dialog box to do that. Updating is only for report parts that have changed on the server since you downloaded them.

To revert to the version on the server, just delete the version you have in your report and add it again.

Updating Report Parts Already on the Server

You can choose to update an existing report part on the server, or to publish it as a new report part without replacing the existing one. When you update the report part on the server, it does not automatically modify copies of the report part in other reports. If other report authors have added that report part to a report, they are informed of the change the next time they open that report. They can choose to accept your changes or not.

If you choose to publish it as a new report part, Report Builder gives it a new unique ID, and it no longer links to the original report part.

If the dataset is embedded in the report part, then every time you publish the report part, the dataset will be displayed in the **Publish Report Parts** dialog box. Shared datasets are not displayed in the **Publish Report Parts** dialog box.

Working with Report Parts in Report Designer

Report parts work a little differently in Report Designer in SQL Server SQL Server Data Tools (SSDT). In Report Designer, publishing is one-way: you can publish a report part from Report Designer, but you cannot reuse an existing report part in Report Designer. For more information, see [Report Parts in Report Designer \(SSRS\)](#).

How-to Topics

[Publish and Republish Report Parts \(Report Builder and SSRS\)](#)

[Browse for Report Parts and Set a Default Folder \(Report Builder and SSRS\)](#)

[Check for Updates or Turn Updates Off \(Report Builder and SSRS\)](#)

See Also

[Report Parts and Datasets in Report Builder](#)
[Troubleshoot Report Parts \(Report Builder and SSRS\)](#)
[Managing Report Parts](#)

Publish and Republish Report Parts (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Report parts are paginated report items that have been published separately to a report server and that can be reused in other paginated reports. You can publish a report part with default settings in a default location, or you can edit report part metadata such as name and description, and save it somewhere else on a report server. You can also save it to a SharePoint site that is integrated with a report server if you have the correct permissions.

You can publish just the report part, with the dataset that it depends on embedded in it, or you can publish the dataset separately. If you publish the dataset separately, it becomes a shared dataset, and the report part links to it. For more information, see [Report Parts and Datasets in Report Builder](#).

You can republish an existing report part. If you have permissions, you can save over the original instance of the report part on the server, even if you didn't create it. You can also publish it as a new report part and save it either in the same or a different location.

To publish a report part

1. On the Report Builder menu, click **Publish Report Parts**.

If you are not connected to a report server, you will be prompted to connect. If you cannot connect to a report server, you cannot publish report parts.

2. To save your report parts with default settings to the default location, click **Publish all report parts with default settings**.

Otherwise, click **Review and modify report parts before publishing**.

3. Edit the report part name and description: Double-click the name to edit it and click in the **Description** field to add a description.

NOTE

It is a good idea to give the report part name and description, to help people identify it when searching. The maximum length for the name of a report part is 260 characters for the entire path, including the names of the folders on the server, followed by the actual name of the report part.

4. To save your report part to a folder other than the default, click the **Browse** button.

5. When you have set the options for all the report parts in the report, click **Publish**.

After you publish the report parts, the dialog box shows which were successfully published and which were not. You can view details in the **Publish Report Parts** dialog box for the report parts that failed to publish.

6. Click **Close**.

To republish a report part

1. Follow the previous procedure for publishing a report part.
2. In the **Publish Report Parts** dialog box, if you click **Publish as a New Copy of the Report Part**, Report

Builder will not save over the existing report part on the server, but will create another report part instead.

NOTE

If you publish it as a new report part, it will have a new unique ID. It will no longer receive updates if the original report part changes.

See Also

[Report Parts \(Report Builder and SSRS\)](#)

[Report Parts and Datasets in Report Builder](#)

[Troubleshoot Report Parts \(Report Builder and SSRS\)](#)

[Check for Updates or Turn Updates Off \(Report Builder and SSRS\)](#)

[Browse for Report Parts and Set a Default Folder \(Report Builder and SSRS\)](#)

Browse for Report Parts and Set a Default Folder (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

The easiest way to create a Reporting Services paginated report is to add existing report parts, such as tables and charts, to your report from the Report Part Gallery. When you add a report part to your report, you are also adding everything it must have to work. For example, any report part that displays data depends on a dataset, that is, a query and a connection to a data source. After you add the report part to your report, you can modify it as much as you need.

You can set a default folder to publish report parts to the report server or SharePoint site integrated with a report server.

For more information, see [Report Parts \(Report Builder and SSRS\)](#).

To browse for report parts

1. On the **Insert** menu, click **Report Parts**.

If you are not already connected, click **Connect to a report server**, and enter the name.

NOTE

You must be connected to a report server to browse for report parts.

2. You can narrow your search by specifying details about the report part. Type all or part of the name and description in the **Search** box, or click **Add Criteria** and add values for any or all of these fields:

- Created by
- Date created
- Date last modified
- Last modified by
- Server folder
- Type

For example, to find an image, click **Add Criteria**, and then click **Type**. In the dropdown box, select the **Image** check box, press ENTER, and then click the Search magnifying glass.

NOTE

For the **Created by** and **Last modified by** values, search for the person's user name as it is represented on the report server.

To set a default folder for report parts

1. Click **Report Builder**, and then click **Options**.

2. In the **Options** dialog box, on the **Settings** tab, type a folder name in the **Publish report parts to this folder by default** textbox.

Report Builder will create this folder if you have permissions to create folders on the report server and the folder does not exist yet.

You do not need to restart Report Builder for this setting to take effect.

See Also

[Check for Updates or Turn Updates Off \(Report Builder and SSRS\)](#)

[Report Parts \(Report Builder and SSRS\)](#)

[Report Parts and Datasets in Report Builder](#)

[Troubleshoot Report Parts \(Report Builder and SSRS\)](#)

[Publish and Republish Report Parts \(Report Builder and SSRS\)](#)

Charts (Report Builder and SSRS)

3/24/2017 • 8 min to read • [Edit Online](#)

Read about using chart data regions to help readers of your Reporting Services paginated reports understand large volumes of aggregated data at a glance.

The more time you spend carefully preparing and understanding your data before you create a chart, the easier it will be to design your charts quickly and efficiently. For help choosing which chart to use, see [Chart Types](#). To start experimenting with charts immediately, see the bar, column, sparkline, and pie chart tutorials in [Report Builder Tutorials](#).

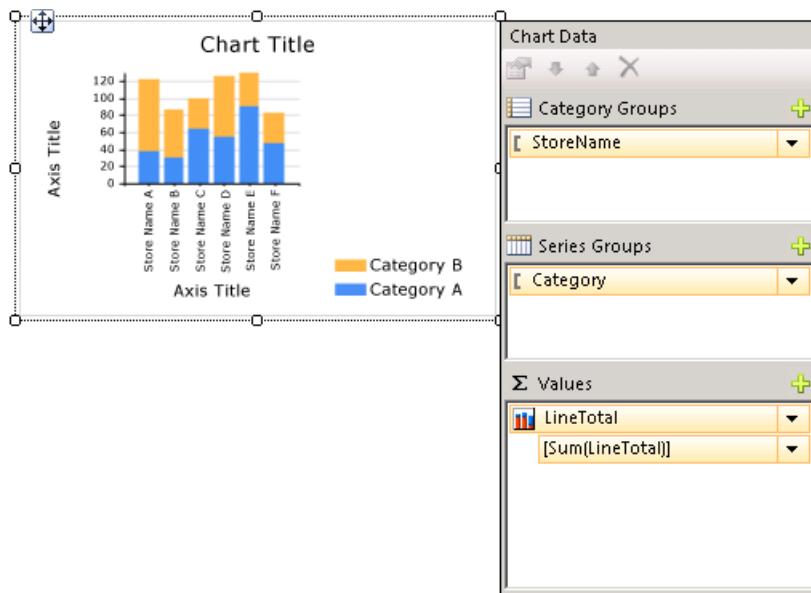
The following illustration shows many of the different elements used in the chart.



You can publish charts separately from a report as *report parts*. For more information, see [Report Parts](#).

Designing a Chart

After you add a chart data region to the design surface, you can drag report dataset fields for numeric and non-numeric data to the Chart Data pane of the chart. When you click the chart on the design surface, the Chart Data pane appears, with three areas—Category Groups, Series Groups, and Values. If the report has a shared or embedded dataset, the fields in the dataset appear in the Report Data pane. Drag fields from the dataset into the appropriate area of the Chart Data pane. By default, when a field is added to one of the areas of the chart, Reporting Services calculates an aggregate for the field. You can also use series grouping to dynamically generate series. A chart is [organized like a matrix](#).



NOTE

The data in the chart at design time is different from the data in the chart when the report is processed. It is not your real data. It is generated data that has been added so that you can design your chart with an idea of what the chart will look like.

How a Chart is like a Matrix

One way to think about how charts work is to compare them to matrices.



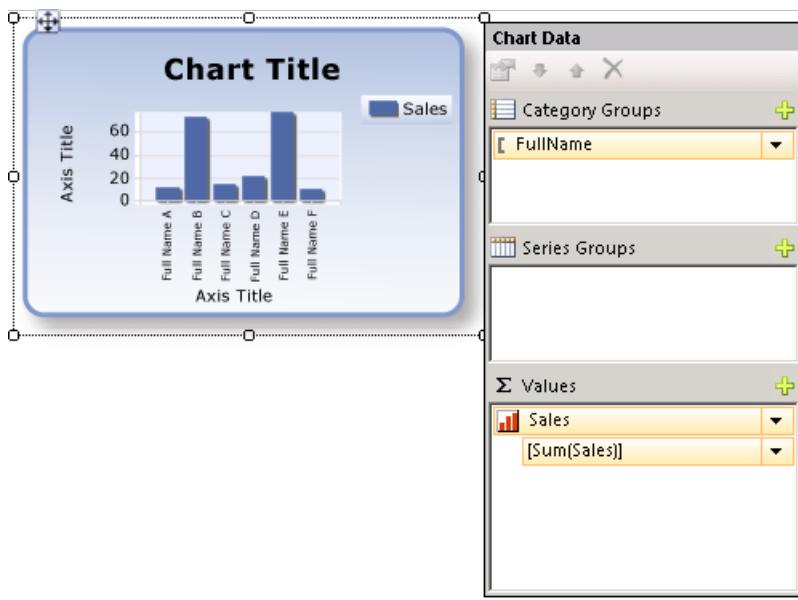
Conceptually, their organization is identical:

- The Columns group in the matrix is like the Category Groups area in the chart.
- The Rows group in the matrix is like the Series Groups area in the chart.
- The Data area in the matrix is like the Values area in the chart.

Adding Data to the Chart

Suppose you have a report that shows Sales by Name. You drop the Full Name field to the Category Groups area and the Sales field to the Values area.

When you add the Sales field to the Values area, the text of the data field appears in the legend, and the data from this numeric field will be aggregated into one value. By default, the value is aggregated using the built-in function Sum. The Chart Data pane will contain a simple expression for your field. In our example, `[Sum(Sales)]` will appear for the field expression `=Sum(Fields!Sales.Value)`. If no groups are specified, the chart will only show one data point. In order to show multiple data points, you must group your data by adding a grouping field. When you add the Name field to the Category Groups area, a grouping field of the same name as the name of the field is automatically added to the chart. When fields that define the values along the x and y axes are added, the chart has enough information to plot the data correctly.



When the Series Groups area is left empty, the number of series is fixed at design time. In this example, Sales is the only series that appears on the chart.

Category and Series Groups in a Chart

A chart supports nested category and series groups. Charts do not display detail data. Add groups to a chart by dragging dataset fields to the category and series drop zones for a selected chart.

Shape charts such as pie charts support category groups and nested category groups. Other charts such as bar charts support category groups and series groups. You can nest groups, but make sure that the numbers of categories or series do not obscure the presentation of information in the chart.

Adding Series Grouping to a Chart

If you add a field to the Series Groups area, the number of series depends on the data that is contained in that field. In our earlier example, suppose you add a Year field to the Series Groups area. The number of values in the Year field will determine how many series will appear on the chart. If the Year field contains the years 2004, 2005, and 2006, the chart will display three series for every field in the Values area.

Dataset Considerations Before Creating a Chart

Charts provide a summary view of your data. However, with large datasets, the information on a chart can become obscured or unreadable. Missing or null data points, data types ill-suited to the type of chart, and advanced applications such as combining charts with tables can all affect the readability of a chart. Before designing a chart, you should carefully prepare and understand your data so that you can design your charts quickly and efficiently.

You can have as many charts in your report as you want. A chart, like any other data region such as a matrix or table, is bound to a single dataset. If you want to display multiple datasets on the same chart, you can create an additional dataset that uses a JOIN or UNION statement in your SQL query before adding data to the chart. For more information about the JOIN and UNION statement, see Books Online or another SQL reference.

Consider pre-aggregating data in the dataset query if detail data is not necessary or useful. To display each data point more clearly, reduce the number of categories in your dataset. You can filter the dataset or add a condition to your query that reduces the number of rows returned.

Best Practices When Displaying Data in a Chart

Charts are most effective when the number of elements that are displayed presents a clear image of the

underlying information. Some charts, like scatter graphs, benefit from numerous data points, while others, like pie charts, are more effective with fewer data points. Choose a chart type carefully based on the values in your dataset and how you want this information to be shown. For more information, see [Chart Types \(Report Builder and SSRS\)](#).

There are several ways you can consolidate data on a chart:

- When using a pie chart, collect small slices into one slice called "Other." This will reduce the number of slices on your pie chart. For more information, see [Collect Small Slices on a Pie Chart \(Report Builder and SSRS\)](#).
- Avoid using data point labels when there are numerous data points. Data point labels are most effective when there are only a few points on the chart.
- Filter unwanted or irrelevant data. This helps you highlight the key data that you are trying to show on the chart. To filter data points in a chart, set a filter on a category group or a series group. By default, the chart uses the built-in function Sum to aggregate values that belong to the same group into an individual data point in the series. If you change the aggregate function of a series, you must also change the aggregate function in the filter expression. For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).
- To display ratio data in a table or matrix template, consider using a linear gauge instead of a bar graph. Gauges are better suited for showing a single value inside a cell. For more information, see [Nested Data Regions \(Report Builder and SSRS\)](#).

Aggregating Values from a Data Field on the Chart

By default, when a field is added to the Values area of the chart, Reporting Services calculates an aggregate for the field. If you drag a field onto the chart without dropping the field into a specific area, the chart will determine whether this field belongs on the category (x) axis or value (y) axis based on the data type of the field. Numeric fields that are dropped in the Values area are aggregated using the SUM function. If the data type of your value field is String in the Values area, the chart cannot display a numeric value, even if there are numbers in the fields, so the chart displays the COUNT function. To avoid this behavior, make sure that the fields that you use have numeric data types, instead of Strings that contain formatted numbers. You can use a Visual Basic expression to convert String values to a numeric data type using the **Cdbl** or **CInt** constant. For example, the following complex expression converts a field that is named `MyField` that contains numeric values that are formatted as Strings.

```
=Sum(CDbL(Fields!MyField.Value))
```

For more information about aggregate expressions, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

In This Section

[Add a Chart to a Report \(Report Builder and SSRS\)](#)

Describes the first steps in adding a chart to your report.

[Chart Types \(Report Builder and SSRS\)](#)

Describes all of the chart types and sub-types available in Reporting Services, including considerations and best practices for using various chart types.

[Formatting a Chart \(Report Builder and SSRS\)](#)

Use formatting to improve the overall appearance and highlight key data points of your chart.

[Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

Describes considerations when working with charts based on fields with empty or null values.

[Displaying a Series with Multiple Data Ranges on a Chart \(Report Builder and SSRS\)](#)

Describes how to add scale breaks to a series that contains more than one range of data.

[Multiple Series on a Chart \(Report Builder and SSRS\)](#)

Describes several methods of showing multiple series on the same chart, including combining chart types, using the secondary axis, specifying different chart types and using multiple chart areas.

[Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

Provide different views of data from the same report dataset.

[Add or Delete a Group in a Chart \(Report Builder and SSRS\)](#)

Describes adding groups and nested groups to a chart.

[Add a Moving Average to a Chart \(Report Builder and SSRS\)](#)

Describes using the Moving Average formula to calculate the average of the data in your series.

[Troubleshoot Charts \(Report Builder and SSRS\)](#)

Describes tips for working with charts.

See Also

[Images, Text Boxes, Rectangles, and Lines \(Report Builder and SSRS\)](#)

[Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#)

[Nested Data Regions \(Report Builder and SSRS\)](#)

[Tutorial: Add a Column Chart to Your Report \(Report Builder\)](#)

[Tutorial: Add a Pie Chart to Your Report \(Report Builder\)](#)

[Tutorial: Add a Bar Chart to Your Report \(Report Builder\)](#)

Add a Chart to a Report (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

When you want to summarize data in a visual format in a Reporting Services paginated report, use a Chart data region. It is important to choose an appropriate chart type for the type of data that you are presenting. This affects how well the data can be interpreted when put in chart form. For more information, see [Charts \(Report Builder and SSRS\)](#).

The simplest way to add a Chart data region to your report is to run the New Chart Wizard. The wizard offers column, line, pie, bar, and area charts. For these and other chart types, you can also add a chart manually.

After you add a Chart data region to the design surface, you can drag report dataset fields for numeric and non-numeric data to the Chart Data pane of the chart. Click the chart to display the Chart Data pane with its three areas: Series Groups, Category Groups, and Values.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a chart to a report by using the Chart Wizard

1. NOTE

The Chart Wizard is only available in Report Builder.

On the **Insert** tab, click **Chart**, and then click **Chart Wizard**.

2. Follow the steps in the **New** Chart wizard.
3. On the **Home** tab, click **Run** to see the rendered report.
4. On the **Run** tab, click **Design** to continue working on the report.

To add a chart to a report

1. Create a report and define a dataset. For more information, see [Report Datasets \(SSRS\)](#).
 2. On the **Insert** tab, click **Chart**, and then click **Insert Chart**.
 3. Click on the design surface where you want the upper-left corner of the chart, and then drag to where you want the lower-right corner of the chart.
- The **Select Chart Type** dialog box appears.
4. Select the type of chart you want to add. Click **OK**.
 5. Click the chart to display the **Chart Data** pane.
 6. Add one or more fields to the **Values** area. This information will be plotted on the value axis.
 7. Add a grouping field to the **Category Groups** area. When you add this field to the **Category Groups** area, a grouping field is automatically created for you. Each group represents a data point in your series.

8. To summarize the data by category, right-click the data field and click **Series Properties**. In the **Category** box, select the category field from the drop-down list. Click **OK**.
9. On the **Home** tab, click **Run** to see the rendered report.
10. On the **Run** tab, click **Design** to continue working on the report.

On charts with axes, such as bar and column charts, the category axis may not display all the category labels. For more information about how to change the axis labels, see [Specify an Axis Interval \(Report Builder and SSRS\)](#).

See Also

- [Charts \(Report Builder and SSRS\)](#)
- [Chart Types \(Report Builder and SSRS\)](#)
- [Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)
- [Tutorial: Adding a Bar Chart to Your Report \(Report Builder\)](#)
- [Tutorial: Adding a Bar Chart to a Report \(Report Designer\)](#)
- [Tutorial: Adding a Pie Chart to Your Report \(Report Builder\)](#)
- [Tutorial: Adding a Pie Chart to a Report \(Report Designer\)](#)

Chart Types (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

It is important to choose an appropriate chart type for the type of data that you are presenting. This will determine how well the data can be interpreted when put in chart form. For example, if your dataset contains a lot of data points relative to the size of the chart, it may be better presented using an area, line, or scatter chart. For discussion on how to prepare your data depending on the chart type selected, see [Charts \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Choosing a Chart Type

Each chart type has unique characteristics to help you visualize your dataset. You can use any chart type to display your data, but your data will be easier to read when you use a chart type that is suitable to your data, based on what you are trying to show in your report. The following table summarizes chart features that affect the suitability of a chart to your particular dataset.

You can change the chart type after you have created it. For more information, see [Change a Chart Type \(Report Builder and SSRS\)](#).

Examples of many of these types of charts are available as sample reports. For more information about downloading sample reports, see [SQL Server 2016 Report Builder and Report Designer sample reports](#).

CHART TYPE	DISPLAY RATIO DATA	DISPLAY STOCK DATA	DISPLAY LINEAR DATA	DISPLAY MULTI-VALUE DATA
Area Charts (Report Builder and SSRS)			✓	
Bar Charts (Report Builder and SSRS)			✓	
Data Bars			✓	
Column Charts (Report Builder and SSRS)			✓	
Line Charts (Report Builder and SSRS)			✓	
Pie Charts (Report Builder and SSRS)	✓			
Polar Charts (Report Builder and SSRS)	✓			

CHART TYPE	DISPLAY RATIO DATA	DISPLAY STOCK DATA	DISPLAY LINEAR DATA	DISPLAY MULTI-VALUE DATA
Range Charts (Report Builder and SSRS)			✓	✓
Scatter Charts (Report Builder and SSRS)	✓		✓	
Shape Charts (Report Builder and SSRS)	✓			
Sparklines	✓	✓	✓	✓
Stock Charts (Report Builder and SSRS)		✓		✓

See Also

[Charts \(Report Builder and SSRS\)](#)

[Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

[Add a Chart to a Report \(Report Builder and SSRS\)](#)

Change a Chart Type (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

When you first insert a chart into a Reporting Services paginated report, the **Select Chart Type** dialog appears. If you cancel this dialog, a Column chart type is added by default.

At any point when designing the report, you can change the chart type to something more suitable to the report. It is important to select the correct chart type for your data so that it can be interpreted correctly. You should understand each chart type's characteristics to determine what chart type is best suited for your data. For more information, see [Chart Types \(Report Builder and SSRS\)](#).

When multiple series are displayed on a chart, you may want to change the chart type of an individual series. You can only change the chart type of an individual series if the chart type is Area, Column, Line, or Scatter. For all other chart types, all series in your chart will be changed to the selected chart type.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To change the chart type

1. In Design view, right-click the chart and then click **Change Chart Type**.

NOTE

When there are multiple series on a chart, you must right-click on the series, not the chart, which you want to change.

2. In the **SelectChart Type** dialog box, select a chart type from the list.

See Also

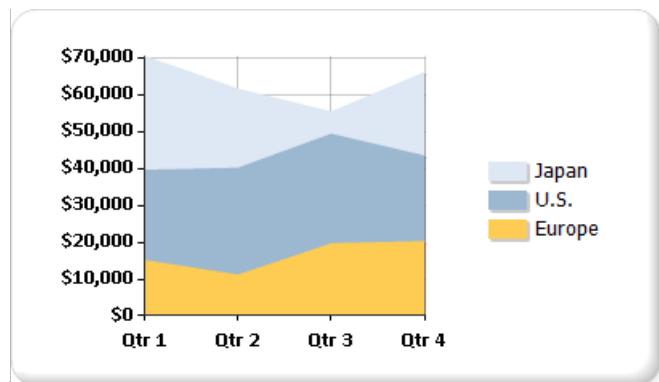
- [Charts \(Report Builder and SSRS\)](#)
- [Gauges \(Report Builder and SSRS\)](#)
- [Add a Chart to a Report \(Report Builder and SSRS\)](#)

Area Charts (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

An area chart displays a series as a set of points connected by a line, with all the area filled in below the line. For more information on how to add data to an area chart, see [Charts \(Report Builder and SSRS\)](#).

The following illustration shows an example of a stacked area chart. The data is well suited for display on a stacked area chart because the chart can display totals for all series as well as the proportion that each series contributes to the total.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Stacked area.** An area chart where multiple series are stacked vertically. If there is only one series in your chart, the stacked area chart will display the same as an area chart.
- **Percent stacked area.** An area chart where multiple series are stacked vertically to fit the entire chart area. If there is only one series in your chart, the stacked area chart will display the same as an area chart.
- **Smooth area.** An area chart where the data points are connected by a smooth line instead of a regular line. Use a smooth area chart instead of an area chart when you are more concerned with displaying trends than with displaying the values of individual data points.

Data Considerations for Area Charts

- Along with the line chart, the area chart is the only chart type that displays data contiguously. For this reason, an area chart is commonly used to represent data that occurs over a continuous period of time.
- A percent stacked area chart is useful for showing proportional data that occurs over time.
- If there is only one series, a stacked area chart will be drawn as an area chart.
- In a plain area chart, if the values in multiple series are similar, the areas may overlap, obscuring important data point values. You can resolve this issue by changing the chart type to a stacked area chart, which is designed to show multiple series on an area chart.
- If your stacked area chart contains gaps, it is possible that your dataset includes empty values, which will be

shown as a vacant section on a stacked area chart. If your dataset includes empty values, consider inserting empty points on the chart. Adding empty points will fill in the empty areas on the chart with a different color to indicate null or zero values. For more information, see [Add Empty Points to a Chart \(Report Builder and SSRS\)](#).

- Area chart types are very similar to column and line charts in behavior. If you are making a comparison between multiple series, consider using a column chart instead. If you are analyzing trends over a period of time, consider using a line chart.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Chart Types \(Report Builder and SSRS\)](#)

[Line Charts \(Report Builder and SSRS\)](#)

[Change a Chart Type \(Report Builder and SSRS\)](#)

[Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

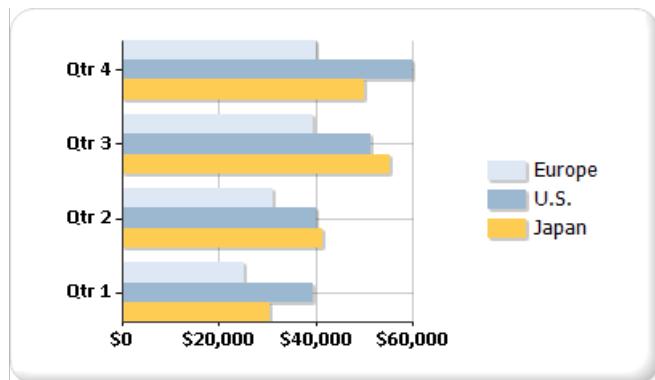
Bar Charts (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

A bar chart displays series as sets of horizontal bars. The plain bar chart is closely related to the column chart, which displays series as sets of vertical bars, and the range bar chart, which displays series as sets of horizontal bars with varying beginning and end points.

The bar chart is the only chart type that displays data horizontally. For this reason, it is popular for representing data that occurs over time, with a finite start and end date. It is also popular for showing categorical information since the categories can be displayed horizontally. For more information about how to add data to a bar chart, see [Charts \(Report Builder and SSRS\)](#).

The following illustration shows a bar chart. The bar chart is well suited for this data because all three series share a common time period, allowing for valid comparisons to be made.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations of the Bar Chart

- **Stacked.** A bar chart where multiple series are stacked vertically. If there is only one series in your chart, the stacked bar chart will display the same as a bar chart.
- **Percent stacked.** A bar chart where multiple series are stacked vertically to fit 100% of the chart area. If there is only one series in your chart, all the bars will fit to 100% of the chart area.
- **3D clustered.** A bar chart that shows individual series in separate rows on a 3D chart.
- **3D cylinder.** A bar chart that shapes the bars as cylinders on a 3D chart.

Data Considerations for Bar Charts

- Bar charts have their axes reversed. The category axis is the vertical axis (or "y-axis") and the value axis is the horizontal axis (or "x-axis"). This means that in a bar chart, you have more space for category labels to be displayed along the y-axis as a list that reads from top to bottom.
- Bar and column charts are most commonly used to show comparisons between groups. If more than three series are present on the chart, consider using a stacked bar or column chart. You can also collect stacked bar or column charts into multiple groups if you have several series on your chart.

- A bar chart displays values from left to right, which may be more intuitive when displaying data related to durations.
- If you are looking to add bars to a table or matrix within the report, consider using a linear gauge instead of a bar chart. The linear gauge is designed to show one value instead of multiple groups, so it is more flexible for use within a list or table data region. For more information, see [Gauges \(Report Builder and SSRS\)](#).
- You can add special drawing styles to the individual bars on a bar chart to increase its visual impact. Drawing styles include wedge, emboss, cylinder, and light-to-dark. These effects are designed to improve the appearance of your 2D chart. If you are using a 3D chart, the drawing styles will still be applied, but may not have the same effect. For more information about how to add a drawing style to a bar chart, see [Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#).
- Stacked bar charts place series on top of each other to create one bar stack. You have the option of separating the stacked bar chart into multiple sets of stacks for each category. The grouped stacked chart is displayed side-by-side. You can have any number of grouped stacked series in a chart.
- When data point labels are shown on a bar chart, the labels are placed on the outside of each bar. This can cause labels to overlap when the bars take up all of the allotted space within the chart area. You can change the position of the data point labels displayed for each bar by setting the **BarLabelStyle** property in the Properties pane.
- If there are a lot of data points in your dataset relative to the size of your chart, the size of the columns or bars and the spacing between them are reduced. To manually set the width of the columns in a chart, modify their width, in pixels, by modifying the **PointWidth** property. By default, this property has a value of 0.8. When you increase the width of the columns or bars in a chart, the space between each column or bar decreases.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Chart Types \(Report Builder and SSRS\)](#)

[Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

[Column Charts \(Report Builder and SSRS\)](#)

[Range Charts \(Report Builder and SSRS\)](#)

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Formatting the Legend on a Chart \(Report Builder and SSRS\)](#)

[Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#)

[Tutorial: Adding a Bar Chart to a Report \(Report Builder\)](#)

[Tutorial: Adding a Bar Chart to a Report](#)

Column Charts (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

A column chart displays a series as a set of vertical bars that are grouped by category. Column charts are useful for showing data changes over a period of time or for illustrating comparisons among items. The plain column chart is closely related to the bar chart, which displays series as sets of horizontal bars, and the range column chart, which displays series as sets of vertical bars with varying beginning and end points. For more information, see [Bar Charts \(Report Builder and SSRS\)](#) and [Range Charts \(Report Builder and SSRS\)](#).

The column chart is well suited for this data because all three series share a common time period, allowing for valid comparisons to be made.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations of a Column Chart

- **Stacked.** A column chart where multiple series are stacked vertically. If there is only one series in your chart, the stacked column chart will display the same as a column chart.
- **Percent stacked.** A column chart where multiple series are stacked vertically to fit 100% of the chart area. If there is only one series in your chart, all the column bars will fit to 100% of the chart area.
- **3D clustered.** A column chart that shows individual series in separate rows on a 3D chart.
- **3D cylinder.** A column chart whose bars are shaped like cylinders on a 3D chart.
- **Histogram.** A column chart which the chart calculates so that its bars are arranged in a normal distribution.
- **Pareto.** A column chart whose bars are arranged from highest to lowest.

Data Considerations for a Column Chart

- Bar and column charts are most commonly used to show comparisons between groups. If more than three series are present on the chart, consider using a stacked bar or column chart. You can also collect stacked bar or column charts into multiple groups if you have several series on your chart. For more information, see [Bar Charts \(Report Builder and SSRS\)](#).
- In a column chart, you have less space for category axis labels to display horizontally. If you have longer category labels, consider using a bar chart or changing the rotation angle of the label.
- You can add special drawing styles to the individual bars on a column chart to increase its visual impact. Drawing styles include wedge, emboss, cylinder and light-to-dark. These effects are designed to improve the appearance of your 2D chart. If you are using a 3D chart, the drawing styles will still be applied, but may not have the same effect. For more information about how to add a drawing style to a bar chart, see [Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#).
- Unique to column charts is the ability to show your chart as a histogram or Pareto chart. To do so, set the ShowColumnAs property to **Histogram** or **Pareto** in the Properties window to **true**.

See Also

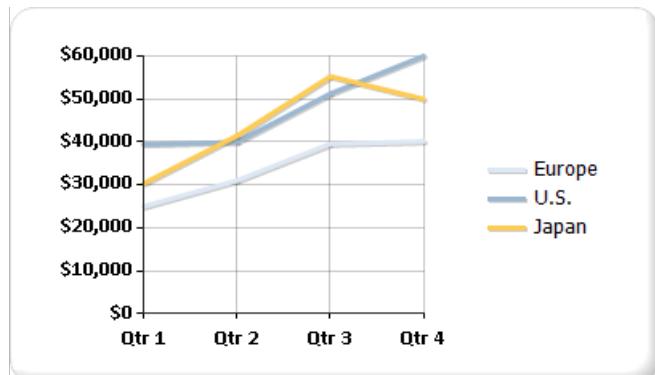
- [Charts \(Report Builder and SSRS\)](#)
- [Chart Types \(Report Builder and SSRS\)](#)
- [Bar Charts \(Report Builder and SSRS\)](#)
- [Range Charts \(Report Builder and SSRS\)](#)
- [Tutorial: Add a Bar Chart to Your Report \(Report Builder\)](#)
- [Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

Line Charts (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

A line chart displays a series as a set of points connected by a single line. Line charts are used to representing large amounts of data that occur over a continuous period of time. For more information about how to add data to a line chart, see [Charts \(Report Builder and SSRS\)](#).

The following illustration shows a line chart that contains three series.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Smooth line.** A line chart that uses a curved line instead of a regular line.
- **Stepped line.** A line chart that uses a stepped line instead of a regular line. The stepped line connects points by using a line that makes it look like steps on a ladder or staircase.
- **Sparkline charts.** Variations of the line chart that show only the line series in the cell of a table or matrix.
For more information, see [Sparklines and Data Bars \(Report Builder and SSRS\)](#).

Data Considerations for Line Charts

- To improve the visual impact of the default line chart, consider changing the width of the series border to 3, and adding a shadow offset of 1. This will create a much bolder line chart. You will need to revert these properties to their original values if you change the chart type from Line to another type.
- If your dataset includes empty values, the line chart will add empty points, in the form of placeholder lines, in order to maintain continuity on the chart. If you would rather not see these lines, consider displaying your dataset using a non-contiguous chart type such as a bar or column chart.
- A line chart requires at least two points to draw a line. If your dataset has only one data point, the line chart will display as a single data point marker.
- A series that is drawn as a line will not take up much space within a chart area. For this reason, line charts are frequently combined with other chart types such as column charts. However, you cannot combine a line chart with bar, polar, pie or shape chart types.

See Also

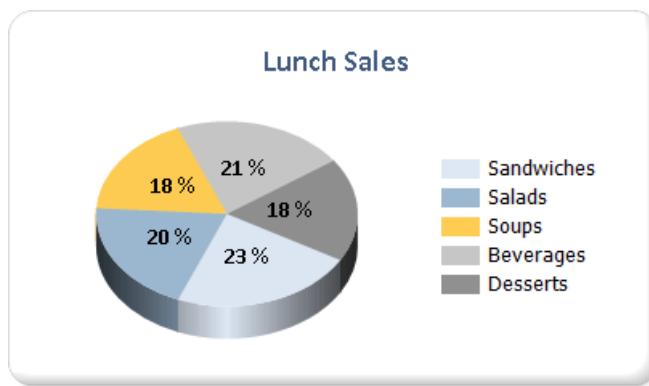
- [Bar Charts \(Report Builder and SSRS\)](#)
- [Column Charts \(Report Builder and SSRS\)](#)
- [Charts \(Report Builder and SSRS\)](#)
- [Chart Types \(Report Builder and SSRS\)](#)
- [Area Charts \(Report Builder and SSRS\)](#)
- [Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)
- [Charts \(Report Builder and SSRS\)](#)

Pie Charts (Report Builder and SSRS)

3/24/2017 • 5 min to read • [Edit Online](#)

Pie charts and doughnut charts display data as a proportion of the whole. Pie charts are most commonly used to make comparisons between groups. Pie and doughnut charts, along with pyramid and funnel charts, comprise a group of charts known as shape charts. Shape charts have no axes. When a numeric field is dropped on a shape chart, the chart calculates the percentage of each value to the total. For more information on shape charts, see [Shape Charts \(Report Builder and SSRS\)](#).

The following illustration shows a 3-D pie chart with data labels formatted as percentages. The legend is positioned in the right-center.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Exploded pie.** A pie chart where all of the slices are moved away from the center of the pie. In addition to the exploded pie chart, in which all slices are separated, you can create an exploded slice chart, in which only one slice is called out.
- **Doughnut.** A pie chart that has an open space in the center.
- **Exploded doughnut.** A doughnut chart where all of the slices are moved away from the center of the doughnut.
- **3D Pie.** A pie chart that has a 3-D style applied.
- **3D Exploded Pie.** An exploded pie chart that has a 3-D style applied.

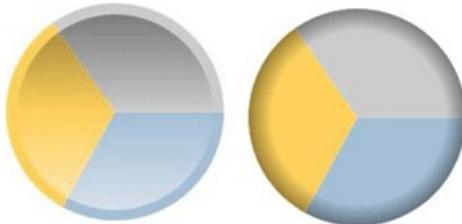
Data Considerations for Display on a Pie Chart

- Pie charts are popular in reports because of their visual impact. However, pie charts are a very simplified chart type that may not best represent your data. Consider using a pie chart only after the data has been aggregated to seven data points or less.
- Pie charts display each data group as a separate slice on the chart. You must add at least one data field and one category field to the pie chart. If more than one data field is added to a pie chart, the pie chart will display both data fields in the same chart.

- Null, empty, negative, and zero values have no effect when calculating ratios. For this reason, these values are not shown on a pie chart. If you want to visually indicate these types of values on your chart, change the chart type to be something other than a pie chart.
- If you are defining your own colors on a pie chart using a custom palette, be sure that you have enough colors in your palette to display each data point with its own unique color. For more information, see [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#).
- Unlike most other chart types, a pie chart will display individual data points, and not individual series, in its legend.
- A pie chart requires at least two values in order to make a valid comparison between proportions. If your pie chart contains only one color, verify that you have added a category field to group by. When the pie chart does not contain categories, it aggregates the values from your data field into one value for display.
- Like all other chart types, the pie chart generates colors based on the color values contained in the default palette. This approach might cause different pie charts to color data points differently when you are using multiple pie charts in a report. If you have several pie charts in your report, you might want to manually set colors for each category group in order to retain the same color across different charts. For more information about how to define colors on a chart, see [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#).

Applying Drawing Styles to a Pie Chart

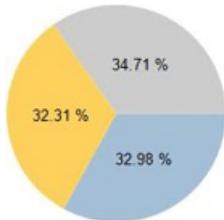
You can add special drawing styles to the pie chart to increase its visual impact. Drawing styles include bevel and concave effects. These effects are available only on a 2-D pie chart. The following illustration shows an example of the bevel and concave drawing styles on a pie chart.



For more information, see [Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#).

Displaying Percentage Values on a Pie Chart

Like other Shape charts, pie charts represent proportions of the total. As a result, it is common to format pie chart labels as percentages. In order to be consistent with other chart types, the chart does not display percentage labels by default. For more information about how to display values as percentages on the chart, see [Display Percentage Values on a Pie Chart \(Report Builder and SSRS\)](#). For more information about how to format numbers as percentages in your report, see [Formatting Numbers and Dates \(Report Builder and SSRS\)](#).



Preventing Overlapped Labels on a Pie Chart

If there are a lot of data points on a pie chart, the data labels will overlap. There are several ways to prevent labels from overlapping:

- Decrease the font size of the data point labels.
- Increase the width and height of your chart to allow more room for the labels.
- Display pie labels outside the chart area. For more information, see [Display Data Point Labels Outside a Pie Chart \(Report Builder and SSRS\)](#).
- Collect the small pie slices into one slice.

Consolidating Small Slices on a Pie Chart

When the pie chart has too many points, the data becomes obscured and hard to read. If your data has many small data points, there are two ways of collecting multiple pie slices:

- Collect smaller data slices into one slice on the pie chart. This is useful in situations where, for example, you want the pie chart to have an "Other" data point that simply collects the remaining data. For more information, see [Collect Small Slices on a Pie Chart \(Report Builder and SSRS\)](#).
- Collect small slices into a supplementary pie chart. The second pie chart does not display in the designer. Instead, during report processing, the chart calculates if a second pie chart is necessary to be shown, based on the values of the data points. If so, the values are added into another pie chart.

See Also

[Display Data Point Labels Outside a Pie Chart \(Report Builder and SSRS\)](#)

[Collect Small Slices on a Pie Chart \(Report Builder and SSRS\)](#)

[Display Percentage Values on a Pie Chart \(Report Builder and SSRS\)](#)

[Tutorial: Add a Pie Chart to Your Report \(Report Builder\)](#)

[Formatting the Legend on a Chart \(Report Builder and SSRS\)](#)

[Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

Polar Charts (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A polar chart displays a series as a set of points that are grouped by category on a 360-degree circle. Values are represented by the length of the point as measured from the center of the circle. The farther the point is from the center, the greater its value. Category labels are displayed on the perimeter of the chart. For more information on how to add data to a polar chart, see [Charts \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Radar chart.** A radar chart displays a series as a circular line or area. Unlike the polar chart, the radar chart does not display data in terms of polar coordinates.

Data Considerations for Polar Charts

- The radar chart is useful for comparisons between multiple series of category data.
- Polar charts are most commonly used to graph polar data, where each data point is determined by an angle and a distance.
- Polar charts cannot be combined with any other chart type in the same chart area.

Example

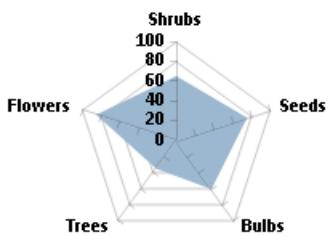
The following example shows how a radar chart can be used. The table below provides sample data for the chart.

NAME	SALES
Shrubs	61
Seeds	78
Bulbs	60
Trees	38
Flowers	81

In this example, the Name field is placed in the Category Groups area. The Sales field is placed in the Values area. The Sales field is automatically aggregated for the chart when you drop it. The radar chart calculates where to place the labels based on the number of values in the Sales field, which contains five values and places labels at five equidistant points on a circle. If the Sales field contained three values, the labels would be placed at three equidistant points on a circle.

The following illustration shows an example of a radar chart based on the data presented.

Garden Center Sales



See Also

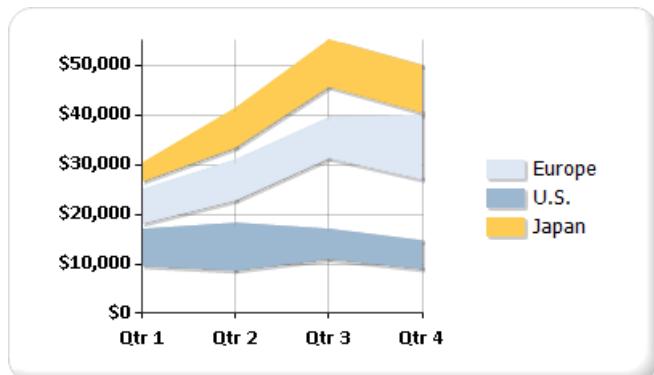
- [Charts \(Report Builder and SSRS\)](#)
- [Formatting a Chart \(Report Builder and SSRS\)](#)
- [Chart Types \(Report Builder and SSRS\)](#)
- [Line Charts \(Report Builder and SSRS\)](#)
- [Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

Range Charts (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A range chart type displays a set of data points that are each defined by multiple values for the same category. Values are represented by the height of the marker as measured by the value axis. Category labels are displayed on the category axis. The plain range chart fills in the area between the top and bottom value for each data point.

The following illustration shows a plain range chart with three series.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Smooth range.** A smooth range chart displays curved lines rather than straight.
- **Column range.** A column range chart uses columns instead of areas to display the ranges.
- **Bar range.** A bar range chart uses bars instead of areas to display the ranges.

Data Considerations for Range Charts

- Range chart types require two values per data point. These values correspond with a high value and a low value that defines the range for each data point.
- Range charts are useful for analysis only if the top values are always higher than the bottom values. If this is not the case, consider using a line chart. If the high value is lower the low value, the range chart will display the absolute value of the difference between these values.
- If only one value is specified, the range chart will display as if it were a regular area chart, with one value per data point.
- Range charts are often used to graph data that contains minimum and maximum values for each category group in the dataset.
- Displaying markers on each data point is not supported on the range chart.
- Like the area chart, in a plain range chart, if the values in multiple series are similar, the series will overlap. In this scenario, you may want to use a column range or bar range chart instead of a plain range chart.

- Gantt charts can be created using a range bar chart.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Chart Types \(Report Builder and SSRS\)](#)

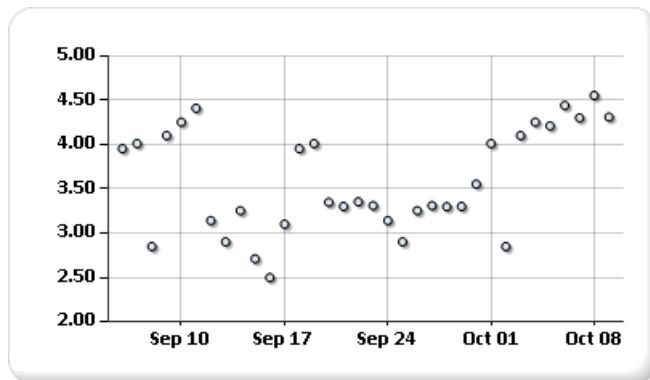
[Formatting a Chart \(Report Builder and SSRS\)](#)

Scatter Charts (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A scatter chart displays a series as a set of points. Values are represented by the position of the points on the chart. Categories are represented by different markers on the chart. Scatter charts are typically used to compare aggregated data across categories. For more information on how to add data to a scatter chart, see [Charts \(Report Builder and SSRS\)](#)

The following illustration shows an example of a scatter chart.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Bubble.** Bubble charts display the difference between two values of a data point based on the size of the bubble. The larger the bubble, the larger the difference between the two values.
- **3-D Bubble.** A bubble chart displayed in 3-D.

Data Considerations for a Scatter Chart

- Scatter charts are commonly used for displaying and comparing numeric values, such as scientific, statistical, and engineering data.
- Use the scatter chart when you want to compare large numbers of data points without regard to time. The more data that you include in a scatter chart, the better the comparisons that you can make.
- The bubble chart requires two values (top and bottom) per data point.
- Scatter charts are ideal for handling the distribution of values and clusters of data points. This is the best chart type if your dataset contains many points (for example, several thousand points). Displaying multiple series on a point chart is visually distracting and should be avoided. In this scenario, consider using a line chart.
- By default, scatter charts display data points as circles. If you have multiple series on a scatter chart, consider changing the marker shape of each point to be square, triangle, diamond, or another shape.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Chart Types \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Line Charts \(Report Builder and SSRS\)](#)

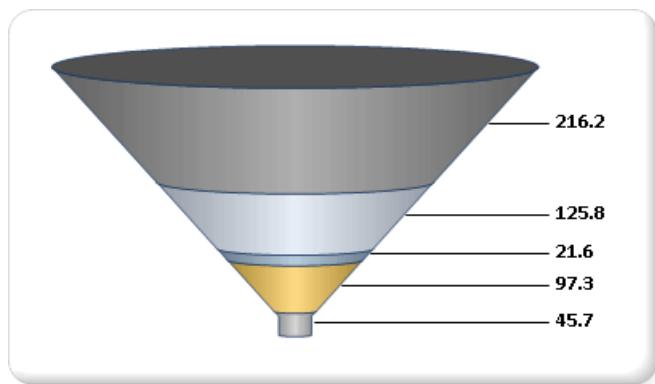
Shape Charts (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

A shape chart displays value data as percentages of a whole. Shape charts are typically used to show proportional comparisons between different values in a set. Categories are represented by individual segments of the shape. The size of the segment is determined by the value. Shape charts are similar in use to pie charts, except that they order categories from largest to smallest.

A funnel chart displays values as progressively decreasing proportions. The size of the area is determined by the series value as a percentage of the total of all values. For example, you might use a funnel chart to display Web site visitor trends. It is likely that the funnel chart will display a wide area at the top, indicating visitor page hits to the homepage, and the other areas will be proportionally smaller. For more information about how to add data to a funnel chart, see [Charts \(Report Builder and SSRS\)](#).

The following illustration shows an example of a funnel chart.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Pyramid.** A pyramid chart displays proportional data so that the chart looks like a pyramid.

Data Considerations for Shape Charts

- Shape charts are popular in reports because of their visual impact. However, shape charts are a very simplified chart type that may not best represent your data. Consider using a shape chart only once the data has been aggregated to seven data points or less. In general, use the shape chart to display only one category per data region.
- Shape charts display each data group as a separate segment of the chart. You must add at least one data field and one category field. If more than one data field is added to a shape chart, the shape chart will display both data fields in the same chart.
- Shape charts are most effective for showing proportional percentages in sorted order. However, in order to maintain consistency, the chart does not sort the values in your dataset by default. Consider ordering your values from highest to lowest to most accurately represent your data as a funnel or a pyramid. For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).

- Null, empty, negative and zero values have no effect when calculating ratios. For this reason, these values are not shown on a shape chart. If you want to visually indicate these types of values on your chart, change the chart type to be something other than a shape chart. For more information about how to add empty points to a non-shape chart, see [Add Empty Points to a Chart \(Report Builder and SSRS\)](#).
- If you are defining your own colors on a shape chart using a custom palette, be sure that you have enough colors in your palette to highlight each data point with its own unique color. For more information, see [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#).
- Unlike all other chart types, a shape chart will display individual data points, and not individual series, in its legend.
- Settings for the value and category axis are ignored for funnel charts. If you have multiple category or series groups, the group labels are displayed in the chart legend.
- Shape chart types cannot be combined with any other chart type in the same chart area. If you have to show comparisons between data displayed on a shape chart, and data displayed on another chart type, you will need to add a second chart area.
- You can apply additional drawing styles to pie and doughnut charts for increased visual impact. See [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#) for more information.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#)

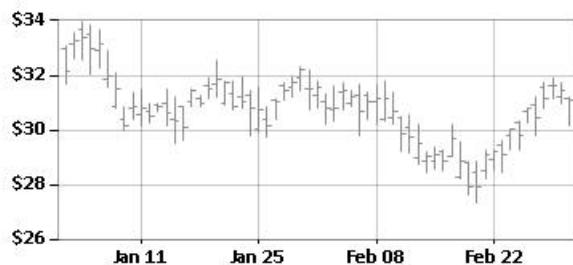
[Pie Charts \(Report Builder and SSRS\)](#)

Stock Charts (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A stock chart is specifically designed for financial or scientific data that uses up to four values per data point. These values align with the high, low, open and close values that are used to plot financial stock data. This chart type displays opening and closing values by using markers, which are typically lines or triangles. In the following example, the opening values are shown by the markers on the left, and the closing values are shown by the markers on the right.

Fiscal Quarter Stock Price



An example of a stock chart is available as a sample SQL Server 2016 Report Builder report. For more information about downloading this sample report and others, see [SQL Server 2016 Report Builder and Report Designer sample reports](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Variations

- **Candlestick.** The candlestick chart is a specialized form of the stock chart, wherein boxes are used to show the range between the open and close values. Like the stock chart, the candlestick chart can display up to four values per data point.

Data Considerations for Stock Charts

- When presenting many stock data points, such as annual stock price trend, it is difficult to distinguish each open, close, high and low value of each data point. In this scenario, consider using a line chart instead of a stock chart.
- When axis labels are generated, labeling usually begins at zero. In general, stock prices do not fluctuate to the same degree as other data sets. For this reason, you may want to disable the axis labels from starting at zero, in order to get a better view of your data. To do this, set **IncludeZero** to **false** in the **Axis Properties** dialog box or the Properties window. For more information about how the chart generates axis labels, see [Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#).
- Reporting Services provides many calculated formulas for use with stock charts, including Price Indicator, Relative Strength Index, MACD and more.

See Also

[Range Charts \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Axis Properties Dialog Box, Axis Options \(Report Builder and SSRS\)](#)

Tree Map and Sunburst Charts in Reporting Services

3/24/2017 • 5 min to read • [Edit Online](#)

 **Need help?** [MSDN Forum](#), [Stackoverflow](#), [Connect](#)

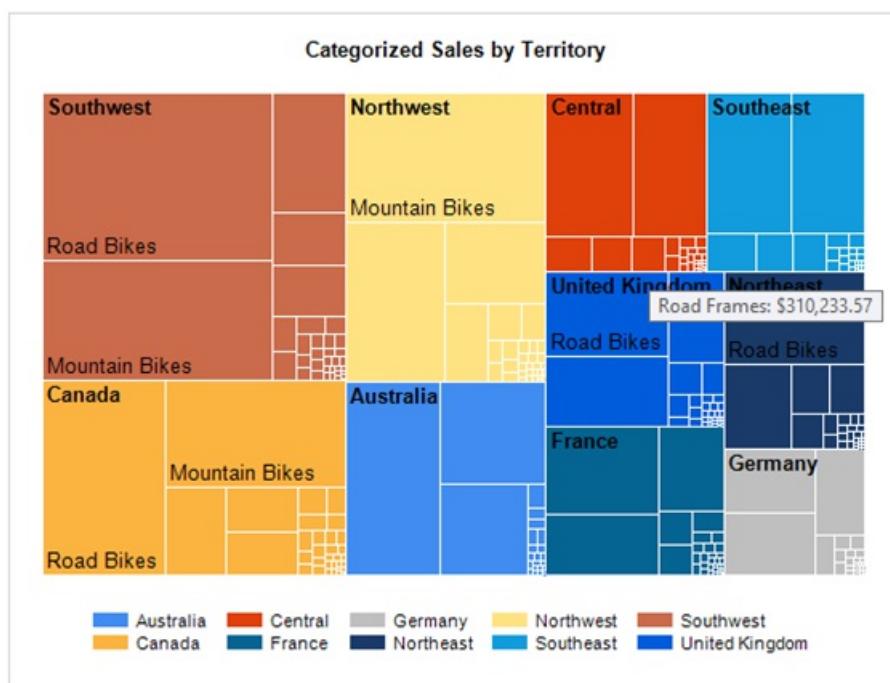
The Reporting Services Tree Map and sunburst visualizations are great for visually representing hierarchical data. This topic is an overview of how to add a Tree Map or Sunburst chart to a Reporting Services report. The topic also includes a sample Adventureworks query to help get you started.

Tree Map Chart



A tree map chart divides the chart area into rectangles that represent the different levels and relative sizes of the data hierarchy. The map is similar to branches on a tree that start with a trunk and divide into smaller and smaller branches. Each rectangle is broken into smaller rectangles representing the next level in the hierarchy. The top level tree map rectangles are arranged with the largest rectangle in the upper left corner of the chart to the smallest rectangle in the lower right corner. Within a rectangle, the next level of the hierarchy is also arranged with rectangles from the upper left to the lower right.

For example, in the following image of the sample tree map, the Southwest territory is the largest and Germany is the smallest. Within the Southwest, Road Bikes are larger than Mountain Bikes.

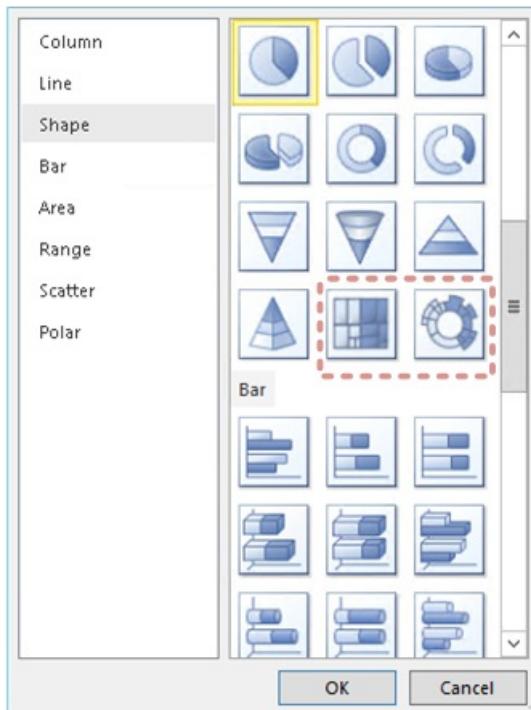


To insert a tree map chart and configure for the sample Adventureworks data

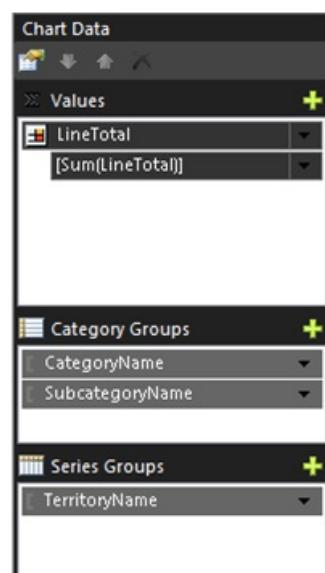
Note: Before you add a chart to your report, create a data source and dataset. For sample data and a sample query, see the section [Sample Adventureworks data](#) in this topic.

1. Right-click the design surface, click **Insert**, and then click **Chart**.

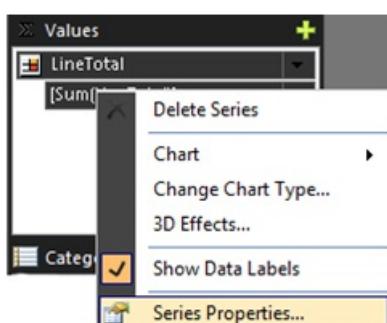
Select Tree Map .



2. Reposition and resize the chart. For use with the sample data, a chart that is 5 inches wide is a good start.
3. Add the following fields from the sample data:

	<p>Values: LineTotal</p> <p>Category Groups: Add them in the order of:</p> <ol style="list-style-type: none"> 1) CategoryName 2) SubcategoryName <p>Series Groups: TerritoryName</p>
--	---

4. To optimize the page size for general shape of a Tree Map, set the legend position to the bottom.
5. To add tool tips that display the subcategory and the line total, right-click **LineTotal** and then click **Series Properties**.



Set the **Tooltip** property to the following:

```
=Fields!SubcategoryName.Value &": " &Format(Sum(Fields!LineTotal.Value),"C")
```

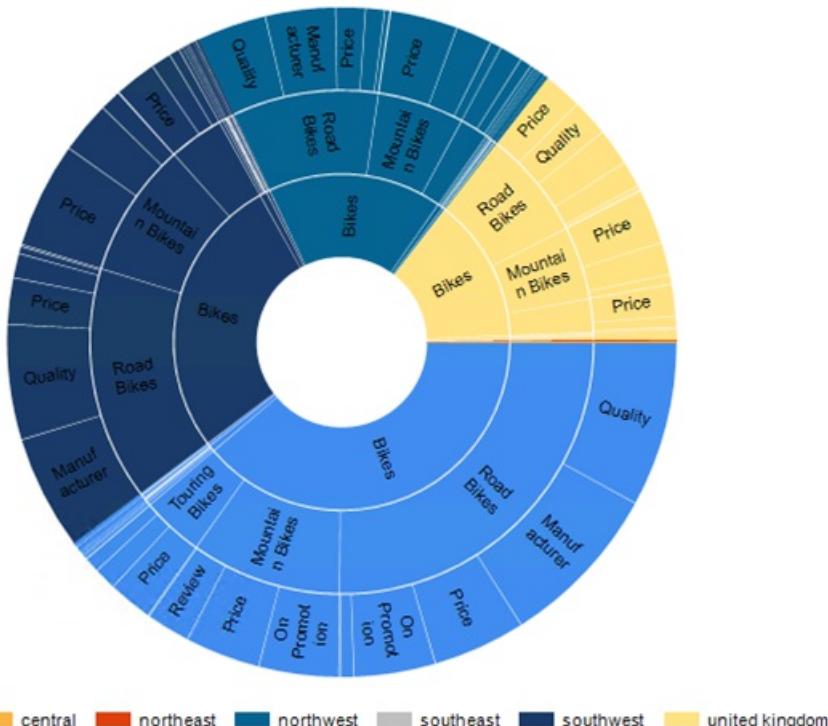
For more information, see [Show ToolTips on a Series \(Report Builder and SSRS\)](#).

6. Change the default chart title to "Categorized Sales by Territory".
 7. The number of label values that are displayed are affected by the size of the font, the size of the overall chart area, and the size of specific rectangles. To see more of the labels, change the Label font property of LineTotal to 10pt instead of the default 8pt.

Sunburst Chart



In a sunburst chart, the hierarchy is represented by a series of circles, with the highest level of the hierarchy in the center and lower levels of the hierarchy as rings displayed outside the center. The lowest level of the hierarchy is the outside ring.



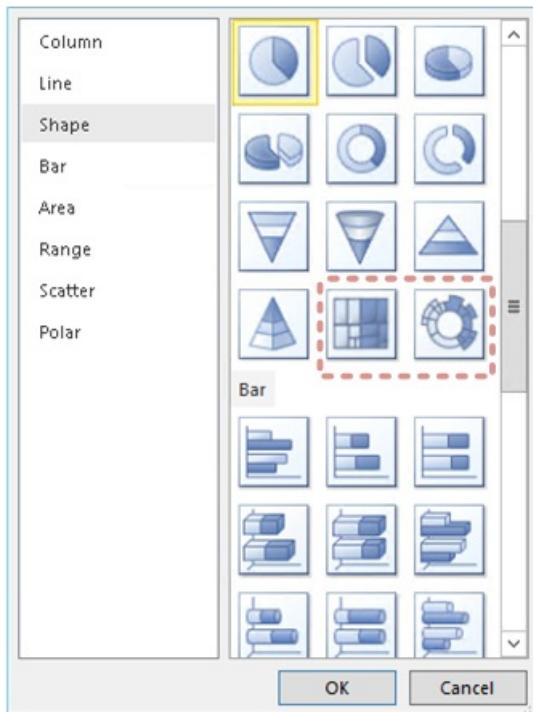
To insert a sunburst chart and configure for the sample Adventureworks data

Note: Before you add a chart to your report, create a data source and dataset. For sample data and a sample query, see the section [Sample Adventureworks data](#) in this topic.

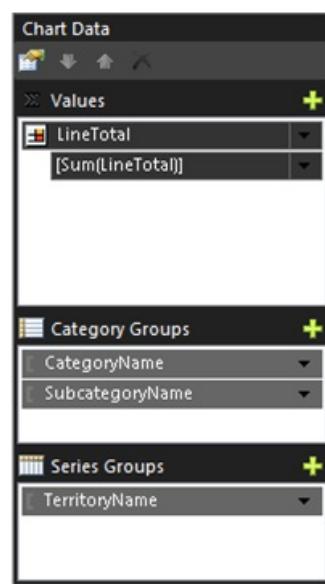
1. Right-click the design surface, click **Insert**, and then click **Chart**.



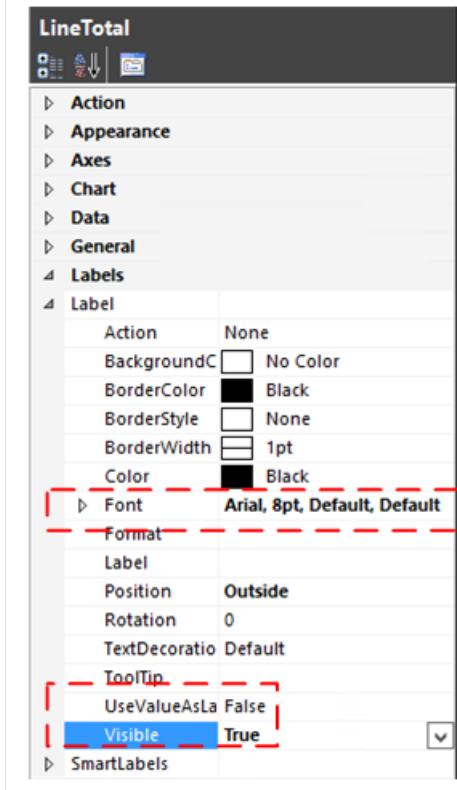
Select Sunburst



2. Reposition and resize the chart. For use with the sample data., a chart that is 5 inches wide is a good start.
3. Add the following fields from the sample data:

	<p>Values: LineTotal</p> <p>Category Groups: Add them in the order of :</p> <ol style="list-style-type: none"> 1) CategoryName 2) SubcategoryName, 3) SalesReasonName <p>Series Groups: TerritoryName .</p>
--	---

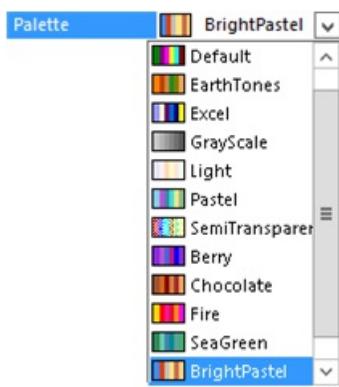
4. To optimize the page size for the general shape of a Sunburst, set the legend position to the bottom.
5. Change the default chart title to "Categorized Sales by Territory, with sales reason".
- 6.



To add the values of the category groups to the sunburst as labels, set the label property **Visible** = true and the **UseValueAsLabel**=False.

The label values that are displayed are affected by the size of the font, the size of the overall chart area, and the size of specific rectangles. To see more of the labels, change the Label font property of LineTotal to 8pt instead of the default 10pt.

- If you want a different range of colors, change the chart **Palette** property.



Sample Adventureworks data

This sections includes a sample query and the basic steps for creating a data source and dataset in Report Builder. If your report already contains a data source and dataset you can skip this section.

The query returns Adventureworks sales order detail data with sales territory, product category, product sub category, and sales reason data.

1. Get the Data:

The query in this section is based on the Adventureworks database which is available for download from [Adventure Works 2014 Full Database Backup](#).

For more information on how to install the database, see [How to install Adventure Works 2014 Sample Databases.pdf](#).

2. Create a data source:

- In the **Report Data** pane, right-click **Data Sources** and click **Add data source**.
- Select **Use a connection embedded in my report**.

- c. Select the connection type of **Microsoft SQL Server**.
- d. Type in the connection string to your server and database, for example the following:

```
Data Source=[server name];Initial Catalog=AdventureWorks2014
```

- e. It is a good idea to verify with the **Test Connection** button and then click **OK**.

For more information on creating a data source, see [Add and Verify a Data Connection \(Report Builder and SSRS\)](#).

3. Create a dataset:

- In the **Report Data** pane, right-click **Datasets** and click **Add dataset**.
- Select **Use a dataset embedded in my report**.
- Select the data source you created in the previous steps.
- Select the **Text** query type and then copy and paste the following query into the **Query:** text box:

```

SELECT      Sales.SalesOrderHeader.SalesOrderID, Sales.SalesOrderHeader.OrderDate,
Sales.SalesOrderDetail.SalesOrderDetailID, Sales.SalesOrderDetail.ProductID,
Sales.SalesOrderDetail.LineTotal,
                           Sales.SalesOrderDetail.UnitPrice, Sales.SalesOrderDetail.OrderQty,
Production.Product.Name, Production.Product.ProductNumber, Sales.SalesTerritory.TerritoryID,
lower(Sales.SalesTerritory.Name) AS TerritoryName,
                           Production.ProductSubcategory.Name AS SubcategoryName,
Production.ProductCategory.Name AS CategoryName, Sales.SalesReason.SalesReasonID,
Sales.SalesReason.Name AS SalesReasonName
FROM          Sales.SalesOrderDetail INNER JOIN
                           Sales.SalesOrderHeader ON Sales.SalesOrderDetail.SalesOrderID =
Sales.SalesOrderHeader.SalesOrderID INNER JOIN
                           Production.Product ON Sales.SalesOrderDetail.ProductID =
Production.Product.ProductID INNER JOIN
                           Sales.SalesTerritory ON Sales.SalesOrderHeader.TerritoryID =
Sales.SalesTerritory.TerritoryID AND Sales.SalesOrderHeader.TerritoryID =
Sales.SalesTerritory.TerritoryID AND
                           Sales.SalesOrderHeader.TerritoryID = Sales.SalesTerritory.TerritoryID
INNER JOIN
                           Production.ProductSubcategory ON Production.Product.ProductSubcategoryID
= Production.ProductSubcategory.ProductSubcategoryID AND
                           Production.Product.ProductSubcategoryID =
Production.ProductSubcategory.ProductSubcategoryID AND
                           Production.Product.ProductSubcategoryID =
Production.ProductSubcategory.ProductSubcategoryID INNER JOIN
                           Production.ProductCategory ON
Production.ProductSubcategory.ProductCategoryID = Production.ProductCategory.ProductCategoryID
AND
                           Production.ProductSubcategory.ProductCategoryID =
Production.ProductCategory.ProductCategoryID AND
                           Production.ProductSubcategory.ProductCategoryID =
Production.ProductCategory.ProductCategoryID INNER JOIN
                           Sales.SalesOrderHeaderSalesReason ON Sales.SalesOrderHeader.SalesOrderID
= Sales.SalesOrderHeaderSalesReason.SalesOrderID AND
                           Sales.SalesOrderHeader.SalesOrderID =
Sales.SalesOrderHeaderSalesReason.SalesOrderID AND Sales.SalesOrderHeader.SalesOrderID =
Sales.SalesOrderHeaderSalesReason.SalesOrderID AND
                           Sales.SalesOrderHeader.SalesOrderID =
Sales.SalesOrderHeaderSalesReason.SalesOrderID INNER JOIN
                           Sales.SalesReason ON Sales.SalesOrderHeaderSalesReason.SalesReasonID =
Sales.SalesReason.SalesReasonID AND
                           Sales.SalesOrderHeaderSalesReason.SalesReasonID =
Sales.SalesReason.SalesReasonID AND Sales.SalesOrderHeaderSalesReason.SalesReasonID =
Sales.SalesReason.SalesReasonID

```

- click **OK**.

For more information on creating a dataset, see [Create a Shared Dataset or Embedded Dataset \(Report Builder and SSRS\)](#).

See Also

- [Shared Dataset Design View \(Report Builder\)](#)
- [Show ToolTips on a Series \(Report Builder and SSRS\)](#)
- [Tutorial: Treemaps in Power BI](#)
- [Treemap: Microsoft Research Data Visualization Apps for Office](#)

Formatting a Chart (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

After you have defined the data for your chart and it is appearing the way you want, you can add formatting to improve the overall appearance and highlight key data points. The most common formatting options can be modified from the Properties dialog box that are found when you right-click a chart element to display its shortcut menu. Each chart element has its own dialog box. For more information about chart elements, see [Charts \(Report Builder and SSRS\)](#).

All properties that relate to the chart are located in the Properties pane, but many of these properties can also be set from a dialog box. If you are formatting the series, you can specify series-specific properties using custom attributes, which can only be found in the **CustomAttributes** category of properties, located in the Properties pane.

To effectively format the chart using a minimal number of steps, change the default border style, palette and drawing style. These three features produce the largest visible change on the chart. Drawing styles are only applicable to pie, doughnut, bar and column charts.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

In This Section

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

Discusses how colors are defined using a palette, how you can define your own color palette, and how to define colors based on an expression.

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

Discusses how to display gridlines, tick marks, and titles, and how to format numbers and dates on the axis scale.

[Formatting the Legend on a Chart \(Report Builder and SSRS\)](#)

Discusses how to re-order and format items in the chart legend.

[Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)

Discusses how to position data point labels and format data point markers for every series on the chart.

[3D, Bevel, and Other Effects in a Chart \(Report Builder and SSRS\)](#)

Discusses various 3D effects that you can apply to a chart.

[Add a Border Frame to a Chart \(Report Builder and SSRS\)](#)

Explains how to add a border frame to a chart.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Add a Border Frame to a Chart \(Report Builder and SSRS\)](#)

[Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)](#)

[Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#)

Formatting Series Colors on a Chart (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Reporting Services provides several built-in palettes for charts, or you can define a custom palette. By default, charts use the built-in **Pacific** color palette to fill each series. These colors also appear in the legend. When multiple series are added to the chart, the chart assigns the series a color in the order that the colors have been defined in the palette.

If there are a greater number of series than there are colors in the palette, the chart will begin reusing colors, and two series may have the same color. This frequently occurs if you are using a Shape chart, where each data point is assigned a color from the palette. To avoid confusion, define a custom palette with at least the same number of colors as you have series on your chart.

You can select a new palette or define a custom palette from the Properties pane. For more information, see [Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Using Built-In Palettes

Reporting Services provides a list of predefined, built-in palettes that you can use to define a color set for series on your chart. All built-in palettes contain between 10 and 16 color values. You cannot extend the built-in palette to include more colors, so if you need more than 16 colors, you must define a custom palette.

If you are printing a report, consider using the **Greyscale** palette. This palette uses shades of black and white to represent each series in a chart.

The palette named Default was used as the default chart palette in earlier versions of Reporting Services. It has been maintained with the same name for consistency. Charts will upgrade seamlessly using the Default palette, but after upgrading, you might consider changing it.

Using Custom Palettes

If you want to apply your own colors to the chart, use a custom palette. A custom palette let you add your own colors in the order you want them to appear on the chart. A custom palette is especially helpful if the number of series in your chart is unknown at design time. For more information, see [Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)](#).

Using a Color Fill on Each Series

You can also define your own colors on the chart by specifying a color for each series on the chart. To do this, open the **Series Properties** dialog box and set the **Color** property for **Fill**. This approach will override all defined palettes. Generally, it is better to use a custom palette to define your own colors because the number of series in your dataset may not be known until report processing.

This approach is best suited when you want to conditionally set the color of the series based on an expression.

For more information, see [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#).

In This Section

[Specify Consistent Colors across Multiple Shape Charts \(Report Builder and SSRS\)](#)

[Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)\)](#)

[Highlight Chart Data by Adding Strip Lines \(Report Builder and SSRS\)](#)

See Also

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Formatting the Legend on a Chart \(Report Builder and SSRS\)](#)

Specify Consistent Colors across Multiple Shape Charts (Report Builder and SSRS)

3/29/2017 • 3 min to read • [Edit Online](#)

On non-shape charts in a paginated report, Reporting Services selects a new color from the palette based on the index of series in the chart. For example, the first series on your chart will be mapped to the first color in the palette. However, this behavior differs for shape charts. On shape charts, each color in the palette is mapped to a data point in the dataset. For example, data point 1 is mapped to the first color in the palette, data point 2 is mapped to the second color palette and so on.

If a data point has no value, it is omitted from display on a shape chart. This means that the data point is skipped from being colored. For example, if point 2 has a value of zero, point 1 will be mapped to the first color in the palette, and point 3 will be mapped to the second color in the palette. This approach is useful because the empty points in the dataset of a pie chart do not unnecessarily use a palette color when the empty point does not need to be drawn.

As a side effect, when multiple pie charts are displayed in a report, the pie charts may display different colors for data points that have the same category grouping. To resolve this, you need to define individual colors that map to a category group instead of individual data values. How you do this depends on if the shape charts are sparklines in a table or matrix, or if they are shape charts in the report itself.

The legend is connected to the series, so any color you specify for the series will automatically be shown on the legend.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To specify consistent colors across multiple sparkline shape charts in a table or matrix

1. Click the chart to display the Chart Data pane.
2. In the **Category Groups** area, right-click a category and click **Category Group Properties**.
3. On the General tab, in the **Synchronize groups in** box, click the name of the category for which you would like to synchronize colors, and then click **OK**.

To specify consistent colors across multiple shape charts

1. Right-click outside the body of the report, and select **Report Properties**.
2. In **Code**, type the following code into the textbox.

```

Private colorPalette As String() = {"Color1", "Color2", "Color3"}
Private count As Integer = 0
Private mapping As New System.Collections.Hashtable()
Public Function GetColor(ByVal groupingValue As String) As String
    If mapping.ContainsKey(groupingValue) Then
        Return mapping(groupingValue)
    End If
    Dim c As String = colorPalette(count Mod colorPalette.Length)
    count = count + 1
    mapping.Add(groupingValue, c)
    Return c
End Function

```

NOTE

You will need to replace the "Color1" strings with your own colors. You can use named colors, for example "Red", or you can use six-digit hexadecimal value that represent the color, such as "#FFFFFF" for black. If you have more than three colors defined, you will need to extend the array of colors so that the number of colors in the array matches the number of points in your shape chart. You can add new colors to the array by specifying a comma-separated list of string values that contain named colors or hexadecimal representations of colors.

3. Click **OK**.
4. Right-click on the shape chart and select **Series Properties**.
5. In **Fill**, click the **Expression** (fx) button to edit the expression for the **Color** property.
6. Type the following expression, where "MyCategoryField" is the field that is displayed in the **Category Groups** area:

```
=Code.GetColor(Fields!MyCategoryField)
```

See Also

- [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)
- [Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#)
- [Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)](#)
- [Add Empty Points to a Chart \(Report Builder and SSRS\)](#)
- [Shape Charts \(Report Builder and SSRS\)](#)
- [Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)
- [Nested Data Regions \(Report Builder and SSRS\)](#)
- [Sparklines and Data Bars \(Report Builder and SSRS\)](#)

Define Colors on a Chart Using a Palette (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

You can change the color palette for a chart by selecting a pre-defined palette or defining a custom palette. Custom palettes are report-specific.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To change the colors on the chart using a built-in color palette

1. Open the Properties pane.
2. On the design surface, click the chart. The properties for the chart object are displayed in the Properties pane. The object name (**Chart1** by default) appears in the drop-down list at the top of the Properties pane.
3. In the **Chart** section, for the **Palette** property, select a new palette from the drop-down list.

NOTE

You cannot change the colors or order in a pre-defined palette.

To define your own colors on the chart using a custom color palette

1. Open the Properties pane.
2. On the design surface, click the chart. The properties for the chart object are displayed in the Properties pane.
3. In the **Chart** section, for the **Palette** property, select **Custom**.
4. In the **CustomPaletteColors** property, click the Edit Collection (...) button. The **ReportColorExpression Collection Editor** opens.
5. Click **Add** to add a color. Select a color from the drop-down list or select Expression and specify a hex value for a specific color, such as ff6600 for "Orange".

For more information about hex values, see [Color Table](#) on MSDN.

6. Click **Add** to add more colors to the palette.
7. When you are done, click **OK**.

If you are using a custom palette, you can change the order of the colors to change the color of different series in the chart.

See Also

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Specify Consistent Colors across Multiple Shape Charts \(Report Builder and SSRS\)](#)

Highlight Chart Data by Adding Strip Lines (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Strip lines, or strips, are horizontal or vertical ranges that shade the background of the chart in regular or custom intervals. You can use strip lines to:

- Improve readability for looking up individual values on the chart. Specify strip lines at regular intervals to help separate data points when reading the chart.
- Highlight dates that occur at regular intervals. For example, in a sales report you might use strip lines to identify weekend data points.
- Highlight a specific key range. Using the previous example, you can use one strip line to highlight the highest range of sales that is between \$80-100.

Strip lines are not applicable to Shape or Polar chart types.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To display interlaced strip lines at regular intervals on a chart

1. To show horizontal strip lines, right-click the vertical chart axis and click **VerticalAxis Properties**.

To show vertical strip lines, right-click the horizontal chart axis and click **Horizontal Axis Properties**.

2. Select the **Use interlacing** option. Grey strip lines will appear on your chart.

3. (Optional) Specify a color for the strip lines using the adjacent **Color** drop-down list.

To display interlaced strip lines at custom intervals on a chart

1. To show horizontal strip lines, right-click the vertical chart axis and click **VerticalAxis Properties**.

To show vertical strip lines, right-click the horizontal chart axis and click **Horizontal Axis Properties**.

The axis properties are displayed in the Properties window.

2. In the **Appearance** section of the Properties pane, for the StripLines property, click the Edit Collection (...) button to open the **ChartStripLine Collection Editor**.

3. Click **Add** to add a new strip line to the collection.

4. Click StripWidth to specify the width of the strip line, measured in inches on the report. If you are highlighting dates or times, click StripWidthType and select a time interval.

5. Type a value or expression for the Interval to specify how often the strip line will repeat. For example, if you specify an interval of 10, and your strip line width is 5, strip lines will display at values of 0 to 5, 15 to 20, 30 to 35, and so on.

NOTE

By default, Interval is set to Auto, which means the chart will not calculate an interval for custom strip lines. The chart only calculates intervals for strip lines if an interval value is set.

See Also

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Add a Moving Average to a Chart \(Report Builder and SSRS\)](#)

Formatting Axis Labels on a Chart (Report Builder and SSRS)

3/24/2017 • 7 min to read • [Edit Online](#)

Coordinate-based chart types (column, bar, area, point, line, and range) have two axes that are used to categorize and display data relationships. Different types of formatting will be applied to each axis.

You can format axes by using the **Axis Properties** dialog box or by using the Properties pane. Right-click the axis you want to format and click **Axis Properties** to change values for the axis text, numeric and date formats, major and minor tick marks, auto-fitting for labels, and the thickness, color, and style of the axis line. To change values for the axis title, right-click the axis title, and click **Axis Title Properties**.

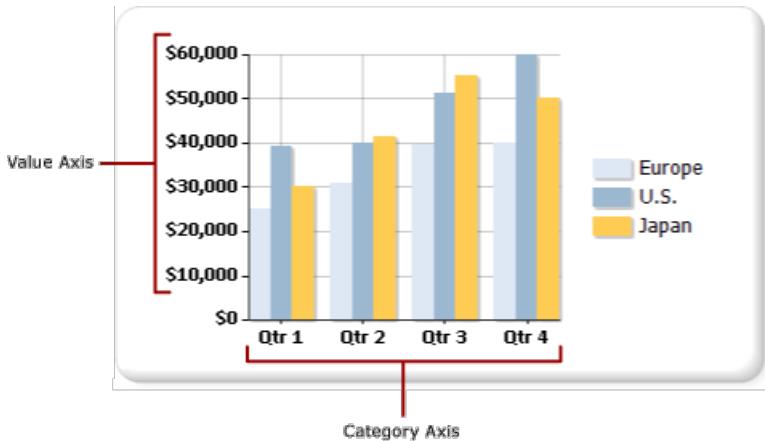
Axis labels identify major intervals on the chart. By default, the chart uses an algorithm to determine how the labels should be optimally placed on the axis to avoid overlapping text.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Types of Axes

The chart has two primary axes: the value axis and the category axis.



When you drag a field from your dataset onto the chart surface, the chart will determine whether this field belongs on the category or value axis.

The value axis is usually the vertical axis, or y-axis, of the chart. It is used to display numeric data values that are being charted. A field that is dragged into the data fields region will be plotted on the value axis. The category axis is usually the horizontal axis, or x-axis, of the chart. For bar charts, these axes are reversed. In bar chart types, the category axis is the vertical axis and the value axis is the horizontal axis. For more information, see [Bar Charts \(Report Builder and SSRS\)](#).

How the Chart Calculates Axis Label Intervals

Before you format axis labels, you should understand how the chart calculates axis label intervals. This will enable you to set the properties necessary to achieve the axis labeling behavior that you want.

The axis scale is bound by a minimum and a maximum value that define the data range to be displayed along the axis. The chart calculates the minimum and maximum value along each axis based on the values in your result set. On the value axis, the scale will always be determined by the smallest and largest number in the value field. On the category axis, the minimum and maximum value types are determined depending on the type of your category field. Any field in a dataset can be categorized into one of three category field types. The following table illustrates these three types of category fields.

CATEGORY FIELD TYPE	DESCRIPTION	EXAMPLE
Numeric	Categories are plotted in numeric order along the x-axis.	A sales report by employee identification number displays the employee identification numbers along the x-axis.
Date/time	Categories are plotted in chronological order along the x-axis.	A sales report by month displays formatted dates along the x-axis.
Strings	Categories are plotted in the order it first appears in the data source along the x-axis.	A sales report by region displays region names along the x-axis.

All chart types with two axes are designed to suppress some axis labels when there are too many categories to fit in order to produce a cleaner image on the chart and avoid label collisions.

The application calculates where labels are placed on an axis according to the following steps:

1. Minimum and maximum values are identified based on the values in the result set.
2. An equidistant number of axis intervals, usually between four and six, are calculated based on these minimum and maximum values.
3. Based on the axis label properties, labels are displayed at these intervals. Properties that affect label placement include font size, the angle at which the labels are displayed, and text wrapping properties. These axis label auto-fit options can be changed.

Example of How the Chart Calculates Axis Labels

The table shown here contains sample sales data to be plotted on a column chart. The Name field is added to the Category Groups area, and the Quantity field is added to the Values area.

NAME	QUANTITY
Michael Blythe	229
Jae Pak	112
Ranjit Varkey Chudukatil	494
Jillian Carson	247
Linda Mitchell	339
Rachel Valdez	194

The Quantity field is plotted along the value -axis. The lowest value is 112 and the highest value is 494. In this case, the chart calculates the scale to start at 0 and end at 500. The chart also calculates five equidistant intervals of 100, and creates labels at 0, 100, 200, 300, 400, and 500.

The Name field is plotted along the category axis. The chart calculates between four and six labels and it calculates auto-fit settings to determine how the labels can fit on the category axis without causing label collisions. As a result, some category labels might be omitted. You can override auto-fitting options for each axis independently.

Displaying All Labels on the Category Axis

On the value axis, axis intervals provide a consistent measure of the data points on the chart. However, on the category axis, this functionality can cause categories to appear without axis labels. Typically, you want all categories to be labeled. You can set the number of intervals to 1 to show all categories. For more information, see [Specify an Axis Interval \(Report Builder and SSRS\)](#).

NOTE

By superseding the automatic labeling features with a manual interval on an axis, the chart must resize all other elements appropriately. As a result, you may encounter unpredictable results with the sizing and positioning of the labels, or the size of other elements on the chart.

Variable Axis Intervals

The chart calculates approximately five axis label intervals regardless of the size of the chart. On wider or taller charts, if you show only five labels on an axis, large gaps can appear between each label. This makes it more difficult to identify the value of each data point against the axis. To avoid this behavior on wider or taller charts, you can set a variable axis interval. The chart will calculate the optimal number of labels that can appear on the axis based on the width or height of the chart, depending on the corresponding axis. For more information, see [Specify an Axis Interval \(Report Builder and SSRS\)](#).

Sorting Axis Values

Categories appear along the x-axis in the order that they appear in the result set. You can change the group order by adding a SORT command to the query or by sorting the dataset using an expression. Chart data regions are sorted the same as all other data regions. For more information about how to sort data, see [Sort Data in a Data Region \(Report Builder and SSRS\)](#).

Specifying Scalar Values on the Category Axis

By default, the chart will only display axis labels for data points in the dataset that contain valid values. For example, if you have values of 1, 2, and 6 on the category axis, the chart will only show categories 1, 2, and 6. To maintain the scale of category values, you can specify the chart to use a scalar axis. In this scenario, the chart will show labels for 1-6 on the x-axis of the chart, even though your dataset does not contain values for 3-5.

There are two ways to set a scalar axis:

- Select the **Scalar axis** option in the **Axis Properties** dialog box. This will add numeric or date/time values to the axis where no data grouping values exist. For more information, see [Axis Properties Dialog Box, Axis Options \(Report Builder and SSRS\)](#).
- Select a field or type an expression for the **Category field** option in the **Series Properties** dialog box. The chart will add axis intervals for all values in the category field you specified.

Adding or Removing Side Margins from the Category Axis

In Bar, Column and Scatter chart types, the chart automatically adds side margins on the ends of the x-axis. You cannot change the size of the margin. In all other chart types, the chart does not add side margins. For more

information, see [Add or Remove Margins from a Chart \(Report Builder and SSRS\)](#).

In This Section

[Format Axis Labels as Dates or Currencies \(Report Builder and SSRS\)](#)

[Position Labels in a Chart \(Report Builder and SSRS\)](#)

[Specify an Axis Interval \(Report Builder and SSRS\)](#)

[Add or Remove Margins from a Chart \(Report Builder and SSRS\)](#)

[Specify a Logarithmic Scale \(Report Builder and SSRS\)](#)

See Also

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)

Format Axis Labels as Dates or Currencies (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

When you show properly formatted DateTime values on an axis in a Reporting Services paginated report, a chart will automatically display these values as days. To specify a date/time interval for the x-axis, such as an interval of months or an interval of hours, you must format the axis labels and set the type of axis interval to a valid date or time interval.

NOTE

In column and scatter charts, the horizontal, or x-axis, is the category axis. In bar charts, the vertical, or y-axis, is the category axis.

In order to format time intervals correctly, the values displayed on the x-axis must evaluate to a [DateTime](#) data type. If your field has a data type of [String](#), the chart will not calculate intervals as dates or times. For more information, see [Charts \(Report Builder and SSRS\)](#).

When a numeric value is added to the y-axis, by default, the chart does not format the number before displaying it. If your numeric field is a sales figure, consider formatting the numbers as currencies to increase the readability of the chart.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To format x-axis labels as monthly intervals

1. Right-click the horizontal, or x-axis, of the chart, and select **HorizontalAxis Properties**.
2. In the **HorizontalAxis Properties** dialog box, select **Number**.
3. From the **Category** list, select **Date**. From the **Type** list, select a date format to apply to the x-axis labels.
4. Select **Axis Options**.
5. In **Interval**, type **1**. In **Interval type** property, select **Months**.

NOTE

If you do not specify an interval type, the chart will calculate intervals in terms of days.

6. Click **OK**.

To format y-axis labels using a currency format

1. Right-click the vertical, or y-axis, of the chart, and select **VerticalAxis Properties**.
2. In the **VerticalAxis Properties** dialog box, select **Number**.

3. From the **Category** list, select **Currency**. From the **Symbol** list, select a currency format to apply to the y-axis labels.
4. Click **OK**.

See Also

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Specify a Logarithmic Scale \(Report Builder and SSRS\)](#)

[Specify an Axis Interval \(Report Builder and SSRS\)](#)

Position Labels in a Chart (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

Because each chart type in a Reporting Services paginated report has a different shape, data point labels are placed in an optimal location so as not to interfere on the chart. The default position of the labels depends varies with the chart type:

- On stacked charts, labels can only be positioned inside the series.
- On funnel or pyramid charts, labels are placed on the outside in a column.
- On pie charts, labels are placed inside the individual slices on a pie chart.
- On bar charts, labels are placed outside of the bars that represent data points.
- On polar charts, labels are placed outside of the circular area that represents data points.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To change the position of point labels in a Pie chart

1. Create a pie chart.
2. On the design surface, right-click the chart and select **Show Data Labels**.
3. Open the Properties pane. On the **View** tab, click **Properties**.
4. On the design surface, click the chart. The properties for the chart are displayed in the Properties pane.
5. In the **General** section, expand the **CustomAttributes** node. A list of attributes for the pie chart is displayed.
6. Select a value for the **PieLabelStyle** property.

To change the position of point labels in a Funnel or Pyramid chart

1. Create a funnel or pyramid chart.
2. On the design surface, right-click the chart and select **Show Data Labels**.
3. Open the Properties pane. On the **View** tab, click **Properties**.
4. On the design surface, click the chart. The properties for the chart are displayed in the Properties pane.
5. In the **General** section, expand the **CustomAttributes** node. A list of attributes for the funnel chart is displayed.
6. For a funnel chart, select a value for the **FunnelLabelStyle** property. For a pyramid chart, select a value for the **PyramidLabelStyle** property.

NOTE

When this property is set to a value **OutsideInColumn**, the labels are drawn in a vertical column. There is no way to change the position of the column.

To change the position of point labels in a Bar chart

1. Create a bar chart.
2. On the design surface, right-click the chart and select **Show Data Labels**.
3. Open the Properties pane. On the **View** tab, click **Properties**
4. On the design surface, click the chart. The properties for the chart are displayed in the Properties pane.
5. In the **General** section, expand the **CustomAttributes** node. A list of attributes for the bar chart is displayed.
6. Select a value for the **BarLabelStyle** property.

When the bar label style is set to **Outside**, the labels will be placed outside of the bar, as long as it fits in the chart area. If the label cannot be placed outside of the bar but inside of the chart area, the label is placed inside the bar at the position closest to the end of the bar.

To change the position of point labels in an Area, Column, Line or Scatter chart

1. Create an Area, Column, Line or Scatter chart.
2. On the design surface, right-click the chart and select **Show Data Labels**.
3. Open the Properties pane. On the **View** tab, click **Properties**
4. On the design surface, click the series. The properties for the series are displayed in the Properties pane.
5. In the **Data** section, expand the **DataPoint** node, then expand the **Label** node.
6. Select a value for the **Position** property.

See Also

[Pie Charts \(Report Builder and SSRS\)](#)

[Bar Charts \(Report Builder and SSRS\)](#)

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Format Axis Labels as Dates or Currencies \(Report Builder and SSRS\)](#)

[Display Data Point Labels Outside a Pie Chart \(Report Builder and SSRS\)](#)

[Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)

Specify an Axis Interval (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Learn to change the number of labels and tick marks on the category (x) axis in a chart by setting the axis interval in a Reporting Services paginated report.

On the value axis (usually the y axis), axis intervals provide a consistent measure of the data points on the chart.

But on the category axis (usually the x axis), sometimes an automatic axis interval results in categories without axis labels. You can specify the number of intervals you want in the axis Interval property. Reporting Services calculates the number of intervals at run time, based on the data in the result set. For more information about how axis intervals are calculated, see [Formatting Axis Labels on a Chart](#).

To try setting the axis interval with sample data, see [Tutorial: Add a Column Chart to Your Report \(Report Builder\)](#).

NOTE

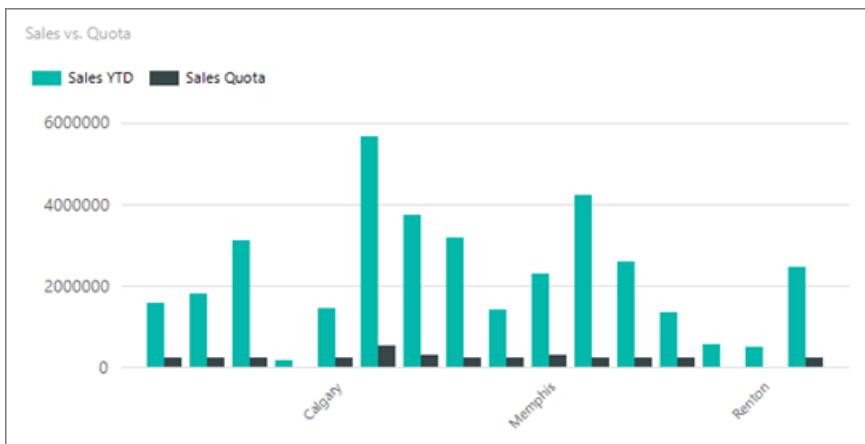
The category axis is usually the horizontal or x-axis. However, for bar charts, the category axis is the vertical or y-axis.

This topic doesn't apply to:

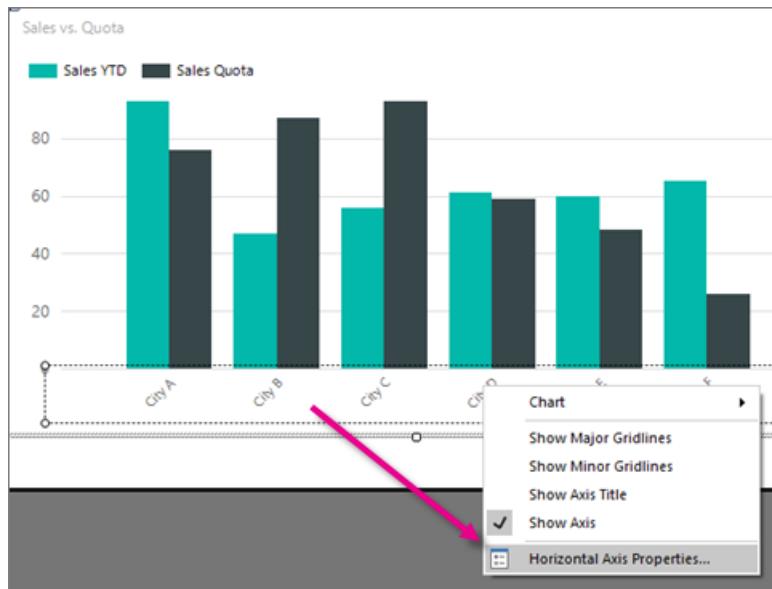
- Date or time values on the category axis. Be default, **DateTime** values appear as days. You can specify a different date or time interval, such as a month or time interval. For more information, see [Format Axis Labels as Dates or Currencies](#).
- Pie, doughnut, funnel or pyramid charts, which do not have axes.

To show all the category labels on the x-axis

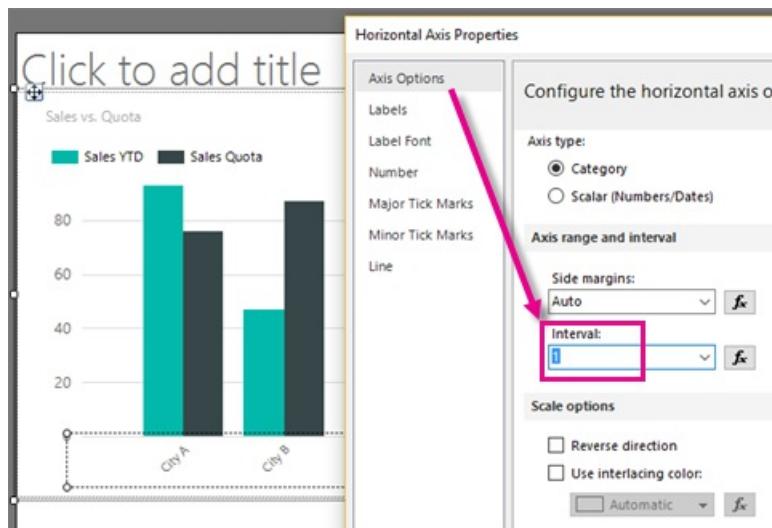
In this column chart, the horizontal label interval is set to Auto.



1. Right-click the category axis and click **Horizontal Axis Properties**.

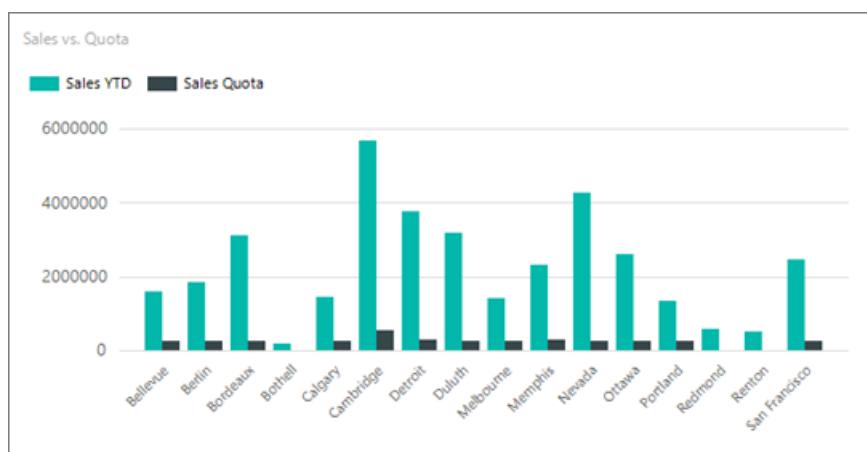


2. In the **Horizontal Axis Properties** dialog box > **Axis Options** tab, set **Interval** to **1** to show every category group label. To show every other category group label on the x-axis, type **2**.



3. Click **OK**.

Now the column chart displays all its horizontal axis labels.



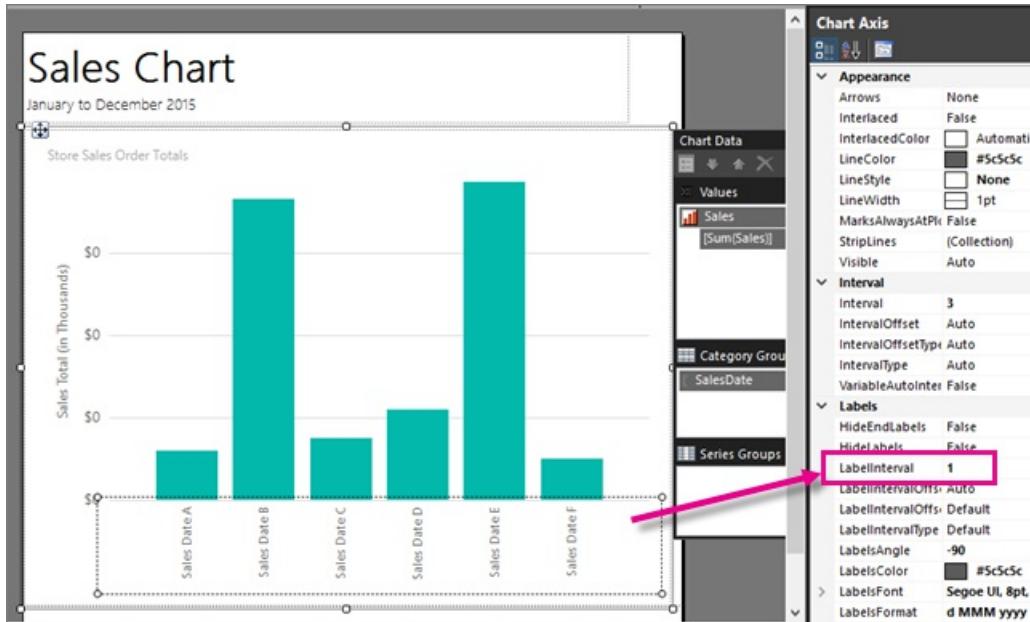
NOTE

When you set an axis interval, all automatic labeling is disabled. If you specify a value for the axis interval, you may see unpredictable labeling behavior, depending on how many categories are on the category axis.

Change the label interval in Properties pane

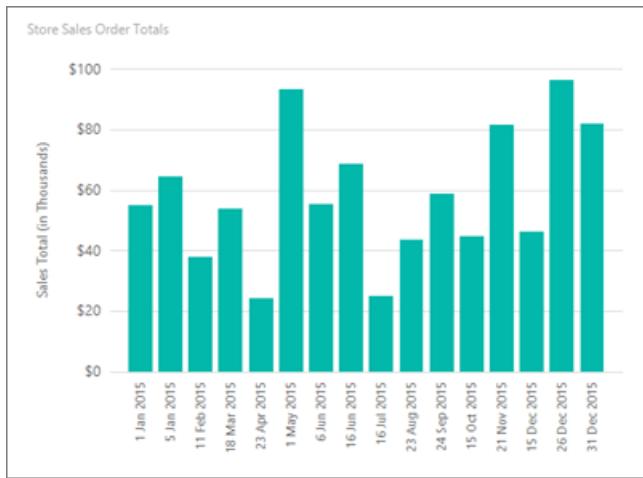
You can also set the label interval in the Properties pane.

1. In report design view, click the chart, then select the horizontal axis labels.
2. In the Properties pane, set LabelInterval to 1.



The chart looks the same in design view.

3. Click **Run** to preview the report.



Now the chart displays all its labels.

To enable a variable interval calculation on an axis

By default, Reporting Services sets the axis interval to Auto. This procedure explains how to set it back to the default.

1. Right-click the chart axis that you want to change, and then click **Axis Properties**.
2. In the **Horizontal Axis Properties** dialog box > **Axis Options** tab, set **Interval** to **Auto**. The chart will display the optimal number of category labels that can fit along the axis.
3. Click **OK**.

See Also

- [Formatting a Chart \(Report Builder and SSRS\)](#)
- [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)
- [Sort Data in a Data Region \(Report Builder and SSRS\)](#)
- [Axis Properties Dialog Box, Axis Options \(Report Builder and SSRS\)](#)
- [Specify a Logarithmic Scale \(Report Builder and SSRS\)](#)
- [Plot Data on a Secondary Axis \(Report Builder and SSRS\)](#)

Add or Remove Margins from a Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

For Column and Scatter chart types in Reporting Services paginated reports, the chart automatically adds side margins on the ends of the x-axis. In Bar chart types, the chart automatically adds side margins on the ends of the y-axis. In all other chart types, the chart does not add side margins. You cannot change the size of the margin.

This topic does not apply to pie, doughnut, funnel, or pyramid chart types.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To enable or disable side margins

1. Right-click the axis and select **Axis Properties**. The **Vertical** or **Horizontal Axis Properties** dialog box appears.
2. On the **Axis Options** page, set the **Side margins** property:
 - **Auto** The chart will determine whether to add a side margin based on the chart type.
 - **Disabled** Bar, column, and scatter charts will have no side margins.
3. Click **OK**.

See Also

- [Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)
[Axis Properties Dialog Box, Axis Options \(Report Builder and SSRS\)](#)
[Specify an Axis Interval \(Report Builder and SSRS\)](#)
[Format Axis Labels as Dates or Currencies \(Report Builder and SSRS\)](#)
[Charts \(Report Builder and SSRS\)](#)

Specify a Logarithmic Scale (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

If you have data that is logarithmically proportional, you may want to consider using a logarithmic scale on a chart in a Reporting Services paginated report. This helps improve the appearance of the chart by making your data more manageable. Most logarithmic scales use a base of 10.

This feature is only available on the value axis. The value axis is usually the vertical, or y-axis. On bar charts, however, it is the horizontal, or x-axis.

If your axis is logarithmic, all other properties relating to the axis will be scaled logarithmically. For example, if you specify a base-10 logarithmic scale on your axis, setting an axis interval of 2 will generate intervals in magnitudes of 10 to the power of 2, or 100. This means your axis values will read 1, 100, 10000, instead of the default result of 1, 10, 100, 1000, 10000.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To specify a logarithmic scale

1. Right-click the y-axis of your chart, and click **VerticalAxis Properties**. The **VerticalAxis Properties** dialog box appears.
2. In **Axis Options**, select **Use logarithmic scale**.
3. In the **Logbase** text box, type a positive value for the logarithmic base. If no value is specified, the logarithmic base defaults to 10.

See Also

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

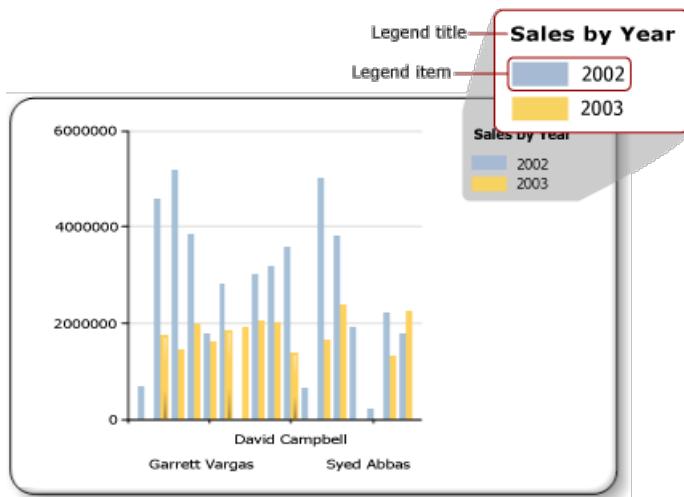
[Format Axis Labels as Dates or Currencies \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

Chart Legend - Formatting (Report Builder)

3/24/2017 • 5 min to read • [Edit Online](#)

The chart legend contains descriptions for each category in a chart. A legend always contains one or more legend items, where each legend item consists of a colored box that represents the series, and a text string that describes the series, as indicated in the following illustration.



A legend item is connected to an individual series on the chart, except for Shape charts, where the legend is connected to individual data points. The chart automatically adds items into the legend based on the series that are generated from your data.

You can format a legend by using the **Legend Properties** dialog box or by using the Properties pane. Right-click the legend and click **Legend Properties** to change values for the legend text, background color, borders, and 3D effects. To change values for the legend title, select the legend, right-click the legend title, and click **Legend Title Properties**.

You cannot add images, extra columns or other supplementary items to the legend.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Ordering Legend Items in the Legend

Series are ordered in the legend according to the order that they are processed by the Reporting Services processing engine. You can change the order by changing the order of fields in the data fields drop-zone. If you are using series grouping, the series data is not known until processing, so that there is no way for you to re-order these items. The changes can be seen in Preview. For more information about series grouping, see [Charts \(Report Builder and SSRS\)](#).

You can hide any series from being shown in the legend. If you are using series grouping, all series related to the data field will be hidden. For more information, see [Hide Legend Items on the Chart \(Report Builder and SSRS\)](#).

Changing the Text or Color of a Legend Item in the Legend

When a field is placed in the data field drop-zone of a chart, a legend item is automatically generated that contains

the name of this field. By default, the text of each legend item is taken from the name of the data field. Every legend item is connected to an individual series on the chart with the exception of Shape charts, where the legend is connected to individual data points instead of individual series. When a category group is defined on a Shape chart, the text of each legend item is taken from the string representation of the category group. You can specify custom label text for pie, doughnut and funnel charts to show information other than the category group label that relates to each individual data point in the legend. To do this, select the legend and specify legend text in either the **Series Properties** dialog box or the **LegendText** property in the Properties pane. For more information, see [Change the Text of a Legend Item \(Report Builder and SSRS\)](#).

You can also specify chart-specific, case-sensitive keywords for commonly referenced attributes in the **LegendText** property or in the **Series Properties** dialog box. The chart control replaces these keywords at run time with their data representation. This approach is useful on Shape charts because you are able to show information relating to specific data points. For more information, see [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#).

The colored box displayed for each legend item is inherited from the fill color of its corresponding series. If you want to change the color displayed in a legend item, you will have to change the color of the corresponding series. For more information, see [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#).

Removing Extra Legend Items from the Legend

The legend is always connected to a series. If a legend item appears in the legend but the corresponding series is not displayed in the chart, the most likely cause is that the series does not contain any values. You must remove this series in order to remove the legend item from the legend. To remove a series from your chart, you can right-click the specified series and select the Delete Series option.

Repositioning the Legend

The chart area is the rectangular area that encompasses the axis labels and the plotting area. You can drag the legend to one of twelve different positions when the legend is placed outside the chart area. By default, the legend is shown outside the chart area. You can also set the position in the **Legend Properties** dialog box.

You cannot drag the legend inside or outside the chart area. If you want to position the legend inside the chart area, on the **Legend Properties** dialog box, under **Docking**, select **Default** from the drop-down list and clear the **Show legend outside chart area** option. By placing the legend inside the chart area, you can maximize space for data points on the chart. However, depending on the dataset, it can cause the legend to overlap some data points on the chart area, making the chart more difficult to read.

Displaying Legend Items Horizontally

By default, the legend is formatted as a list of one or more rows containing one legend item each. The legend area expands to accommodate the number of legend items. If the legend cannot expand, an ellipsis (...) is displayed. Depending on the specified legend style, the legend can expand vertically or horizontally. You can change the layout style on the **Legend Properties** dialog box or change the allocated space to display all the legend items.

To display the legend horizontally, dock the legend to the top or bottom of the chart. This causes the legend to expand horizontally. You can also set the Layout property to **Row** or **Wide Table**. Set the MaxAutoSize property in the Properties pane to control the vertical space allocated to the legend when it is docked at the top or bottom of the chart area.

Formatting the Legend Text

You can change the font, size, style, and color of legend text on the **Font** page of the **Legend Properties** dialog box.

By default, the legend text is not optimized to fit the legend area. To cause the legend text to automatically fit the allocated space, set the AutoFitTextDisabled property to **False** and set a minimum font size for the MinFontSize property to the lowest font size that you think will be presentable and still allow for legend optimization.

See Also

[Legend Properties Dialog Box, General \(Report Builder and SSRS\)](#)

[Change the Text of a Legend Item \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Hide Legend Items on the Chart \(Report Builder and SSRS\)](#)

[Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)](#)

Chart Legend - Hide Items (Report Builder)

3/24/2017 • 1 min to read • [Edit Online](#)

By default, any series added to a non-Shape chart in a Reporting Services paginated report will be added as an item in the legend. For pie, doughnut, funnel, and pyramid charts, any series added to the chart will add individual data points in the legend.

You can hide any item on the legend. When you hide a legend item, it will still appear in the chart. This is useful in situations where you do not want to show more information for a series that is added to the chart. For example, if you have added a calculated series like a moving average to the chart, you may want to hide this entry in the legend so that more information is only shown for the original series.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To hide an item from display in the legend

1. Right-click on the series you want to hide and select **Series Properties**.
2. Click **Legend**. Select the **Do not show this series in a legend** option.

NOTE

You cannot hide a series for one group and not for the others. If you have added a field to the **Series Groups** area, all series belonging to this group will be hidden.

See Also

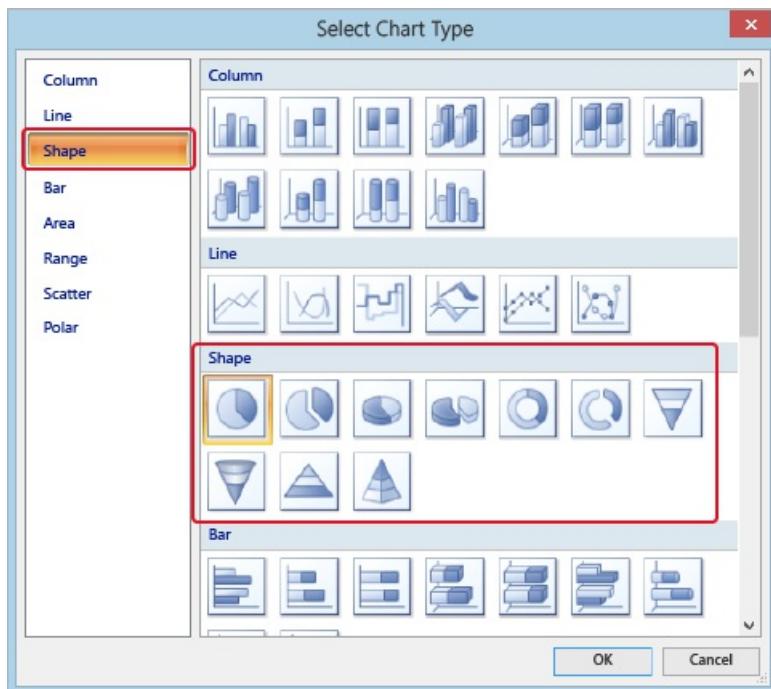
- [Formatting the Legend on a Chart \(Report Builder and SSRS\)](#)
- [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)
- [Change the Text of a Legend Item \(Report Builder and SSRS\)](#)
- [Add a Moving Average to a Chart \(Report Builder and SSRS\)](#)
- [Legend Properties Dialog Box, General \(Report Builder and SSRS\)](#)

Chart Legend - Change Item Text (Report Builder)

3/24/2017 • 2 min to read • [Edit Online](#)

When a field is placed in the Values area of the chart, a legend item is automatically generated that contains the name of this field. Every legend item is connected to an individual series on the chart, with the exception of shape charts, where the legend is connected to individual data points instead of individual series.

On shape charts, you can change the text of a legend item to show more information about the individual data points. For example, if you want to show the values of the data points as percentages in the legend, you can use a keyword such as **#PERCENT**. You can append .NET Framework format codes in conjunction with keywords to apply numeric and date formats. For more information about keywords, see [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#).



On non-shape charts, you can change the text of a legend item. For example, if your series name is "Series1", you may want to change the text to something more descriptive like "Sales for 2008".

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To modify the text that appears in the legend on a Shape chart

1. Right-click on a series, or right-click on a field in the **Values** area, and select **Series Properties**.
2. Click **Legend** and in the **Custom legend text** box, type a keyword.

The following table provides examples of chart-specific keywords to use for the **Custom Legend Text** property. For more information about keywords, see [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#).

KEYWORD	DESCRIPTION	EXAMPLE OF WHAT APPEARS AS TEXT IN THE LEGEND
#PERCENT{P1}	Displays the percentage of the total value to one decimal place.	85.0%
#VALY	Displays the actual numeric value of the data field.	17000
#VALY{C2}	Displays the actual numeric value of the data field as a currency to two decimal places.	\$17000.00
#AXISLABEL (#PERCENT{P0})	Displays the text representation of the category field, followed by the percentage that each category displays on the chart.	Michael Blythe (85%)

NOTE

The #AXISLABEL keyword can only be evaluated at run time when there is not a field specified in the **Series Groups** area.

To modify the text that appears in the legend on a non-Shape chart

1. Right-click on a series, or right-click on a field in the **Values** area, and select **Series Properties**.
2. Click **Legend** and in the **Custom legend text** box, type a legend label. The series is updated with your text.

See Also

[Formatting the Legend on a Chart \(Report Builder and SSRS\)](#)

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

[Hide Legend Items on the Chart \(Report Builder and SSRS\)](#)

Formatting Data Points on a Chart (Report Builder and SSRS)

3/24/2017 • 6 min to read • [Edit Online](#)

In a Reporting Services paginated report, a data point is the smallest individual entity on the chart. On non-Shape charts, data points are represented depending on their chart type. For example, a Line series consists of one or more connected data points. On Shape charts, data points are represented by individual slices or segments that add up to the whole chart. For example, on a pie chart, each piece is a data point. For more information, see [Chart Types \(Report Builder and SSRS\)](#).

One or more data points form a series. By default, all formatting options are applied to all data points in the series. If you want to specify properties for individual data points, you can specify a field or expression on the series that formats individual data point at run time based on the dataset.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Adding ToolTips and Drillthrough Actions to Data Points

You can add ToolTips to each data point by setting the value of the **ToolTip** property on the series. By displaying ToolTips, you can give your users the ability to see any information related to the data point, such as the group name, the value of the data point and the percentage of the data point relative to the series total. For more information, see [Show ToolTips on a Series \(Report Builder and SSRS\)](#).

You can also specify a drillthrough action for data points on the series to display another report or a URL. You can pass parameters to show information relating to the data point that has been clicked. For more information, see [Add a Drillthrough Action on a Report \(Report Builder and SSRS\)](#).

Highlighting Individual Data Points in a Series

On any non-Shape chart, you can highlight individual data points by specifying an expression for the **Color** property. For example, to highlight the highest data point value in a series that is named `MyField` with a different color than the other data points, the expression would be similar to the following:

```
=IIf(Fields!MyField.Value >= Max(Fields!MyField.Value, "MyDataSet"), "Red", "Green")
```

In this example, the highest value for `MyField` will have the color Red and all other data points will have the color Green. When you specify a color for the series using the **Fill** property on the series, the chart overrides the colors that are specified in the palette. For more information, see [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#).

Positioning Data Point Labels on a Chart

For all chart types, you can show data point labels when you right-click the chart and select **Show Data Labels**. The position of the data point labels is specified depending on the chart type:

- On a bar chart, you can reposition the data point label using the **BarLabelStyle** custom attribute. There are four possible positions: Outside, Left, Center and Right. When the bar label style is set to Outside, the labels will be positioned outside the bar, as long as it fits in the chart area. If the label cannot be positioned outside the bar and inside the chart area, the label is positioned inside the bar.
- On a pie chart, you can reposition the data point label using the **PieLabelStyle** custom attribute. There are many considerations when positioning data point labels around a pie chart, including the size of the pie chart, the available space between the pie chart and its corresponding legend, and the size of the labels. For more information, see [Display Data Point Labels Outside a Pie Chart \(Report Builder and SSRS\)](#).
- On a pyramid or funnel chart, you can reposition the data point labels using the **FunnelLabelStyle** and **PyramidLabelStyle** custom attributes. You can set these attributes in the **Properties** pane when you have selected a pyramid or funnel chart type.
- On stacked charts, data point labels are always positioned inside the series and the **Position** property on the series label is ignored.
- On all other chart types, you can reposition the data point label using the **Position** property on the series label. By default, the chart automatically calculates the position for data point labels to avoid label collisions. When you set a value for **Position**, all data point labels will be positioned the same way, which may cause the labels to overlap. Consider using this approach only when you have fewer data points.

For more information, see [Position Labels in a Chart \(Report Builder and SSRS\)](#).

Adding Keywords for Data Point Labels, ToolTips, and Legend Text

You can use case-sensitive, chart-specific keywords to represent an item that exists in the chart. These keywords are only applicable to ToolTips, custom legend text, and data point label properties. In many cases, a chart keyword has an equivalent simple expression, but the keyword is faster and easier to type. The following is a list of chart keywords.

CHART KEYWORD	DESCRIPTION	APPLICABLE TO CHART TYPE	EXAMPLE OF AN EQUIVALENT SIMPLE EXPRESSION
#VALY	Y value of the data point.	All	=Fields!MyDataField.Value
#VALY2	Y value #2 of data point.	Range, Bubble	None
#VALY3	Y value #3 of data point.	Stock, Candlestick	None
#VALY4	Y value #4 of data point.	Stock, Candlestick	None
#SERIESNAME	Series name.	All	None
#LABEL	Data point label.	All	None
#AXISLABEL	Axis data point label.	Shape	=Fields!MyDataField.Value
#INDEX	Data point index.	All	None
#PERCENT	Percentage of the data point Y value.	All	=FormatPercent(Fields!MyDataField.Value/Sum(Fields!MyDataSet"),2)
#TOTAL	Total of all Y values in the series.	All	=Sum(Fields!MyDataField.Value)
#LEGENDTEXT	The text that corresponds to the text of the legend item.	All	None
#AVG	Average of all Y values in the series.	All	=Avg(Fields!MyDataField.Value)
#MIN	Minimum of all Y values in the series.	All	=Min(Fields!MyDataField.Value)
#MAX	Maximum of all Y values in the series.	All	=Max(Fields!MyDataField.Value)
#FIRST	First of all Y values in the series.	All	=First(Fields!MyDataField.Value)

To format the keyword, enclose a .NET Framework format string in parentheses. For example, to specify the value of the data point in a ToolTip as a number with two decimal places, include the format string "N2" in braces, such as "#VALY(N2)" for the **ToolTip** property on the series. For more information about .NET Framework format strings, see [Formatting Types](#) on MSDN. For more information about formatting numbers in Reporting Services, see [Formatting Numbers and Dates \(Report Builder and SSRS\)](#).

For more information about adding keywords to a chart, see [Show ToolTips on a Series \(Report Builder and SSRS\)](#), [Change the Text of a Legend Item \(Report Builder and SSRS\)](#).

Increasing Readability in a Chart with Multiple Data Points

If you have multiple series on your chart, it may reduce the readability of the chart data points. When adding multiple series to the chart, consider using a technique that distinguishes how to read and understand each series in the chart effectively. For more information, see [Multiple Series on a Chart \(Report Builder and SSRS\)](#).

For simplicity, when you are using a Shape chart, consider adding only one data field and one category field. For more information, see [Shape Charts \(Report Builder and SSRS\)](#). If your chart needs more than one data field and category field, consider changing the chart type. You can right-click the series and select **Change Chart Type**.

Inserting Data Point Markers

A data point marker is a visual indicator used to draw attention to each data point in a series. On a scatter chart, the marker is used to determine the shape and size of the individual data points. The size of the marker is specified based on the chart type. You can change the size, color, or style of the marker. Markers are not available for range and shape chart types, or any stacked subtypes.

In This Section

[Show ToolTips on a Series \(Report Builder and SSRS\)](#)

[Display Data Point Labels Outside a Pie Chart \(Report Builder and SSRS\)](#)

[Display Percentage Values on a Pie Chart \(Report Builder and SSRS\)](#)

See Also

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Format Axis Labels as Dates or Currencies \(Report Builder and SSRS\)](#)

[Tutorial: Add a Pie Chart to Your Report \(Report Builder\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

Show ToolTips on a Series (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

You can add a ToolTip to each data point on the series of a chart to display information related to the data point, such as the group name, the value of the data point, or the percentage of the data point relative to the series total. When users hover over the data point in a rendered paginated report, they'll see this information.

You cannot add a ToolTip to a calculated series.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To specify a ToolTip on each data point

1. Right-click on the series or right-click on the field in the **Values** area, and click **Series Properties**.
2. Click **Series Data** and, for the **ToolTip** property, type in a string or expression. You can use any chart-specific keyword to represent another element on the chart. For more information, see [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#).

See Also

[Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)

[Change the Text of a Legend Item \(Report Builder and SSRS\)](#)

[Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#)

[Add a Drillthrough Action on a Report \(Report Builder and SSRS\)](#)

Display Data Point Labels Outside a Pie Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In Reporting Services, pie chart labeling is optimized to display labels on only several slices of data. Labels may overlap if the pie chart contains too many slices. One solution is to display the labels outside the pie chart, which may create more room for longer data labels. If you find that your labels still overlap, you can create more space for them by enabling 3D. This reduces the diameter of the pie chart, creating more space around the chart.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To display data point labels inside a pie chart

1. Add a pie chart to your report. For more information, see [Add a Chart to a Report \(Report Builder and SSRS\)](#).
2. On the design surface, right-click on the chart and select **Show Data Labels**.

To display data point labels outside a pie chart

1. Create a pie chart and display the data labels.
2. Open the Properties pane.
3. On the design surface, click on the pie itself to display the **Category** properties in the Properties pane.
4. Expand the **CustomAttributes** node. A list of attributes for the pie chart is displayed.
5. Set the **PieLabelStyle** property to **Outside**.
6. Set the **PieLineColor** property to **Black**. The PieLineColor property defines callout lines for each data point label.

To prevent overlapping labels displayed outside a pie chart

1. Create a pie chart with external labels.
2. On the design surface, right-click outside the pie chart but inside the chart borders and select **Chart Area Properties**. The **Chart AreaProperties** dialog box appears.
3. On the **3D Options** tab, select **Enable 3D**.
4. If you want the chart to have more room for labels but still appear two-dimensional, set the **Rotation** and **Inclination** properties to **0**.

See Also

[Pie Charts \(Report Builder and SSRS\)](#)

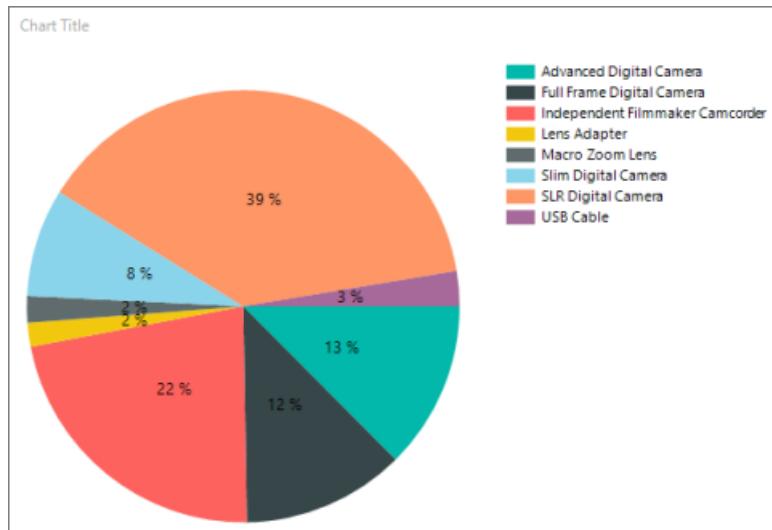
[Collect Small Slices on a Pie Chart \(Report Builder and SSRS\)](#)

[Display Percentage Values on a Pie Chart \(Report Builder and SSRS\)](#)

Display Percentage Values on a Pie Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In Reporting Services paginated reports, by default the legend shows categories. You may also want percentages in the legend or the pie slices themselves.



The [Tutorial: Add a Pie Chart to Your Report \(Report Builder\)](#) walks you through adding percentages to pie slices, if you'd like to try this with sample data first.

To display percentage values as labels on a pie chart

1. Add a pie chart to your report. For more information, see [Add a Chart to a Report \(Report Builder and SSRS\)](#).
2. On the design surface, right-click on the pie and select **Show Data Labels**. The data labels should appear within each slice on the pie chart.
3. On the design surface, right-click on the labels and select **Series Label Properties**. The **Series Label Properties** dialog box appears.
4. Type **#PERCENT** for the **Label data** option.
5. (Optional) To specify how many decimal places the label shows, type "#PERCENT{Pn}" where *n* is the number of decimal places to display. For example, to display no decimal places, type "#PERCENT{P0}".

To display percentage values in the legend of a pie chart

1. On the design surface, right-click on the pie chart and select **Series Properties**. The **Series Properties** dialog box displays.
2. In **Legend**, type **#PERCENT** for the **Custom legend text** property.

See Also

- [Tutorial: Add a Pie Chart to Your Report \(Report Builder\)](#)
- [Pie Charts \(Report Builder and SSRS\)](#)

- [Formatting the Legend on a Chart \(Report Builder and SSRS\)](#)
- [Display Data Point Labels Outside a Pie Chart \(Report Builder and SSRS\)](#)

Chart Effects - 3D, Bevel, and Other (Report Builder)

3/24/2017 • 3 min to read • [Edit Online](#)

Three-dimensional (3D) effects can be used to provide depth and add visual impact to charts in your Reporting Services paginated reports. For example, if you want to emphasize a particular slice of an exploded pie chart, you can rotate and change the perspective of the chart so that people notice that slice first. When 3D effects are applied to your chart, all gradient colors and hatching styles are disabled.

Three-dimensional effects can be applied to individual charts, and it is possible to display both two-dimensional and three-dimensional charts on the same report.

For all chart types, you can add three-dimensional effects to a chart area in the **Chart Area Properties** dialog box by selecting **Enable 3D**. For more information, see [Add 3D Effects to a Chart \(Report Builder and SSRS\)](#).

Another way to add visual impact to charts is by adding bevel, emboss and texture styles in bar, column, pie and doughnut charts. For more information, see [Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Coordinate-Based Three-Dimensional Charts

When working with coordinate-based chart types (Column, Bar, Area, Point, Line and Range), three-dimensional effects display the chart with a third axis, known as the "z-axis". The introduction of this third axis allows you to apply a variety of visual enhancements to your chart.

Changing the White Space in a 3D Chart

When you display a chart area in three-dimensional mode, each series is shown in a separate row along the z-axis of the chart. To change the amount of space between each series, modify the chart's point gap depth by changing the **Point Gap Depth** property in the 3D Effects dialog box.

Changing the Projection of a 3D Chart

There are two types of 3D projections: oblique and perspective. An oblique projection to the chart adds a depth dimension to a two-dimensional chart. The z-axis is drawn at equal angles from the horizontal and vertical axes, which remain perpendicular to each other just as in a two-dimensional chart.

Perspective projection transforms the chart by estimating a view plane and re-drawing the chart as if it were being viewed from that point. The **Rotation** value shifts the view vertically from "ground level" at 0 to overhead at 90. The **Inclination** value shifts the viewing angle to the left or right. A value of 0 is equivalent to a two-dimensional view of the chart. The **Perspective** value defines the percentage of distortion that will be used when displaying the projection. This type of projection maintains the proportions of your chart, but the chart's appearance becomes distorted, so it is most effective to use a lower degree of perspective.

NOTE

The oblique and perspective projections are separate types of projections, so they cannot be used together on the same chart.

Clustering Data

In 2D charts, multiple series of data appear side-by-side. Clustering shows individual series in separate rows on a 3D chart. For example, if you have a chart that contains three series of data points, clustering will display each of the three series on a separate row along the z-axis. By default, all chart types shown in 3D are clustered.

Clustering can be disabled for bar and column charts. When clustering is disabled, multiple bar and column series are displayed side-by-side in one row.

Shape-Based Three-Dimensional Charts

Shape-based chart types (Pie, Doughnut, Funnel, Pyramid) have fewer three-dimensional effects available. When working with shape-based chart types, you can change the rotation and inclination values only.

Rotations

Charts can be rotated horizontally and vertically from -90 to 90 degrees. A positive horizontal angle will rotate the chart counter-clockwise around the x-axis, while a positive vertical angle will rotate the chart clockwise around the y-axis.

Highlighting 3D Effects

You can add highlighting styles to a 3D chart via the **Shading** property, which appears under Area3DStyle in the Properties pane when the chart area is selected. A simple lighting style applies the same hue to the chart area elements. A realistic style changes the hues of the chart area elements depending on a specified lighting angle.

See Also

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Add 3D Effects to a Chart \(Report Builder and SSRS\)](#)

Chart Effects - Add 3D Effects (Report Builder)

3/24/2017 • 2 min to read • [Edit Online](#)

Three-dimensional (3D) effects can be used to provide depth and add visual impact to charts in your Reporting Services paginated reports. For example, if you want to emphasize a particular slice of an exploded pie chart, you can rotate and change the perspective of the chart so that people notice that slice first. When 3D effects are applied to your chart, all gradient colors and hatching styles are disabled.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To apply 3D effects to a Range, Area, Line, Scatter or Polar chart

1. Right-click anywhere inside the chart area and select **3D Effects**. The **Chart Area Properties** dialog box appears.
2. In **3D Options**, select the **Enable 3D** option.
3. (Optional) In **3D Options**, you can set a variety of properties relating to 3D angles and scene options. For more information about these properties, see [3D, Bevel, and Other Effects in a Chart \(Report Builder and SSRS\)](#).
4. Click **OK**.

To apply 3D effects to a Funnel chart

1. Right-click anywhere inside the chart area and select **3D Effects**. The **Chart Area Properties** dialog box appears.
2. In **3D Options**, select the **Enable 3D** option. Click **OK**.
3. (Optional) To customize the funnel chart visual appearance, you can go to the Properties pane and change properties specific to the funnel chart.
 - a. Open the Properties pane.
 - b. On the design surface, click anywhere on the funnel. The properties for the series of the funnel chart are displayed in the Properties pane.
 - c. In the **General** section, expand the **CustomAttributes** node.
 - d. For the **Funnel3DDrawingStyle** property, select whether the funnel will be shown with as circular or square.
 - e. For the **Funnel3DRotationAngle** property, set a value between -10 and 10. This will determine the degree of tilt that will be displayed on the 3D funnel chart.

To apply 3D effects to a Pie chart

1. Right-click anywhere inside the chart area and select 3D Effects. The **Chart Area Properties** dialog box appears.

2. In **3D Options**, select the **Enable 3D** option. Click **OK**.
3. (Optional) In **Rotation**, type an integer value that represents the horizontal rotation of the pie chart.
4. (Optional) In **Inclination**, type an integer value that represents the vertical tilt rotation of the pie chart.
5. Click **OK**.

To apply 3D effects to a Bar or Column chart

1. Right-click anywhere inside the chart area and select **3D Effects**. The **Chart Area Properties** dialog box appears.
2. Select the **Enable 3D** option. Click **OK**.
3. (Optional) Select the **Enable series clustering** option. If your chart contains multiple bar or column chart series, this option will display the series as clustered. By default, multiple bar or column series are shown side-by-side.
4. Click **OK**.
5. (Optional) To add cylinder effects to a bar or column chart, follow these steps:
 - a. Open the Properties pane.
 - b. On the design surface, click any of the bars in the series. The properties for the series are displayed in the Properties pane.
 - c. In the **General** section, expand the **CustomAttributes** node.
 - d. For the **DrawingStyle** property, specify the **Cylinder** value.

See Also

[3D, Bevel, and Other Effects in a Chart \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

Chart Effects - Add Bevel, Emboss, or Texture (Report Builder)

3/24/2017 • 1 min to read • [Edit Online](#)

When using certain chart types, you can specify a drawing effect to increase the visual impact of your chart. These drawing effects are only applied to the series of your chart. They have no effect on any other chart element.

When you are using any variant of a pie or doughnut chart, you can specify a soft edge or concave drawing style, similar to bevel or emboss effects that can be applied to an image.

When you are using any variant of a bar or column chart, you can apply texture styles, such as cylinder, wedge, and light-to-dark, to the individual bars or columns.

In addition to these drawing styles, you can add borders and shadows to many chart elements to give the illusion of depth. For more information on other ways to format the chart, see [Formatting a Chart \(Report Builder and SSRS\)](#).

NOTE

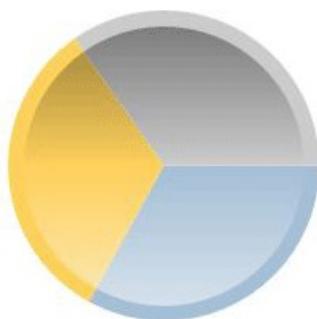
You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add bevel or emboss styles to a pie or doughnut chart

1. On the **View** tab, select **Properties** to open the Properties pane.
2. Select the pie or doughnut chart that you want to enhance. Select a data field in the chart, not the entire chart.
3. In the Properties pane, expand the **CustomAttributes** node.
4. For **PieDrawingStyle**, select a style from the drop-down list.

NOTE

You can't have 3D and bevel or emboss styles on the same chart. If you have enabled 3D for the chart, you will not see the **PieDrawingStyle** property.



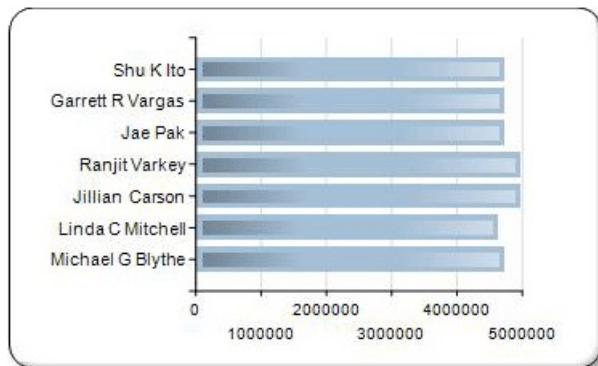
To add texture styles to a bar or column chart

1. Select the bar or column chart that you want to enhance. Select a data field in the chart, not the entire chart.

2. Open the Properties pane.
3. Expand the **CustomAttributes** node.
4. For DrawingStyle, select a style from the drop-down list.

NOTE

You can't have 3D and bevel or emboss styles on the same chart. If you have enabled 3D for the chart, you will not see the PieDrawingStyle property.



See Also

- [Bar Charts \(Report Builder and SSRS\)](#)
[Column Charts \(Report Builder and SSRS\)](#)
[Pie Charts \(Report Builder and SSRS\)](#)
[Formatting a Chart \(Report Builder and SSRS\)](#)

Add a Border Frame to a Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

To give a chart more visual impact, consider using a border frame around the outside of the chart. You can select a border frame by using the **Chart Properties** dialog box or by using the Properties pane. The chart border frames cannot be applied to any other data region.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To apply a border to a chart

1. Right-click anywhere on the chart and select **Chart Properties**.

NOTE

If you do not see the **Chart Properties**, point to **Chart** on the shortcut menu and select **Chart Properties**.

2. Select **Border**, and click the type of border to apply to the chart.
3. (Optional) Select a value or type an expression that represents the style of the border.
4. (Optional) Specify the color of the line that will be drawn around the chart as the border.

NOTE

The **Line color** list contains common colors. If you want to select from a list of more colors, click **More colors** in the list or click the expression (**fx**) button next to the list to bring up the **Expression** editor.

5. (Optional) Specify the width of the border. Valid values are between 0.25pt and 20pt. Consider keeping the size of your border to between 1 and 3pt for the best visual effect.
6. (Optional) If your report contains a background color other than white, consider defining a page color that is the same color. The page color is the background color outside of the border line.
7. (Optional) If you choose a Frame type, specify a style and color for the frame. The **Frame fill color** list contains common colors.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Add Bevel, Emboss, and Texture Styles to a Chart \(Report Builder and SSRS\)](#)

Start Pie Chart Values at the Top of the Pie (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In pie charts in Reporting Services paginated reports, by default the first value in the dataset starts at 90 degrees from the top of the pie.

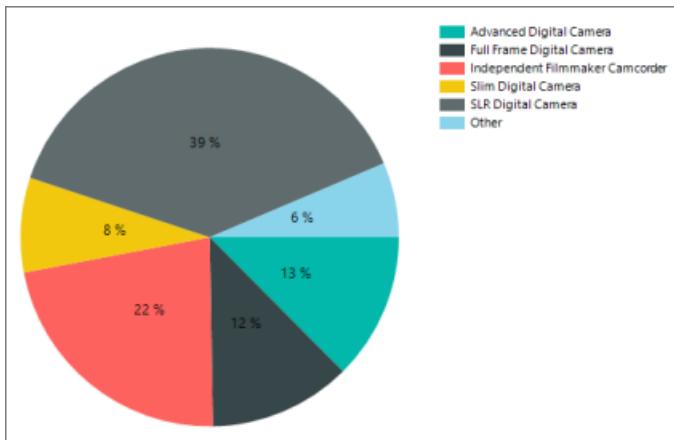


Chart values start at 90 degrees.

You may want the first value start at the top instead.

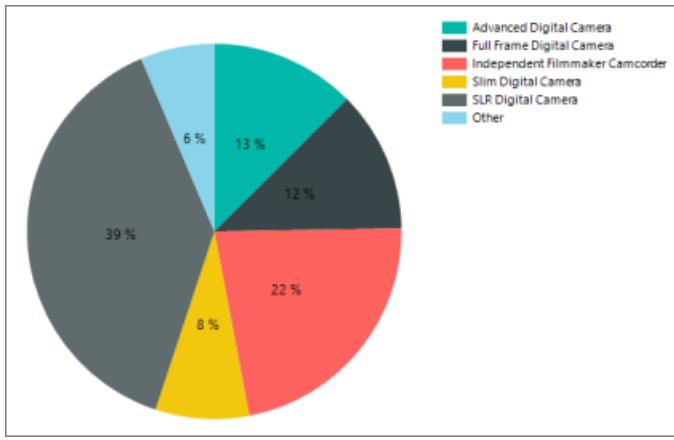


Chart values start at the top of the chart.

To Start the Pie Chart at the Top of the Pie

1. Click the pie itself.
2. If the **Properties** pane is not open, on the **View** tab, click **Properties**.
3. In the **Properties** pane, under **Custom Attributes**, change **PieStartAngle** from **0** to **270**.
4. Click **Run** to preview your report.

The first value now starts at the top of the pie chart.

See Also

[Formatting a Chart \(Report Builder and SSRS\)](#)

Pie Charts (Report Builder and SSRS)

Empty and Null Data Points in Charts (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

If you are displaying fields with empty or null values in a chart in your Reporting Services paginated report, the chart may not look as you expect. Charts process empty values differently depending on the specified chart type:

- If the chart type is a linear chart type (bar, column, scatter, line, area, range), empty values are displayed as empty spaces or "gaps" in the chart. If you want to indicate empty points, you must add empty point placeholders. For more information, see [Add Empty Points to a Chart \(Report Builder and SSRS\)](#).
- If the chart type is a contiguous, linear chart type (area, bar, column, line, scatter), empty data points are added to the chart to maintain continuity in the series.
- If the chart type is a nonlinear chart type (polar, pie, doughnut, funnel or pyramid), empty values are omitted from display on the chart.
- In shape chart types, null values are omitted.

An example of a chart with empty data points is available as a sample report. For more information about downloading this sample report and others, see [SQL Server 2016 Report Builder and Report Designer sample reports](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Removing Empty or Null Values

To avoid obscuring important data, consider removing empty values from your dataset. To filter nulls, you can use the NOT IS NULL clause in your query. Alternatively, you can add a filtering expression that specifies that you only want to display values not equal to zero. For more information, see [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#).

Fields with No Values in a Chart

If a field does not contain any values in the returned dataset, the chart displays an empty chart with no data points, but the series name (typically the field name) is added as a legend item.

This behavior differs from the case where there are zero rows of data in the returned dataset, which can occur when the report is parameterized and the selected value returns an empty result set. If your dataset query returns zero rows of data, a message is displayed at run time to indicate that no data can be shown. You can customize this message by modifying the NoDataMessage caption for the report in the **Properties** pane. For more information, see [Report Embedded Datasets and Shared Datasets \(Report Builder and SSRS\)](#).

See Also

[Charts \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Add a Chart to a Report \(Report Builder and SSRS\)](#)

[Troubleshoot Charts \(Report Builder and SSRS\)](#)

Add Empty Points to a Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Null values are shown on the chart as empty spaces or gaps between data points in a series. In Reporting Services paginated reports, empty points are data points that can be inserted in the empty space created by null values.

By default, empty points are calculated by taking the average of the previous and next data points that contain a value. You can change this so that all empty points are inserted at zero.

For more information, see [Empty and Null Data Points in Charts \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To specify empty points on a chart

1. Open the Properties pane.
2. On the design surface, right-click the series that contains null values. The properties for the series are displayed in the Properties pane.
3. Expand the **EmptyPoint** node.
4. Select a color value for the Color property.
5. In the **EmptyPoint** node, expand the Marker node.
6. Select a marker type for the MarkerType property.

NOTE

You must select a marker type to add empty points to a bar, column or scatter chart. However, for area, line and radar charts, selecting a marker type is optional because the chart fills in the empty space or gap without requiring a marker to be specified.

7. Set the value of the empty point.
 - a. In the Properties pane, expand the **CustomAttributes** node.
 - b. Set the EmptyPointValue property. To insert empty points at an average of the previous and next data points, select **Average**. To insert empty points at zero, select **Zero**.

See Also

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Chart Types \(Report Builder and SSRS\)](#)

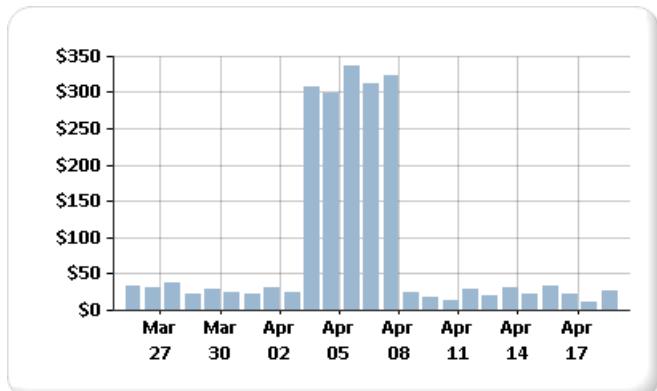
[Add Scale Breaks to a Chart \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

Displaying a Series with Multiple Data Ranges on a Chart

3/29/2017 • 3 min to read • [Edit Online](#)

Chart will use the minimum and maximum values of a series to calculate the axis scale. When a series on your chart contains more than one range of data, the data points can become obscured, and only a few data points to be seen easily on the chart. For example, suppose a report displays daily sales totals over a period of 30 days.



For most of the month, the sales are between 10 and 40. However, a one-week sales marketing campaign has caused a sudden sales increase at the beginning of April. This change in sales data produces an uneven distribution of data points that reduces the overall readability of the chart.

There are different ways to improve readability:

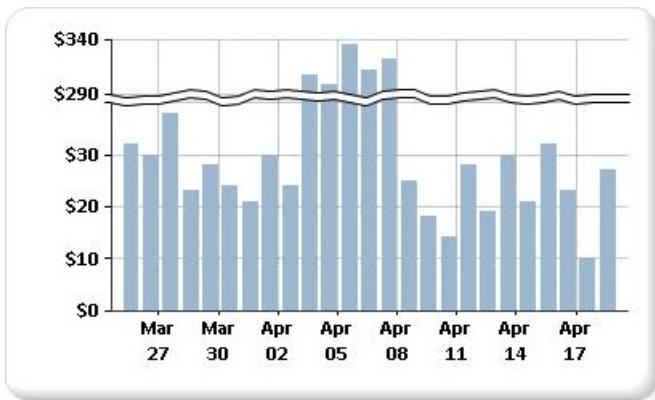
- **Enable scale breaks.** If your data forms two or more sets of data ranges, use a scale break to remove the gap between the ranges. A scale break is a stripe drawn across the plotting area to denote a break between the high and low values of a series.
- **Filter out unnecessary values.** If you have data points that are obscuring the important data range to be displayed on the chart, remove the unwanted points using a report filter. For information on how to add a filter to the chart in Reporting Services, see [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#).
- **Plot each data range as a separate series for multiple series comparison.** If you have more than two data ranges, consider separating the data ranges into separate series. For more information, see [Multiple Series on a Chart \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Displaying Multiple Data Ranges Using Scale Breaks

When you enable a scale break, the chart calculates where to draw a line across the chart. You must have sufficient separation between ranges to draw a scale break. By default, a scale break can be added only if there is a separation between the data ranges of at least 25% of the chart.



NOTE

You cannot specify where to place a scale break on a chart. You can, however, modify how the scale break is calculated, described later in this topic.

If you enable a scale break but it does not appear, even though there is sufficient distance between the data ranges, you can set the CollapsibleSpaceThreshold property to a value less than 25. The CollapsibleSpaceThreshold specifies the percent of collapsible space required between the data ranges. For more information, see [Add Scale Breaks to a Chart \(Report Builder and SSRS\)](#).

Charts support up to five scale breaks per chart; however, displaying more than one scale break can cause the chart to become unreadable. If you have more than two data ranges, consider using a different method for displaying this data. For more information, see [Multiple Series on a Chart \(Report Builder and SSRS\)](#).

Unsupported Scale Break Scenarios

Scale breaks are not supported in the following chart scenarios:

- The chart is 3-D enabled.
- A logarithmic value axis has been specified.
- The value axis minimum or maximum has been explicitly set.
- The chart type is polar, radar, pie, doughnut, funnel, pyramid, or any stacked chart.

An example of chart with scale breaks is available as a sample report. For more information about downloading this sample report and others, see SQL Server 2016[Report Builder and Report Designer sample reports](#).

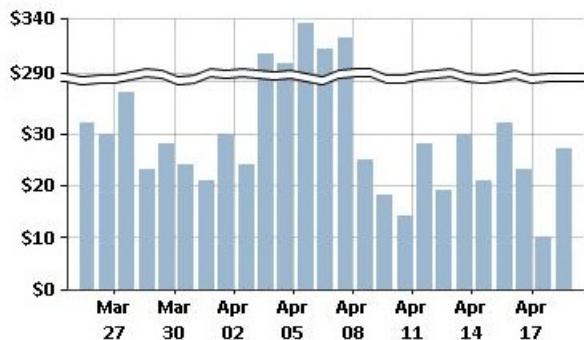
See Also

- [Multiple Series on a Chart \(Report Builder and SSRS\)](#)
- [Formatting a Chart \(Report Builder and SSRS\)](#)
- [3D, Bevel, and Other Effects in a Chart \(Report Builder and SSRS\)](#)
- [Charts \(Report Builder and SSRS\)](#)
- [Axis Properties Dialog Box, Axis Options \(Report Builder and SSRS\)](#)
- [Collect Small Slices on a Pie Chart \(Report Builder and SSRS\)](#)

Add Scale Breaks to a Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A scale break is a stripe drawn across the plotting area of a chart to denote a break in continuity between the high and low values on a value axis (usually the vertical, or y-axis). Use a scale break to display two distinct ranges in the same chart area.



NOTE

You cannot specify where to place a scale break on your chart. The chart uses its own calculations based on the values in your dataset to determine whether there is sufficient separation between data ranges to draw a scale break on the value axis (y-axis) at run time.

An example of a chart with scale breaks is available as a sample report. For more information about downloading this sample report and others, see SQL Server 2016[Report Builder and Report Designer sample reports](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To enable scale breaks on the chart

1. Right-click the vertical axis and then click **Axis Properties**. The **VerticalAxis Properties** dialog box opens.
2. Select the **Enable scale breaks** check box.

To change the style of the scale break

1. Open the Properties pane.
2. On the design surface, right-click on the y-axis of the chart. The properties for the y-axis object (named **Chart Axis** by default) are displayed in the Properties pane.
3. In the **Scale** section, expand the **ScaleBreakStyle** property.
4. Change the values for **ScaleBreakStyle** properties, such as **BreakLineType** and **Spacing**. For more information about scale break properties, see [Displaying a Series with Multiple Data Ranges on a Chart \(Report Builder and SSRS\)](#).

See Also

[Charts \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Axis Properties Dialog Box, Axis Options \(Report Builder and SSRS\)](#)

Multiple Series on a Chart (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

When multiple series are present on a chart, you must determine the best way to compare the series. You can use a stacked chart to show relative proportions of each series. If you are comparing only two series that share a common category (x) axis, use the secondary axis. This is useful when showing two related series of data, for example, price and volume, or income and tax. If the chart becomes unreadable, consider using multiple chart areas to create more visual separation between each series.

In addition to using chart features, it is important to decide which chart type should be used for your data. If the fields in your dataset are related, consider using a range chart.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Using Stacked and 100% Stacked Charts

Stacked charts are commonly used to display multiple series in one chart area. Consider using stacked charts when the data that you are trying to show is closely related. It is also a good practice to show four or less series on a stacked chart. If you want to compare the proportion that each series contributes to the whole, use a 100% stacked area, bar, or column chart. These charts calculate the relative percentage that each series contributes to the category. For more information, see [Area Charts \(Report Builder and SSRS\)](#), [Bar Charts \(Report Builder and SSRS\)](#) and [Column Charts \(Report Builder and SSRS\)](#).

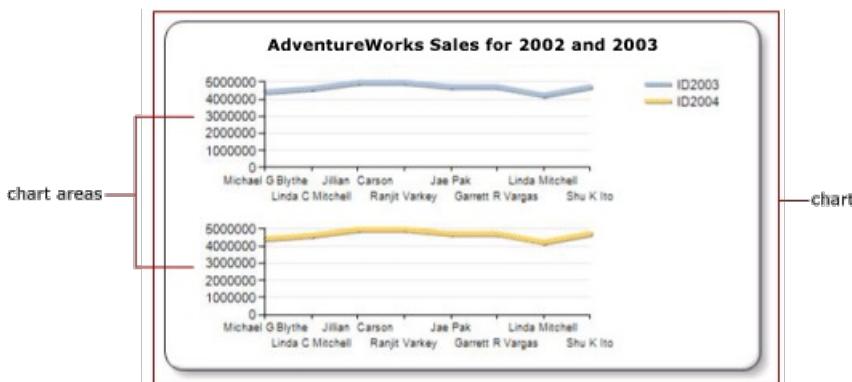
Using the Secondary Axis

When a new series is added to the chart, it is plotted using the primary x and y axes. When you want to compare values that are of a different unit of measure, consider using the *secondary axis* so that you can plot two series on separate axes. The secondary axis is useful when comparing values that are of a different unit of measure. The secondary axis is drawn on the opposite side of the primary axis. The chart only supports a primary and a secondary axis. The secondary axis has the same properties as the primary axis. For more information, see [Plot Data on a Secondary Axis \(Report Builder and SSRS\)](#).

If you want to display more than two series that have different ranges of data, consider putting the series in separate chart areas.

Using Chart Areas

The chart is the top-level container that includes the outer border, the chart title, and the legend. By default, the chart contains one default chart area. The chart area is not visible on the chart surface, but you can think of the chart area as a container that includes only the axis labels, the axis title, and the plotting area of one or more series. The following illustration shows the concept of chart areas within a single chart.



Using the **Chart Area Properties** dialog box, you can specify the 2D and 3D orientation of all series contained in the chart area, align multiple chart areas within the same chart, and format the colors of the plotting area. When a new chart area is defined on a chart that contains only one default chart area, the available space for a chart area is horizontally divided by two and the new chart area is positioned below the first chart area.

Each series can be connected to only one chart area. By default, all series are added to the default chart area. When using area, column, line, and scatter charts, any combination of these series can be displayed on the same chart area. For example, you can display a column series and a line series on the same chart area. The advantage of using the same chart area for multiple series is that end users can make comparisons easily.

Bar, radar, and shape charts cannot be combined with any other chart type in the same chart area. If you want to make comparisons with multiple series that are of type bar, radar, or shape, you will need to do one of the following:

- Change all series in the chart area to be of the same chart type.
- Create a new chart area and move one or more of the series from the default chart area into the newly created chart area.

The multiple chart area on a single chart feature is also useful if you are trying to compare data that has different scales of values. For example, if your first series contains data in the range of 10 to 20 and your second series contains data in the range of 400 to 800, the values in your first series may become obscured. Consider separating each series into a different chart area. For more information, see [Specify a Chart Area for a Series \(Report Builder and SSRS\)](#).

Using Range Charts

Range charts have two values per data point. If your chart contains two series that share the same category (x) axis, you can use a range chart to show the difference between the two series. Range charts are best suited for displaying high-low or top-bottom information. For example, if your first series contains the highest sale for each day in January, and your second series contains the lowest sale for each day in January, you can use a range chart to show the difference between the highest and lowest sale for each day. For more information, see [Range Charts \(Report Builder and SSRS\)](#).

See Also

[Charts \(Report Builder and SSRS\)](#)

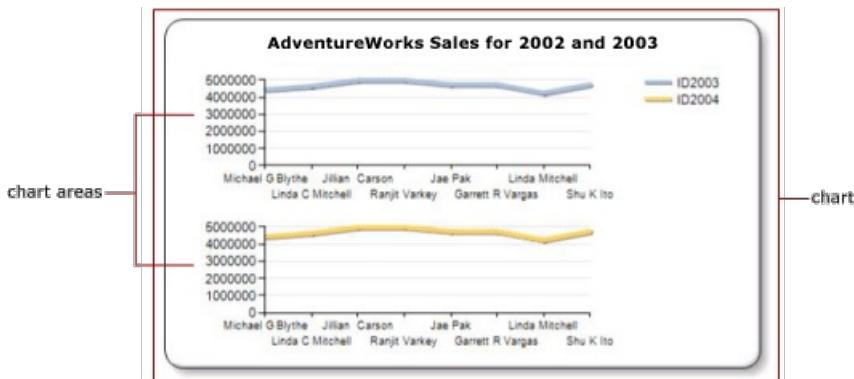
[Displaying a Series with Multiple Data Ranges on a Chart \(Report Builder and SSRS\)](#)

[Chart Types \(Report Builder and SSRS\)](#)

Specify a Chart Area for a Series (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

In Reporting Services paginated reports, the *chart* is the top-level container that includes the outer border, the chart title, and the legend. By default, the chart contains one *chart area*. The chart area is not visible on the chart surface, but you can think of the chart area as a container that encompasses only the axis labels, the axis title and the plotting area of one or more series. The following illustration shows the concept of multiple chart areas within a single chart.



By default, all series are added to the default chart area. When using area, column, line, and scatter charts, any combination of these series can be displayed on the same chart area. If you have several series in the same chart area, the readability of the chart is reduced. You may want to separate the chart types into multiple chart areas. Using multiple chart areas will increase readability for easier comparisons. For example, price-volume stock charts often have different ranges of values, but comparisons can be made between the price and volume data over the same period of time.

The bar, polar, or shape series can only be combined with series of the same chart types in the same chart area. If you are using a Polar or Shape chart, consider using a separate chart data region for each field that you wish to show.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To associate a series with a new chart area

1. Right-click anywhere on the chart and select **Add New Chart Area**. A new, blank chart area appears on the chart.
2. Right-click the series on the chart or right-click a series or data field in the appropriate area in the Chart Data pane, and then click **Series Properties**.
3. In **Axes and Chart Areas**, select the chart area that you want the series to be shown in.
4. (Optional) Align the chart areas vertically. To do this, right-click the chart and select **Chart Area Properties**. In **Alignment**, select another chart area that you want to align the selected chart area with.

See Also

- [Multiple Series on a Chart \(Report Builder and SSRS\)](#)
- [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)
- [Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)](#)
- [Polar Charts \(Report Builder and SSRS\)](#)
- [Shape Charts \(Report Builder and SSRS\)](#)
- [Pie Charts \(Report Builder and SSRS\)](#)

Plot Data on a Secondary Axis (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

The chart has two axis types: primary and secondary. The secondary axis is useful when comparing two value sets with two distinct data ranges that share a common category.

For example, suppose you have a chart that calculates Revenue vs. Tax for the year 2008. In this case, the 2008 time period is common to both value sets. However, when both series are plotted on the same y-axis, we cannot make a useful comparison because the scale of the y-axis is optimized for the largest values in the dataset. If we show Revenue on the primary axis, and Tax on the secondary axis, we can display each series on its own y-axis with its own scale of values. The series still share a common x-axis.

In situations where there are more than two series to be compared, consider a different approach for comparing and displaying multiple series in a chart. For more information, see [Multiple Series on a Chart \(Report Builder and SSRS\)](#).

An example of this chart is available as a sample report. For more information about downloading this sample report and others, see [SQL Server 2016 Report Builder and Report Designer sample reports](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To plot a series on the secondary axis

1. Right-click the series in the chart or right-click on a field in the **Values** area that you want to display on the secondary axis and click **Series Properties**. The **Series Properties** dialog box appears.
2. Click **Axes and Chart Area**, and select which of the secondary axes you want to enable, the value axis or the category axis.

See Also

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Specify an Axis Interval \(Report Builder and SSRS\)](#)

Linking Multiple Data Regions to the Same Dataset (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can add multiple data regions to a Reporting Services paginated report to provide different views of data from the same report dataset. For example, you might want to display data in a table and also display it visually in a chart. To do so, you must use identical expressions and scopes for the appropriate filter expressions, sort expressions, and group expressions.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To use a chart and a table or matrix to display the same data, it helps to understand the similarities between a table and shape charts, and a matrix and area, bar, and column charts. A table with a single row group can easily be displayed as a pie chart. As you add multiple row groups, you can choose different types of charts to best display the nested groups. Adding nested row groups to a pie chart increases the number of slices in the pie. You must decide if the number of group instances for the parent group and child group combined is too many to display in a single pie chart. For multiple group values that display as small slices on a pie chart, you can set a property so that all values below a certain threshold display as one pie slice. For more information, see [Collect Small Slices on a Pie Chart \(Report Builder and SSRS\)](#).

A table with multiple row groups can be shown as a column chart with multiple category groups. For more information, see [Display the Same Data on a Matrix and a Chart \(Report Builder\)](#). For an example of a table and chart that present different views of the same report dataset, see the Product Line Sales report in AdventureWorks Report Samples. Because both the table and the chart are linked to the same dataset in this report, when you click the interactive sort button for Employee Name in the sort the Top Employees table, the Top Employees chart also automatically shows the new sort order. For more information about downloading this sample report and others, see [SQL Server 2016 Report Builder and Report Designer sample reports](#).

A matrix with multiple row and column groups can best be displayed by using an area, bar, or column chart with both category and series groups. Use the same group expressions for column groups on the matrix and category groups on the chart, and the same group expressions for row groups on the matrix and series groups on the chart. You must keep in mind that the number of group instances affects the readability of the chart. You can define groups based on range values to reduce the number of group instances in a report. For more information, see [Group Expression Examples \(Report Builder and SSRS\)](#).

See Also

[Charts \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Nested Data Regions \(Report Builder and SSRS\)](#)

Display the Same Data on a Matrix and a Chart (Report Builder)

3/24/2017 • 4 min to read • [Edit Online](#)

When you want to show the same data in a matrix and a chart, you must set properties on both data regions to specify the same dataset, and also the same expressions for filters, groups, sorts, and data.

Because both data regions will have the same ancestor for data (the report dataset), you can add an interactive sort button to the matrix that, when the user clicks it, changes the sort order for both the matrix and the chart. For more information, see [Add Interactive Sort to a Table or Matrix \(Report Builder and SSRS\)](#).

To use the matrix column group values as a legend for the chart, you must specify the colors for the series data on the chart, and then use the same colors as the fill colors for the background of the text boxes in the matrix cell that displays the group values. For more information, see [Specify Consistent Colors across Multiple Shape Charts \(Report Builder and SSRS\)](#).

At run-time, your report may appear cluttered if there are too many group values for your group definitions. You might need to filter values, combine groups, or adjust the threshold for the chart to combine groups for you. For more information, see [Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a matrix and chart to display the same data

1. Open a report in design view.
2. From the **Insert** tab, in the **Data Regions** group, click **Matrix**, and then click the report body or in a rectangle in the report body. A matrix is added to the report.
3. From the **Insert** tab, in the **Data Regions** group, click **Chart**, and then select the chart type. A chart is added to the report.
4. (Optional) From the **Insert** tab, in the **Report Items** group, click **Rectangle**, and then click the report. A rectangle is added to the report. Drag the matrix and chart from steps 2 and 3 into the rectangle.

By placing multiple data regions in the rectangle container, you help control how the matrix and chart render when you view the report.

In the next few steps, you will choose the same dataset field to display in the matrix and to display in the chart.

5. From the Report Data pane, drag a numeric dataset field to the Data cell in the matrix.

By default, the aggregate function Sum is used for calculating the group value. If you change the aggregate function in the matrix, you must change in the chart also.

6. In the matrix, right-click the cell with data, click **Text Box Properties**, and then click **Number**. Choose an appropriate format for the dataset field value.
7. Click **OK**.

8. Drag the same dataset field you chose in step 3 to the **Values** area on the chart.
9. In the chart, right-click the Y axis, click **Axis Properties**, and then click **Number**. Choose the same format for the data that you chose in step 4.
10. Click **OK**.

In the next few steps, you will set the matrix row group and the chart series group to the same expression, and also set the sort order for the chart series group.

11. From the Report Data pane, drag the dataset field that you want to group by for matrix rows to the Row Groups pane.

By default, the matrix row group adds a sort expression that is the same as the group expression.

12. Drag the same dataset field that you used in step 9 to the **Series Groups** area for the chart.

13. Right-click the group in the **Series Groups** area, and then click **Series Group Properties**.

14. Click **Sorting**.

15. Click **Add**. A new row appears in the sort expressions grid.

16. In **Sort by**, from the drop-down list, choose the dataset field that you chose to group by in step 9.

17. Click **OK**.

In the next few steps, you will set the matrix column group and the chart category group to the same expression, and also set the sort order for the chart category group.

18. From the Report Data pane, drag the dataset field that you want to group by for matrix columns to the Column Groups pane.

By default, the matrix column group adds a sort expression that is the same as the group expression.

19. Drag the same dataset field that you used in step 16 to the **Category Groups** area for the chart.

20. Right-click the group in the **CategoryGroups** area, and then click **Category Group Properties**.

21. Click **Sorting**.

22. Click **Add**. A new row appears in the sort expressions grid.

23. In **Sort by**, from the drop-down list, choose the dataset field that you chose to group by in step 16.

24. Click **OK**.

25. Preview the result. The matrix row and column groups display the same data as the chart series and category groups.

See Also

[Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

Add or Delete a Group in a Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In Reporting Services paginated reports, click in the chart data region to display the **Chart Data** pane. Create groups by dragging dataset fields to the **Category Groups** and **Series Groups** areas. To add nested groups, add multiple fields to the area.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a parent or child group to a chart

1. On the report design surface, click anywhere in the chart to select it. The **Chart Data** pane appears.
2. Drag a field from the **Report Data** window to the **Category Groups** or **Series Groups** area. To add a parent group, position the cursor in front of an existing group. To add a child group, position the cursor after an existing group.

To edit a category group on a chart

1. On the report design surface, click anywhere in the chart to select it. The **Chart Data** pane appears.
2. Right-click the group in the **Category Groups** area, and then click **Category Group Properties**.
3. Add or remove group expressions, filters, sort expressions, and group variables.
4. Click **OK**.

To edit a series group on a chart

1. On the report design surface, click anywhere in the chart to select it. The **Chart Data** pane appears.
2. Right-click the group in the **Series Groups** area, and then click **Series Group Properties**.
3. Add or remove group expressions, filters, sort expressions, and group variables.
4. Click **OK**.

To delete a group from a chart

1. On the report design surface, click anywhere in the chart to select it. The **Chart Data** pane appears.
2. Right-click the group in the **Category Groups** or **Series Groups** area, and then click **Delete**.

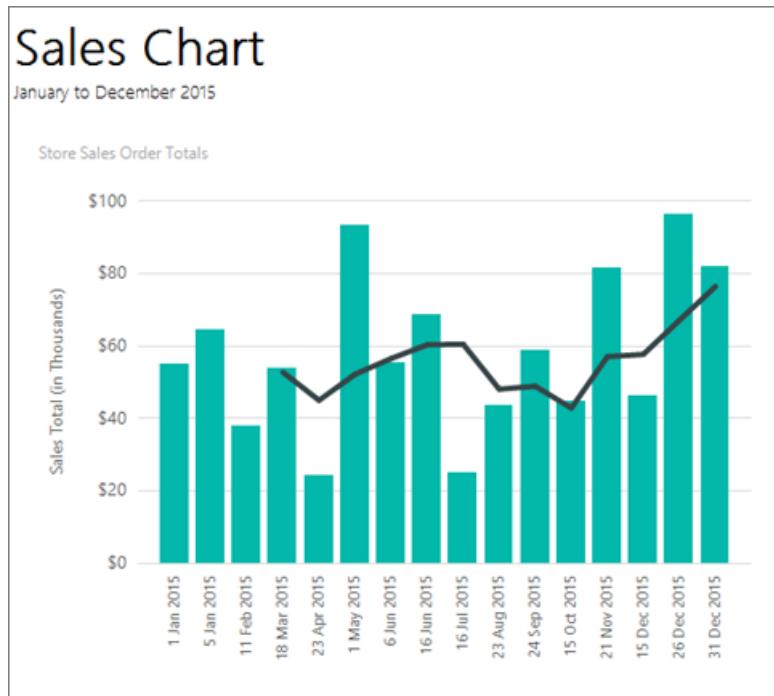
See Also

[Charts \(Report Builder and SSRS\)](#)

Add a Moving Average to a Chart (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A moving average is an average of the data in your series, calculated over a defined period of time. In Reporting Services paginated reports, the moving average can be shown on the chart to identify significant trends.



The Moving Average formula is the most popular price indicator used in technical analyses. Many other formulas, including mean, median and standard deviation, can also be derived from a series on the chart. When specifying a moving average, each formula may have one or more parameters that must be specified.

The [Tutorial: Add a Column Chart to Your Report \(Report Builder\)](#) walks you through adding a moving average to a chart, if you'd like to try it with sample data.

When a moving average formula is added in Design mode, the line series that is added is only a visual placeholder. The chart will calculate the data points of each formula during report processing.

Built-in support for trend lines is not available in Reporting Services.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a calculated moving average to a series on the chart

1. Right-click on a field in the **Values** area and click **Add Calculated Series**. The **Calculated Series Properties** dialog box opens.
2. Select the **Moving average** option from the **Formula** drop-down list.
3. Specify an integer value for the **Period** that represents the period of the moving average.

NOTE

The period is the number of days used to calculate a moving average. If date/time values are not specified on the x-axis, the period is represented by the number of data points used to calculate a moving average. If there is only one data point, the moving average formula does not calculate. The moving average is calculated starting at the second point. If you specify the **Start from first point** option, the chart will start the moving average at the first point. If there is only one data point, the point in the calculated moving average will be identical to the first point in your original series.

See Also

- [Tutorial: Add a Column Chart to Your Report \(Report Builder\)](#)
- [Formatting a Chart \(Report Builder and SSRS\)](#)
- [Charts \(Report Builder and SSRS\)](#)
- [Add Empty Points to a Chart \(Report Builder and SSRS\)](#)

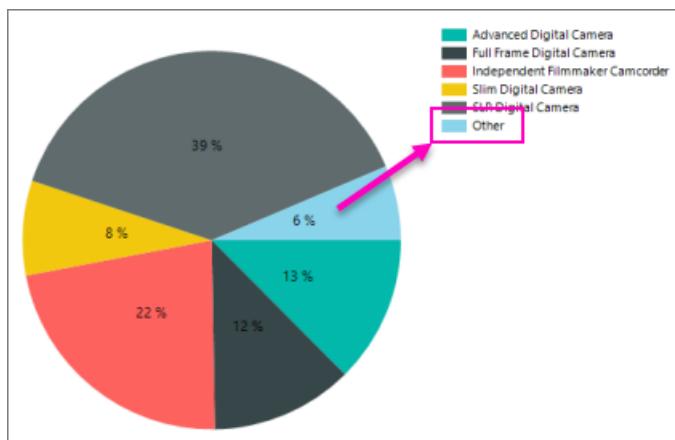
Collect Small Slices on a Pie Chart (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Pie charts with too many slices can look cluttered. Learn to collect many small slices in a pie chart into one single slice in Reporting Services paginated reports.

To collect small slices into one slice, first decide whether your threshold for collecting small slices is measured as a percentage of the pie chart or as a fixed value.

The [Tutorial: Add a Pie Chart to Your Report \(Report Builder\)](#) walks you through collecting small slices into a single slice, if you'd like to try this with sample data first.

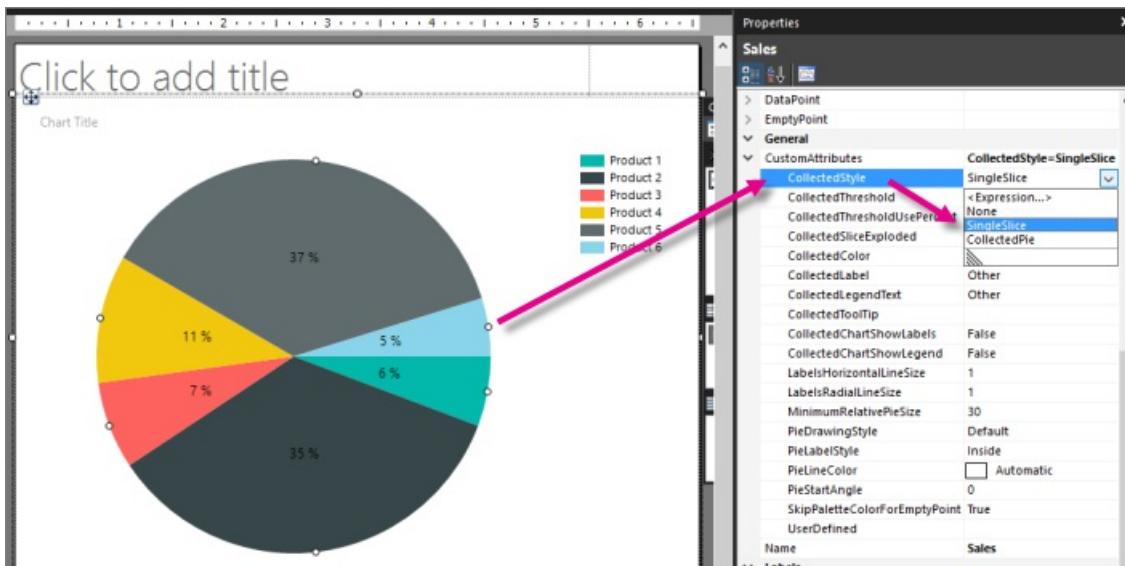


You can also collect small slices into a second pie chart that is called out from a collected slice of the first pie chart. The second pie chart is drawn to the right of the original pie chart.

You cannot combine slices of funnel or pyramid charts into one slice.

To collect small slices into a single slice on a pie chart

1. Open the Properties pane.
2. On the design surface, click on any slice of the pie chart. The properties for the series are displayed in the Properties pane.
3. In the **General** section, expand the **CustomAttributes** node.
4. Set the **CollectedStyle** property to **SingleSlice**.



- Set the collected threshold value and type of threshold. The following examples are common ways of setting collected thresholds.

- By percentage.** For example, to collect any slice on your pie chart that is less than 10% into a single slice:

Set the CollectedThresholdUsePercent property to **True**.

Set the CollectedThreshold property to **10**.

NOTE

If you set CollectedStyle to **SingleSlice**, CollectedThreshold to a value greater than **100**, and CollectedThresholdUsePercent to **True**, the chart will throw an exception because it cannot calculate a percentage. To resolve this issue, set CollectedThreshold to a value less than **100**.

- By data value.** For example, to collect any slice on your pie chart that is less than 5000 into a single slice:

Set the CollectedThresholdUsePercent property to **False**.

Set the CollectedThreshold property to **5000**.

- Set the CollectedLabel property to a string that represents the text label that will be shown on the collected slice.
- (Optional) Set the CollectedSliceExploded, CollectedColor, CollectedLabelText and CollectedToolTip properties. These properties change the appearance, color, label text, legend text and tooltip aspects of the single slice.

To collect small slices into a secondary, callout pie chart

- Follow Steps 1-3 from above.
- Set the CollectedStyle property to **CollectedPie**.
- Set the CollectedThreshold property to a value that represents the threshold at which small slices will be collected into one slice. When the CollectedStyle property is set to **CollectedPie**, the CollectedThresholdUsePercent property is always set to **True**, and the collected threshold is always measured in percent.
- (Optional) Set the CollectedColor, CollectedLabel, CollectedLabelText and CollectedToolTip properties. All

other properties named "Collected" do not apply to the collected pie.

NOTE

The secondary pie chart is calculated based on the small slices in your data so it will only appear in Preview. It does not appear on the design surface.

NOTE

You cannot format the secondary pie chart. For this reason, it is strongly recommended to use the first approach when collecting pie slices.

See Also

- [Tutorial: Add a Pie Chart to Your Report \(Report Builder\)](#)
- [Pie Charts \(Report Builder and SSRS\)](#)
- [Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)
- [Display Data Point Labels Outside a Pie Chart \(Report Builder and SSRS\)](#)
- [Display Percentage Values on a Pie Chart \(Report Builder and SSRS\)](#)

Troubleshoot Charts (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

These issues can be helpful when working with charts.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Why does my chart count, not sum, the values on the value axis?

Most chart types require numeric values along the value axis, which is typically the y-axis, in order to draw correctly. If the data type of your value field is **String**, the chart cannot display a numeric value, even if there are numerals in the fields. Instead, the chart displays a count of the total number of rows that contain a value in that field. To avoid this behavior, make sure that the fields that you use for value series have numeric data types, as opposed to strings that contain formatted numbers.

See Also

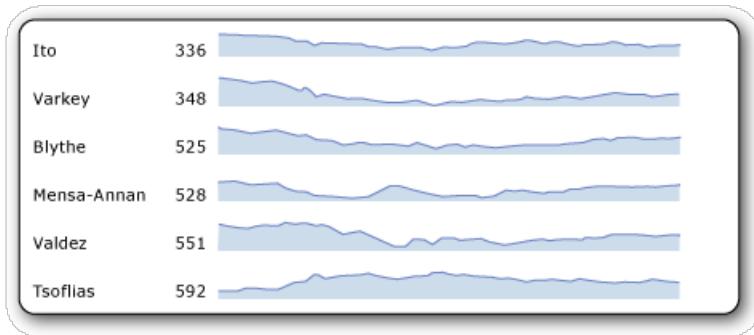
[Charts \(Report Builder and SSRS\)](#)

Sparklines and Data Bars (Report Builder and SSRS)

3/24/2017 • 5 min to read • [Edit Online](#)

Sparklines and data bars are small, simple charts that convey a lot of information in a little space, often inline with text.

In Reporting Services reports, sparklines and data bars are often used in tables and matrices. Their impact comes from viewing many of them together and being able to quickly compare them one above the other, rather than viewing them singly. They make it easy to see the outliers, the rows that are not performing like the others. Although they are small, each sparkline often represents multiple data points, often over time. Data bars can represent multiple data points, but typically illustrate only one. Each sparkline typically presents a single series. You cannot add a sparkline to a detail group in a table. Because sparklines display aggregated data, they must go in a cell associated with a group. Sparklines and data bars have the same basic chart elements of categories, series, and values, but they have no legend, axis lines, labels, or tick marks.



To quickly get started with sparklines, see [Tutorial: Add a Sparkline to Your Report \(Report Builder\)](#) and the videos [How to: Create a Sparkline in a Table](#) and [Sparklines, Bar Charts, and Indicators in Report Builder](#).

NOTE

You can publish sparklines and data bars with their parent table, matrix, or list, separately from a report as a report part. Read more about [Report Parts](#).

Types of Sparklines

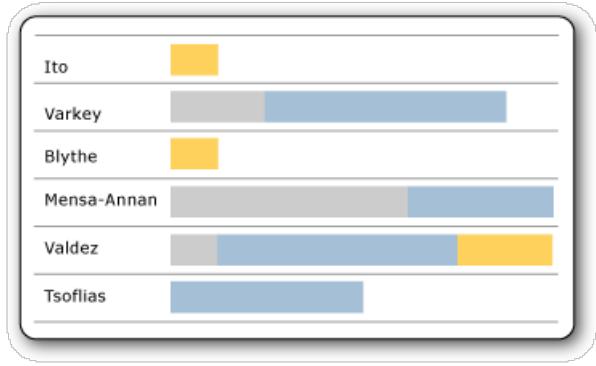
You can create almost as many types of sparklines as there are regular charts. In general, you cannot make 3D sparklines. You can make sparkline versions of these full charts:

- [Column Charts \(Report Builder and SSRS\)](#): The basic, stacked, and 100% stacked column charts.
- [Line Charts \(Report Builder and SSRS\)](#): All except the 3D line chart.
- [Area Charts \(Report Builder and SSRS\)](#): All except the 3D area charts
- [Pie Charts \(Report Builder and SSRS\)](#): And doughnut charts, both flat and 3D, but not the other shapes such as funnel and pyramid charts.
- [Range Charts \(Report Builder and SSRS\)](#): The stock, candlestick, error bar, and box plot charts.

Data Bars

Data bars typically represent a single data point, though they can represent multiple data points, just like regular

bar charts. They often contain several series with no category, or have series grouping.



In this example using stacked data bars, each data bar, although only one bar, illustrates more than one data point. For example, the three different colors of the bar could represent tasks of three levels of priority with the length of the bar representing the total number of tasks assigned to each person. If you made these 100% stacked data bars instead, each bar would fill the cell, and the different colors would represent the percentage of the whole for each priority level.

You can make data bar versions of these full charts:

- [Bar Charts \(Report Builder and SSRS\)](#): Basic, stacked, and 100% stacked bar charts.
 - [Column Charts \(Report Builder and SSRS\)](#): Basic, stacked, and 100% stacked column charts. Column charts can be either sparklines or data bars.

Aligning Sparkline Data in a Table or Matrix

When you insert a sparkline in a table or matrix, it is usually important for the data points in each sparkline to align with the data points of the other sparklines in that column. Otherwise it is hard to compare the data in the different rows. For example, when you compare sales data by month for different salespeople in your company, you would want the months to align. If an employee was out for the month of April, there would be no data for that employee for that month. You would want to see a gap for that month, and see the data for subsequent months align with the data for the other employees. You can do this by aligning the horizontal axis. For more information, see the section about sparklines in [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#), and see [Align the Data in a Chart in a Table or Matrix \(Report Builder and SSRS\)](#).

Likewise, to be comparable across rows, data must also align vertically, meaning that the height of the bars or lines in one sparkline or data bar must be relative to the height of the bars and lines in all the other sparklines or data bars. Otherwise, you can't compare the rows to each other.



In this image, the column chart shows daily sales for each employee. Note that for days that an employee has no sales, the chart leaves a blank and aligns subsequent days. This is an example of horizontal alignment. Also note that for some employees, every bar is short, and no bar reaches the top of the cell. This is an example of vertical alignment; without it, in the rows with no tall bars, the short bars would expand to fill the height of the cell.

Understanding the Data Supplied to a Sparkline or Data Bar

When you add a sparkline or data bar to a table or matrix, this is referred to as *nesting* one data region inside another. Nesting means that the data supplied to the sparkline or data bar is controlled by the dataset that the table or matrix is based on, and by where you put it in the table or matrix. For more information, see [Nested Data Regions \(Report Builder and SSRS\)](#).

Converting a Sparkline or Data Bar to a Full Chart

Because sparklines and data bars are just a kind of chart, if you decide you would rather have the full chart functionality, you can convert one to a full chart by right-clicking the chart and clicking **Convert to Full Chart**. When you do, the axis lines, labels, tick marks, and legend are added automatically.

NOTE

You cannot convert a full chart to a sparkline or data bar with one click. However, you can make a sparkline or data bar from a full chart just by deleting all the chart elements that are not in sparklines and data bars.

How-to Topics

[Add Sparklines and Data Bars \(Report Builder and SSRS\)](#)

[Align the Data in a Chart in a Table or Matrix \(Report Builder and SSRS\)](#)

Other how-to topics for charts

Because sparklines and data bars are a type of chart, you might also find the following how-to topics for charts helpful and relevant:

[Add a Chart to a Report \(Report Builder and SSRS\)](#)

[Add Empty Points to a Chart \(Report Builder and SSRS\)](#)

[Add or Remove Margins from a Chart \(Report Builder and SSRS\)](#)

[Change a Chart Type \(Report Builder and SSRS\)](#)

[Define Colors on a Chart Using a Palette \(Report Builder and SSRS\)](#)

[Show ToolTips on a Series \(Report Builder and SSRS\)](#)

[Specify a Logarithmic Scale \(Report Builder and SSRS\)](#)

[Specify an Axis Interval \(Report Builder and SSRS\)](#)

[Specify Consistent Colors across Multiple Shape Charts \(Report Builder and SSRS\)](#)

See Also

[Charts \(Report Builder and SSRS\)](#)

[Tutorial: Add a Sparkline to Your Report \(Report Builder\)](#)

Add Sparklines and Data Bars (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Sparklines and data bars are small, spare charts that convey a lot of information with little extraneous detail. For more information about them, see [Sparklines and Data Bars \(Report Builder and SSRS\)](#).

In Reporting Services paginated reports, sparklines and data bars are most commonly placed in cells in a table or matrix. Sparklines usually display only one series each. Data bars can contain one or more data points. Both sparklines and data bars derive their impact from repeating the series information for each row in the table or matrix.

To add a sparkline or data bar to a table or matrix

1. If you have not done so already, create a [table](#) or [matrix](#) with the data you want to display.
2. Insert a column in your table or matrix. For more information, see [Insert or Delete a Column \(Report Builder and SSRS\)](#).
3. On the **Insert** tab, click **Sparkline** or **Data Bar**, and then click in a cell in the new column.

NOTE

You cannot place sparklines in a detail group in a table. They must go in a cell associated with a group.

4. In the **Change Sparkline/Data Bar Type** dialog box, click the kind of sparkline or data bar you want, and then click **OK**.
5. Click the sparkline or data bar.

The **Chart Data** pane opens.

6. In the **Values** area, click the **Add Fields** plus sign (+), and then click the field whose values you want charted.
7. In the **Category Groups** area, click the **Add Fields** plus sign (+), and then click the field whose values you want to group by.

Typically for sparklines and data bars, you will not add a field to the **Series Group** area because you only want one series for each row.

See Also

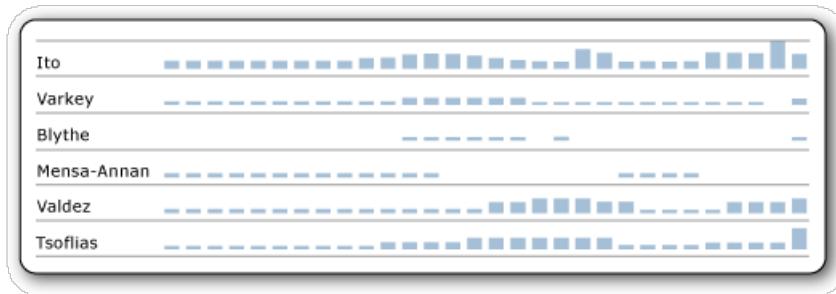
[Charts \(Report Builder and SSRS\)](#)

[Align the Data in a Chart in a Table or Matrix \(Report Builder and SSRS\)](#)

Align the Data in a Chart in a Table or Matrix (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Sparklines and data bars are small, simple charts that convey a lot of information with little extraneous detail. In a Reporting Services paginated report, when you check this option the values in your sparklines and data bars will align across the different cells in the table or matrix, even if there are missing values in the data they are based on.



In this image, the column chart shows daily sales for each employee. Note that for days that an employee has no sales, the chart leaves a blank and aligns subsequent days horizontally. It also aligns the charts vertically by making the sizes of the different charts relative to each other. For more information, see [Sparklines and Data Bars \(Report Builder and SSRS\)](#).

Align the data in a sparkline or data bar

1. [Add a sparkline or data bar](#) to a table or matrix.
2. Click in the sparkline or data bar, and then click **Horizontal Axis Properties** or **Vertical Axis Properties**.
3. On the **Axis Options** tab, check the **Align axes in** box, and then in the dropdown box, select the group on which to align the axis.
4. Click **OK**.

See Also

[Charts \(Report Builder and SSRS\)](#)

[Add Sparklines and Data Bars \(Report Builder and SSRS\)](#)

Gauges (Report Builder and SSRS)

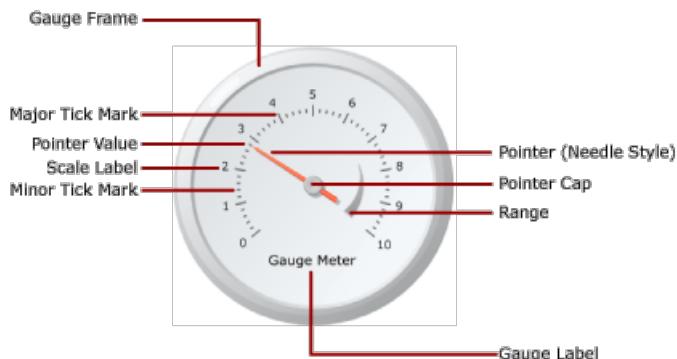
3/24/2017 • 12 min to read • [Edit Online](#)

In Reporting Services paginated reports, a gauge data region displays a single value from your dataset. An individual gauge is always positioned inside a gauge panel, where you can add child or adjacent gauges. Inside a single gauge panel, you can create multiple gauges that share common functions such as filtering, grouping, or sorting.

Gauges can perform many tasks in a report:

- Display key performance indicators (KPIs) in a single radial or linear gauge.
- Place a gauge inside a table or matrix to illustrate values inside each cell.
- Use multiple gauges in a single gauge panel to compare data between fields.

There are two types of gauges: radial and linear. The following illustration shows the basic elements of a single radial gauge in the gauge panel.



For more information about using gauges as KPIs, see [Tutorial: Adding a KPI to Your Report \(Report Builder\)](#).

NOTE

You can publish gauges separately from a report as report parts. Read more about [Report Parts](#).

Gauge Types

Reporting Services provides two gauge types: radial and linear. The radial gauge is typically used when you want to express the data as a velocity. The linear gauge is used to express the data as a temperature or scale value.

The key differences between the two types are the overall shape of the gauge and the available gauge pointers. Radial gauges are circular, or degrees of circular, and resemble speedometers. The gauge pointers are often needles, but can be markers or bars.

Linear gauges are rectangular, oriented horizontally or vertically, and resemble rulers. The gauge pointers are often thermometers, but can be markers or bars. Because of its shape, this gauge type is useful for integrating into the table or matrix data regions to show progress data.

Other than these differences, the two gauge types are interchangeable. However, if you have to use a simple gauge in your report, you should consider using an indicator instead of a gauge. For more information, see

Indicators (Report Builder and SSRS).

The following illustrations show radial and linear gauges. The radial gauge is round and uses the needle pointer. The linear gauge is horizontal and uses the thermometer pointer.

Radial Gauge



Radial gauge options: Radial, Radial with Mini Gauge, Two Scales, 90 Degrees Northeast, 90 Degrees Northwest, 90 Degrees Southwest, 90 Degrees Southeast, 180 Degrees North, 180 Degrees South, 180 Degrees West, 180 Degrees East, and Meter.

Linear Gauge



Linear gauge options: Horizontal, Vertical, Multiple Bar Pointers, Two Scales, Three Color Range, Logarithmic, Thermometer, Thermometer Fahrenheit/Celsius, and Bullet Graph.

Adding Data to a Gauge

After you add a gauge to the design surface, drag a dataset field to the gauge data pane. By default, the gauge aggregates field values into one value that is shown on the gauge. That value is attached to the pointer by using the Value property. Depending on the data type of the field, the gauge uses the SUM or COUNT aggregate. When you use numeric data, appropriate to add, the gauge uses the SUM function. Otherwise, it uses the COUNT aggregate. The value of the pointer can use a different aggregate, or no aggregate.

You can add grouping to the gauge to view individual groups or individual rows on the gauge. When grouping and filtering is applied, the gauge uses the pointer value to display the last group or row in the returned dataset.

You can add multiple values to an individual gauge by adding another pointer. This pointer can belong to the same scale, or you can add another scale and then associate the pointer with that scale.

Unlike the chart types available in the **Select Chart Type** dialog box, the gauge types available in the **Select Gauge Type** dialog box are created by using a combination of gauge properties. Therefore, you cannot change the gauge type the same way you change a chart type. To change the gauge type, you must remove the gauge and re-add it to the design surface. A gauge has at least one scale and one pointer. You can have multiple scales by right-clicking the gauge and selecting **Add Scale**. By default, this creates a smaller scale that is positioned inside the first scale. The scale displays labels and tick marks. There are two sets of tick marks, minor and major.

You can have multiple pointers by right-clicking the gauge and selecting **Add Pointer**. This will create another pointer on the same scale, but if you have multiple scales, you can associate a pointer with any scale on the gauge.

Considerations When Adding Data to the Gauge

Like all other data regions, the Gauge data region can be bound to only one dataset. If you have multiple datasets, consider using a JOIN or UNION to create one dataset, or use separate gauges for each dataset.

Numeric data types are aggregated with the SUM function. Non-numeric data types are aggregated with the COUNT function, which counts the number of instances for a particular value or field within the dataset or group.

After data is added, when you right-click the pointer, you get Clear Pointer Value and Delete Pointer options. The

Clear Pointer Value option removes the field attached to the gauge, but the pointer will still appear on the gauge. The Delete Pointer option removes the field from the gauge and deletes the pointer from view. If you re-add a field to the gauge, the default pointer reappears. After you have added the field to the gauge, you must set the maximum and minimum values on the corresponding scale to give context to the value on the gauge. You can also set the minimum and maximum values on a range, which shows a critical area on the scale. The gauge will not automatically set the minimum or maximum values on the scale or the range because it cannot determine how the value should be perceived.

Methods of Adding Data to a Gauge

After you have defined a dataset for your report, you can add a data field to the gauge by using one of the following approaches:

- Drag a field from your dataset into the data pane. Click the gauge and drag a field to it. You can open the data pane by clicking on the gauge or dragging a field across the gauge. If a pointer is not already on the gauge, a pointer is added to the gauge and bound to the field you have added.
- Display the data pane and point to the field placeholder. Click the down arrow next to the field placeholder and select the field that you want to use. If there is a field already selected, click the down arrow, and then select a different field.

NOTE

This approach is not applicable when there is no pointer on the gauge or the report contains more than one dataset and the gauge panel is not associated with a dataset.

- Right-click on the gauge pointer and select **Pointer Properties**. For **Value**, select a field from the drop-down list or define a field expression by clicking the **Expression** (fx) button.

Aggregating Fields into a Single Value

When a field is added to a gauge, Reporting Services calculates an aggregate for the field by default. Numeric data types are aggregated with the SUM function. Non-numeric data types are aggregated with the COUNT function, which counts the number of instances for a particular value or field within the dataset or group. If the data type of your value field is String, the gauge cannot display a numeric value, even if there are numerals in the fields. Instead, the gauge aggregates string fields using the COUNT function. To avoid this behavior, make sure that the fields you use have numeric data types, as opposed to strings that contain formatted numbers. You can use a Visual Basic expression to convert String values to a numeric data type using the CDbl or CInt constant. For example, the following expression converts a string field called MyField to numeric values.

```
=Sum(CDbl(Fields!MyField.Value))
```

For more information about aggregate expressions, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

Defining a Group on a Gauge

After you have added a field to the gauge, you can add one data group. The Gauge differs from all other data regions in Reporting Services, which can display multiple groups in one data region. When you add a group by defining a group expression on the gauge, it is the same as when you add a row group on the Tablix data region. However, when the group is added, only the value of the last group is displayed as the pointer value on the gauge. For example, if you add a grouping expression on Year, the pointer will point to the value that represents the aggregate sales value for the last year in the dataset. For more information about groups, see [Understanding Groups \(Report Builder and SSRS\)](#).

You might want to add a group on the gauge if, for example, you are displaying multiple gauges in a table or list and you want to display data aggregated by group. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#).

Positioning Elements in a Gauge

The gauge panel is the top-level container that holds one or more gauges. You can view the **Gauge Panel Properties** dialog box by clicking just outside the gauge. Each gauge, in turn, contains several gauge elements: a gauge scale, a gauge range, and a gauge pointer. When you are using the gauge, you will need to understand how elements are measured in the gauge panel in order to modify the size and location of these elements.

Understanding Size and Position Measurements

All size and position measurements on the gauge are calculated as a percentage of their parent element. When parent elements have different width and height values, the size of the gauge element is calculated as a percentage of the smaller of the two values. For example, on a linear gauge, all pointer measurements are calculated as a percentage of the width or height of the linear gauge, whichever is smaller.

Position measurements are also calculated as a percentage of their parent element using a coordinate system. The origin of this coordinate system is in the upper-left corner with the X axis pointing to the right and the Y axis pointing down. Coordinate values should be between 0 and 100, and all measurements are represented as a percentage. For example, when the X and Y positions of the linear gauge are set to 50 and 50, the linear gauge is positioned in the middle of the gauge panel.

Positioning Multiple Gauges inside the Gauge Panel

There are two ways to add a new gauge to a gauge panel that already contains one gauge. You can add a gauge as a child of the first one or you can add another gauge adjacent to the first one.

When a new gauge is added to a gauge panel, it is sized and positioned in equal proportion to all other gauges in the gauge panel. For example, if a radial gauge is added to a gauge panel that already contains a radial gauge, the two gauges will automatically be resized to each fit half of the panel.

You can add a new gauge to a gauge panel that already contains a gauge. To do this, right-click anywhere on the gauge panel, hover over **Add New Gauge** and select **Child**. The **Select Gauge Type** dialog box will appear. When the new gauge is added as a child, it is added in one of two ways. In a radial gauge, the child gauge is positioned in the top-left corner of the first gauge. In a linear gauge, the child gauge is positioned in the middle of the first gauge. You can position the child gauge, relative to the parent gauge, by using the **Position** properties. As with all other elements, position measurements are calculated as a percentage of their parent element.

Positioning Gauge Scale Labels and Gauge Ranges

There are two properties that determine the position of labels on a gauge scale. You can set the gauge scale's **Placement** property to specify whether the labels are displayed inside, outside or across the scale bar. You can also specify a numeric value for the **Distance from scale** property, which specifies the number of units that are added or subtracted from the placement to determine the label position. For example, if **Placement** is set to **Outside** and you have set **Distance from scale** to 10, the labels will be positioned 10 units from the outer edge of the gauge scale, where 1 unit is either:

- 1% of the gauge diameter on a radial gauge, or
- 1% of the smallest value of the gauge height or width on a linear gauge.

The **Position** and **Distance from scale** properties also apply to gauge ranges.

Maintaining Aspect Ratio on a Linear Gauge

The radial gauge assumes a circular form, so this gauge type usually maintains the same width and height values. However, on a linear gauge, which assumes a rectangular form, the proportion between the width and height is usually uneven. The aspect ratio of a gauge determines the proportion of width to height that should be maintained when the gauge is resized. For example, if this value is set to 2, the width of the gauge will always be twice the height of the gauge, no matter how the gauge is resized. To set the aspect ratio, you can set the **AspectRatio** property from the **Linear Gauge Properties** dialog box.

How-To Topics

This section lists procedures that show you, step by step, how to work with gauges in your reports; how to get data to display effectively in gauges and how to add and configure gauges and their elements.

- [Add a Gauge to a Report \(Report Builder and SSRS\)](#)
- [Set a Minimum or Maximum on a Gauge \(Report Builder and SSRS\)](#)
- [Set a Snapping Interval on a Gauge \(Report Builder and SSRS\)](#)
- [Specify an Image as a Pointer on a Gauge \(Report Builder and SSRS\)](#)

In This Section

The following topics provide additional information about working with gauges.

Term	Definition
Formatting Scales on a Gauge (Report Builder and SSRS)	Provides general information about formatting scales on gauges and detailed information about the formatting options for scales on radial and linear gauges.
Formatting Pointers on a Gauge (Report Builder and SSRS)	Provides general information about formatting pointers on gauges and detailed information about the formatting options for pointer styles available for radial and linear gauges.
Formatting Ranges on a Gauge (Report Builder and SSRS)	Provides information about formatting ranges on gauges to indicate an important subsection of values on the gauge or visually indicate when the pointer value has gone into a certain span of values.

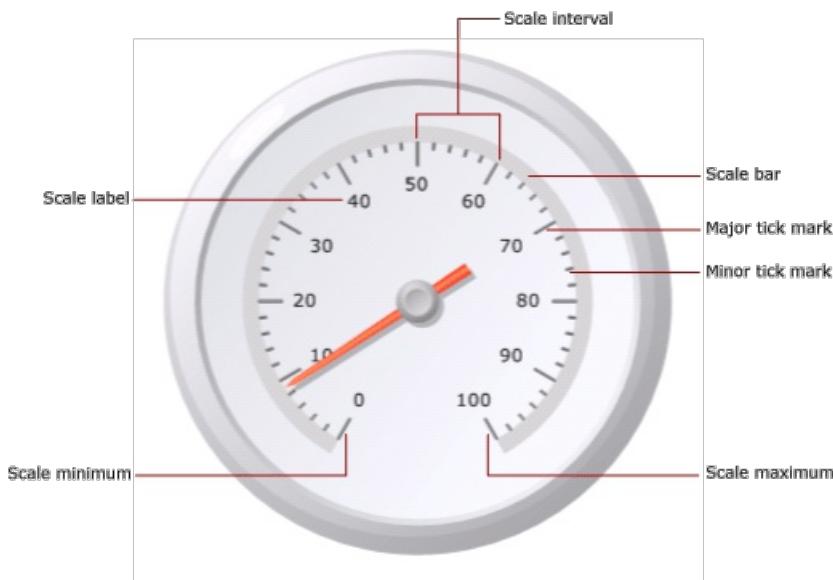
See Also

- [Expressions \(Report Builder and SSRS\)](#)
[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)
[Report Parameters \(Report Builder and Report Designer\)](#)
[Charts \(Report Builder and SSRS\)](#)
[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Formatting Scales on a Gauge (Report Builder and SSRS)

3/24/2017 • 7 min to read • [Edit Online](#)

In a Reporting Services paginated report, the gauge scale is the range of numbers, bound by a minimum and a maximum, shown on a gauge. Typically, a gauge scale contains gauge labels and tick marks to give an accurate reading of what the gauge pointer shows. A gauge scale is generally associated with one or more gauge pointers. You can have more than one scale on the same gauge.



Unlike the chart, where multiple groups are defined, the gauge only shows one value. You must define the minimum and maximum values of the scale. The interval numbers are calculated automatically based on the values specified for the minimum and maximum.

When you add a second scale on a gauge that already contains one scale, the appearance properties of the first scale are cloned on the second scale.

You can set properties on the scale by right-clicking on the scale labels or tick marks and selecting **Radial Scale Properties** or **Linear Scale Properties**. Each gauge type contains at least one scale with the same set of properties. There are also properties unique to each gauge type:

- On a radial gauge, you can specify the radius, start angle, and sweep angle of the radial scale.
- On a linear gauge, you can specify the width of the start and end margins relative to the endpoints on a linear gauge.

To quickly get started with formatting scales, see [Set a Minimum or Maximum on a Gauge \(Report Builder and SSRS\)](#).

Defining Minimum, Maximum, and Intervals on a Scale

A gauge is frequently used to display KPIs that are measured in percentages from 0 to 100, so these are the default values given to the minimum and maximum properties on the gauge. However, these values may not represent the scale of values that you are trying to show. Because there is no built-in logic to determine what the KPI data field represents, the gauge does not automatically calculate minimum and maximum values. If your KPI data field is not a value between 0 and 100, you must explicitly set values for the minimum and maximum

properties in order to give context to the one value that is being displayed on the gauge.

On the scale are major and minor tick marks. In addition, the scale has labels that are typically associated with the major tick marks. For example, a scale might have major tick marks at 0, 20, 40, 60, 80, and 100. The labels should correspond to those tick marks. The difference between the label values is called the scale interval. In this example, the scale interval is set to 20. You can set the Interval property in the **Radial Scale Properties** or **Linear Scale Properties** dialog box.

The application calculates scale intervals according to the following steps:

1. Specify minimum and maximum values. These values are not calculated automatically based on your dataset, so you must provide the values on the gauge's **Properties** dialog box.
2. If you do not specify a value for Interval, the default value is Auto. This means the application will calculate an equidistant number of intervals based on the minimum and maximum values specified in the first step. If you do specify a value for Interval, the gauge will calculate the difference between the maximum and minimum value, and divide that number by the value specified in the Interval property.

There are also properties to define label and tick mark intervals. If you specify a value for these properties, they will override the value specified for the scale interval property. For example, if the scale interval is Auto, but you specify 4 for the label interval, the labels will be displayed as 0, 4, 8 and so on, but the major tick marks will still be calculated by the gauge based on its own calculation. This may cause situations where the labels are not in sync with the tick marks. Consider hiding tick marks from display if you set a label interval.

The interval offset determines the number of units that will be skipped before the first label is shown. All successive major tick marks and labels that appear on the scale will use the interval specified. A value of 0 for label or tick mark intervals is the same as resetting the interval to Auto.

Reducing Label Collisions with Multipliers

If your values contain many digits, they may begin to obscure the readability of the gauge. You can use a scale multiplier to increase or reduce the scale of the values. When a scale multiplier is specified, each original value on the scale is multiplied by the multiplier before being displayed on the scale. To reduce the scale of values, you must specify a decimal number. For example, if your scale goes from 0 to 10000 but you want to show the numbers 0 to 10 on the gauge, you can use a multiplier value of 0.001.

NOTE

Using a multiplier does not multiply the actual value of the aggregated field that the gauge uses. It only multiplies the values of the labels displayed on the gauge after the minimum, maximum, and intervals are defined. Consider keeping the interval calculations as automatic when using a multiplier.

Specifying the Scale Bar Width, Radius, and Angles on a Radial Scale

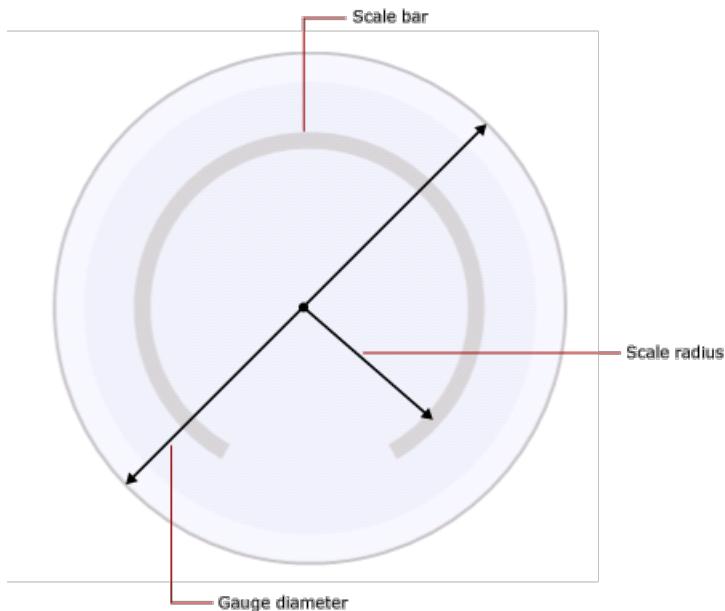
Use the **Layout** page of the **Radial Scale Properties** dialog box to set the scale bar width, scale radius, start angle, and sweep angle of the scale. You can use these properties to customize the size and format of the scale. For example, if you position scale labels outside the scale, you will need to resize the radius of the scale so that the labels fit inside the gauge.

NOTE

When you click the scale of a gauge, a dotted outline appears around the scale. This outline is not the scale bar, and it is not used when calculating measurements on the gauge. It is available only at design time so that you can highlight the scale in order to access the scale properties.

All measurements are based on the scale bar. When you select a gauge, the scale bar width is not shown. If you specify a value for the scale bar, it will help you with all other measurements related to the scale. To view the scale bar, set the **Scale bar width** property on the **Layout** page of the **Radial Scale Properties** dialog box to a value greater than 0. On a radial gauge, the scale bar is measured as a percentage of the gauge diameter. On a linear gauge, the scale bar is measured as a percentage of the gauge width or height, whichever is smallest.

The scale radius is the distance from the center of the gauge to the middle of the scale bar. The value of the scale radius is measured as a percentage of the gauge diameter. It is a good practice to keep the scale radius below a value of 35. If you specify a value higher than 35, the scale will likely be drawn outside the boundaries of the gauge. The following illustration shows how the scale radius is measured, relative to the diameter of the gauge, on the scale bar.



The start angle is the angle of rotation, between 0 and 360, at which the scale will begin. The zero (0) position is located at the bottom of the gauge, and the start angle rotates clockwise. For example, a start angle of 90 degrees starts the scale at the 9 o'clock position.

The sweep angle is the number of degrees, between 0 and 360 that the scale will sweep in a circle. A sweep angle of 360 degrees produces a scale that is a complete circle. This is useful if you want to design a gauge to look like a clock.

Positioning Labels on a Linear or Radial Scale

There are two properties that determine the position of labels. The label placement property specifies whether the labels are displayed on the inside, outside, or across the scale bar. The distance property sets the label distance from the scale, starting at the scale bar. If you want to position labels within the scale bar, specify a negative number. For example, if your labels are outside the scale and you have set a distance from scale of 10, the labels will be shown at 10 units outside where the labels would typically be placed, where 1 unit is either:

- 1% of the gauge diameter on a radial gauge, or
- 1% of the smallest value of the gauge height or width on a linear gauge.

See Also

- [Formatting Ranges on a Gauge \(Report Builder and SSRS\)](#)
- [Formatting Pointers on a Gauge \(Report Builder and SSRS\)](#)
- [Format Axis Labels as Dates or Currencies \(Report Builder and SSRS\)](#)
- [Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)
- [Gauges \(Report Builder and SSRS\)](#)

Formatting Pointers on a Gauge (Report Builder and SSRS)

3/24/2017 • 5 min to read • [Edit Online](#)

In a Reporting Services paginated report, the gauge pointer indicates the current value of the gauge.

By default, when a field is added, the values that are contained in the field are aggregated into one value that is shown by the pointer on the gauge. You can add multiple pointers to the gauge to point at multiple values on the same scale, or add multiple scales and a pointer for every scale you have added. After you add a field to a gauge, you must set the maximum and minimum values on the corresponding scale to give context to the pointer value. You also have the option of setting the minimum and maximum values on a range, which shows a critical area on the scale.

You can set appearance properties on the pointer by right-clicking on the pointer and selecting **Radial Pointer Properties** or **Linear Pointer Properties**. Each gauge pointer contains the same set of properties. There are also corresponding appearance properties unique to each gauge type:

- On a radial gauge, you can specify a needle pointer and a needle cap.
- On a linear gauge, you can specify a thermometer pointer, which is a variation of the bar pointer. The thermometer pointer lets you specify the shape of the bulb.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

How the Pointer is Connected to Data

By default, when a gauge is added, it contains one pointer that has no associated field. This is known as an empty pointer. It will display zero until a field is added to the data pane. When you add a field to the data pane, the pointer is connected to that field. If you delete a field from the data pane, the pointer that is associated with that field is also deleted.

After data is added, when you right-click on the pointer, you will get **Clear Pointer** and **Delete Pointer** options. The **Clear Pointer Value** option will remove the field that is attached to the gauge, but the pointer will still appear on the gauge. The **Delete Pointer** option will remove the field from the gauge and delete the pointer from view. If you re-add a field to the gauge, the default pointer will reappear. When you set the pointer's **Hidden** property to **True**, the pointer is not hidden on the design surface, but it is hidden at run time.

Displaying Multiple Pointers on the Gauge

You can add multiple pointers to the gauge to point at multiple values on the same scale. This can be useful for showing a low and a high value at the same time. To specify more than one pointer on the gauge for the same scale, right-click anywhere inside the gauge and click **Add Pointer** on the shortcut menu. Alternatively, you can add a scale by right-clicking anywhere in the gauge and clicking **Add Scale**. Then you can add a new pointer, and it will automatically be associated with the last scale.

When pointers overlap, the drawing order of the pointers is determined by the order in which they are added to the gauge. You cannot reorder the drawing order of the pointers by changing the order of the fields in the data

pane. To change the order of drawing for multiple pointers, open the Properties pane and click **Pointers (...)**. Then, change the order of the pointers in the Pointer collection.

Setting Gradients on a Needle Cap

You can specify a needle cap that can be drawn on top or underneath the pointer on a radial gauge only. All needle cap styles are drawn by using built-in gradients that cannot be modified. The exception is the **RoundedDark** style, where you can specify a gradient color and gradient style.

Setting a Snapping Interval

A snapping interval defines the multiple to which values are rounded. By default, the gauge will point to the exact value of the field you have specified in the data pane. However, you might want to round the exact value up or down so that the pointer will snap to a pre-set interval. For example, if the value on your gauge is 34.2 and you specify a snapping interval of 5, the gauge pointer will point to 35. If the value on your gauge is 31.2 and you specify a snapping interval of 5, the gauge pointer will point to 30. For more information, see [Set a Snapping Interval on a Gauge \(Report Builder and SSRS\)](#).

Specifying an Image as a Pointer on a Radial Gauge

In addition to the built-in list of pointer styles, you can specify an image as a pointer. This is most effective when you use an image to replace an existing needle pointer style. The image is superimposed over the pointer, but all pointer functionality is applicable. Color and gradient options are not applicable when an image is used for the pointer.

If the pointer image is an irregular shape, you should define the color as transparent to hide the areas of your image that should not appear on the gauge. When you define a transparent color, the gauge transposes the image on top of your existing pointer and trims the image so that only the shape of the pointer appears. The gauge rescales the image to fit the size of your pointer. When you specify an image for a pointer, any subsequent pointer that is added on top of the gauge will be drawn underneath the image. For this reason, it is better not to specify an image for the pointer if there are multiple pointers on the gauge. For more information, see [Specify an Image as a Pointer on a Gauge \(Report Builder and SSRS\)](#).

See Also

[Formatting Scales on a Gauge \(Report Builder and SSRS\)](#)

[Formatting Ranges on a Gauge \(Report Builder and SSRS\)](#)

[Gauges \(Report Builder and SSRS\)](#)

Formatting Ranges on a Gauge (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a Reporting Services paginated report, the gauge range is a zone or area on the gauge scale that indicates an important subsection of values on the gauge. Using a gauge range, you can visually indicate when the pointer value has gone into a certain span of values. Ranges are defined by a start value and an end value.

You can also use ranges to define different sections of a gauge. For example, on a gauge with values from 0 to 10, you can define a red range that has a value from 0 to 3, a yellow range that has a value from 4 to 7 and a green range that has a value from 8 to 10. If the start value that you specified is greater than the end value that you specified, the values are swapped so that the start value is the end value and the end value is the start value.

You can position the range in the same way that you position pointers on a scale. The **Position** and **Distance from scale** properties determine the position of the range. For more information, see [Gauges \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

See Also

- [Formatting Scales on a Gauge \(Report Builder and SSRS\)](#)
- [Formatting Pointers on a Gauge \(Report Builder and SSRS\)](#)
- [Set a Minimum or Maximum on a Gauge \(Report Builder and SSRS\)](#)
- [Tutorial: Adding a KPI to Your Report \(Report Builder\)](#)
- [Gauges \(Report Builder and SSRS\)](#)

Add a Gauge to a Report (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a Reporting Services paginated report, when you want to summarize data in a visual format, you can use a gauge data region. After you add a gauge data region to the design surface, you can drag report dataset fields to a data pane on the gauge.

To add a gauge to your report

1. Create a report or open an existing report.
2. On the Insert tab, double-click Gauge. The **Select Gauge Type** dialog box opens.
3. Select the type of gauge you want to add. Click **OK**.

NOTE

Unlike charts, gauges have only two types: linear and radial. The available gauges in the **Select a Gauge Type** dialog box are templates for these two types of gauges. For this reason, you cannot change the gauge type after you add the gauge to your report. You must delete and re-add a gauge to change its type.

If the report has no data source and dataset the **Data Source Properties** dialog box opens and takes you through the steps to create both. For more information see, [Add and Verify a Data Connection \(Report Builder and SSRS\)](#).

If the report has a data source, but no dataset the **Dataset Properties** dialog box opens and takes you through the steps to create one. For more information, see [Create a Shared Dataset or Embedded Dataset \(Report Builder and SSRS\)](#).

4. Click the gauge to display the data pane. By default, a gauge has one pointer corresponding to one value. You can add additional pointers.
5. Add one field from the dataset to the data field drop-zone. You can add only one field. If you want to display multiple fields, you must add additional pointers, one for each field.

Right-click the gauge scale, and select **Scale Properties**. Type a value for the **Minimum** and **Maximum** of the scale. For more information, see [Set a Minimum or Maximum on a Gauge \(Report Builder and SSRS\)](#).

See Also

[Nested Data Regions \(Report Builder and SSRS\)](#)

[Gauges \(Report Builder and SSRS\)](#)

Set a Minimum or Maximum on a Gauge (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Unlike charts in a Reporting Services paginated report, where multiple groups are defined, gauges show only one value. Because Report Builder and Report Designer determine the context or relative significance of the one value that you are trying to show on the gauge, you must define the minimum and maximum of the scale.

For example, if your data values are rankings between 0 and 10, you will want to set the minimum to 0 and maximum to 10. The interval numbers are calculated automatically based on the values specified for the minimum and maximum. By default, the minimum and maximum are set to 0 and 100, but this is an arbitrary value that you should change. The application does not calculate your value as a percentage.

If the range of your values is large, for example from 0 to 10000, consider using a multiplier to reduce the number of zeroes on the gauge. The multiplier will only reduce the scale of the numbers on the gauge, not the value itself.

You can use expressions to set the values of the **Minimum** and **Maximum** options. For more information, see [Expressions \(Report Builder and SSRS\)](#).

To set the minimum and maximum on the gauge

1. Right-click on the scale and select **Scale Properties**. The **Scale Properties** dialog box appears.
2. In **General**, specify a value for **Minimum**. By default, this value is 0. Optionally, click the **Expression (fx)** button to edit the expression that sets the value of the option.
3. Specify a value for **Maximum**. By default, this value is 100. Optionally, click the **Expression (fx)** button to edit the expression that sets the value of the option.
4. (Optional) If the values for your minimum and maximum are large, specify a value for the **Multiply scale labels by** option. To specify a multiplier that reduces your scale, use a decimal number. For example, if you have a scale from 0 to 1000, you can specify a multiplier value of 0.01 to reduce the scale to read 0 to 10.

See Also

[Formatting Scales on a Gauge \(Report Builder and SSRS\)](#)

[Formatting Pointers on a Gauge \(Report Builder and SSRS\)](#)

[Gauges \(Report Builder and SSRS\)](#)

Indicators (Report Builder and SSRS)

3/24/2017 • 10 min to read • [Edit Online](#)

In Reporting Services paginated reports, indicators are small gauges that convey the state of a single data value at a glance. The icons that represent indicators and their states are simple and visually effective even when used in small sizes.

You can use state indicators in your reports to show the following:

- **Trends** by using trending-up, flat (no change), or trending-down arrows.
- **State** by using commonly recognized symbols such as checkmarks and exclamation marks.
- **Conditions** by using commonly recognized shapes such traffic lights and signs.
- **Ratings** by using common recognized shapes and symbols that show progress such number of quadrants in a square and stars.

You can use indicators by themselves in dashboards or free-form reports, but they are most commonly used in tables or matrices to visualize data in rows or columns. The following diagram shows a table with a traffic light indicator that conveys year to date sales by sales person and territory.

Territory	First Name	Last Name	Sales YTD	Indicator
<input checked="" type="checkbox"/> Northeast	Sales for: Northeast		\$4,557,045	
<input checked="" type="checkbox"/> Northwest	Pamela	Ansam-Wolfe	\$0	
	David	Campbell	\$3,587,378	
	Tete	Mensa-Annan	\$1,931,620	
	Sales for: Northwest		\$5,518,999	
<input checked="" type="checkbox"/> Southeast	Tsvi	Reiter	\$2,811,013	
	Sales for: Southeast		\$2,811,013	
<input checked="" type="checkbox"/> Southwest	Sales for: Southwest		\$8,219,201	

Reporting Services provides built-in indicator sets and indicator icons to use as is, and you can also customize individual indicator icons and indicators sets to suit your needs.

For more information about using indicators as KPIs, see [Tutorial: Adding a KPI to Your Report \(Report Builder\)](#).

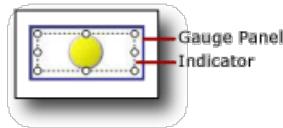
NOTE

You can publish indicators separately from a report as report parts. Read more about [Report Parts](#).

Comparing Indicators to Gauges

Although they look very different, indicators are just simple gauges. Both indicators and gauges display a single data value. The key differences are that gauges have elements such as frames and pointers. Indicators have only states, icons, and (optionally) labels. Indicator states are similar to gauge ranges.

Like gauges, indicators are positioned inside a gauge panel. When you want to configure an indicator by using the **Indicators Properties** dialog box or the Properties pane, you need to select the indicator instead of the panel. Otherwise, the available options apply to the gauge panel options and you cannot configure the indicator. The following picture shows a selected indicator in its gauge panel.



Depending on how you want to depict the data value, gauges might be more effective than indicators. For more information, see [Gauges \(Report Builder and SSRS\)](#).

Choosing the Indicator Type to Use

Using the right indicator set is key to instantly communicating the meaning of the data, whether the data is in a detail row or row or column group in a table or matrix, or by itself in the report body or dashboard. The built-in indicator sets have three or more icons. The icons can vary by shape, color, or both. Each icon communicates a different data state.

The following table lists the built-in indicator sets and describes some common uses of them.

INDICATOR SET	INDICATOR TYPE
	Directional: indicates trends using up, down, flat (no change), up-trending, or down-trending arrows.
	Symbols: indicates states using commonly recognized symbols such as checkmarks and exclamation marks.
	Shape: indicates conditions using commonly shapes such as traffic signs and diamond shapes.
	Ratings: indicates ratings by using common recognized shapes and symbols that show progressive values such as number of quadrants in a square.

After you choose a indicator set, you can customize the appearance of each indicator icon in the set by setting its properties in the dialog boxes for indicators or the Properties pane. You can use the built-in colors, icons, and sizes or expressions to configure indicators.

Customizing Indicators

Indicators can be customized to suit your needs. You can modify the indicators sets as well as individual indicator icon within a set in the following ways:

- Change the colors of indicator icons. For example, you might want the color scheme of an indicator set to be monochromatic or use colors other than the default ones.

- Change the icon in the indicator set. For example, you might want to use the star, circle, and square icons in one indicator set.
- Specify the start and end values for an indicator. For example, you might want to skew data display by using one icon for 75 percent of the indicator values.
- Add icons to the indicator set. For example, you might want to add additional icons to indicator sets to differentiate the indicator values in a more detailed way.
- Delete icons from the indicator set to make the data display simpler by using only a few icons.

For more information, see [Change Indicator Icons and Indicator Sets \(Report Builder and SSRS\)](#).

Using Indicators in Tables and Matrices

The simple shapes of indicators make them ideal to use in tables and matrices. Indicators are effective even in small sizes. This makes them useful in detail or group rows of reports.

The following diagram shows a report with a table that uses the directional indicator set, **Four Arrows (Colored)**, to indicate sales. The indicator icons in the report are configured to use shades of blue instead of the default colors: red, yellow, and green.

Territory	First Name	Last Name	Sales YTD	Indicator
<input checked="" type="checkbox"/> Northeast	Michael	Blythe	\$4,557,045	↑
	Sales for: Northeast		\$4,557,045	↗
<input checked="" type="checkbox"/> Northwest	Pamela	Ansam-Wolfe	\$0	↓
	David	Campbell	\$3,587,378	↑
	Tete	Mensa-Annan	\$1,931,620	↗
	Sales for: Northwest		\$5,518,999	↗
<input checked="" type="checkbox"/> Southeast	Tsvi	Reiter	\$2,811,013	↗
	Sales for: Southeast		\$2,811,013	↓
<input checked="" type="checkbox"/> Southwest	Sales for: Southwest		\$8,219,201	↑

For more information about adding, changing, and deleting indicators, see [Add or Delete an Indicator \(Report Builder and SSRS\)](#).

When you first add an indicator to a report, it is configured to use default values. You can then change the values so the indicator depicts data the way you want. You can change the appearance of the indicator icons, the way the indicator chooses which icon to use, and change the icons used by an indicator set. For more information, see [Change Indicator Icons and Indicator Sets \(Report Builder and SSRS\)](#).

By default, indicators are configured to use percentages as the measurement unit and automatically detect the minimum and maximum values in the data. Each icon in the indicator set has a percentage range. The number of percentage ranges depends on the number of icons in the icon set, but the ranges are the same size and sequential. For example, if the icon set has five icons, there are five percentage ranges, each 20 percent in size. The first one starts at 0 and ends at 20, the second starts at 20 and ends at 40, and so forth. The indicator on the report uses the icon from the indicator set that has a percentage range within which the indicator data value falls. You can change the percentage range for each icon in the set. The minimum and maximum values can be set explicitly by providing a value or an expression. You can change the measurement unit to be a numeric value instead. In this situation, you do not specify minimum or maximum for the data. Instead, you provide only the start and end values for each icon that the indicator uses. For more information, see [Set and Configure Measurement Units \(Report Builder and SSRS\)](#).

Indicators convey data values by synchronizing across indicator data values within a specified scope. By default, the scope is the parent container of the indicator such as the table or matrix that contains the indicator. You can change the synchronization of the indicator by choosing a different scope, depending on the layout of your report. The indicator can omit synchronization. For more information, see [Set Synchronization Scope \(Report Builder and SSRS\)](#).

SSRS).

For general information about understanding and setting scope within reports, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Indicators use only a single value. If you have to show multiple data values, use a sparkline or data bar instead of an indicator. They can represent multiple data values but are also simple, easy to understand at small sizes, and work well in tables and matrices. For more information, see [Sparklines and Data Bars \(Report Builder and SSRS\)](#).

Sizing Indicators to Maximize Visual Impact

In addition to color, direction, and shape you can use size to maximize the visual impact of indicators. Imagine a report that uses indicators to show customer satisfaction with different types of bicycles. The icon that the indicator uses can be configured to be different sizes depending on customer satisfaction. The greater the satisfaction, the larger the icon that appears in the report. The following picture shows a report of bicycle sales, and the sizes of the icon reflect the sales amount.

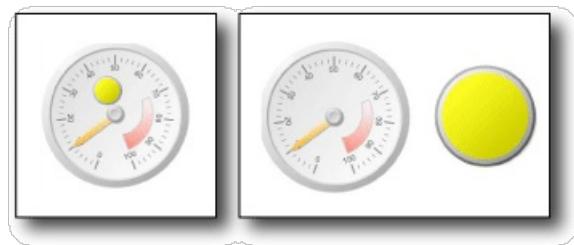
You use expressions to dynamically set the size of the stars based on values of field used by the indicator. For more information, see [Specify the Size of an Indicator Using an Expression \(Report Builder and SSRS\)](#).

To learn more about writing and using expressions, see [Expressions \(Report Builder and SSRS\)](#).

Including Indicators and Gauges in Gauge Panels

Indicators are always positioned inside a gauge panel. The gauge panel is a top-level container that can include one or more gauges and state indicators. The gauge panel can contain child or adjacent gauges or indicators. If you use an indicator as a child to a gauge, you can further visualize the data by showing the state of the data value displayed in the gauge. For example, an indicator within a gauge can display a green circle to tell you that the value in the gauge points to the upper 33 percentage of the value range. Using a gauge and indicator side by side, you can represent the data in different ways. In either case, the indicator and gauge can use the same or different data fields.

The following diagram shows an indicator side by side and within a gauge.



For more information, see [Include Indicators and Gauges in a Gauge Panel \(Report Builder and SSRS\)](#).

For more information about using gauges, see [Gauges \(Report Builder and SSRS\)](#).

Sequence of Indicator States

The sequence of the indicator states in the **Value and States** tab of the **Indicator Properties** dialog box affect which indicator icon displays for a data value when the start and end values of indicator states overlap.

This might happen whether you use the percentage or numeric state measurement unit. It is more likely to occur when you use the numeric measurement unit because you provide specific values for this measurement. It is also more likely to occur when you round report data values because this tends to make values less discrete.

The following scenarios describe how the data visualization of data is affected when you change the sequence of the three states in the **3 Arrows (Colored)** directional indicator. By default the sequence is:

1. Red down arrow
2. Yellow horizontal arrow
3. Green up arrow

The following scenarios show for four different state sequences and their value ranges and how the sequences affect data visualization.

In these scenarios, the **3 Arrows (Colored)** indicator uses numeric state measurements.

STATE SEQUENCE	START VALUE	END VALUE
Red	0	3500
Yellow	3500	5000
Green	5000	10000

The red down arrow depicts the value 3500 and the yellow horizontal arrow 5000.

STATE SEQUENCE	START VALUE	END VALUE
Green	5000	10000
Yellow	3500	5000
Red	0	3500

The yellow horizontal arrow depicts the value 3500 and the green up arrow 5000.

STATE SEQUENCE	START VALUE	END VALUE
Green	5000	10000
Red	0	3500
Yellow	3500	5000

The red down arrow depicts the value 3500 and the green up arrow 5000.

STATE SEQUENCE	START VALUE	END VALUE
Yellow	3500	5000
Red	0	3500
Green	5000	10000

The yellow down arrow now depicts both the value 3500 and 5000.

In summary, evaluation starts and the top of the indicator state list and the report displays the indicator icon associated with the first one indicator state that has a value range that the data fits into. By changing the sequence of the indicator states you can therefore affect the visualization of data values.

How-to Topics

This section lists procedures that show you how to add, change, and delete indicators; how to configure and customize indicators; and how to use indicators in gauges.

- [Add or Delete an Indicator \(Report Builder and SSRS\)](#)
- [Change Indicator Icons and Indicator Sets \(Report Builder and SSRS\)](#)
- [Set and Configure Measurement Units \(Report Builder and SSRS\)](#)
- [Set Synchronization Scope \(Report Builder and SSRS\)](#)
- [Specify the Size of an Indicator Using an Expression \(Report Builder and SSRS\)](#)
- [Include Indicators and Gauges in a Gauge Panel \(Report Builder and SSRS\)](#)

See Also

[Gauges \(Report Builder and SSRS\)](#)

[Sparklines and Data Bars \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

Add or Delete an Indicator (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

In a Reporting Services paginated report, indicators are minimal gauges that convey the state of a single data value at a glance. For more information about them, see [Indicators \(Report Builder and SSRS\)](#).

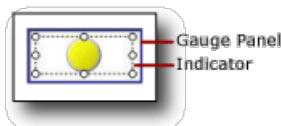
Indicators are commonly placed in cells in a table or matrix, but you can also use indicators by themselves, side-by-side with gauges, or embedded in gauges.

When you first add an indicator, it is by default configured to use percentages as measurement units. The percentage ranges are evenly distributed across members of the indicator set, and the scope of values shown by the indicator is the parent of the indicator such as a table or matrix.

You can update the values and states of indicators. For more information, see the following topics:

- [Change Indicator Icons and Indicator Sets \(Report Builder and SSRS\)](#)
- [Set and Configure Measurement Units \(Report Builder and SSRS\)](#)
- [Set Synchronization Scope \(Report Builder and SSRS\)](#)

Because an indicator is positioned inside the gauge panel, you need to select the indicator instead of the panel when you want to configure the indicator by using the **Indicators Properties** dialog box or the **Properties** pane. The following picture shows a selected indicator in its gauge panel.



NOTE

Depending on column width and the length of data values, the text in table or matrix cells might wrap and display text on multiple lines. When this occurs, the indicator icon might be stretched and change shape. This can make the indicator icon less readable. Place the indicator inside a rectangle to ensure that the icon is never stretched.

To add an indicator to a table or matrix

1. Open an existing report or create a new report that contains a table and matrix with the data you want to display. For more information, see [Tables \(Report Builder and SSRS\)](#) or [Matrices](#).
2. Insert a column in your table or matrix. For more information, see [Insert or Delete a Column \(Report Builder and SSRS\)](#).
3. Optionally, on the **Insert** tab, click **Rectangle**, and then click a cell in the new column.
4. On the **Insert** tab, click **Indicator**, and then click a cell in the new column.

If you added a rectangle to a cell, click that cell.

5. In the **Select Indicator Style** dialog box, in the left pane, click the indicator type you want, and then click the indicator set.

6. Click **OK**.
7. Click the indicator. The **Gauge Data** pane opens.
8. In the **Values** area, in the **(Unspecified)** drop-down list, click the field whose values you want to display as an indicator.

The indicator is configured to use default values. By default, indicators are configured to use percentages as measurement units and the percentage ranges are evenly distributed across the members of the indicator and the value that the indicator conveys uses the scope of the nearest group.

To delete an indicator to a table or matrix

1. Right-click the indicator to delete and click **Delete**.

NOTE

An indicator might be positioned inside a gauge panel that contains other indicators or gauges. If the gauge panels contain multiple items, be sure to click the indicator to delete it, not the gauge panel. If you click and then delete the gauge panel, the gauge panels and all the items in it are deleted.

2. Click **Delete**.

See Also

[Indicators \(Report Builder and SSRS\)](#)

Change Indicator Icons and Indicator Sets (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

The preconfigured indicators sets that Reporting Services provides for paginated reports might not always depict your data effectively and work well in the delivered report. This topic provides procedures to change the appearance of indicator icons and change the indicator sets to include different, more, or fewer indicator icons.

Options such as colors can be set by using expressions. For more information, see [Expressions \(Report Builder and SSRS\)](#).

To change the color of an indicator icon

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. Click the down arrow in the **Color** column next to the icon that you want to change and click the color to use, **No Color**, or **More colors**.

Optionally, click the **Expression** (fx) button to edit an expression that sets the value of the **Color** option.

If you clicked **More Colors**, the **Select Color** dialog box opens, where you can choose from a wide array of colors. For more information about its options, see [Select Color Dialog Box \(Report Builder and SSRS\)](#). Click **OK** to close the **Select Color** dialog box.

4. Click **OK**.

To change the icon

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. Click the down arrow next to the icon that you want to change and select a different icon.

Optionally, click the **Expression** (fx) button to edit an expression that sets the value of the **Icon** option.

4. Click **OK**.

To use a custom image as an indicator icon

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. Click the down arrow next to the icon that you want to change and select **Image**.
4. In the **Select the image source** list, click **External**, **Embedded**, or **Database**.
5. Depending on the source of the image, do the one of the following:
 - To use an image that is stored externally to the report, click **Browse** and locate the image. The report will include a reference to the image.

- To use an image that is embedded in the report, click **Import** and locate the image. The image will be added to the report definition and saved with it.
- To use an image that is in a database, in the **Use this field** list, select the field from the list and then in the **Use this MIME type** list, select the MIME type of the image.

6. Click **OK**.

To add an icon to the indicator set

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. Click **Add**. An indicator is added, using the default icon and the **No Color** option.

Configure the indicator to use the icon and color you want. Procedures earlier in this topic describe the steps to do this.

4. Click **OK**.

To delete an icon to the indicator set

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. Select the icon to delete, and click **Delete**.
4. Click **OK**.

See Also

[Indicators \(Report Builder and SSRS\)](#)

Set and Configure Measurement Units (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

In a Reporting Services paginated report, indicators use one of two measurement units: percentage or numeric.

By default, indicators are configured to use percentages as the measurement unit. This means that the indicator values assigned to each icon in the indicator set are determined by a percentage range. The percentage ranges are evenly divided across the icons in the indicator set. Each icon represents an indicator state. You can change the percentages for each icon in the indicator set by specifying different start and end percentages. Indicators also automatically detect the minimum and maximum values in the data.

You can change the measurement unit to be a numeric value. In this case, you do not specify minimum or maximum for the data; instead, you provide only the start and end values for each icon that the indicator uses.

Options such as measurement units can be set by using expressions. For more information, see [Expressions \(Report Builder and SSRS\)](#).

To use the numeric state measurement unit

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. In the **States Measurement Unit** list, click **Numeric**.

Optionally, click the **Expression (fx)** button to edit an expression that sets the value of the option.

4. For each icon in the indicator set, update the values in the **Start** and **End** text boxes.

Optionally, click the **Expression (fx)** button to edit an expression that sets the values of the **Start** and **End** options.

NOTE

The values in the **Start** and **End** text boxes must be numeric.

5. Click **OK**.

To use the percentage measurement unit

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. In the **States Measurement Unit** list, click **Percentage**.

Optionally, click the **Expression (fx)** button to edit an expression that sets the value of the option.

4. Optionally, change the **Minimum** and **Maximum** options to use specific values instead of automatically detecting the minimum and maximum values of the data that the indicator uses. The value of **Minimum** must be smaller than the value of **Maximum**.

NOTE

If you explicitly set minimum and maximum values, that value range is used by the indicator regardless of the actual the minimum and maximum values in the data. This means that values lower than the minimum and higher than the maximum are excluded from the evaluation that determine what indicator icon to show in the report.

Optionally, click the **Expression (fx)** button to edit an expression that sets the values of the option.

5. For each icon in the indicator set, update the values in the **Start** and **End** text boxes.

Optionally, click the **Expression (fx)** button to edit an expression that sets the values of the **Start** and **End** options.

NOTE

The values in the **Start** and **End** text boxes must be numeric.

6. Click **OK**.

See Also

[Indicators \(Report Builder and SSRS\)](#)

Set Synchronization Scope (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a Reporting Services paginated report, indicators convey data values by synchronizing across the range of indicator values within a specified scope.

By default, the scope is the parent container of the indicator such as the table or matrix that contains the indicator. You can change the synchronization of the indicator depending on the layout of your report. For example, if a data region such as a table has a row group, you can specify the group as the indicator scope. The indicator can also omit synchronization.

Options such as measurement units can be set by using expressions. For more information, see [Expressions \(Report Builder and SSRS\)](#).

For general information about understanding and setting scope within reports, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

To change the synchronization scope of an indicator

1. Right-click the indicator you want to change and click **Indicator Properties**.
2. Click **Values and States** in the left pane.
3. In the **Synchronization scope** list, click the scope that you want to apply.

The **(None)** option, which removes synchronization scope from the indicator, is always available. Other options depend on the layout of your report.

Optionally, click the **Expression (fx)** button to edit an expression that sets the scope.

4. Click **OK**.

See Also

[Indicators \(Report Builder and SSRS\)](#)

Specify the Size of an Indicator Using an Expression (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In addition to color, direction, and shape, you can use size to maximize the visual impact of indicators.

An indicator has a collection of indicator states named `IndicatorStates`. The `IndicatorStates` collection typically have multiple states. Each state is a member of the collection and is represented by an icon. Together the states constitute the `IndicatorStates` collection.

To dynamically configure the sizes of icons, you set properties of members of the `IndicatorStates` collection in the Properties pane of Report Builder. If the **Properties** pane is not visible, click the **View** tab and select **Properties**.

NOTE

In SQL Server Data Tools (SSDT), you use the **Properties** window to set the member properties. If the **Properties** window is not open, press the F4 key.

The **Properties** pane provides access to the properties of the `IndicatorStates` collection of an indicator. You configure the icons to be different sizes by setting the `ScaleFactor` property of the `IndicatorStates` collection members using an expression. For more information, see [Expressions \(Report Builder and SSRS\)](#).

The expression used in this procedure was also used to generate the report with different sizes of indicators, shown in [Indicators \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To specify the indicator icon size using an expression

1. Click the indicator you want to change.
2. In the Properties pane, locate the `IndicatorStates` property.

If the Properties pane is organized by category, you will find `IndicatorStates` in the **States** category.

3. Click the ellipsis (...) button next to `IndicatorStates`. The **IndicatorState Collection Editor** dialog box opens.

Select all members of the collection.

4. In the **Multi-Select Properties** list, click the down arrow next to `ScaleFactor` and then click **Expression**.
5. In the **Expression** dialog box write the expression.

The following sample expression makes the icon a different size based on the value of the `SalesYTD` field.

```
=IIF(Fields!SalesYTD.value = 0,0,Fields!SalesYTD.value/Max(Fields!SalesYTD.value,"Indicator"))
```

For more information, see [Expression Examples \(Report Builder and SSRS\)](#).

6. Click **OK**.

7. Click **OK**.

See Also

[Indicators \(Report Builder and SSRS\)](#)

Include Indicators and Gauges in a Gauge Panel (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

The gauge panel is the top-level container that holds one or more gauges and indicators. Indicators can be embedded in gauges or placed next to them in the gauge panel.

If the indicator and gauge are adjacent in the gauge panel and show data from different fields, you might want to add labels to make it clear what data the gauge and indicator convey.

Gauge and indicator options can be set by using expressions. For more information, see [Expressions \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To embed an indicator in a gauge

1. Open an existing report or create a new report that contains a table and matrix with the data you want to display.
2. Insert a column in your table or matrix. For more information, see [Insert or Delete a Column \(Report Builder and SSRS\)](#).
3. On the **Insert** tab, in the **Data Regions** group, click **Gauge**, and then, and then click a cell in the new column. The **Select Gauge Type** dialog box appears.
4. Click **Radial**. The first radial gauge is selected.
5. Click **OK**.
6. Click the gauge. The **Gauge Data** pane opens.
7. In the **Values** area, in the **(Unspecified)** list box, click the field whose values you want to display in the gauge. Alternatively, drag the field to use from the report dataset.
8. Right-click the gauge, click **Add Indicator**, and then click **Child**. The **Select Indicator Style** dialog box opens.
9. In the **Select Indicator Style** dialog box, in the left pane, click the indicator type you want, and then click the indicator set.
10. Click **OK**.
11. Click the indicator. The **Gauge Data** pane opens.
12. In the **Values** area, in the **(Unspecified)** list box, click the field whose values you want to display as an indicator. Alternatively, drag the field to use from the report dataset.

The field can be the same or a different field from the one you use in the gauge.
13. Click **OK**.

To show an indicator and gauge side by side

1. Open an existing report or create a new report that contains a table and matrix with the data you want to display.
2. Insert a column in your table or matrix. For more information, see [Insert or Delete a Column \(Report Builder and SSRS\)](#).
3. On the **Insert** tab, in the **Data Regions** group, click **Gauge**, and then click a cell in the column you inserted. The **Select Gauge Type** dialog appears.
4. Click **Radial**. The first radial gauge is selected.
5. Click **OK**.
6. Click the gauge. The **Gauge Data** pane opens.
7. In the **Values** area, in the **(Unspecified)** list box, click the field whose values you want to display in the gauge. Alternatively, drag the field to use from the report dataset.
8. Right-click the gauge, click **Add Indicator**, and then click **Adjacent**. The **Select Indicator Style** dialog box opens.
9. In the **Select Indicator Style** dialog box, in the left pane, click the indicator type you want, and then click the indicator set.
10. Click **OK**.
11. Click the indicator. The **Gauge Data** pane opens.
12. In the **Values** area, in the **(Unspecified)** list box, click the field whose values you want to display as an indicator. Alternatively, drag the field to use from the report dataset.

The field can be the same or a different field from the one you use in the gauge.
13. Click **OK**.
14. Right-click the gauge panel and click **Add Label**. A label is added to the gauge panel. Repeat one more time.

The gauge panel has two labels.
15. Drag each label to a location near the gauge or indicator.
16. Right-click the label near the gauge, click **Label Properties**, and type the text you want in the **Text** box.
17. Right-click the label near the indicator, click **Label Properties**, and type the text you want in the **Text** box.
18. Click **OK**.

See Also

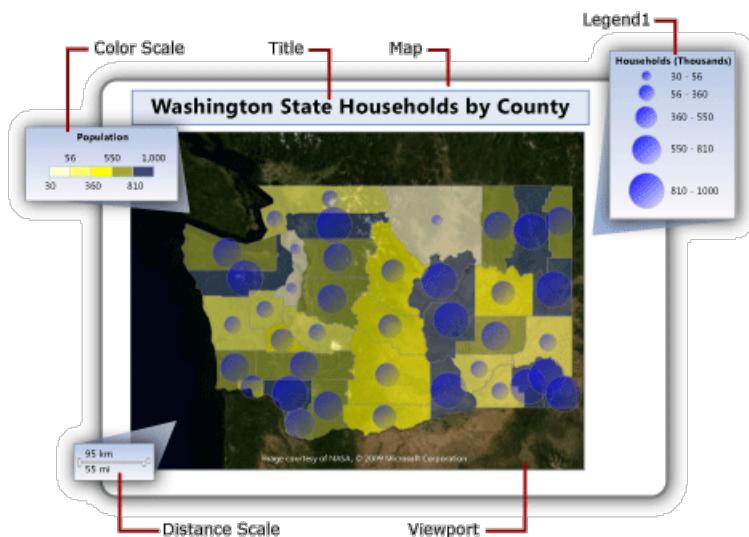
[Indicators \(Report Builder and SSRS\)](#)

Maps (Report Builder and SSRS)

3/24/2017 • 17 min to read • [Edit Online](#)

To visualize business data against a geographical background, you can add a map to your Reporting Services paginated report. The type of map that you select depends on what information that you want to communicate in your report. You can add a map that displays locations only, or a bubble map that varies bubble size based on number of households for an area, or a marker map that varies marker style based on the most profitable product for each store, or a line map that displays routes between stores.

A map contains a title, a viewport that specifies the center point and scale, an optional Bing map tile background for the viewport, one or more layers that display spatial data, and a variety of legends that help users interpret the data visualizations. The following illustration shows the basic parts of a map.



To start to use a map immediately, see [Tutorial: Map Report \(Report Builder\)](#) or [Report Samples \(Report Builder and SSRS\)](#).

NOTE

You can save maps separate from a report as report parts. Read more about [Report Parts](#).

Adding a Map to Your Report

To add a map to your report, here is a list of the general steps to follow:

- Determine which analytical data that you want to display and what types of spatial data that you need. For example, to display relative annual store sales on a bubble map, you need store name and store sales for analytical data and store name and store location as latitude and longitude for spatial data.
- Decide on the style of map you want. Basic maps display locations only. Bubble maps vary bubble size based on a single analytical value. Analytical color maps vary map elements based on ranges of analytical data. The style that you select depends both on the data that you want to visualize and the type of spatial data that you use.
- Collect the information that you must have to specify spatial data sources, spatial data, analytical data sources, and analytical data. This includes connection strings to spatial data sources, specifying the type of spatial data that you need, and making sure that your report data includes match fields that associate the

spatial data and analytical data.

- Run the Map wizard to add a map to your report. This adds the first map layer to the map. Run the Map Layer wizard to create additional layers or modify existing layers. The wizards provide an easy way to get started. For more information, see [Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#).
- After you preview the map in your report, you will probably want to adjust the map view, change the way your data varies the display of each layer, provide legends to help your users interpret the data, and adjust the resolution to provide a good viewing experience for your users.

For more information, see [Plan a Map Report \(Report Builder and SSRS\)](#).

Adding Data to a Map

A map uses two types of data: spatial data and analytical data. Spatial data defines the appearance of the map whereas analytical data provides the values that are associated with the map. For example, spatial data defines the locations of cities in an area whereas analytical data provides the population for each city.

A map must have spatial data; analytical data is optional. For example, you can add a map that displays just store locations in a city.

To visualize data on a map, the analytical data and the spatial data must have a relationship. When the spatial data and the analytical data come from the same source, the relationship is known. When the spatial data and the analytical data come from different sources, you must specify match fields to relate them.

Spatial Data

Spatial data consists of sets of coordinates. Spatial data from a data source can be a single point, multiple points, a single line, multiple lines, or a set of polygons. Each set of coordinates defines a *map element*, for example, a polygon that represents the outline of a county, a line that represents a road, or a point that represents the location of a city.

Spatial data is based on one of the following coordinate systems:

- **Geographic** Specifies geodesic coordinates on a spherical surface by using longitude and latitude. When spatial data is geographic, a projection must be specified. A projection is a set of rules that specifies how to draw objects that have spherical coordinates onto a planar surface. Only geographic data with the same projection can be compared or combined.
- **Planar** Specifies geometric coordinates on a planar surface by using X and Y.

Each map layer displays one type of spatial data: polygons, lines, or points. To display multiple types of spatial data, add multiple layers to the map. You can also add a layer of Microsoft Bing map tiles. The tile layer does not depend on spatial data. The tile layer displays image tiles that correspond to the coordinates of the map viewport.

Sources of Spatial Data

The following sources of spatial data are supported:

- **Map Gallery reports.** Spatial data is embedded in reports located in the map gallery. By default, the Map Gallery is installed in <drive>:\Program Files\Microsoft SQL Server\Report Builder \MapGallery.

NOTE

This Reporting Services mapping feature uses data from TIGER/Line Shapefiles provided courtesy of the U.S. Census Bureau (<http://www.census.gov/>). TIGER/Line Shapefiles are an extract of selected geographic and cartographic information from the Census MAF/TIGER database. TIGER/Line Shapefiles are available without charge from the U.S. Census Bureau. To obtain more information about the TIGER/Line Shapefiles go to <http://www.census.gov/geo/www/tiger>. The boundary information in the TIGER/Line Shapefiles are for statistical data collection and tabulation purposes only; their depiction and designation for statistical purposes does not constitute a determination of jurisdictional authority or rights of ownership or entitlement and they are not legal land descriptions. Census TIGER and TIGER/Line are registered trademarks of the U.S. Bureau of the Census.

- **ESRI Shapefiles.** ESRI Shapefiles contain data that complies with the Environmental Systems Research Institute, Inc. (ESRI) Shapefile spatial data format. ESRI Shapefiles refer to a set of files. Data in the .shp file specifies the geographical or geometrical shapes. Data in the .dbf file provides attributes for the shapes. To view a map in design view or to run a map from the report server, both files must be in the same folder. When you add spatial data from a .shp file on your local file system, the spatial data is embedded in your report. To retrieve spatial data dynamically at run time, upload the Shapefiles to your report server, and then specify them as the source for spatial data. For more information, see [Finding ESRI Shapefiles for a Map](#).
- **SQL Server spatial data stored in a database.** You can use a query that specifies **SQLGeometry** or **SQLGeography** data types from a SQL Server relational database. For more information, see [Spatial Data Types Overview in SQL Server Books Online](#).

In the result set that you see in the query designer, each row of spatial data is treated as a unit and stored in a single map element. For example, if there are multiple points that are defined in one row in the result set, display properties apply to all points in that map element.

- **Custom locations that you create.** You can manually add locations as embedded points to an embedded point layer. For more information, see [Add Custom Locations to a Map \(Report Builder and SSRS\)](#).

Spatial Data in Design View

In Design view, the report processor displays sample spatial data to help you design the map layer. The data that you see depends on the availability of the spatial data:

- **Embedded data.** The sample data is retrieved from map elements embedded in map layers in your report.
- **Link to ESRI Shapefile .** If the ESRI Shapefile (.shp) and the support file (.dbf) are available, the sample data is loaded from the Shapefile. Otherwise, the report processor generates sample data and displays the message **No spatial data available**.
- **SQL Server spatial data.** If the data source is available and the credentials are valid, the sample data is loaded from the spatial data in the database. Otherwise, the report processor generates sample data and displays the message **No spatial data available**.

Embedding Spatial Data in the Report Definition

Unlike analytical data, you have the option to embed spatial data for a map layer in the report definition. When you embed spatial data, you embed map elements that are used in the map layer.

Embedded elements increase the size of the report definition but ensure that the spatial data is always available when the report runs, either in preview or on the report server. More data means more storage and longer processing times. It is always a best practice to limit spatial data, in addition to other report data, to just the information that is needed for your report.

Controlling Map Resolution at Run Time

When you change the resolution for spatial data, you are specifying how detailed you want the lines drawn on a map. For example, for areas, do you need granularity down to a hundred meters of surface area on the earth, or is one mile enough detail?

If the spatial data is embedded in the report, the resolution that you use affects the number of map elements in the report definition. A higher resolution increases the number of elements that are required to draw borders at that resolution. If the spatial data is not embedded in the report, the report server calculates the lines that are required to draw the borders at that resolution every time you view the report. To design a report that balances display resolution and acceptable report rendering time, simplify the map resolution to the level of detail that you need in your report to visualize your analytical data.

Analytical Data

Analytical data is the data that you want to visualize on the map, for example, population for a city or sales total for a store. Analytical data can come from one of the following sources:

- **Dataset field.** A field from a dataset in the Report Data pane.
- **Spatial data source field.** A field from the spatial data source that is included with the spatial data. For example, an ESRI Shapefile frequently includes both spatial and analytical data. Field names from the spatial data source begin with # and appear in the drop-down list of fields when you are specifying the data field for rules for a layer.
- **Embedded data for a map element.** After you embed polygons, lines, or points in a report, you can override the data fields for individual map elements and set custom values.

When you specify rules for a layer and select the analytical data field, if the data type is numeric, the report processor automatically uses the default function Sum to calculate aggregate values for the map element. If the field is not numeric, no aggregate function is specified, and the implicit aggregate function First is used. To change the default expression, change the options for the rules for the layer. For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#).

Match Fields

To relate analytical data to map elements on a layer, you must specify *match fields*. Match fields are used to build a relationship between map elements and analytical data. You can use one or more fields to match on as long as they specify a unique analytical value for each spatial location.

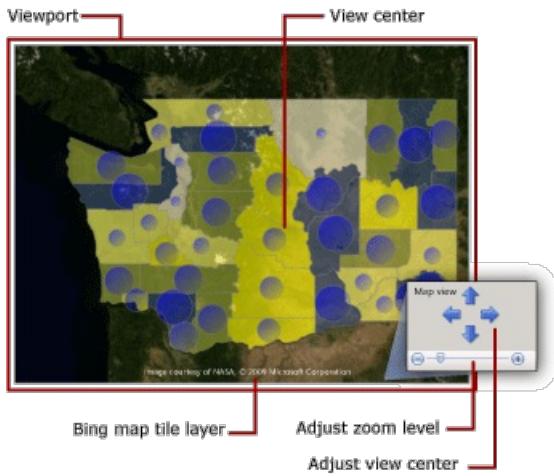
For example, for a bubble map that varies bubble size by city population, the following data is needed:

- From the spatial data source:
 - **SpatialData.** A field that has spatial data that specifies the latitude and longitude of the city.
 - **Name.** A field that has the name of the city.
 - **Area.** A field that has the name of the region.
- From the analytical data source:
 - **Population.** A field that has the city population.
 - **City.** A field that has the name of the city.
 - **Area.** A field that has the name of the territory, state, or region.

In this example, the name of the city alone is not enough to uniquely identify the population. For example, there are many cities named Albany in the United States. To name a specific city, you must specify the area in addition to the city name.

Understanding the Map Viewport

After you specify map data for a report, you can limit the display area of the map by specifying a map *viewport*. By default, the viewport is the same area as the whole map. To crop the map, you can specify the center, zoom level, and maximum and minimum coordinates that define the area that you want to include in your report. To improve the display of the map in the report, you can move the legends, distance scale, and color scale outside the viewport. The following figure shows a viewport:



Adding a Bing Map Tiles Layer

You can add a layer for Bing map tiles that provides a geographic background for the current map view as defined by the viewport. To add a tile layer, you must specify the coordinate system **geographic** and the projection type **Mercator**. Tiles that match the viewport center and zoom level that you select are automatically retrieved from Bing Maps Web Services.

You can customize the layer by specifying the following options:

- Tile type. The following styles are supported:
 - **Road**. Displays a road map style that has a white background, roads, and label text.
 - **Aerial**. Displays an aerial image style without text.
 - **Hybrid**. Displays a combination of the **Road** and **Aerial** styles.
- The language for the display text on the tiles.
- Whether to use a secure connection to retrieve the tiles from the Bing Maps Web service.

For step-by-step instructions, see [Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#).

For more information about tiles, see [Bing Maps Tile System](#). For more information about the use of Bing map tiles in your report, see [Additional Terms of Use](#).

Understanding Map Layers and Map Elements

A map can have multiple layers. There are three types of layers. Each layer displays one type of spatial data:

- **Polygon Layer**. Displays outlines of areas or markers for the polygon center point, which is automatically calculated for each polygon.
- **Line Layer**. Displays lines for paths or routes.
- **Point Layer**. Displays markers for point locations.

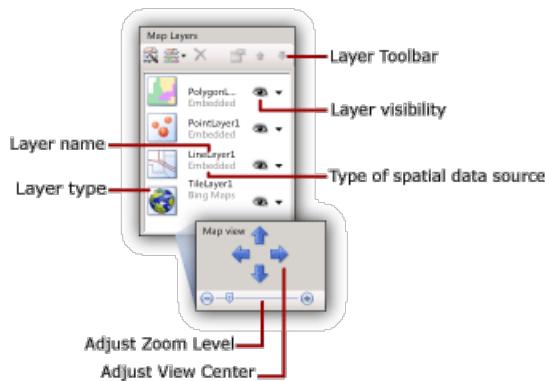
When you specify the source of spatial data for a layer, the wizard checks the spatial data field and sets the

layer type based on its type. A map element is added to the layer for each value from the data source.

For example, to display delivery routes from a central warehouse to your stores, you might add two layers: a point layer with pushpin markers to display store locations and a line layer to display delivery routes to each store from the warehouse. The point layer needs Point spatial data that specifies store locations and the line layer needs Line spatial data that specifies the delivery routes.

The fourth type of layer is a tile layer. A tile layer adds a background of Bing map tiles that corresponds to the map viewport center and zoom level.

To work with layers, select a map on the report design surface to display the Map pane. The Map pane displays the list of layers that are defined for the map. Use this pane to select a layer to change the options, to change the drawing order of layers, to add a layer or run the Map Layer wizard, to hide or show a layer, and to change the view center and zoom level for the map viewport. The following figure shows a viewport:



For more information about map layers, see [Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#).

Varying Display Properties for Points, Lines, and Polygons

Display options for a map elements can be set at the layer level, by using rules for the layer, or on individual elements. For example, you can set display properties for all points on a layer, or you can set rules that control the display properties for all points on a layer whether or not they are embedded, or you can override display property settings for specific embedded points.

When you view a report, the display values that you see are controlled by this hierarchy, listed in ascending order. The higher numbers take precedence:

1. **Layer properties.** Properties that apply to the whole layer. For example, use layer properties to set the source of analytical data or the visibility for the whole layer.
2. **Polygon, Line, Point properties and Embedded Polygon, Line, Point properties.** Properties that apply to all map elements on a layer, whether the elements are from dynamic spatial data or embedded spatial data. For example, use polygon center point properties to set the fill color for bubbles to a gradient that fills bubble areas from dark blue to light blue and from top to bottom.
3. **Color Rules, Size Rules, Width Rules, Marker Type Rules.** Rules apply properties to a layer when the layer has map elements that have a relationship to analytical data. The type of rules vary based on layer type. For example, use point size rules to vary bubble size based on population.
4. **Override for Embedded Polygon, Line, or Point properties.** For embedded map elements, you can select the override option and change any property or data value. Any changes that you make to override rules for individual elements are irreversible. For example, you can highlight a specific store by using a pushpin marker.

For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report](#)

[Builder and SSRS](#)).

In addition to varying the appearance of map elements, you can add interactivity to points, lines, and polygons, or to layers, in the following ways:

- Create tooltips to provide additional details for a map element when the user hovers a pointer over the map.
- Add drillthrough actions to link to other locations in the report, to other reports, or to Web pages.
- Add parameters in expressions that define layer visibility to enable a user to show or hide specific map layers.

For more information, see [Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#).

Understanding Map Legends, Color Scale, and Distance Scale

You can add a variety of legends to your report to help users interpret a map. Maps can include the following items:

- **Legends.** You can create multiple legends. Items that are listed in a legend are generated automatically based on the rules that you specify for map elements on each layer. For each rule, you specify the legend to use to display its related items. In this manner, you can assign items from multiple layers to the same legend or to different legends.
- **Color scale.** You can create one color scale. As an alternative to providing a legend for a color rule, you can display items for a color rule in the color scale. Multiple color rules can apply to the color scale.
- **Distance scale.** You can display one distance scale. The distance scale displays a scale for the current map view in both kilometers and miles.

You can position the legends, color scale, and distance scale in discrete locations inside or outside the viewport. For more information, see [Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#).

Troubleshooting Maps

Map reports use spatial and analytical data from a variety of data sources. Each map layer can use different sources of data. The display properties for each layer follow a specific precedence based on layer properties, rules, map element properties.

If you do not see the result that you want when you view a map report, the root causes can come from a variety of issues. To help you isolate and understand each issue, it helps to work with one layer at a time. Use the Map pane to select a layer and easily toggle its visibility.

For more information about map report issues, see [Troubleshoot Reports: Map Reports \(Report Builder and SSRS\)](#)

How-To Topics

This section lists procedures that show you, step by step, how to work with maps and map layers in your reports.

- [Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#)
- [Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#)
- [Add Custom Locations to a Map \(Report Builder and SSRS\)](#)

In This Section

[Plan a Map Report \(Report Builder and SSRS\)](#)

[Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#)

[Customize the Data and Display of a Map or Map Layer \(Report Builder and SSRS\)](#)

[Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#)

[Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#)

[Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#)

[Add Custom Locations to a Map \(Report Builder and SSRS\)](#)

[Troubleshoot Reports: Map Reports \(Report Builder and SSRS\)](#)

Plan a Map Report (Report Builder and SSRS)

3/24/2017 • 9 min to read • [Edit Online](#)

Good reports present information that leads to actions or insights. To present analytical data such as sales totals or demographics against a geographic background, you can add a map to your Reporting Services paginated report. A map can contain multiple layers, where each layer displays map elements that are defined by a specific type of spatial data: points that represent locations, lines that represent routes, or polygons that represent areas. You can associate your analytical data with map elements on each layer.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Specify the Purpose of the Map

Good report design provides information that help users take actions to address issues. To create a useful, easily understood map display, decide what questions you want the map to help answer. For example, on a map you can visualize the following types of data to identify market opportunities:

- Relative sales for each store.
- Sales categorized by customer demographics, based on customer location relative to store locations.
- Comparative sales or other financial data by sales territory.
- Comparative sales for different discount strategies across multiple stores.

After you identify the purpose of the map display, you must analyze what data you need. Analytical data comes from report datasets. Location data comes from spatial data sources that you must specify.

Specify the Spatial and Analytical Data

You must specify which spatial and analytical data that you need.

Analytical data comes from a report dataset, from sample data included with a map from the map gallery, or from analytical data included with spatial data in an ESRI Shapefile.

Spatial data comes in three forms: points, lines, and polygons.

- **Points.** Points specify locations, for example, a city or an address for a store, restaurant, or convention center. For every location that you want to display on a map, you must provide the spatial data for that location. After you add points to a map, you can display a marker at the point location and vary the marker type, size, and color.
- **Lines.** Lines specify paths or routes, for example, delivery routes or flight paths. After you add a line to a map, you can vary the line color and line width.
- **Polygons.** Polygons specify areas, for example, regions, territories, countries, states, provinces, counties, cities, or areas covered by cities, postal codes, telephone exchanges, or census districts. After you add polygons to a map, in addition to displaying the outline, you can display a marker at the center point of the area of the polygon.

After you identify which spatial data that you need, you must find a source for it.

Find a Source for Spatial Data

To find spatial data to use in your map, you can use the following sources:

- ESRI Shapefiles, including publicly available Shapefiles that you can search for on the Internet.
- Spatial data from SQL Server spatial data sources.
- Maps from reports in the Map Gallery.
- Third-party sites that offer spatial data as ESRI Shapefiles or SQL Server spatial data.
- Bing map tiles, which provide a background for the map view. To display tiles in a map, the report server must be configured to support Bing Maps Web Services.

For more information, see "Where can I get ESRI shapefiles?" in [Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#).

Spatial data can be politically sensitive and possibly copyrighted. Check the terms of use and privacy statements for spatial data sources to understand how you can use spatial data in your report.

After you find the data that you want, you can embed the data in the report definition or retrieve the data dynamically when the report is processed. For more information, see [Balance Report Definition Size and Report Processing Time](#) later in this topic.

Determine the Spatial Data and the Spatial Data Match Fields

To display analytical data on a map and to vary the size, color, or marker type, you must specify fields that relate the spatial data and analytical data.

Spatial data must contain the following fields:

- **Spatial data.** A spatial data field that has the sets of coordinates that define each point, line, or polygon.
- **Match fields.** One or more fields that uniquely identify each spatial data field. For example, for a store location point, you might use the name of the store. If the store name is not unique in the spatial data, you might include the name of the city as well as the store.

The match fields are used to relate the spatial data with the analytical data.

Determine the Analytical Data and the Analytical Data Match Fields

After you identify the spatial data, you must identify the analytical data. Analytical data can come from the following sources:

- An existing report dataset. Fields are specified as simple field expressions, for example, [Sales] or =Fields!Sales.Value.
- Data fields provided by the spatial data source. Fields are specified as keywords that begin with # followed the field name from the source of spatial data.

You must then determine the names of the match fields:

- Match fields. One or more fields that specify the same information as the spatial data match fields to build a relationship between the spatial data and the analytical data. Match fields should match data type as well as formatting.

When you have identified the spatial data source, the spatial data, the analytical data source, the analytical data, and the match fields, you are ready to decide which type of map to add to your report.

Choose a Map Type

When you run the Map wizard, you add a map and the first map layer to your report. The wizard enables you to add one of the following types of maps to your report:

- A basic map that displays locations without associated analytical data.
- A map that has one analytical value associated with each map element, for example, sales total for each store location.
- A map that has more than one analytical value associated with a map element, for example, number of customers and sales total for each store location.

The following table describes each map type. All map types allow you to select a theme that controls palette, border style, and font, and to display labels.

WIZARD ICON	LAYER STYLE	LAYER TYPE	DESCRIPTION AND OPTIONS
	Basic Map	Polygon	A map that displays areas only, for example, sales territories. Options: Vary color by palette or use a single color. A palette is a predefined set of colors. When all colors in a palette have been assigned, shades of colors are assigned.
	Color Analytical Map	Polygon	A map that displays analytical data by varying color, for example, sales data by area.
	Bubble Map	Polygon	A map that displays analytical data by varying bubble size centered on areas, for example, sales data by area. Options: Vary area colors based on a second analytical field and specify color rules.
	Basic Line Map	Line	A map that displays lines only, for example, delivery routes. Options: Vary color by palette or use a single color.
	Analytical Line Map	Line	A map that varies line color and width, for example, number of packages delivered and on-time metrics by route. Options: Vary line width by one analytical field, vary line color by a second analytical field, and specify color rules.

WIZARD ICON	LAYER STYLE	LAYER TYPE	DESCRIPTION AND OPTIONS
	Basic Marker Map	Point	A map that displays a marker at each location, for example, cities. Options: Vary color by palette or use a single color, and change marker style.
	Bubble Marker Map	Point	A map that displays a bubble for each location and varies bubble size by one analytical data field, for example, sales data by city. Options: Vary bubble color by a second analytical field, and specify color rules.
	Analytical Marker Map	Point	A map that displays a marker at each location and varies marker color, size, and type based on analytical data, for example, top selling products, profit range, and discount strategy. Options: Vary marker type by one analytical field, vary marker size by a second analytical field, vary marker color by a third analytical field, and specify color rules.

After you add a map with the Map wizard, you can create additional layers or change options for a layer by using the Layer wizard. For more information about the wizards, see [Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#).

You can customize the display or data options for each layer independently. For more information about customizing a map after you run a wizard, see [Customize the Data and Display of a Map or Map Layer \(Report Builder and SSRS\)](#).

Plan for Legends

To help your users interpret a map, you can add multiple map legends, a color scale, and a distance scale. When you design a map, plan where you want the legends to display. You can specify the following information about each legend:

- **Legend location.** For example, legends can display inside or outside the viewport, and in 12 discrete locations relative to the viewport.
- **Legend styles.** For example, you can specify the font style, border style, separator line, and fill properties.
- **Legend title.** For example, you can specify title text, and independently control whether to display the title for a map legend or the color scale.
- **Map legend layout.** For example, map legends can display as tall tables or wide tables.

The contents of the legend are created automatically during report processing based on rule options that

you set for each layer.

By default, all layers display the results of rules in the first map legend. You can create multiple legends and then, for each rule, assign which legend to use to display the results.

For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#) and [Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#).

Balance Report Definition Size and Report Processing Time

Good report design for maps require that you balance the options that control report performance and report definition size. Map elements that are based on spatial data, or Bing map tiles, can be static and embedded in the report definition or dynamic and created every time the report is processed. You must assess the trade-offs for static or dynamic map data and find the balance that works for your circumstances. Consider the following information to make this decision:

- Embedded map elements can significantly increase the size of the report definition, but reduce the time that is required to view the map in the report. Your report server might have size limits that you need to work with.
- The report definition specifies limits to the number of spatial data points that can be processed and a separate value that specifies the number of map elements that can be included in the report definition.
- Dynamic map elements reduce the report definition size but increase the time that is required to process and render the map.
- When the source of spatial data is located on the your computer, map elements are always embedded in the report definition. This includes spatial data from the Map Gallery or ESRI Shapefiles that have been installed locally.

To use dynamic spatial data, the spatial data source must be on the report server. When reports are designed in SQL Server Data Tools (SSDT), spatial data sources can be added to a project and published to the report server together with the report definition. If you are using Report Builder to design reports, you must upload the spatial data to the report server first, and then, in the wizard or in the layer properties, specify that source of spatial data for the map layer.

See Also

[Customize the Data and Display of a Map or Map Layer \(Report Builder and SSRS\)](#)

[Tutorial: Map Report \(Report Builder\)](#)

[Maps \(Report Builder and SSRS\)](#)

[Troubleshoot Reports: Map Reports \(Report Builder and SSRS\)](#)

Map Wizard and Map Layer Wizard (Report Builder and SSRS)

3/24/2017 • 14 min to read • [Edit Online](#)

In Reporting Services paginated reports, the Map Wizard and Map Layer Wizard automate the task of creating a map, adding a map layer, or changing map layer options on an existing layer.

Before you add a map to a report or a map layer to a map, gather the following information:

- **Spatial data source.** The location or connection to a source that provides spatial data, for example, the name of a SQL Server instance and a database that contains spatial data, or the name of an Environmental Systems Research Institute, Inc. (ESRI) Shapefile.
- **Spatial data.** From the spatial data source, a field that contains sets of coordinates that specify locations.
- **Analytical data.** Analytical data to use to vary the map display, for example, annual store sales.
- **Match fields.** Match fields that define the relationship between spatial data and analytical data, for example, the name of a region and city to uniquely identify each city.

The following sections provide information about options that you specify when in the Map and Map Layer Wizards.

- When you first open Report Builder, click the **Map** wizard icon in the center of the design surface.
- On the **Insert** tab, click **Map**, and then click **Map Wizard**.

To open the Map Layer wizard, do the following action:

- Click the map to display the Map pane and on the toolbar, click the **New layer wizard** button.

Click the title of the wizard page for the corresponding help content. The pages that you see vary depending on your choices for the type of map, the source of spatial data, and the source of analytical data.

1. **Choose a source of spatial data.** Spatial data can come from the map gallery, an Environmental Systems Research Institute, Inc. (ESRI) Shapefile, or from spatial data in a SQL Server relational database.
 - [What is spatial data?](#)
 - [What is the Map gallery?](#)
 - [What is an ESRI Shapefile?](#)
 - [Where can I get ESRI shapefiles?](#)
 - [What is a SQL Server spatial query?](#)
2. **Choose spatial data and map view options.** Set the map view, the map resolution, specify whether to embed spatial data in the report, and specify whether to include a tile background of Microsoft Bing map tiles.
 - [What is the map view or viewport?](#)
 - [What is map resolution or optimization?](#)
 - [What does embedding spatial data do?](#)

- What is a Bing maps tile background?
3. Choose map visualization. Choose the type of map to create.
- What is the difference between a Basic Map, a Bubble Map, and an Analytical Map?
 - Choose map visualization: Polygons
 - Choose map visualization: Lines
 - Choose map visualization: Points
4. Choose a connection to a data source Choose map visualization: Points. Choose a data source connection or create one to an external data source that contains analytical data to display on the map.
5. Design a query. Build a query that specifies the analytical data.
6. Choose the analytical dataset. Specify a data source for the analytical data.
- What is the difference between spatial data and analytical data?
7. Specify the match fields for spatial and analytical data. Build a relationship between the spatial data and analytical data so that the appearance of map elements can vary based on data.
- What is a match field?
8. Choose color theme and data visualization. To specify how to visualize your data against the map background, specify the map theme, the fields to visualize, and what to vary: color, size, and/or marker type.
- What does the theme do?
 - What are the legends and scales in Map Preview for?
 - What are rules?

After you add a map or map layer and preview the report, you can change map and map layer options that you set in the wizards. For more information, see [Customize the Data and Display of a Map or Map Layer \(Report Builder and SSRS\)](#).

For more information about maps, see [Maps \(Report Builder and SSRS\)](#). For step-by-step instructions to add a map to a report, see [Tutorial: Map Report \(Report Builder\)](#).

Choose a source of spatial data

On this page, specify the spatial data source and which spatial data to include. Spatial data can come from the map gallery, an ESRI Shapefile, or a dataset query that specifies SQL Server spatial data from a SQL Server 2008 or later version database.

You can use the same source or a different source of spatial data for each layer, but you must specify the source every time you add a layer. When the spatial data is from the map gallery or an ESRI Shapefile, the spatial data source is not a separate report item. It does not appear in the Report Data pane.

What is spatial data?

Spatial data contains coordinates that define geographic or geometric elements. In a map, spatial data defines *map elements*: polygons that define areas or shapes, lines that define routes or paths, and points that define markers or pushpins. Spatial data is stored in binary format on the data source and is specified as sets of coordinates. For example, a point is an X and Y coordinate (X Y), a line is two sets of coordinates ((X1 Y1), (X2 Y2)), a polygon is four or more sets of coordinates where the first and last set of coordinates are the same ((X1 Y1), (X2 Y2), (X3 Y3), (X1 Y1)).

For more information, see the documentation for the type of spatial data that you use.

What is the map gallery?

The map gallery contains maps from reports that are located in the map gallery folder for the report authoring environment. Maps from the gallery provide a quick start to add a map to your report. The predefined maps in the gallery are provided by a map provider.

NOTE

This Reporting Services mapping feature uses data from TIGER/Line Shapefiles provided courtesy of the U.S. Census Bureau (<http://www.census.gov/>). TIGER/Line Shapefiles are an extract of selected geographic and cartographic information from the Census MAF/TIGER database. TIGER/Line Shapefiles are available without charge from the U.S. Census Bureau. To obtain more information about the TIGER/Line Shapefiles go to <http://www.census.gov/geo/www/tiger>. The boundary information in the TIGER/Line Shapefiles are for statistical data collection and tabulation purposes only; their depiction and designation for statistical purposes does not constitute a determination of jurisdictional authority or rights of ownership or entitlement and they are not legal land descriptions. Census TIGER and TIGER/Line are registered trademarks of the U.S. Bureau of the Census.

To extend the map gallery, you can add or remove reports from the map gallery directory, and add folders to organize the maps. For more information, see [Maps \(Report Builder and SSRS\)](#).

What is an ESRI shapefile?

An ESRI Shapefile is a set of files with data that conforms to the Environmental Systems Research Institute, Inc. (ESRI) shapefile spatial data format. The set of files typically includes the *<filename>.shp* file that contains the spatial data and a support file, *<filename>.dbf*.

When you specify a shape file as a spatial data source and it is located on your local computer, the spatial data is automatically embedded in the report. To use spatial data from an ESRI file dynamically, you must do the following:

In Report Builder, upload both the .shp file and the .dbf file to the same folder on a report server, and then link to the .shp file as the spatial data source.

In Report Designer in SQL Server Data Tools (SSDT), add both the .shp file and the .dbf file to the report project, and then specify the name of the .shp file as the spatial data source.

Where can I get ESRI shapefiles?

ESRI shapefiles are available on the Web. For more information, see [Finding ESRI Shapefiles for a Map](#).

What is a SQL Server spatial query?

A SQL Server spatial query is a dataset query that specifies data that is either a SQLGeometry or a SQLGeography data type from a SQL Server relational database.

NOTE

When you define a data source in the wizard, you will see different query designers in the Design a Query page, depending on what type of data source you are connecting to. For more information, see [Query Designers \(Report Builder\)](#).

When you run the query in the query designer, the result set displays a column with spatial data that appears as text. For example, one row might contain spatial data that is a single point and the next row might contain spatial data that defines a set of points. Each row becomes one map element. You can vary the display of each map element as an indivisible unit.

For more information, see [Spatial Data Types](#).

Choose spatial data and map view options

On this page you can set the following options:

- Set the view center and zoom level for the spatial data that you selected in the previous wizard page. The view that you set applies to the whole map.
- Set the map resolution.
- Specify whether to embed the spatial data in the report.
- For embedded data, specify whether to include all data or just the data in the current view.
- Specify whether to include a Microsoft Bing map tiles background.

What is the map view or viewport?

The map viewport defines the map area to display for all layers in your report.

By default, color scale and distance scale appear inside the viewport, and the map legend appears outside the viewport. You can change these options for the viewport after you complete the wizard.

What is map resolution and optimization?

When you change the resolution for spatial data that represents lines or polygons, you are specifying how detailed you want the map to be drawn. For example, for aerial views of a region, do you need granularity down to a hundred meters of surface area on the earth, or is one mile resolution enough?

When spatial data is embedded in the report, a higher resolution increases the number of elements that are needed to draw details at that resolution. When the spatial data is not embedded in the report, a higher resolution increases the amount of time required by the report processor to calculate the lines for the map at that resolution every time you view the report.

When you lower the resolution, the report processor applies an algorithm to reduce the number of points needed to draw the map elements. The lower the resolution, the less data is required to display the map elements, which can lead to better display performance.

As you adjust the slider, the preview data in the wizard pane updates to give you an indication of the effect. After you add the map to the report, you can adjust this value by changing map viewport options.

What does embedding spatial data do?

When you embed map elements or Bing map tiles in a report, the spatial data is stored in the report definition.

A report with a map can use spatial data or Bing map tiles that are retrieved dynamically either when the report is processed or at design time and then embedded in the report definition. Embedded map elements can significantly increase the size of the report definition, but reduce the time it takes to view the map in the report. Dynamic map elements reduce the report definition size but increase the time it takes to process and view the map.

Good report design requires that you assess the trade-offs for static or dynamic map data and find the balance that works for your circumstances. In general, more data means the report definition and the compiled report requires more storage on the report server and longer processing times. It is always a best practice to crop spatial data, as well as limit other report data, to include just what is needed for the report.

What is a Bing map tile background?

To add a geographic image background to your map, select the Bing map tile background option. The report processor downloads tiles from Bing Maps Web Services for the map area and resolution that you specify on this wizard page. You can specify one of the following tile types:

- **Road.** Display a road map style with white background.

- **Aerial.** Display an aerial view only. No text is displayed in this mode.

- **Hybrid.** Display the combination of **Road** and **Aerial** views.

For more information about tiles, see [Bing Maps Tiles System](#). For more information about the use of Bing map tiles in your report, see [Additional Terms of Use](#).

To see a tile background in Design view, you must have Internet access. To see the tile background in preview from a report on a report server, the report server must be configured to support Bing map tiles.

For more information, see [Troubleshoot Reports: Map Reports \(Report Builder and SSRS\)](#) and [Plan a Map Report](#).

For more information on other ways to customize a tile layer, see [Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#).

Choose map visualization

On this page, choose the type of map or map layer to add to your report. The first time you run the wizard, you are adding the map and the first map layer to the report. A map can contain multiple map layers. Each map layer displays a specific type of spatial data: polygons, lines, or points.

The type of map that you choose depends on the purpose of the map and the data that you have available.

What is the difference among a Basic Map, a Bubble Map, and an Analytical Map?

A **Basic Map** displays locations only. You can vary the colors of the areas on the map by shade, but the color does not represent analytical data values.

A **Bubble Map** conveys the relative value for a single analytical data aggregate as bubble size, for example, store sales. You can create bubble maps for either polygons or points. For polygons, set the polygon center point properties; for points, set the marker properties.

An **Analytical Map** conveys the relative value of one or more analytical data aggregates for each map element. For example, store sales as marker size, profit range for product categories as marker color, and top selling product as marker type.

For more information, see [Plan a Map Report \(Report Builder and SSRS\)](#).

Choose the analytical dataset

On this page, specify where to get the analytical data to display on this map layer.

To display your report data or any analytical data against the map background, you must specify where the data is and how it is related to the spatial data. Your data can come from an existing report dataset, or from a new dataset that you build a query for. Existing analytical data might be included in the ESRI Shapefile that contains the spatial data.

What is the difference between spatial data and analytical data?

Spatial data consists of sets of coordinates that specify points, lines, and polygons. Map elements are based on spatial data.

Analytical data is numeric or categorical data that you want use to vary the appearance of the map. Analytical can come from a report dataset or might be included with spatial data from a map from the map gallery or from an ESRI shape file.

Specify the match fields

On this page, build a relationship between the spatial data and analytical data.

What are match fields?

Match fields enable the report processor to build a relationship between the analytical data and the spatial data. Match fields specify unique values within the analytical data. For example, the store name might not be unique within the data, so you could specify both a city and the store name.

Choose color theme and data visualization

On this page, specify how to visualize your data against the map background, the map theme, the fields to visualize, and what to vary: color, size, and/or marker type.

What does the theme do?

The theme that you choose sets default values for color, border, and font. You can change these options after you complete the wizard.

What are the legends and scales in Map Preview for?

Legends help a user interpret the data that is displayed on a map. A map provides a color range, a distance scale, and a legend.

- **Color range.** The color range displays a color bar with a scale that provides a guide to the data intervals that are determined by the report processor based on the rules that you specify for the layer.
- **Distance scale.** The distance scale provides a guide to the distance units on the map. Distance units are determined automatically based on the map projection and zoom level.
- **Legend.** The legend provides a guide to help interpret the meaning of colors, sizes, and marker types on a map. By default, all the rules for all the layers display data intervals in the first legend. You can customize this legend and add legends after you add the map to the report.

What are rules?

Rules are calculations that the report processor uses to divide analytical data into ranges. You can specify different rules for each layer. The type of rules you can specify depend on the type of spatial data on the layer:

- **Polygons.** You can specify color rules.
- **Center points for polygons.** You can specify color, size, and marker type rules.
- **Lines.** You can specify color and width rules.
- **Points.** You can specify color, size, and marker type rules.

The report processor applies the rules that you set and automatically determines the list of items to display in a legend. By default, the results of all rules for all layers display in the first legend. You can adjust this after you complete the wizard. For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#).

See Also

[Troubleshoot Reports: Map Reports \(Report Builder and SSRS\)](#)

[Plan a Map Report \(Report Builder and SSRS\)](#)

[Maps \(Report Builder and SSRS\)](#)

Customize the Data and Display of a Map or Map Layer (Report Builder and SSRS)

3/24/2017 • 8 min to read • [Edit Online](#)

After you add a map or map layer to a Reporting Services paginated report by using a wizard, you might want to change the way the map looks in the report. You can make improvements by considering the following ideas:

- To help your users understand how to interpret the data display on a map, you can add legends and a color scale, and add labels and tooltips.
- To make the map easier to read, change the center and zoom level, add a distance scale, and display a background grid.
- To help control map drawing time when you run the report, you can adjust the resolution to simplify the map elements.
- You can embed map elements in the report definition, and then change how individual elements appear. For example, you can display the primary office location with a pushpin and other office locations with circles.
- You can add customized regions by specifying your own data group expressions.
- You can add a custom location at a point that you specify on a map layer that has embedded points. You can set the value and display properties for custom points independently from other points on a point layer.
- To provide more detail, you can add links to map elements on each layer that a user can click to open related reports.

For more ideas about improving a report, see [Planning a Report \(Report Builder\)](#).

Display options affect the way a map or the parts of a map appear when you view the report. Some options control the appearance of the map, such as the borders and fonts or the area represented on the map. Other options control the content of each layer, such as bubble sizes, marker types, labels, or tooltips.

A map report item includes the following parts: the map itself, a map viewport, a set of titles, a set of legends (legend, color scale, and distance scale), a set of layers, a set of map elements on each layer (polygons or lines or points). Use the information in the following sections to understand which property dialog box controls the display options for different parts of a map.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Change Options for the Map

On a map report item, you can control the following:

- Add multiple titles.
- Add multiple legends. To change the contents of a legend, you need to create additional legends and then change the rules to specify in which legend to enter the legend items created by each rule.

- Add more layers.
- Hide or show the distance scale or color scale.
- Provide the illusion of depth by specifying a shadow.

To change these options, right-click the map, click **Map**, and change the options.

Change Options for the Viewport

Use the viewport options to change the view of the map that appears in your report.

The source of spatial data might provide more area than you need to display in the report. You can use the viewport to set the center, the zoom level, and to crop the area for the map.

The following options can be set for the viewport:

- Choose the coordinate system and, for a geographic coordinate system, choose the projection.
- Choose the center for the map.
- Crop the view for the map.
- Set the zoom level.
- Resolution and simplification. Choose a balance between drawing time and detailed outlines for lines and polygons.

To change these options, right-click the map viewport, use the [Map Viewport Properties Dialog Box, General](#) page and related pages.

Change Options for the Legends

Legends help users interpret the data on a map.

By default, all rules that you specify for a layer add items to the first legend. In addition, all color rules display values in the color scale.

- To change the display options for the appearance of the legend, including its position relative to the viewport, set options on the legend itself.
- To change the contents or the format of contents for a legend, change the legend options for the corresponding rules for a layer.

For more information, see [Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#).

Change Options for the Layer

To display the layers for a map, click the map to select it. The Map pane appears. To change options for a layer, right-click the layer and use the shortcut menu.

A layer can be one of three types based on the spatial data that is returned by the spatial data source: a polygon layer, a line layer, or a point layer.

The following options can be set for the layer:

- The associated analytical data and match fields. The source of the spatial data is listed on the Map pane under the name of the layer. When the source is embedded, the map elements for the layer are part of the report definition. If the source is not embedded, then the spatial data is retrieved at run time and the report

processor creates the map elements for the layer when the report is processed. To change data options on the layer, use the Analytical Data page in the Map Layer dialog box.

- Layer drawing order. The order that you see the layers listed in the Map pane is the reverse drawing order for the layers. The last layer in the list is drawn first. For example, if you want the points on a point layer to appear on top of the polygons in the polygon layer, the point layer should be first and the polygon layer second.
- Layer visibility, including transparency. To have one layer show through another layer, set the transparency to a value higher than 0. A value of 100% means the layer is not visible. To work with an individual layer, you can easily show or hide each layer independently by using the **Visibility** icon in the Map pane. You can also set zoom level options to specify when to show or hide map elements on the layer based on the zoom level.
- Add a Bing map tile layer for the current viewport center and zoom level. You do not need to specify the geographic coordinates for a tile layer. Tiles are automatically loaded to match the viewport area when the coordinate system is Geographic, the projection is Mercator, the Bing Maps servers are available, and when the report server has been configured to support this feature. For each report, you can specify whether to use a secure connection to retrieve tiles.

For more information about layers, see [Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#).

Change Data Grouping for the Layer

You can customize the way to aggregate spatial data for your own shapes. To set the group properties for a layer, select the layer in the Map pane, and in the Properties pane for the layer, click **Group**, and then click the ellipsis (...) to open the Group properties. In this dialog box, you can specify group expressions, create group variables, and filter data that is used for grouping.

The group expression specifies how analytical data that has a relationship to spatial data is aggregated for each map element on the layer. By default, the group expression is the set of match fields that was specified for the relationship between the spatial data and the analytical data. For example, for a bubble map that displays city locations and population size for a country or region, the match fields include city name [City] and region name [Region] because there can be multiple cities with the same name. The corresponding group expression includes two fields: [City] and [Region].

For more information, see [Map Tips: How To Import Shapefiles Into SQL Server and Aggregate Spatial Data](#).

Change Options for the Map Elements on the Layer

Map elements are the points, lines, or polygons on a layer that are based on the spatial data. For map elements, the following options can be set. These options apply to all map elements on the layer, whether or not they are embedded:

- Labels, label visibility, label offset, and formatting.
- Borders and fill.
- Drillthrough actions.
- Display options.

Display options for map elements follow a precedence order based on the layer, the map element, rules for the map element, and override options for embedded map elements.

To change these options, right-click the map element, use the embedded properties dialog box. For example, for an embedded polygon, use the [Map Embedded Polygon Properties Dialog Box, General page](#)

and related pages.

Understanding Display Option Precedence

When you want to control the display appearance of a point, line, or polygon on a map layer, you must understand where display options can be set and which options have a higher precedence. The following display options are listed from lowest to highest. Higher display options override lower display options in this list:

- Layer options.
- Points, Lines, or Polygons options on each layer. This applies whether the map elements are dynamically retrieved when the report is processed or whether they are embedded in the report definition. For example, you specify a fill color for all elements on a layer.
- Rules. You can set rules to control color, size, width, or marker type for all map elements on a layer. The rules that you can set depend on the type of map element.
 - Color Rules. Apply to markers for points, lines, polygons, and markers for polygon center points.
 - Size Rules. Apply to markers for points and markers for polygon center points.
 - Width Rules. Applies to line widths.
 - Marker Type Rules. Applies to markers for points and markers for polygon center points.
- Override options for individual embedded points, lines, or polygons on a layer. Changes that you make are permanent. To revert these changes, you must reload the data for the layer.

For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#).

See Also

[Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#)

[Maps \(Report Builder and SSRS\)](#)

Vary Polygon, Line, and Point Display by Rules and Analytical Data

3/29/2017 • 7 min to read • [Edit Online](#)

The display options for polygons, lines, and points on a map layer are controlled by setting options for the layer, by setting rules for the map elements on the layer, or by overriding options for specific embedded map elements on a layer.

Display options apply in a specific precedence, listed from lowest to highest precedence:

1. Options set on a polygon layer, a 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.
2. Options 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. A data visualization option requires you to specify a data field to base display variations on. You must have set the match fields for the analytical and spatial data before you can apply data visualization rules. For more information, see [Maps \(Report Builder and SSRS\)](#).
3. Options that you set for selected embedded map elements. Note that, when you override the layer options, the changes that you make to the report definition are permanent. You can change the data field values as well as override display options to customize the way specific polygons, lines, and points appear on a layer.

In addition to controlling the display of map elements on a layer, you can control the layer transparency to allow layers that are drawn earlier to show through layers that are drawn later. For more information about changing options that affect the map or the entire map layer, see [Customize the Data and Display of a Map or Map Layer \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Rules

You can set four types of rules that enable the report processor to automatically adjust display properties for map elements on a layer. Rules vary depending on the map element type: polygons, lines, or points.

- **Polygons.** Vary polygon color.
 - **Polygon Center Points.** For markers that display at the center point of each polygon, vary marker color, marker size, and marker type.
- **Lines.** Vary line color and line width.
- **Points.** For markers that display for each point, vary marker color, marker size, and marker type.

Understanding Color Rules

Color rules apply to fill colors for polygons, lines, and markers that represent points or polygon center points.

Color rules support four options:

- Apply template style. The theme that you chose in the wizard defines the template style for the layer. Theme sets the style for font, the border style, and the palette.
- Visualize data by using color palette. You specify a palette by name. The report processor sets the color of each map element in a layer by stepping through each color in the palette, and then applying successively lighter shades of each color in the palette.
- Visualize data by using color ranges. You specify a beginning, middle, and end color. Then you specify distribution options. The report processor uses the distribution option values to create a set of colors that produces a display similar to a heat map. A heat map displays color as related to temperature. For example, 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. You specify a set of colors. The report processor sets the color of each map element in the layer by stepping through the values that you specify.

The default palette includes the color white. To avoid the strong contrast between white and other colors in the palette, specify a start color that is a light color in the palette.

To display map elements that are not associated with data as colorless, set **No Color** for the default color for map elements on the layer.

Color Scale

By default, all color rule values appear in the color scale, in addition to appearing in the first legend. The color scale is designed to display one range of colors. Choose the most important data to display in the color scale.

To remove the values that you do not want in the color scale, clear the color scale option for every color rule on every layer.

Understanding Line Width Rules

Width rules apply to lines. Width rules support two options:

- Use a default line width. You specify the line width in points.
- Visualize data by using line width. You set the minimum and maximum widths for the line, specify the data field to use to vary the width, and then specify the distribution options to apply to that data.

Understanding Marker Size Rules

Size rules apply to markers that represent points or polygon center points. Size rules support two options:

- Use a default marker size. You specify the size in points.
- Visualize data by using size. You set the minimum and maximum sizes for the marker, specify the data field to use to vary the size, and then specify the distribution options to apply to that data.

Understanding Marker Type Rules

Marker type rules apply to markers that represent points or polygon center points. Marker type rules support two options:

- Use a default marker type. You specify one of the available marker types.
- Visualize data by using markers. You specify a set of markers and specify the order in which you want them used. Marker types include **Circle**, **Diamond**, **Pentagon**, **PushPin**, **Rectangle**, **Star**, **Triangle**, **Trapezoid**, and **Wedge**.

Understanding Distribution Options

To create a distribution of values, you can divide your data into ranges. You specify the distribution type, the number of subranges, and the minimum and maximum range values.

In the following list, assume that you have three map elements and six related analytical values that range from 1 to 9999 with the following values: 1, 10, 200, 2000, 4777, 8999.

- **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. Subrange 1: 1, 10, 200, 500. Subrange 2: 4777. Subrange 3: 8999. This method does not take into account how the data is distributed. Very large values or very small values can skew the distribution results.
- **EqualDistribution.** Create ranges that divide the data so that each range has an equal number of items. For the example data, the three ranges would be 0-10, 11-500, 501-8999. Subrange 1: 1, 10. Subrange 2: 200, 500. Subrange 3: 4777, 8999. This method can skew the distribution by creating divisions that span very large or very small ranges.
- **Optimal.** Create ranges that automatically adjust distribution to create balanced subranges. The number of subranges is determined by the algorithm.
- **Custom.** Specify your own number of ranges to control the distribution of values. For the example data, you can specify ranges 3 ranges: 1-2, 3-8, 9.

The distribution values are used by the rules to vary the map element display values.

Understanding Legends and Legend Items

Legend items are created automatically from the rules that you specify for each layer. Rule options control how many items are created and which legend they appear in. By default, all items for all rules are added to the first legend. To move items out of the first legend, create as many additional legends as you need, and for each rule, specify the legend to use to display the items that result from the rule. To hide items based on a rule, specify a blank legend name.

To control where a legend appears, use the Legend Properties dialog box to specify a position relative to the map viewport. For more information, see [Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#).

Legends automatically expand to display the legend title or legend text. To format the text of legend items, use map legend keywords and custom formats. For more information, see [To change the format of content in a legend](#).

The following tables shows examples of different formats that you can use.

KEYWORD AND FORMAT	DESCRIPTION	EXAMPLE OF WHAT APPEARS AS TEXT IN THE LEGEND
#FROMVALUE {C0}	Displays the currency of the total value with no decimal places	\$400
#FROMVALUE {C2}	Displays the currency of the total value to two decimal places.	\$400.55
#TOVALUE	Displays the actual numeric value of the data field.	10000

KEYWORD AND FORMAT	DESCRIPTION	EXAMPLE OF WHAT APPEARS AS TEXT IN THE LEGEND
#FROMVALUE{N0} - #TOVALUE{N0}	Displays the actual numeric values of the beginning of the range and end of the range.	10 - 790

See Also

[Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#)

[Maps \(Report Builder and SSRS\)](#)

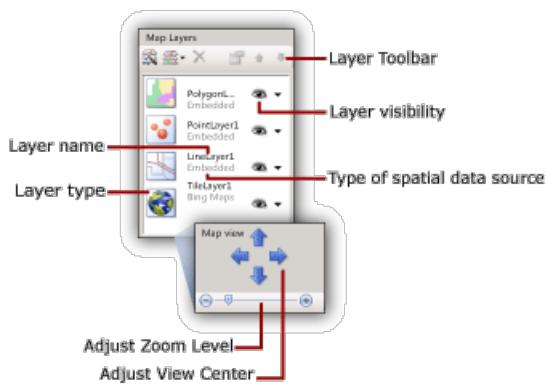
[Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#)

Add, Change, or Delete a Map or Map Layer (Report Builder and SSRS)

3/24/2017 • 9 min to read • [Edit Online](#)

A map is a collection of layers. When you add a map to a Reporting Services paginated report, you define the first layer. You can create additional layers by using the map layer wizard.

The easiest way to add, remove, or change options for a layer is to use the map layer wizard. You can also change options manually from the Map pane. To display the **Map** pane, click in the map on the report design surface. The following figure displays the parts of the pane:



Map layers are drawn from bottom to top in the order that they appear in the Map pane. In the previous figure, the tile layer is drawn first and the polygon layer is drawn last. Layers that are drawn later might hide map elements on layers that are drawn earlier. You can change the order of layers by using the arrow keys on the Map pane toolbar. To show or hide layers, toggle the visibility icon. You can change the transparency of a layer on the **Visibility** page of the **Layer Data** properties dialog box.

The following table displays the toolbar icons for the **Map** pane.

SYMBOL	DESCRIPTION	WHEN TO USE
	Map Layer Wizard	To add a layer by using a wizard, click New layer wizard .
	Add Layer	To manually add a layer, click Add Layer , and then click the type of map layer to add.
	Polygon Layer	Add a map layer that displays areas or shapes that are based sets of polygon coordinates.
	Line Layer	Add a map layer that displays paths or routes that are based on sets of line coordinates.
	Point Layer	Add a map layer that displays locations that are based on sets of point coordinates.

SYMBOL	DESCRIPTION	WHEN TO USE
	Tile Layer	Add a map layer that displays Bing Map tiles that correspond to the current map view area that is defined by the viewport.

At the bottom of the Map pane is the Map view area. To change the center or zoom options for the map, use the arrow keys to adjust the view center and the slider to adjust the zoom level.

For more information about layers, see [Maps \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a layer from the map layer wizard

- From the Ribbon, on the **Insert** menu, click **Map**, and then click **Map Wizard**. The wizard enables you to add a layer to the existing map. Most wizard pages are identical between the map wizard and the map layer wizard.

For more information, see [Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#).

To change options for a layer by using the map layer wizard

- Run the map layer wizard. This wizard enables you to change options for a layer that you created by using the map layer wizard. In the Map pane, right-click the layer, and on the toolbar, click the layer wizard button (

For more information, see [Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#).

To add a point, line, or polygon layer from the Map pane toolbar

- Click the map until the Map pane appears.
- On the toolbar, click the **Add Layer** button, and from the drop-down list, click the type of layer that you want to add: **Point**, **Line**, or **Polygon**.

NOTE

Although you can add a map layer and configure it manually, we recommend that you use the map layer wizard to add new layers. To launch the wizard from the Map pane toolbar, click the layer wizard button (

- Right-click the layer, and then click **Layer Data**.

- In **Use spatial data from**, select the source of spatial data. Options vary based on your selection.

If you want to visualize analytical from your report on this layer, do the following:

- Click **Analytical data**.
- In **Analytical dataset**, click the name of the dataset that contains analytical data and the match fields to build a relationship between analytical and spatial data.

- c. Click **Add**.
- d. Type the name of the match field from the spatial dataset.
- e. Type the name of the match field from the analytical dataset.

For more information about linking spatial and analytical data, see [Customize the Data and Display of a Map or Map Layer \(Report Builder and SSRS\)](#).

5. Click **OK**.

To filter analytical data for the layer

1. Click the map until the Map pane appears.
2. Right-click the layer in the Map pane, and then click **Layer Data**.
3. Click **Filters**.
4. Define a filter equation to limit the analytical data that is used in the map display. For more information, see [Filter Equation Examples \(Report Builder and SSRS\)](#).

To control point properties for a point layer or for polygon center points

1. Select **General** on the **Map Point Properties** dialog box to change label, tooltip, and marker type options for the following map elements:
 - All dynamic or embedded points on a point layer. Color rules, size rules, and marker type rules for points override these options. To override options for a specific embedded point, use the [Map Embedded Point Properties Dialog Box, Marker](#) page.
 - The center point for all dynamic or embedded polygons on a polygon layer. Color rules, size rules, and marker type rules for center points override these options. To override options for a specific center point, use the [Map Embedded Point Properties Dialog Box, Marker](#) page.

To specify embedded data as a source of spatial data

1. Click the map until the Map pane appears.
2. Right-click the layer, and then click **Layer Data**.
3. In **Use spatial data from**, select **Data embedded in report**.
4. To load map elements from an existing report or to create map elements based on an ESRI file, click **Browse**, point to the file, and then click **Open**. The map elements are embedded in this report definition. The spatial data that you point to must match the layer type. For example, for a point layer, you must point to spatial data that specifies sets of point coordinates.
5. In **Spatial field**, specify the name of the field that contains spatial data. You might need to determine this name from the source of spatial data.

NOTE

If you do not know the name of the field and you browsed to an ESRI Shapefile, use the **Link to ESRI shape file option** instead of this option.

6. Click **OK**.

To specify an ESRI Shapefile as a source of spatial data

1. Click the map until the Map pane appears.
2. Right-click the layer, and then click **Layer Data**.
3. In **Use spatial data from**, select **Link to ESRI Shapefile**.
4. In **File name**, type the location of an ESRI Shapefile, or click **Browse** to select an ESRI Shapefile.

NOTE

If the Shapefile is on your local computer, the spatial data is embedded in the report definition. To retrieve the data dynamically when the report is processed, you must upload the ESRI .shp file and its .dbf support file to the report server. For more information, see " How to: Upload a File or Report (Report Manager)" in the [Reporting Services documentation](#) in SQL Server Books Online.

5. Click **OK**.

To specify a report dataset field as a source of spatial data

1. Click the map until the Map pane appears.
2. Right-click the layer, and then click **Layer Data**.
3. In **Use spatial data from**, select **Spatial field in a dataset**.
4. In **Dataset name**, click the name of a dataset in the report that contains that spatial data that you want.
5. In **Spatial field name**, click the name of the field in the dataset that contains spatial data.
6. Click **OK**.

To add a tile layer

1. Click the map until the Map pane appears.
2. On the toolbar, click the **Add Layer** button, and from the drop-down list, click **Tile Layer**.

NOTE

For more information about the use of Bing map tiles in your report, see [Additional Terms of Use](#).

3. Right-click the tile layer in the Map pane, and then click **Tile Properties**.
4. In **Tile options**, select a tile style. If the Bing map tiles are available, the layer on the design surface updates with the style that you select.

NOTE

A tile layer can also be added when you add a polygon, line, or point layer in the Map or Map Layer wizard. On the **Choose spatial data and map view options** page, select the option **Add a Bing Maps background for this map view**.

To change the drawing order of a layer

1. Click the map until the Map pane appears.

2. Click the layer in the Map pane to select it.
3. On the Map pane toolbar, click the up or down arrow to change the drawing order of each layer.

To change the transparency of a polygon, line, or point layer

1. Click the map until the Map pane appears.
2. Right-click the layer, and then click **Layer Data**.
3. Click **Visibility**.
4. In **Transparency options**, type a value that represents the percentage transparency, for example, **40**. Zero (0) % transparency means that the layer is opaque. 100% transparency means that you will not see the layer in the report.
5. Click **OK**.

To change the transparency of a tile layer

1. Click the map until the Map pane appears.
2. Right-click the layer, and then click **Tile Properties**.
3. Click **Visibility**.
4. In **Transparency options**, type a value that represents the percentage transparency, for example, **40**.
5. Click **OK**.

To specify a secure connection for a tile layer

1. Click the map until the Map pane appears.
2. In the Map pane, click the tile layer to select it. The Properties pane displays the tile layer properties.
3. In the Properties pane, set UseSecureConnection to **True**.

The connection for the Bing Maps Web service will use the HTTP SSL (Secure Sockets Layer) service to retrieve Bing map tiles for this layer.

To specify the language for tile labels

1. By default, for tile styles that display labels, the language is determined from the default locale for Report Builder. You can customize the language setting for tile labels in the following ways.
 - Click the map outside the viewport to select the map. In the Properties pane, for the TileLanguage property, select a culture value from the drop-down list.
 - Click the report background to select the report. In the Properties pane, for the Language property, select a culture value from the drop-down list.

The order of precedence for setting the tile label language is: report property Language, default locale for Report Builder, and map property TileLanguage.

To conditionally hide a layer based on viewport zoom level

1. Set **Visibility** options to control the display for a map layer.
 - In the Map Layers pane, right-click a layer to select it, and on the Map Layers toolbar, click

Properties to open **Map Layer Properties**.

- Click **Visibility**.
- In Layer visibility, select **Show or hide based on zoom value**.
- Enter minimum and maximum zoom values for when display the layer.
- Optional. Enter a value for transparency.

You can also conditionally hide the layer. For more information, see [Hide an Item \(Report Builder and SSRS\)](#).

See Also

[Maps \(Report Builder and SSRS\)](#)

[Troubleshoot Reports: Map Reports \(Report Builder and SSRS\)](#)

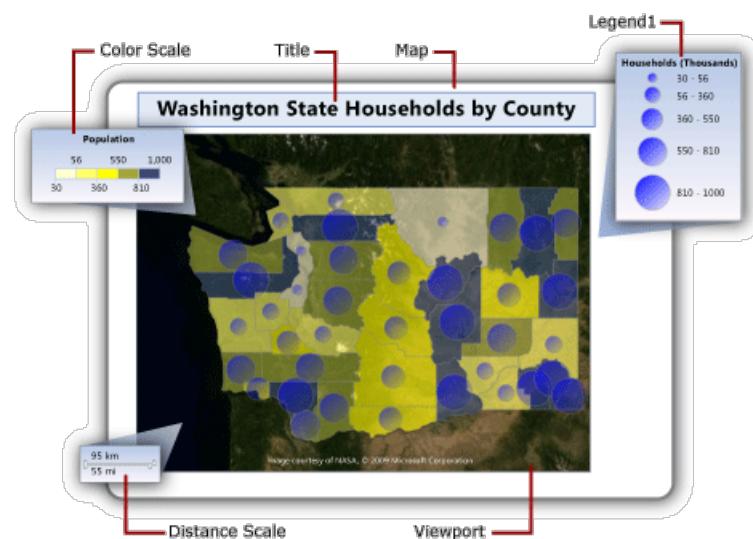
Change Map Legends, Color Scale, and Associated Rules (Report Builder and SSRS)

3/29/2017 • 8 min to read • [Edit Online](#)

In a Reporting Services paginated report, a map can contain map legends, a color scale, and a distance scale. These parts of a map help users interpret the data visualization on the map.

Legends include the following parts of a map:

- **Map legend** Displays a guide to help interpret the analytical data that varies the display of a map elements on a map layer. A map can have multiple legends. For each map layer, you specify which legend to use. A legend can provide a guide to more than one map layer.
- **Color scale** Displays a guide to help interpret colors on the map. A map has one color scale. Multiple layers can provide the data for the color scale.
- **Distance scale** Displays a guide to help interpret the scale of the map. A map has one distance scale. The current map viewport zoom value determines the distance scale.



To change the position of a legend relative to the viewport

To change the position of a legend relative to the viewport

1. In Design view, right-click the legend and open the <report item>**Properties** page.
2. In **Position**, click the location that specifies where to display the legend relative to the viewport.
3. To display the legend outside the viewport, select **Show <report item> outside viewport**.
4. Click **OK**.

NOTE

In preview, map legends and the color scale appear only when there are results from the rules related to that legend. If there are no items to display, the legend does not appear in the rendered report.

To change the layout of a map legend

To change the layout of a map legend

1. In Design view, right-click the legend and open the **Legend Properties** page.
2. In **Legend layout**, click the table layout that you want to use for the legend. As you click different options, the layout on the design surface changes.
3. Click **OK**.

To show or hide a map legend title

To show or hide a map legend title

- Right-click the map legend on the design surface, and then click **Show Legend Title**.

To show or hide a color scale title

To show or hide a color scale title

- Right-click the color scale on the design surface, and then click **Show Color Scale Title**.

To move items out of the first legend

Create as many additional legends as you need and then update the rules for each map layer specify which legend to display the rule results in.

To create a new legend

- In Design view, right-click the map outside the map viewport, and then click **Add Legend**.

A new legend appears on the map.

To display rule results in a legend

1. In Design view, click the map until the Map pane appears.
2. Right-click the layer that has the data that you want and then click <*map element type*>**Color Rule**.
3. Click **Legend**.
4. In the **Show in this legend** drop-down list, click the name of the legend to display the rule results in.
5. Click **OK**.

To vary map element colors based on a template style

To vary map element colors based on a template style

1. In Design view, click the map until the Map pane appears.
2. Right-click the layer that has the data that you want and then click <*map element type*>**Color Rule**.
3. Click **Apply template style**.

A template style specifies font, border style, and color palette. Each map element is assigned a different color from the color palette for the theme that was specified in the Map Wizard or Map Layer Wizard. This is the only option that applies to layers that do not have associated analytical data.

4. Click **OK**.

To vary map element colors based on color palette

To vary map element colors based on color palette

1. In Design view, click the map until the Map pane appears.
2. Right-click the layer that has the data that you want, and then click <*map element type*>**Color Rule**.

3. Click **Visualize data by using color palette**.

This option uses a built-in or custom palette that you specify. Based on related analytical data, each map element is assigned a different color or shade of color from the palette.

4. In **Data field**, type the name of the field that contains the analytical data that you want to visualize by color.

5. In **Palette**, from the drop-down list, select the name of the palette to use.

6. Click **OK**.

To vary map element colors based on color ranges

To vary map element colors based on color ranges

1. In Design view, click the map until the Map pane appears.

2. Right-click the layer that has the data that you want, and then click <map element type>**Color Rule**.

3. Click **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, divide the related analytical data into ranges. The report processor assigns the appropriate color to each map element based on its associated data and the range that it falls into.

4. In **Data field**, type the name of the field that contains the analytical data that you want to visualize by color.

5. In **Start color**, specify the color to use for the lowest range.

6. In **Middle color**, specify the color to use for the middle range.

7. In **End color**, specify the color to use for the highest range.

8. Click **OK**.

To vary map element colors based on custom colors

To vary map element colors based on custom colors

1. In Design view, click the map until the Map pane appears.

2. Right-click the layer that has the data that you want, and then click <map element type>**Color Rule**.

3. Click **Visualize data by using custom colors**.

This option uses the list of colors that you specify. Based on related analytical data, each map element is assigned a color from the list. If there are more map elements than colors, no color is assigned.

4. In **Data field**, type the name of the field that contains the analytical data that you want to visualize by color.

5. In **Custom colors**, click **Add** to specify each custom color.

6. Click **OK**.

To set distribution options for a legend

To set distribution options for a legend

1. In Design view, click the map until the Map pane appears.

2. Right-click the layer that has the data that you want, and then click <*map element type*>**Color Rule**.
 3. Select the **Visualize data by using** <rule type> option. To use distribution options, you must create ranges on the **Distribution** page based on analytical data that is associated with the layer.
 4. Click **Distribution**.
 5. Select one of the following distribution types:
 - **EqualInterval**. Specifies ranges that divide the data into equal range intervals.
 - **EqualDistribution**. Specifies ranges that divide the data so that each range has an equal number of items.
 - **Optimal**. Specifies ranges that automatically adjust distribution to create balanced subranges.
 - **Custom**. Specify your own number of ranges to control the distribution of values.
- For more information about distribution options, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#).
6. In **Number of subranges**, type the number of subranges to use. When the distribution type is **Optimal**, the number of subranges is automatically calculated.
 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 change the contents of a rule legend

To change the contents of a color, size, width, or marker type legend

1. In Design view, click the map until the Map pane appears.
2. Right-click the layer that has the data that you want, and then click <*map element type*>**Rule**.
3. Verify that **Visualize data by using** <rule type> is selected.
4. In **Data field**, verify that the analytical data that you are visualizing on the layer is selected.

NOTE

If no fields appear in the drop-down list, right-click the layer, and then click **Layer Data** to open the Map Layer Data Properties Dialog Box, Analytical Data page and verify that you have specified analytical data for this layer.

5. Click **Legend**.
6. In **Show in this legend**, select the map legend to use to display the rule results.
7. Click **OK**.

To change the contents of the color scale

To change the contents of the color scale or a color legend

1. In Design view, click the map until the Map pane appears.
2. Right-click the layer that has the data that you want, and then click <*map element type*>**Color Rule**.

3. Select the color rule option to use. To display items in a map legend or color scale, you must select one of the **Visualize data by using** <rule type> options.
4. In **Data field**, verify that the analytical data that you are visualizing on the layer is selected.

NOTE

If no fields appear in the drop-down list, right-click the layer, and then click **Layer Data** to open the Map Layer Data Properties Dialog Box, Analytical Data page and verify that you have specified analytical data for this layer.

5. Click **Legend**.
6. In **Color scale options**, select **Show in color scale** to display the rule results in the color scale. You can specify this option for more than one color rule.
7. Click **OK**.

To remove all items from a legend

To hide items based on a rule

1. In Design view, click the map until the Map pane appears.
2. Right-click the layer that has the data that you want, and then click <*map element type*>**Rule**.
3. Click **Legend**.
4. Click **OK**.

To change the format of content in a legend

Set legend options for the rule that is associated with the map legend.

To change the format of content in a legend

1. In Design view, click the map until the Map pane appears.
2. Right-click the layer that has the data that you want, and then click <*map element type*>**Rule**.
3. Click **Legend**.
4. **Legend text** displays keywords that specify which data appears in the legend. Use map keywords and custom formats to help control the format of legend text. For example, #FROMVALUE {C2} specifies a currency format with two decimal places. For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#).
5. Click **OK**.

See Also

[Maps \(Report Builder and SSRS\)](#)

[Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#)

[Customize the Data and Display of a Map or Map Layer \(Report Builder and SSRS\)](#)

[Troubleshoot Reports: Map Reports \(Report Builder and SSRS\)](#)

[Map Wizard and Map Layer Wizard \(Report Builder and SSRS\)](#)

Add Custom Locations to a Map (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

After you add a map to a Reporting Services paginated report, you can add your own point locations.

Display properties for all points on a layer are controlled by setting options for the point properties for the layer. For a selected embedded point, you can override the display properties.

NOTE

When you override the layer display properties for the embedded point, the changes that you make are not reversible.

For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a point layer

1. On the report design surface, click the map to select it and display the Map pane.
2. On the toolbar, click **Add Layer**.
3. From the drop-down list, click **Add Point Layer**. A point layer with no points is added to the map. By default, the point layer is ready for embedded points.

To add a custom point

1. On the report design surface, click the map to select it and display the Map pane.
2. In the Map pane, right-click a point layer that has type **Embedded**, and then click **Add Point**. The cursor changes to crosshairs.
3. To add a point, click a location on the map. An embedded point is added to the selected layer at the location where you click.

To customize the display for an embedded point

1. Right-click the point, and then click **Point Properties**. The **Map Embedded Point Properties** dialog box opens.
2. Click **Override point options for this layer**. Multiple property pages appear in the left pane.
3. Click the pages and set the display properties that you want to apply to this point.

See Also

[Maps \(Report Builder and SSRS\)](#)

[Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#)

Troubleshoot Reports: Map Reports (Report Builder and SSRS)

3/24/2017 • 14 min to read • [Edit Online](#)

Issues with maps in a Reporting Services paginated report might occur when you add a map or map layer to your report, when you customize an existing map or map layer in your report, when you preview a map in a report, or when you publish a report with a map. Use this topic to help troubleshoot these issues.

Need more help?

Try:

- MSDN forum for [SQL Server 2016](#)
- [SQL Server 2016](#) on Stack Overflow
- Log an issue or suggestion at [Microsoft Connect](#)

Report Definition Size Issues

Use this section to help solve issues that relate to report definition size.

How do I reduce the report definition size?

A map layer contains map elements that are created from spatial data. In some cases, map elements are embedded in the report definition. This happens in the following ways:

- If the source of spatial data is from a map in the Map Gallery or from an ESRI Shapefile on your local computer, map elements are automatically embedded in the report definition.

If you publish a report to the report server and there is a spatial data source reference to a local file, the spatial data cannot be retrieved at report processing time. To avoid this issue, the map data is embedded in the report definition.

- In the Map wizard or Layer wizard, if you select the option to embed spatial data, map elements based on the spatial data are embedded in the map layer in the report definition.
- In the Map pane, if you right-click the layer, and then click one of the **Embed Spatial Data** options, map elements based on the spatial data are embedded in the map layer in the report definition.
- To remove embedded data that is based on an ESRI Shapefile from a report definition, you must do the following:

1. Upload or publish the ESRI .shp and .dbf files to the report server.
2. In the report, in the Map pane in Design view, select the layer that has embedded data, and open the **Layer Data** properties. In **Use spatial data from**, select **Link to ESRI Shapefile**, and then browse to the folder on the report server that contains the ESRI Shapefiles, select it, and click OK.
3. Save your report. The embedded data for the layer that you changed has been removed from the report definition.

Map elements from a report in the Map Gallery will always be embedded in a map layer.

Spatial Data Issues

Use this section to help solve issues that relate to spatial data.

On the design surface, I see sample spatial data

At design time, the design surface might show the message about sample spatial data for the following reasons:

- Spatial data comes from an ESRI .shp file, but the corresponding .dbf file is not available. ESRI Shapefiles usually include both a .shp file with spatial data and a support file .dbf. Verify that the .dbf file is in the same directory as the .shp file.
- Spatial data comes from a dataset and the data connection for the query is not valid or the current credentials are not valid.
- The map layer contains a property with an expression. Expressions are not evaluated until the report runs. To see the map, you must preview the report.
- Spatial data comes from a dataset that has a specific scope. For example, when a map is nested in a tablix data region, or the map uses the same dataset scope for analytical and for spatial data, the data scope is not calculated until the report runs.

When I set an offset for an individual map element, a cluster of map elements move

Spatial data defines the map elements that are displayed on each map layer. A map element can be based on spatial data that is a single point, a set of points, a single line, a set of lines, a single polygon, or a set of polygons. Each map element is a unit. If a map element contains multiple points, and you move the element, all points for that map element will move.

The data for each map element is determined by the format of the spatial data from the external source. For example, when a query specifies spatial data from a SQL Server database, each row in the result set can contain multiple sets of point or line or polygon coordinates. All map elements that are defined by a single row in the result set are treated as a unit. If you want to vary the display of specific sets of coordinates, you must do one of the following:

- Change the query to return the coordinate sets as separate rows in the result set.
- Select the map elements to vary and set the corresponding embedded point, line, or polygon properties by overriding the default display properties for the corresponding layer type.

My layer that uses spatial data from an ESRI Shapefile always has embedded data.

To ensure that reports with maps can run on a report server, ESRI Shapefiles must be available as resources on the report server. If you add a layer to a map and specify a Shapefile that is on your local file system, the spatial data is automatically embedded in the report.

To replace the embedded data with a link to the ESRI Shapefile, you must upload the .shp file and its matching .dbf file to the report server, and then change the source of spatial data for the layer.

I renamed a data source or dataset to a friendly name and now no data appears in my map.

The report definition is not automatically updated when you manually change the name of any report item.

When you change the name of a dataset, any data region or map layer that refers to that dataset must be manually updated. To rebind a tablix, chart, or gauge to a dataset, select the item on the design surface, open the data region properties, and select the name of the appropriate dataset. To rebind a map layer to a dataset, select the layer, open the layer properties, and select the name of the appropriate dataset.

My spatial data contains nulls and empty strings.

In spatial data for the map report item, nulls are set to zero (0) and empty strings are set to blank ("").

For spatial that comes from a SQL Server database, to change this behavior, you must change the query that returns the spatial data.

My map exceeds the maximum number of spatial elements

By default, a map can have 20,000 map elements or 1,000,000 points. If your map exceeds these limits, you can use one of the following approaches:

- Remove a layer.
- Decrease the map resolution.
- Decrease the map viewport coordinates to view a smaller area.
- If the spatial data comes from a report dataset, set a filter to limit the data from the dataset. The filter must be set on a field that is not a spatial data type.
- If the spatial data comes from a SQL Server database, change the query to use spatial functions to limit the data to a smaller area.

Viewport Center and View Issues

Use this section to help solve issues that relate to viewport options.

I cannot set the center and view on an embedded map element.

To center a viewport on a specific map element, you must have associated the spatial data on a layer with analytical data.

I set the map center and view in my report. When I reopen the report, why isn't the map view the same?

If the user credentials that are needed to read spatial data are not available to the report when you open it, placeholder spatial data is used. Depending on the center and zoom options set for the map viewport, the map view might center on a different layer.

To reload the spatial data and use the map view center saved in the report, right-click the map viewport, and then click **Reload**. After you enter the credentials for the spatial data source, the layer loads the spatial data and the map view is restored.

The center and view for a map layer option does not work.

When the viewport is set to center on the spatial data for a specific layer, and the center of the view does not appear to be the center for the layer, there are probably small islands or areas that are included in the spatial data that are too small to be seen in the viewport. For example, spatial data for a country might include small islands or other small territories as part of the territory. The viewport uses all spatial data to calculate the center for the layer.

To override calculations for the layer, you can do one of the following:

- Specify a custom center for the viewport.
- Change the zoom level for the viewport to eliminate the locations that you do not want to include.
- Embed the spatial data in the report and delete the locations that you do not want to include.

Layer Issues

Use this section to help solve issues that relate to layer options.

I do not see one or more layers in my map.

Whether you see a map layer in a report depends on the availability of the spatial data, the relationship between the spatial data and the analytical data, the type of spatial data and the corresponding layer type, the visibility and transparency options on the layer, and the layer drawing order. If you do not see data from a layer, check the following options:

- **Layer type and spatial data type.** The layer type displays only spatial data that matches the layer type. For example, if the layer type is Point but the spatial data is Line, no data will appear.
- **Match field values.** The values in the fields that you specify to relate analytical data and spatial data must uniquely identify each map element. The fields must have the same data type. The values in the fields must be identical. For more information, see [Legend, Color Scale, and Distance Scale Issues](#).
- **Layer order.** The order of layers in the Map pane is the order in which layers are drawn in the report renderer. Spatial data on layers which are drawn first might be overwritten by spatial data for layers that are drawn later. Layers that appear at the top of the list are drawn first. When you change the order layers in the list, you are changing the drawing order of the layers.
- **Transparency.** You can specify the transparency for each map layer independently. Default values for transparency differ depending on how you add a layer. A transparency of 0% means the layer is opaque and no other layer data will show through this one. To allow other data to show through an existing layer, adjust the value to a higher percentage that gives you the effect that you want.
- **Visibility.** Visibility for a layer is either **Visible**, **Hidden**, or **ZoomBased**, based on the zoom level of the map viewport. The maximum and minimum range for zoom level can also be specified. Visibility can be based on an expression that evaluates to one of these values.

TIP

You can toggle visibility for each layer in the Map pane. When you are designing each layer, toggle all other layers off to determine whether the issue is for an individual layer or is for transparency issues among layers.

I set a filter on the map layer and it has no effect.

To filter data for a layer, the data type in the filter expression must be specified. Verify that you have specified the correct underlying data type so that the filter equation correctly evaluates the specified condition. For more information, see [Filter Equation Examples \(Report Builder and SSRS\)](#).

Legend, Color Scale, and Rule Issues

Use this section to help solve issues that relate to rules, legend, and color scale options.

How do I control the values in the map legend?

Legend values are determined automatically from map element type rules that you specify for each map layer and by distribution rules that you specify for the legend.

By default, all items generated by all rules display in the first legend. Values for all polygon, line, and point rules for each layer contribute to the combined legend range. To display items in different legends, you must first create multiple legends and then, for each rule, specify in which legend to display the related items.

To associate a rule with a specific legend, open the rule properties, and on the Legend page, select the name of the legend to use. To remove items from a legend, in legend options, select the blank line for the name of the legend. If you rename legend elements in the report, you must manually associate each layer with the appropriate legend item.

To control the title and content for each legend, use the Legend properties for the rule. You can specify how many divisions to create, change the calculations that assign values to each division, set minimum and maximum range values, and change the format of the legend text.

For more information, see [Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#).

The rules that I set do not give the results that I expect.

Rules apply to the analytical data that is associated with map elements on a layer. Use the following list to help identify issues with all color rules, size rules, width rules, and marker type rules:

- The precedence for applying style to each map element (polygon, line, point) is, from lowest to highest precedence: layer properties; map element properties for all map elements on the layer; rules that you specify; and then, for embedded map elements that you select the override option for, the values that you specify. Once you select the override option for an embedded element, rules no longer apply, even if you later change values back to their original setting.
- Match field issues. Match fields enable data binding between map elements and analytical data. The spatial data and analytical data fields that correspond to match fields must have the same data type and the same format. If the match field does not exactly match the spatial data and analytical data, the rule has no effect. For example, if the match field for spatial data has extra blanks or extra punctuation compared to the match field for the analytical data, no match occurs.
- For more information, see [Vary Polygon, Line, and Point Display by Rules and Analytical Data \(Report Builder and SSRS\)](#).

What is the value NaN on the Color Scale?

NaN stands for Not a Number. Color scales values are expected to be numeric. Check the distribution settings and the legend text value for the rules that are associated with the color scale. If you created custom distribution ranges, verify that you specified the lower bound on the first range and the upper bound on the last range.

My color scale does not appear when I run the report.

The color scale displays information to the user when a map layer specifies color rules for polygons, lines, or points for the whole layer or for embedded map elements. If no map element specifies a color rule, or if the color rules specify by using a legend instead of the color map, then the color map does not appear in the rendered report.

To display the color scale, specify color rules for a layer or an embedded map element. For more information, see [Change Map Legends, Color Scale, and Associated Rules \(Report Builder and SSRS\)](#).

Tile Issues

Use this section to help solve issues that relate to tile background options.

I cannot see the Bing maps tile background.

The following settings affect whether a Bing maps tile background displays in local preview or on a report that runs from the report server:

- The map tile layer must exist. In the Map wizard or Layer wizard, select **Add a Bing Maps background for this map view**. This adds a tile layer for the current map viewport view center and zoom level. You can also add a tile layer from the Map pane toolbar.
- The map coordinate system for the viewport must be **Geographic**, not **Planar**.
- The map projection must be **Mercator**.
- For local preview, you must have internet access. For a report that runs from the report server, the report server must be configured to support tile background. For more information, see "Planning for Map Support" in the [Reporting Services documentation](#) in SQL Server Books Online.

For more information about adding a tile layer, see [Add, Change, or Delete a Map or Map Layer \(Report Builder and SSRS\)](#).

How do I control the text on a tile layer?

Both **Road** and **Hybrid** views include text. The text is part of the tiles that come from Bing Maps Web Services.

To include a tile layer without text, select **Aerial** view.

ToolTip and Label Issues

Use this section to help solve issues that relate to label or ToolTip options.

I get an expression error about dataset scope when I set a label or ToolTip to an expression.

When your spatial data comes from a map gallery or an ESRI Shapefile, the associated data is not part of a report dataset. You cannot use expression syntax for a dataset field reference to specify this data for a label or tooltip.

To specify data that is related to spatial data that is not part of a report dataset, you must use the symbol # followed by a label that specifies the name of the data.

See Also

[Maps \(Report Builder and SSRS\)](#)

[Troubleshoot Report Builder](#)

Images, Text Boxes, Rectangles, and Lines (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In addition to data regions like tables, matrices, and charts, Reporting Services paginated reports use other report items like images, text boxes, and rectangles to add visual interest, highlight key information, or provide related information. You can change the formatting of a report item. For example, you can add a border or padding, change the initial visibility or direction, or specify an exact size and location for the item.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

In This Section

[Text Boxes \(Report Builder and SSRS\)](#)

Text boxes can be placed anywhere on a report and can contain labels, fields, or calculated data. You use expressions to define the value to display in a text box when you view a report.

Every cell in a table or matrix is also a text box, which you can format in the same way that you format stand-alone text boxes.

[Rectangles and Lines \(Report Builder and SSRS\)](#)

Lines display horizontally, vertically, or diagonally. A line is defined with a start and end point and can have various styles (for example, weight and color) assigned to it. A line has no data associated with it.

Rectangles can be used as a graphical element, or as a container for other report items. As a graphical element, a rectangle has the same properties as a line. As a container, a rectangle acts as a parent container for all report items inside it. Placing report items in a parent container helps control how they appear on each report page.

[Images \(Report Builder and SSRS\)](#)

Images display binary image data in a report. You provide the source for the image. The source can be a URL reference to an image stored on a Web server, a reference to embedded image data, or a reference to binary image data in a database. Report Builder and Report Designer support .bmp, .jpeg, .gif, and .png files.

See Also

[Formatting Report Items \(Report Builder and SSRS\)](#)

Text Boxes (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

When you think of a text box, you probably think of a stand-alone box containing text on a surface like in Microsoft Office PowerPoint. In Reporting Services paginated reports, some text boxes are like that, and they can display static text for titles, descriptions, and labels, or dynamic text based on expressions. But every cell in a table or matrix (a tablix data region) also contains a text box, which you can format the same way you format stand-alone text boxes in your report.

NOTE

If you drag a report dataset field value directly to the report design surface, or to a text box on the report design surface, you only see the first value in the result set when you run the report. To see all the values for a field, you need to create a table, matrix, or list data region first, and drag the field to a cell in the data region. That way, when you run the report, you will see all the values in that field.

To show repeating text in a free-form layout, create a list data region and place the text box in it. Use a list when you want to repeat a form for multiple values, for example, a customer invoice form repeated once for each customer. Read more about [creating invoices and forms with lists](#).

Use a rectangle container when you want to control the text box layout and white space below the last text box. For more information, see [Rectangles and Lines \(Report Builder and SSRS\)](#).

The expressions in a text box can contain literal text, point to a field in the database, or calculate data. All expressions are shown as placeholder text so that you can format numbers, colors, and other appearance properties. You can also combine placeholders with literal text in the same text box.

You can format text in any single text box with multiple fonts, colors, styles, and actions. For more information, see [Formatting Text and Placeholders \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Growing and Shrinking a Text Box

By default, text boxes are a fixed size. You can allow a text box to shrink or expand vertically based on its contents. For more information, see [Allow a Text Box to Grow or Shrink \(Report Builder and SSRS\)](#).

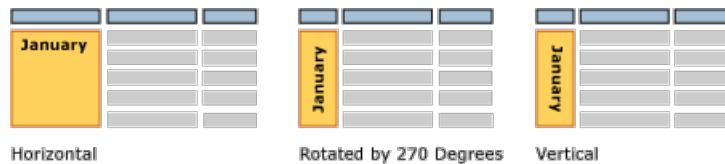
Rotating a Text Box

Rotating text boxes can help you create more readable reports, support locale-specific text orientation, fit more columns on a printed report that has fixed page size, and create reports with more graphical appeal. A text box can be rotated in different directions: horizontal, vertical (rotated 90 degrees), or rotated by 270 degrees. The vertical option is most commonly used for East Asian languages that are written top to bottom. In most renderers the vertical option handles the glyph rotation properly so that the text is written top to bottom, but the characters are not on their sides. For other languages, in the vertical and 270-degree options the text is written sideways.

You can rotate text boxes that contain static text, fields from a report dataset, or calculated data. The text box can

be stand-alone in the report body, in a table or matrix, or in a report header and footer.

The following picture shows three versions of a table report that groups data by month. The text box that contains the month value uses a different text box orientation.



Orientation is set on the text box and applies to all the text in the box. You cannot specify a different orientation for parts of the text box.

To get started, see the section on rotating text in the [Tutorial: Format Text \(Report Builder\)](#), and see [Set Text Box Orientation \(Report Builder and SSRS\)](#).

How-To Topics

[Add, Move, or Delete a Text Box \(Report Builder and SSRS\)](#)

[Format Text in a Text Box \(Report Builder and SSRS\)](#)

[Set Text Box Orientation \(Report Builder and SSRS\)](#)

[Allow a Text Box to Grow or Shrink \(Report Builder and SSRS\)](#)

See Also

[Formatting Text and Placeholders \(Report Builder and SSRS\)](#)

[Formatting Numbers and Dates \(Report Builder and SSRS\)](#)

Add, Move, or Delete a Text Box (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Text boxes are the most commonly used report item in Reporting Services paginated reports. You can add a text box to the report body to display information such as titles, parameter choices, built-in fields, and dates.

Every cell in a table or matrix is really a text box. Almost all report data displayed in a report with tables and matrices is the result of the report processor evaluating the contents of each text box in the report. As such, you can format cells in the same way you would format other text boxes outside the data region.

To add a text box to a list data region, you must first add the text box and then drag it into the list.

NOTE

When you click a text box, you're immediately editing the text in the text box. To select the text box itself, not the text in it, press ESC.

To add a text box

1. On the **Insert** tab in Design view, click **Text Box**.
2. On the design surface, click and then drag a box to the desired size of the text box.

To add a text box in a list

1. On the **Insert** tab in report design view, click **List**.
2. On the design surface, click and then drag a box to the desired size of the list.
3. On the **Insert** tab, click **Text Box**.
4. On the design surface, click and then drag a box to the desired size of the text box inside the list you added in step 1.
5. To confirm the text box is correctly nested inside the list, select the text box.

NOTE

If you click in the text box and are in edit mode, press ESC to select the text box.

6. In the Properties pane, verify that the **Parent** property is the rectangle that was automatically added to the list data region.

NOTE

If the Properties pane is not visible, check **Properties** on the **View** tab.

To move a text box

1. In report design view, click any empty space within the text box to select the text box.

NOTE

If you click in the text box and are in edit mode, press ESC to select the text box.

2. Click the text box handle and drag the text box to the new location.

Alternatively, use the arrow keys to move a selected text box horizontally or vertically. To move the text box in smaller increments on the design surface, hold down CTRL plus the arrow keys.

To delete a text box

1. In report design view, right-click any empty space within the text box to select it, and then click **Delete**.

Alternatively, click any empty space within the text box, and then press DELETE.

NOTE

If you click in the text box and are in edit mode, press ESC to select the text box.

See Also

[Text Boxes \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Keyboard Shortcuts \(Report Builder\)](#)

Set Text Box Orientation (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a Reporting Services paginated report, you can rotate a text box in different directions:

- Horizontally
- Vertically (rotated 90 degrees, with text reading from top to bottom)
- Rotated by 270 degrees (text reading from bottom to top).

Because you rotate the text box not the text, the rotation applies to all the text in the text box. You cannot specify different directions for parts of the text. Size the column width and the row height manually to accommodate the rotated text.

The WritingMode property, which you use to specify text orientation, isn't in the **Text Box Properties** dialog box. It's in the Properties pane and set the property there.

To rotate text

1. Create a report or open an existing report, and [add a text box](#) to the design surface.
2. Select the text box that you want to rotate.
3. If the Properties pane is not open, on the **View** tab, select the **Properties** check box.
4. In the Properties pane, find the WritingMode property and select the text orientation to apply to the text box.

NOTE

When the properties in the Properties pane are organized into categories, WritingMode is in the **Localization** category.

5. In the list box, select **Horizontal**, **Vertical**, or **Rotate270**.

See Also

[Text Boxes \(Report Builder and SSRS\)](#)

[Tutorial: Format Text \(Report Builder\)](#)

Allow a Text Box to Grow or Shrink (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a Reporting Services paginated report, text boxes aren't just the stand-alone boxes on the report design surface. Every cell in a table or matrix (a tablix data region) also contains a text box, which can be formatted in the same way as stand-alone text boxes. By default, text boxes are a fixed size. You can set options that let a text box expand or shrink based on its contents. These options correspond to the **CanGrow** or **CanShrink** properties in the Properties pane.

To Allow a Text Box to Grow or Shrink

1. Right-click the text box and click **Text Box Properties**.
2. Click the **General** tab.
 - To allow the text box to expand vertically based on its contents, select **Allow height to increase**.
 - To allow the text box to shrink based on its contents, select **Allow height to decrease**.

See Also

[Text Boxes \(Report Builder and SSRS\)](#)

Rectangles and Lines (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Rectangles and lines can create visual effects within a Reporting Services paginated report. You can set display properties on these report items from the Border section of the Home tab, and set other properties in the Properties pane. You can add features like a background color or image, a tooltip, or a bookmark to a rectangle.

Rectangles and Lines as Report Parts

You can publish rectangles with the items that they contain separately from the report as report parts. Report parts are self-contained report items that are stored on the report server and can be included in other reports.

You cannot publish the report items within a rectangle as report parts. When people add the rectangle to a report, they get the rectangle and the items it contains. Read more about [Report Parts](#).

Using a Rectangle as a Container

You can use a rectangle as a container for other items. When you move the rectangle, the items that are contained within the rectangle move along with it. An item within the rectangle shows the name of the rectangle in its **Parent** property. For more information about using a rectangle as a container, see [Add a Rectangle or Container \(Report Builder and SSRS\)](#) and [Display the Same Data on a Matrix and a Chart \(Report Builder\)](#).

NOTE

A rectangle is only a container for items that you either create in the rectangle or drag into the rectangle. If you draw a rectangle around an item that already exists on the design surface, the rectangle will not act as its container. The rectangle will not be listed in the item's Parent property.

When using rectangles to contain report items, consider how the items will be affected as a whole during report rendering. Report items that contain repeated rows of data (for example, tables) will expand to accommodate the data that is returned by a query, and this affects the positioning of other items in the rectangle. A table will push items down if they are positioned below the data region. To anchor an item in place, you can place the report item inside of a rectangle that has an upper edge above the lower edge of the table. For more information, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

Adding a Report Border

You can add a border to a report by adding borders to the headers, footers, and report body themselves, without adding lines or rectangles. For more information, see [Add a Border to a Report \(Report Builder and SSRS\)](#).

How-To Topics

[Add a Border to a Report \(Report Builder and SSRS\)](#)

[Add a Rectangle or Container \(Report Builder and SSRS\)](#)

[Add and Modify a Line \(Report Builder and SSRS\)](#)

See Also

[Add a Rectangle or Container \(Report Builder and SSRS\)](#)

Add a Rectangle or Container (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Add a rectangle to a Reporting Services paginated report when you want a graphical element to separate areas of the report, emphasize areas of a report, or provide a background for one or more report items. Rectangles are also used as containers to help control the way data regions render in a report. You can customize the appearance of a rectangle by editing rectangle properties such as the background and border colors. For more information about using a rectangle as a container, see [Rectangles and Lines \(Report Builder and SSRS\)](#) and [Display the Same Data on a Matrix and a Chart \(Report Builder\)](#).

To add a rectangle

1. On the **Insert** tab, in the **Report Items** group, click **Rectangle**.
2. On the design surface, click the location where you want the upper left corner of the rectangle, and drag to where you want the lower-right corner.

Note that as you move the cursor, "snap lines" appear as the cursor lines up with other objects on the design surface. These help you if you want objects to be aligned.

To create a container

1. Add a rectangle report item to the report.
2. Drag other report items into the rectangle.

NOTE

A rectangle is only a container for items that you either create in the rectangle or drag into the rectangle. If you draw a rectangle around an item that already exists on the design surface, the rectangle will not act as its container.

To change rectangle properties such as color, style, or weight

1. Select the rectangle, and then click the line color, style, or weight options in the **Border** section of the Home tab.
2. Click the arrow next to the **Border** button to determine which sides of the rectangle to change.

NOTE

If you set the line style to **Double** and the line width is 1 1/2 pt or narrower, the line may not appear double when you run the report in Report Builder, Report Designer, or in the Reporting Services web portal. It appears double when you export the report to other formats such as Microsoft Word and Acrobat PDF.

See Also

- [Rectangles and Lines \(Report Builder and SSRS\)](#)
- [Rendering Behaviors \(Report Builder and SSRS\)](#)

Add and Modify a Line (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

You can add a line to a Reporting Services paginated report when you want a graphical element to separate sections of the report. You can customize the appearance of the line by editing line properties such as color or style. For example, you might want to incorporate company colors into the report.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a line

1. On the **Insert** tab, click **Line**.
2. On the design surface, click where you want one end of the line, and then click where you want the other end.
3. To change the line properties, select the line on the design surface and then edit its properties in the **Border** section of the **Home** tab.

NOTE

If you set the line style to **Double** and the line width is 1 1/2 pt or narrower, the line may not appear double when you run the report in Report Builder, Report Designer, or a Reporting Services web portal. It appears double when you export the report to other formats such as Microsoft Word and Acrobat PDF.

See Also

[Rectangles and Lines \(Report Builder and SSRS\)](#)

Add a Border to a Report (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

You can add a border to a Reporting Services paginated report by adding borders to the headers, footers, and report body themselves, without adding lines or rectangles.

If you add a report border that appears on the page header and footer, do not suppress the header and footer on the first and last pages of the report. If you do, the border may appear cut off at the top or bottom of the first and last pages of the report. For more information, see [Page Headers and Footers \(Report Builder and SSRS\)](#).

To add a border to a report

1. Right-click in the header outside any items in the header, and click **Header Properties**. On the **Border** tab, add a left, top, and right border with the style you want.

NOTE

If your report doesn't have headers, you can place borders around just the report body, or you can add headers from the **Insert** tab.

2. Right-click in the body outside any items on the design surface, and click **Body Properties**. On the **Border** tab, add a left and right border with the style you want.
3. Right-click in the footer outside any items in the footer, and click **Footer Properties**. On the **Border** tab, add a left, bottom, and right border with the style you want.

See Also

[Rectangles and Lines \(Report Builder and SSRS\)](#)

Images (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

An image is a report item that contains a reference to an image that is embedded in the report, stored in a database, stored on the report server, or stored elsewhere on the Web. An image can be a picture that is repeated with rows of data. You can also use an image as a background for certain report items.

Storing logos on a server is a good idea because you can use the same logo in many reports.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Comparing External, Embedded, and Data-Bound Images

When you use a server-based or other external image in a report, the image item contains a path that points to an image on the report server or wherever it exists on the Web. When you use an embedded image, however, the image data is stored within the report definition and does not exist as a separate file.

Server-based images work well for logos and static pictures that are shared among several reports or Web pages. Embedded images ensure that the images are always available to the report, but they cannot be shared. Report definitions with external images are smaller than definitions with embedded images.

Data-bound images can also be displayed from binary data stored in a database. For example, the pictures that appear alongside product names in a product list are database images. In the following picture, the images of bicycles are stored in a database and retrieved in the report to illustrate each product.

Category	Product	Color	Picture
Mountain bikes	Mountain-300	Black	
	Mountain-100	Silver	
	Mountain-400-W	Red	
Road bikes	Road-450	Red	
	Road-250	Orange	
	Road-550-W	Yellow	
Touring bikes	Touring-1000	Yellow	
	Touring-3000	Blue	

Images as Report Parts

You can save images separately from a report as report parts. Report parts are self-contained report items that are stored on the report server and can be included in other reports. Use Report Builder to browse and select parts from the Report Part Gallery to add to your reports. Use Report Designer or Report Builder to save report parts for use in the Report Part Gallery. For more information, see **Report Parts (Report Builder and SSRS)** and **Report Parts in Report Designer (SSRS)** on the Web at microsoft.com.

Embedding Images

You can embed images in a report so that all image data is stored within the report definition. When you embed an image, the image is MIME-encoded and stored as text in the report definition. Using an embedded image ensures that the image is always available to the report, but it also increases the size of the report definition.

For more information about embedding an image, see [Embed an Image in a Report \(Report Builder and SSRS\)](#).

External Images

You can include stored images in a report by specifying a URL to the image. When you use an external image in a report, the image source is set to **External** and the value for the image is the URL address or path to the image.

For more information, see [Specifying Paths to External Items \(Report Builder and SSRS\)](#).

When the report is run in Report Builder or Report Designer, preview uses the credentials of the user to display the image. When the report is run on the report server, the image in the report may not be displayed if the server credentials are not sufficient to access the image. In that case, contact your system administrator.

For more information about adding an external image to a report, see [Add an External Image \(Report Builder and SSRS\)](#).

Background Images

You can use an image as a background image in the body of the report or in a rectangle, text box, list, matrix, or table. A background image and an image have similar properties. You can also specify how the image is repeated to fill the background of the item.

NOTE

Some rendering extensions, like the HTML rendering extension, render the background image for the report body in the body, the page header, and the page footer. You can define a separate background image for the page header and footer, but if no image is defined, the report uses the background image of the body. Other rendering extensions, like the Image rendering extension, do not render the body background image in the page header and footer.

For more information about adding a background image, see [Add a Background Image \(Report Builder and SSRS\)](#).

Data-bound Images

You can add images that are stored in a database to your report. You use the same image report item as the one used for static images, but with a set of properties that indicate that the image is stored in a database. To view instructions about working with data-bound images, see [Add a Data-Bound Image \(Report Builder and SSRS\)](#).

How-to Topics

[Add an External Image \(Report Builder and SSRS\)](#)

[Embed an Image in a Report \(Report Builder and SSRS\)](#)

[Add a Background Image \(Report Builder and SSRS\)](#)

[Add a Data-Bound Image \(Report Builder and SSRS\)](#)

See Also

[Exporting to an Image File \(Report Builder and SSRS\)](#)

Rendering Behaviors (Report Builder and SSRS)

Add an External Image (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

External images can be on a report server in native mode or SharePoint integrated mode, or on any other Web site. When you include external images in your report, you must verify that the image exists and that the report reader has permissions to access the image. For more information, see [Images \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add an external image

1. In report design view, on the **Insert** tab, click **Image**.
2. On the design surface, click and then drag a box to the desired size of the image.
3. On the **General** tab of the **Image Properties** dialog box, type a name in the **Name** text box or accept the default.
4. (Optional) In the **Tooltip** text box, type text to display when the user hovers over the image in a report rendered for HTML.
5. In **Select the image source**, select **External**.

For an image on a report server in native mode, type a relative path to the image in the **Use this image** box—for example, `./images/image1.jpg`.

For an image on a report server in SharePoint integrated mode, or any other Web site, type a full URL to the image in the **Use this image** box—for example,

`http://<SharePointservername>/<site>/Documents/images/image1.jpg`.

For more information, see [Specifying Paths to External Items \(Report Builder and SSRS\)](#).

6. (Optional) Click **Size**, **Visibility**, **Action**, or **Border** to set additional properties for the image report item.
7. Click **OK**.

See Also

[Embed an Image in a Report \(Report Builder and SSRS\)](#)

[Add a Background Image \(Report Builder and SSRS\)](#)

[Image Properties Dialog Box, General \(Report Builder and SSRS\)](#)

Embed an Image in a Report (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A report can include an embedded image. Embedding an image ensures that the image is always available to a report, but can affect the size of the report definition, the file that defines the report. The images embedded in a report are listed in the Report Data pane.

You might want to embed an image in the report definition before adding the image to the design surface. For more information, see [Add a Background Image \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To embed an image in a report

1. In report design view, on the **Insert** tab, click **Image**.
2. On the design surface, click and then drag a box to the desired size of the image.
3. In the **General** page of the **Image Properties** dialog box, type a name in the **Name** text box or accept the default.
4. (Optional) In the **ToolTip** text box, type the text that you want to appear when the user hovers over the image in the rendered report.
5. In **Select the image source**, select **Embedded**.
6. Click the **Import** button next to the **Use this image** text box
7. In **Files of type**, select the image file type, navigate to the file, and then click **Open**.
8. In the **Image Properties** dialog box, click **OK**.

The image is displayed in the box you drew on the design surface, and the file is displayed under the Images folder in the Report Data pane.

NOTE

The MIME type (for example, bmp) is derived automatically when the image is imported. To change the MIME type, see the next procedure.

(optional) To change the MIME type of an imported image

1. Open the report in Design view.
2. Select the image on the design surface. The **Properties** pane displays the image properties.

NOTE

If the Properties pane is not visible, on the **View** tab, click **Properties**.

3. Click in the text box next to the **MIMEType** property and select a new MIME type from the drop-down list.

See Also

[Images \(Report Builder and SSRS\)](#)

[Add a Data-Bound Image \(Report Builder and SSRS\)](#)

[Image Properties Dialog Box, General \(Report Builder and SSRS\)](#)

Add a Background Image (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can add a background image to a report item such as a rectangle, text box, list, matrix, table, and some parts of a chart, or a report section such as the page header, page footer, or report body. You can define a background image for any selected item on the report design surface that displays **BackgroundImage** in the Properties pane. Like other images, the background image can be a URL to an image on the report server, an image from a dataset field, or an image embedded in the report definition. To use an image embedded in the report, you must first add the image to the report definition before you can add the image to the design surface.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To embed an image in the report definition

1. In the Report Data pane, right-click the **Images** node, and then click **Add Image**.

NOTE

If the Report Data pane is not visible, on the **View** tab, click **Report Data**.

2. Navigate to the image you want to embed in your report definition, and then click **OK**.

To add a background image

1. In report design view, select the report item to which you want to add a background image.
2. If the Properties pane is not visible, on the **View** tab, select **Properties**.
3. In the Properties pane, expand **BackgroundImage**, and then do the following:

- For an embedded image:

Set **Source** to **Embedded**.

Set **Value** to the name of an image that is embedded in the report.

- For an external image:

Set **Source** to **External**.

Set **Value** to a valid path to an image. This can be on a report server in native mode or SharePoint integrated mode, or it can be on any other Web site. For more information, see [Add an External Image \(Report Builder and SSRS\)](#).

- For an image that is contained in a field in the database to which the report item is connected:

Set **Source** to **Database**.

Set **Value** to the name of a field in the report dataset. For more information, see [Add a Data-Bound Image \(Report Builder and SSRS\)](#).

For **MIMEType**, or file format, select the appropriate MIME type for the image—for example, .bmp.

NOTE

MIMETYPE applies only if the **Source** property is set to **Database**. If the **Source** property is set to **External** or **Embedded**, the value of **MIMETYPE** is ignored.

- For **BackgroundRepeat**, select an expression, **Default**, **Repeat**, **RepeatX**, or **RepeatY**, or **Clip**.

For background images in a chart, **BackgroundRepeat** can be set to **Default**, **Repeat**, **Fit**, and **Clip**, but not **RepeatX** or **RepeatY**.

See Also

[Images \(Report Builder and SSRS\)](#)

[Image Properties Dialog Box, General \(Report Builder and SSRS\)](#)

Add a Data-Bound Image (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A report can include a reference to an image that is stored in a database. Such an image is known as a *data-bound image*. The pictures that appear alongside product names in a product list are examples of data-bound images.

Adding a data-bound image to a page header or page footer requires additional steps. For more information, see [Page Headers and Footers \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a data-bound image

1. In report design view, create a table with a data source connection and a dataset with a field that contains binary image data. For more information, see [Tables \(Report Builder and SSRS\)](#).
2. Insert a column in your table. For more information, see [Insert or Delete a Column \(Report Builder and SSRS\)](#).
3. On the **Insert** menu, click **Image**, and then click in the data row of the new column.
4. On the General page of the **Image Properties** dialog box, type a name in the **Name** text box or accept the default.
5. (Optional) In the **Tooltip** text box, type text to display when the user hovers over the image in the report rendered for HTML.
6. In **Select the image source**, select **Database**.
7. In **Use this Field**, select the field that contains images in your report.
8. In **Use this MIME type**, select the MIME type, or file format, of the image—for example, bmp.
9. Click **OK**.

An image placeholder appears on the report design surface.

See Also

[Images \(Report Builder and SSRS\)](#)

[Embed an Image in a Report \(Report Builder and SSRS\)](#)

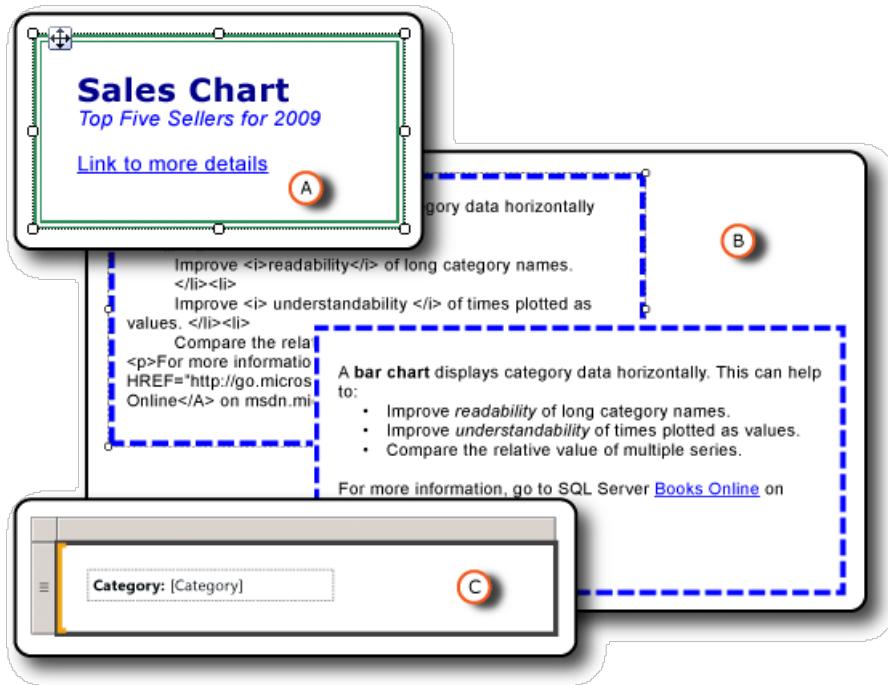
[Add an External Image \(Report Builder and SSRS\)](#)

[Image Properties Dialog Box, General \(Report Builder and SSRS\)](#)

Formatting Report Items (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Formatting the items in your report makes the report look more attractive and enhances its readability. You can format the text boxes and individual items within text boxes, the images, the expressions, and the data while in report design mode.



A. Text box with a double-line border and a variety of formatting styles, including a link.

B. Text box with a dashed border and raw HTML and rendered HTML.

C. Text box with a text label and a placeholder.

You can change formatting options by selecting the item that you want to format and then opening the item's Properties dialog box. For example, if you want to format the contents of an entire text box or a selected word within the text box, right-click the item and select **Text Box Properties**. Then, you can apply the formatting styles that you want.

To quickly get started, see [Tutorial: Format Text \(Report Builder\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

In This Section

[Formatting Text and Placeholders \(Report Builder and SSRS\)](#)

Describes how to format text and customize format options for different blocks of text within a text box.

[Importing HTML into a Report \(Report Builder and SSRS\)](#)

Describes how to insert and use HTML in a report.

[Formatting Numbers and Dates \(Report Builder and SSRS\)](#)

Describes how to use standard and custom formatting strings that are supported by Reporting Services.

[Formatting Lines, Colors, and Images \(Report Builder and SSRS\)](#)

Describes how to format lines, gridlines, colors, and images within report items and data regions.

[Set the Locale for a Report or Text Box \(Reporting Services\)](#)

Describes how to change the setting for data display formats that differ by language and region, such as date, currency, and number values.

See Also

[Formatting a Chart \(Report Builder and SSRS\)](#)

Formatting Text and Placeholders (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

A text box can be a report item or an individual cell within a data region that contains text, a calculated field, a pointer to a field in a database, or a combination of all three items. You can mix fonts and colors, add bold and italic styles, and use paragraph styles such as alignment and hanging indents. You can format an entire text box or you can format specific text, numbers, expressions, or fields within the text box.

Font, size, color, and effects all contribute to the readability of a report. Font, font style, font size, and underline effects can be applied to text within a text box or data region. By default, the report font that is used is Arial, 10 points, and black. By using the **Text Box** and **Text Properties** dialog boxes, you can specify how the text appears when the report is rendered.



In this illustration, the text box itself has a border, and all the text is in the same text box, but the text has a variety of formatting.

To quickly get started, see [Tutorial: Format Text \(Report Builder\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Creating Placeholder Text in a Text Box

When a simple or complex expression is defined inside a text box, the resulting UI representation of this expression is known as a *placeholder*. You can define colors, fonts, actions, and other behavior on any number of placeholders or sections of text within a single text box.

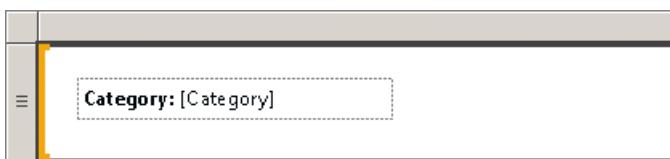
The value of a placeholder is always a simple or complex expression. You can add a placeholder to a text box by creating an expression using one of the following methods:

- Drag a field from the **Report Data** pane and drop it into the text box. If you drag the expression anywhere else on the report body, a new text box will be created that contains the placeholder. The value of this placeholder will be the field expression that corresponds to the field that was dropped.
- Right-click anywhere in the text box and select **Insert Placeholder**. In the **Placeholder Properties** dialog box, you can specify an expression as the value of your placeholder. For more information, see [Placeholder Properties Dialog Box, General \(Report Builder and SSRS\)](#).
- Type any simple or complex expression into the text box. For example, if you type **Name: [Name]** into the text box, the **[Name]** text will be displayed as a placeholder that represents the expression
`=Fields!Name.Value .`

- Type an expression in an empty text box by starting with an equal sign (=). When you change the focus off the text box, the resulting expression is converted into a placeholder that you can edit. If the text box is not empty, or the equal sign is inserted anywhere but as the first character in the text box, the equal sign is treated as a string literal and a placeholder is not created. For more information about defining simple and complex expressions, see [Expression Uses in Reports \(Report Builder and SSRS\)](#).

Formatting Placeholders and Static Text in a Text Box

You can format placeholders using the **Placeholder Properties** dialog box. You can format only the entire placeholder, not sections of the placeholder. If you want to see the underlying expression, you can pause your pointer on the placeholder. You can change the underlying expression by double-clicking the placeholder or right-clicking the placeholder and selecting **Placeholder Properties**. You can also specify a UI label using the **Label** property in **General** of the **Placeholder Properties** dialog box. This will be the text that is shown at design-time for the placeholder.



In this illustration, a text box in a list contains both a label with bold formatting and a placeholder with no formatting.

Unlike placeholder text, you can align individual text in a text box separately, use multiple paragraphs within a single text box, and define other behavior for any subset of text.

You can define colors, fonts, actions, and other behavior on any subset of text within a single text box to create a mail merge or template for text in your report. You can also use multiple paragraphs inside a single text box. For example, if you have two separate paragraphs of text, you can separate the paragraphs by pressing ENTER in the text box. You can also set an alignment value for any individual string of text. You can also define an action for individual text in a text box. This can be useful if you want to add a hyperlink for a string of text that is contained inside a text box.

NOTE

Actions defined on the text box have a higher priority than actions defined for individual text in a text box.

For more information about mixed formatting, see [Format Text in a Text Box \(Report Builder and SSRS\)](#).

Aligning Horizontal Text using General

In **Alignment** on the **Text Box Properties** dialog box, you can specify how the text should be aligned horizontally. If you do not specify a value for alignment, the default value of the alignment is **Default**. This means that the text is aligned based on the field type of your placeholder value. If you specify an expression that evaluates to a non-string value, i.e., not a number, the text is aligned to the right. If your expression evaluates to a string value, such as a number, the text is aligned to the left.

See Also

[Expressions \(Report Builder and SSRS\)](#)

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Formatting Scales on a Gauge \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Placeholder Properties Dialog Box, General \(Report Builder and SSRS\)](#)

[Exporting to Microsoft Excel \(Report Builder and SSRS\)](#)

[Text Boxes \(Report Builder and SSRS\)](#)

Format Text in a Text Box (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can format any part of the text within a text box independently, and mix placeholder text and static text in one text box. This ability to mix formats and add placeholder text enables you to create mail merges or templates for text in your report. Any expression can be defined and formatted separately using a placeholder.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To combine multiple formats in a text box

1. On the **Insert** tab, click **Text Box**. Click the design surface, and then drag to create a box that is the size you want.
2. Inside the text box, select the text you want to format.
3. Right-click the selected text, and click **Text Properties**.
4. Set formatting options. For example, on the **General** tab:
 - **Tooltip** Type text or an expression that evaluates to a ToolTip. The ToolTip appears when the user pauses the pointer over the item in a report
 - **Markup type** Select an option to indicate how the selected text will be rendered:
 - Plain Text** Display the selected text as simple text. HTML will be treated as literal text.
 - HTML** Display the selected text as HTML. If the expression value of the placeholder contains valid HTML tags, these tags will be rendered as HTML. For more information, see [Importing HTML into a Report \(Report Builder and SSRS\)](#).
5. Click **OK**.
6. Repeat steps 2 through 5 for the remaining text you want to format.

To format text and placeholders differently in the same text box

1. On the **Insert** tab, click **List**. Click the design surface, and then drag to create a box that is the size you want. The **Dataset Properties** dialog box opens. You can use a shared dataset or a dataset embedded in your report. For more information, click [Dataset Properties Dialog Box, Query \(Report Builder\)](#) or [Dataset Properties Dialog Box, Query](#).
2. On the **Insert** tab, click **Text Box**. Click in the list, and then drag to create a box that is the size you want.
3. Type a label in the text box — for example, **My Field**:
4. Drag a field from your dataset into the text box. A placeholder is created for your field.
5. For basic formatting, select the placeholder text and then click one of the formatting options in the **Font** group on the **Home** tab. For example, click the **Bold** button.

For more formatting options, right-click the placeholder text, and then click **Placeholder Properties**.
6. Click **OK**. In report design view, the text box should contain "**My Field**: [FieldName]", where *FieldName* is

the name of your field.

7. Click **Run**.

The list repeats one time for every value in the field, and the *FieldName* placeholder is replaced each time by the value of that field in the dataset.

See Also

[Text Boxes \(Report Builder and SSRS\)](#)

[Formatting Text and Placeholders \(Report Builder and SSRS\)](#)

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Add HTML into a Report \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

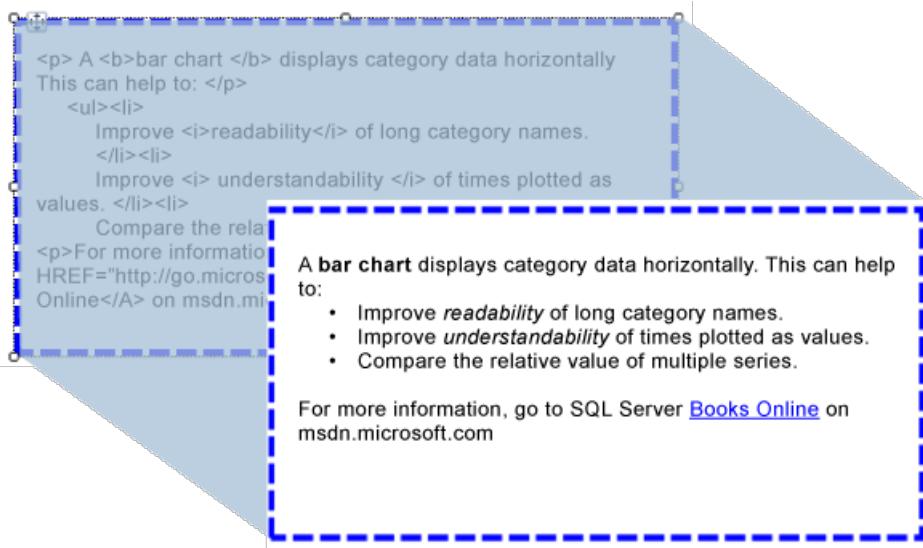
[Formatting Numbers and Dates \(Report Builder and SSRS\)](#)

[Placeholder Properties Dialog Box, General \(Report Builder and SSRS\)](#)

Importing HTML into a Report (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can use a text box to insert HTML-formatted text that you have retrieved from a field in your dataset into a report. The text can come from any simple or complex expression that evaluates to correctly formatted HTML. Formatted text can be rendered to all supported output formats, including PDF.



This illustration shows text with HTML formatting in report design view, and the same text as it is rendered when the report is run.

NOTE

When you import text that contains HTML markup, the data must always be parsed by the text box first. Because only a subset of HTML tags is supported, the HTML that is shown in the rendered report may differ from your original HTML.

To quickly get started, see [Tutorial: Format Text \(Report Builder\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Supported HTML Tags

The following is a complete list of tags that will render as HTML when defined as placeholder text:

- Hyperlinks: <A HREF>
- Fonts:
- Header, style and block elements: <H{n}>, <DIV>, , <P>, <DIV>, , <HN>
- Text format: , <I>, <U>, <S>

- List handling: , ,

Any other HTML markup tags will be ignored during report processing. If the HTML represented by the expression in the placeholder text is not well formed, the placeholder is rendered as plain text. All HTML tags are case-insensitive.

If the text in your text box contains only one block of text, any HTML in the placeholder that defines block elements will render correctly. However, if the text box has multiple blocks of text, the HTML tags are ignored and the structure of the text is defined by the blocks of text.

If more than one tag is defined for text, and Reporting Services detects a conflict between the HTML and existing report constraints, only the innermost HTML tag will be treated as HTML.

For more information, see [Add HTML into a Report \(Report Builder and SSRS\)](#).

Limitations of Cascading Style Sheet Attributes

When using cascading style sheet (CSS) attributes, only a basic set of tags are defined. The following is a list of attributes that are supported:

- text-align, text-indent
- font-family
- font-size
 - Only valid RDL size values, in absolute CSS length units are supported. Supported units are: in, cm, mm, pt, pc.
 - Relative CSS length units are ignored and not supported. Unsupported units include em, ex, px, %, rem.

For more information on CSS units, see: [CSS Values and Units Reference](#)
([http://msdn.microsoft.com/library/ms531211\(VS.85\).aspx](http://msdn.microsoft.com/library/ms531211(VS.85).aspx)).

- color
- padding, padding-bottom, padding-top, padding-right, padding-left
- font-weight

Here are some considerations for using CSS:

- Malformed CSS values are ignored in the same way as malformed HTML.
- When both attribute and CSS style attributes exist in the same tag, the CSS property has a higher precedence. For example, if your text is <p style="text-align: right" align="left">, only the text-align attribute will be applied and the text will be right-aligned.
- For attributes and CSS styles, if a property is specified more than once, only the last instance of the property is applied. For example, if your text is <p align="left" align="right">, the text will be right-aligned.

See Also

[Rendering to HTML \(Report Builder and SSRS\)](#)

Add HTML into a Report (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Using a placeholder, you can import HTML from a field in your dataset for use in the report. By default, a placeholder represents plain text, so you will need to change the placeholder mark-up type to HTML. For more information, see [Importing HTML into a Report \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add HTML from a field in your dataset into a text box

1. On the **Insert** tab, click **List**. Click the design surface, and then drag to create a box that is the size you want.

The **Dataset Properties** dialog box opens. You can use a shared dataset or a dataset embedded in your report. For more information, click [Dataset Properties Dialog Box, Query \(Report Builder\)](#) or [Dataset Properties Dialog Box, Query](#).

2. On the **Insert** tab, click **Text Box**. Click in the list, and then drag to create a box that is the size you want.
3. Drag an HTML field from your dataset into the text box. A placeholder is created for your field.
4. Right-click the placeholder, and then click **Placeholder Properties**.
5. On the **General** tab, verify that the **Value** box contains an expression that evaluates to the field you dropped in step 3.
6. Click **HTML - Interpret HTML tags as styles**. This causes the field to be evaluated as HTML.
7. Click **OK**.

See Also

- [Formatting Numbers and Dates \(Report Builder and SSRS\)](#)
- [Formatting Lines, Colors, and Images \(Report Builder and SSRS\)](#)
- [Placeholder Properties Dialog Box, General \(Report Builder and SSRS\)](#)

Formatting Numbers and Dates (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

You can format numbers and dates in data regions by selecting a format from the **Number** page of the corresponding data region's **Properties** dialog box.

To specify format strings within a text box report item, you need to select the item that you want to format, right-click, select **Text Box Properties**, and then click **Number**. You can format individual cells in a table or matrix data region in the same manner, because cells in a table or matrix are individual text boxes.

A chart data region commonly shows dates along the category (x) axis, and values along the value (y) axis. To specify formatting in a chart, right-click an axis and select **Axis Properties**. On the value axis, you can specify formats only for numbers. For more information, see [Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#).

To specify formatting in a Gauge data region, right-click the scale of the gauge and select **Radial Scale Properties** or **Linear Scale Properties**.

NOTE

If some formatting options you want to use are grayed out, it means that those formatting options are not compatible with the field's data type, which is set in the data source. For example, if the field contains number values but the field's data type is String, you cannot apply numerical data formatting options such as currency or decimals.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Considerations for Formatting Numbers and Dates

Before you format numbers and dates in your report, consider the following:

- By default, numbers are formatted to reflect the cultural settings on the client computer. Use formatting strings to specify how numbers are displayed so that formatting is consistent regardless of where the person who is viewing the report is located.
- The formats provided on the **Number** page are a subset of the .NET Framework standard numeric format strings. To format a number or date using a custom format that is not shown in the dialog box, you can use any .NET Framework format strings for numbers or dates. For more information about custom format strings, see the [Formatting Types](#) topic on MSDN.
- If a custom format string has been specified, it has a higher priority over default settings that are culture-specific. For example, suppose you set a custom format string of "#,###" to show the number 1234 as 1,234. This may have different meaning to users in the United States than it does to users in Europe. Before you specify a custom format, consider how the format you choose will affect users of different cultures who may view the report.
- If you specify an invalid format string, the formatted text is interpreted as a literal string which overrides

the formatting.

- If you are formatting a mix of numbers and characters in the same text box, consider using a placeholder to format the number separately from the rest of the text. For more information, see [Formatting Text and Placeholders \(Report Builder and SSRS\)](#). If an invalid format string is specified for the Format property on the text box, the format string is ignored. If an invalid format string is specified for the Format property on the chart or gauge, the format string that you specified is interpreted as a string and formatting is not applied.
- If you select **Currency** under **Category** and you check **Show values in**, you can select **Thousands**, **Millions**, or **Billions** to display numbers using financial formats. For example, if the field value is 1,789,905,394 and you select **Billions** and specify 2 decimal places, the value displayed in the report is 1.78.

See Also

[Formatting Text and Placeholders \(Report Builder and SSRS\)](#)

[Formatting Lines, Colors, and Images \(Report Builder and SSRS\)](#)

[Formatting a Chart \(Report Builder and SSRS\)](#)

[Format Axis Labels as Dates or Currencies \(Report Builder and SSRS\)](#)

[Formatting Scales on a Gauge \(Report Builder and SSRS\)](#)

Formatting Lines, Colors, and Images (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Reporting Services gives you the ability to format lines, colors, data regions, images, and other report items.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Borders, Lines and Gridlines

Borders, lines, and gridlines can visually tie items together on the page and help your report readers easily read the contents of the report. Using the predefined border styles, you can quickly add a border around a text box, a group of text boxes, or an image. In addition, you can change the style, width, and color for the borders, lines, and gridlines. Borders are added around the entire item selected or around a border along an edge of the item, for example, a border along the bottom of a text box.

To format borders and gridlines in a text box, report layout, or around an image, use the **Border** tab of the report item's **Properties** dialog box. For example, if you want to add a border around an image, right-click the image and then in the **Image Properties** dialog box, click **Border**.

In addition to the standard border frames, additional border frames can be applied to charts. For more information, see [Add a Border Frame to a Chart \(Report Builder and SSRS\)](#).

You can also add a border to the report itself. For more information, see [Add a Border to a Report \(Report Builder and SSRS\)](#).

Applying Background Colors

A solid color can be added to the background of the entire report, a text box within the report, or to a cell or group of cells within a data region. By default, the background color is white; however, you can select a color from the **Fill** tab of the report item's **Properties** dialog box. For example, if you want to change the background color of a text box, right-click the text box and select **Text Box Properties**. Click **Fill** and then select the color you want. In this dialog box, you can select a background color for the selected item or you can add an image that appears in the background.

When you use the chart, you can also specify gradients and pattern styles for background colors. For more information, see [Formatting a Chart \(Report Builder and SSRS\)](#).

Using Images as Formatting

Fields that contain images can be added to a data region. If you use an image field, the images appear in the report with the report is run.

You can also add images such as logos to the background of your report or to a rectangle, text box, table, matrix, or some parts of a chart, or to the body and page sections of a report. For more information, see [Images \(Report Builder and SSRS\)](#).

See Also

[Formatting Text and Placeholders \(Report Builder and SSRS\)](#)

[Formatting Numbers and Dates \(Report Builder and SSRS\)](#)

[Format Text in a Text Box \(Report Builder and SSRS\)](#)

[Fill Dialog Box \(Report Builder and SSRS\)](#)

Set the Locale for a Report or Text Box (Reporting Services)

3/24/2017 • 1 min to read • [Edit Online](#)

The **Language** property on a report or a text box contains the locale setting, which determines the default formats for displaying report data that differ by language and region, for example, date, currency, or number values. The **Language** property on a text box overrides the **Language** property on the report. If no value is specified for **Language**, Reporting Services uses the locale of the operating system on the report server for published reports or of the report authoring computer for report preview.

For HTML reports, you can override the default **Language** value and use the language specified by the HTTP header of the browser client by using the built-in field User!Language in an expression for the **Language** property of a report or a text box.

You can also specify the **Language** property for a report in a URL. For more information, see [Set the Language for Report Parameters in a URL](#).

To set the locale for a report

1. In Design view, click outside the report design surface to select the report.
2. In the Properties pane, for the **Language** property, type or select the language that you want to use for the report.

To set the locale for a text box

1. In Design view, select the text box to which you want to apply the locale settings.
2. In the Properties pane, do the following:
 - For the **Calendar** property, type or select the calendar that you want to use for dates.
 - For the **Direction** property, type or select the direction in which the text is written.
 - For the **Language** property, type or select the language that you want to use for the text box. This value overrides the **Language** property for the Report.
 - For the **NumeralLanguage** property, type or select the format to use for numbers in the text box.
 - For the **NumeralVariant** property, type or select the variant of the format to use for numbers in the text box.
 - For the **UnicodeBiDi** property, select the level of bidirectional embedding to use in the text box.

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Solution Design Considerations for Multi-Lingual or Global Deployments \(Reporting Services\)](#)

Report Parameters (Report Builder and Report Designer)

3/24/2017 • 17 min to read • [Edit Online](#)

This topic describes the common uses for Reporting Services report parameters, the properties you can set, and much more. Report parameters enable you to control report data, connect related reports together, and vary report presentation. You can use report parameters in paginated reports you create in Report Builder and Report Designer, and also in mobile reports you create in SQL Server Mobile Report Publisher. Read more about [Report Parameters Concepts](#).

| **Applies to:** Report Builder, Reporting Services SharePoint mode and Native mode

To try adding a parameter to a report yourself, see [Tutorial: Add a Parameter to Your Report \(Report Builder\)](#).

Common Uses for Parameters

Here are some of the most common ways to use parameters.

Control Paginated and Mobile Report Data

- Filter paginated report data at the data source by writing dataset queries that contain variables.
- Filter data from a shared dataset. When you add a shared dataset to a paginated report, you cannot change the query. In the report, you can add a dataset filter that includes a reference to a report parameter that you create.
- Filter data from a shared dataset in a SQL Server mobile report. See [Create mobile reports with SQL Server Mobile Report Publisher](#) for more information.
- Enable users to specify values to customize the data in a paginated report. For example, provide two parameters for the start date and end date for sales data.

Connect Related Reports

- Use parameters to relate main reports to drillthrough reports, to subreports, and to linked reports. When you design a set of reports, you can design each report to answer certain questions. Each report can provide a different view or a different level of detail for related information. To provide a set of interrelated reports, create parameters for the related data on target reports.

For more information, see [Drillthrough Reports \(Report Builder and SSRS\)](#), [Subreports \(Report Builder and SSRS\)](#), and [Create a Linked Report](#).

- Customize sets of parameters for multiple users. Create two linked reports based on a sales report on the report server. One linked report uses predefined parameter values for sales persons and the second linked report uses predefined parameter values for sales managers. Both reports use the same report definition.

Vary Report Presentation

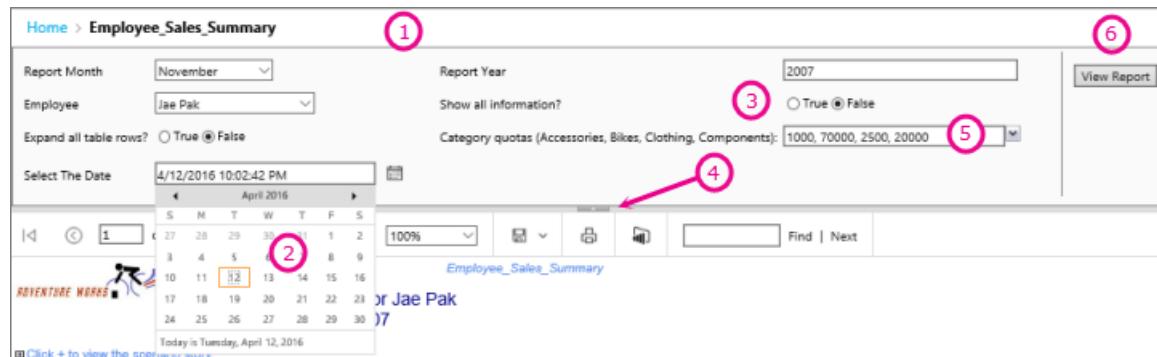
- Send commands to a report server through a URL request, to customize the rendering of a report. For more information, see [URL Access \(SSRS\)](#) and [Pass a Report Parameter Within a URL](#).

- Enable users to specify values to help customize the appearance of a report. For example, provide a Boolean parameter to indicate whether to expand or collapse all nested row groups in a table.
- Enable users to customize report data and appearance by including parameters in an expression.

For more information, see [Parameters Collection References \(Report Builder and SSRS\)](#).

Viewing a Report with Parameters

When you view a report that has parameters, the report viewer toolbar displays each parameter so you can interactively specify values. The following illustration shows the parameter area for a report with parameters @ReportMonth, @ReportYear, @EmployeeID, @ShowAll, @ExpandTableRows, @CategoryQuota, and @SalesDate.



- Parameters pane** The report viewer toolbar displays a prompt and default value for each parameter. You can customize the layout of parameters in the parameters pane. For more information, see [Customize the Parameters Pane in a Report \(Report Builder\)](#).
- @SalesDate parameter** The parameter @SalesDate is data type **DateTime**. The prompt Select the Date appears next to the text box. To modify the date, type a new date in the text box or use the calendar control.
- @ShowAll parameter** The parameter @ShowAll is data type **Boolean**. Use the radio buttons to specify **True** or **False**.
- Show or Hide Parameter Area handle** On the report viewer toolbar, click this arrow to show or hide the parameters pane.
- @CategoryQuota parameter** The parameter @CategoryQuota is data type **Float**, so it takes a numeric value. @CategoryQuota is set to allow multiple values.
- View Report** After you enter parameter values, click **View Report** to run the report. If all parameters have default values, the report runs automatically on first view.

Creating Parameters

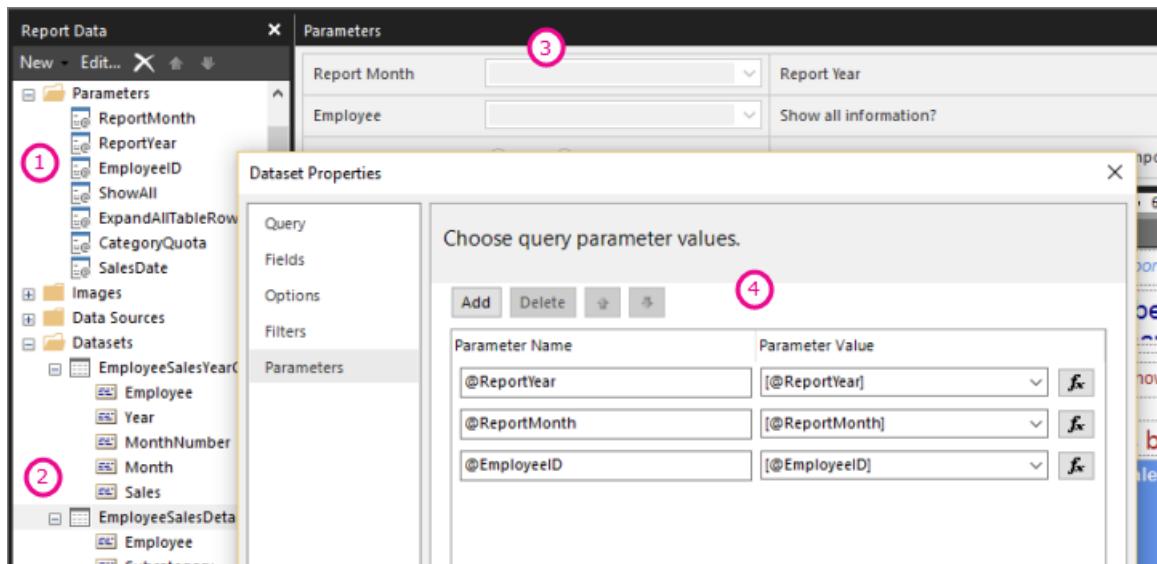
You can create report parameters in a few different ways.

NOTE

Not all data sources support parameters.

Dataset query or stored procedure with parameters

Add a dataset query that contains variables or a dataset stored procedure that contains input parameters. A dataset parameter is created for each variable or input parameter, and a report parameter is created for each dataset parameter.



This image from Report Builder shows:

1. The report parameters in the Report Data pane.
2. The dataset with the parameters.
3. The Parameters pane.
4. The parameters listed in the Dataset Properties dialog box.

The dataset can be embedded or shared. When you add a shared dataset to a report, dataset parameters that are marked internal cannot be overridden in the report. You can override dataset parameters that are not marked internal.

For more information, see [Dataset Query](#) in this topic.

Create a parameter manually

Create a parameter manually from the Report Data pane. You can configure report parameters so that a user can interactively enter values to help customize the contents or appearance of a report. You can also configure report parameters so that a user cannot change preconfigured values.

NOTE

Because parameters are managed independently on the server, republishing a main report with new parameter settings does not overwrite the existing parameters settings on the report.

Report part with a parameter

Add a report part that contains references to a parameter or to a shared dataset that contains variables.

Report parts are stored on the report server and available for others to use in their reports. Report parts that are parameters cannot be managed from the report server. You can search for parameters in the Report Part Gallery and after you add them, configure them in your report. For more information, see [Report Parts \(Report Builder and SSRS\)](#).

NOTE

Parameters can be published as a separate report part for data regions that have dependent datasets with parameters. Although parameters are listed as a report part, you cannot add a report part parameter directly to a report. Instead, add the report part, and any necessary report parameters are automatically generated from dataset queries that are contained or referenced by the report part. For more information about report parts, see [Report Parts \(Report Builder and SSRS\)](#) and [Report Parts in Report Designer \(SSRS\)](#).

Parameter Values

The following are options for selecting parameter values in the report.

- Select a single parameter value from a drop-down list.
- Select multiple parameter values from a drop-down list.
- Select a value from a drop-down list for one parameter, which determines the values that are available in the drop-down list for another parameter. These are cascading parameters. Cascading parameters enables you to successively filter parameter values from thousands of values to a manageable number.

For more information, see [Add Cascading Parameters to a Report \(Report Builder and SSRS\)](#).

- Run the report without having to first select a parameter value because a default value has been created for the parameter.

Report Parameter Properties

You can change the report parameter properties by using the Report Properties dialog box. The following table summarizes the properties that you can set for each parameter:

PROPERTY	DESCRIPTION
Name	Type a case-sensitive name for the parameter. The name must begin with a letter and can have letters, numbers, an underscore (_). The name cannot have spaces. For automatically-generated parameters, the name matches the parameter in the dataset query. By default, manually-created parameters are similar to ReportParameter1.
Prompt	The text that appears next to the parameter on the report viewer toolbar.

PROPERTY	DESCRIPTION
Data type	<p>A report parameter must be one of the following data types:</p> <p>Boolean. The user selects True or False from a radio button.</p> <p>Date Time. The user selects a date from a calendar control.</p> <p>Integer. The user types values in a text box.</p> <p>Float. The user types values in a text box.</p> <p>Text. The user types values in a text box.</p> <p>Note that when available values are defined for a parameter, the user chooses values from a drop-down list, even when the data type is DateTime.</p> <p>For more information about report data types, see RDL Data Types.</p>
Allow blank value	<p>Select this option if the value of the parameter can be an empty string or a blank.</p> <p>If you specify valid values for a parameter, and you want a blank value to be one of the valid values, you must include it as one of the values that you specify. Selecting this option does not automatically include a blank for available values.</p>
Allow null value	<p>Select this option if the value of the parameter can be a null.</p> <p>If you specify valid values for a parameter, and you want null to be one of the valid values, you must include null as one of the values that you specify. Selecting this option does not automatically include a null for available values.</p>
Allow multiple values	<p>Provide available values to create a drop-down list that your users can choose from. This is a good way to ensure that only valid values are submitted in the dataset query.</p> <p>Select this option if the value for the parameter can be multiple values that are displayed in a drop-down list. Null values are not allowed. When this option is selected, check boxes are added to the list of available values in a parameter drop-down list. The top of the list includes a check box for Select All. Users can check the values that they want.</p> <p>If the data that provides values changes rapidly, the list the user sees might not be the most current.</p>
Visible	<p>Select this option to display the report parameter at the top of the report when it is run. This option allows users to select parameter values at run time.</p>
Hidden	<p>Select this option to hide the report parameter in the published report. The report parameter values can still be set on a report URL, in a subscription definition, or on the report server.</p>

PROPERTY	DESCRIPTION
Internal	Select this option to hide the report parameter. In the published report, the report parameter can only be viewed in the report definition.
Available values	<p>If you have specified available values for a parameter, the valid values always appear as a drop-down list. For example, if you provide available values for a DateTime parameter, a drop-down list for dates appears in the parameter pane instead of a calendar control.</p> <p>To ensure that a list of values is consistent among a report and subreports, you can set an option on the data source to use a single transaction for all queries in the datasets that are associated with a data source.</p> <p>** Security Note ** In any report that includes a parameter of data type Text, be sure to use an available values list (also known as a valid values list) and ensure that any user running the report has only the permissions necessary to view the data in the report. For more information, see Security (Report Builder).</p>
Default values	<p>Set default values from a query or from a static list.</p> <p>When each parameter has a default value, the report runs automatically on first view.</p>
Advanced	<p>Set the report definition attribute UsedInQuery, a value that indicates whether this parameter directly or indirectly affects the data in a report.</p> <p>Automatically determine when to refresh Choose this option when you want the report processor to determine a setting for this value. The value is True if the report processor detects a dataset query with a direct or indirect reference to this parameter, or if the report has subreports.</p> <p>Always refresh Choose this option when the report parameter is used directly or indirectly in a dataset query or parameter expression. This option sets UsedInQuery to True.</p> <p>Never refresh Choose this option when the report parameter is not used directly or indirectly in a dataset query or parameter expression. This option sets UsedInQuery to False.</p> <p>** Caution ** Use Never Refresh with caution. On the report server, UsedInQuery is used to help control cache options for report data and for rendered reports, and parameter options for snapshot reports. If you set Never Refresh incorrectly, you could cause incorrect report data or reports to be cached, or cause a snapshot report to have inconsistent data. For more information, see Report Definition Language (SSRS).</p>

Dataset Query

To filter data in the dataset query, you can include a restriction clause that limits the retrieved data by specifying values to include or exclude from the result set.

Use the query designer for the data source to help build a parameterized query.

- For Transact-SQL queries, different data sources support different syntax for parameters. Support ranges from parameters that are identified in the query by position or by name. For more information, see topics for specific external data source types in [Report Datasets \(SSRS\)](#). In the relational query designer, you must select the parameter option for a filter to create a parameterized query. For more information, see [Relational Query Designer User Interface \(Report Builder\)](#).
- For queries that are based on a multidimensional data source such as Microsoft SQL Server Analysis Services, SAP NetWeaver BI, or Hyperion Essbase, you can specify whether to create a parameter based on a filter that you specify in the query designer. For more information, see the query designer topic in [Query Designers \(Report Builder\)](#) that corresponds to the data extension.

Parameter Management for a Published Report

When you design a report, report parameters are saved in the report definition. When you publish a report, report parameters are saved and managed separately from the report definition.

For a published report, you can use the following:

- Report parameter properties.** Change report parameter values directly on the report server independently from the report definition.
- Cached reports.** To create a cache plan for a report, each parameter must have a default value. For more information, see [Caching Reports \(SSRS\)](#).
- Cached shared datasets.** To create a cache plan for a shared dataset, each parameter must have a default value. For more information, see [Caching Reports \(SSRS\)](#).
- Linked reports.** You can create linked reports with preset parameter values to filter data for different audiences. For more information, see [Create a Linked Report](#).
- Report subscriptions.** You can specify parameter values to filter data and deliver reports through subscriptions. For more information, see [Subscriptions and Delivery \(Reporting Services\)](#).
- URL access.** You can specify parameter values in a URL to a report. You can also run reports and specify parameter values using URL access. For more information, see [URL Access \(SSRS\)](#).

Parameter properties for a published report are generally preserved if you republish the report definition. If the report definition is republished as the same report, and parameter names and data types remain the same, your property settings are retained. If you add or delete parameters in the report definition, or change the data type or name of an existing parameter, you may need to change the parameter properties in the published report.

Not all parameters can be modified in all cases. If a report parameter gets a default value from a dataset query, that value cannot be modified for a published report and cannot be modified on the report server. The value that is used at run time is determined when the query runs, or in the case of expression-based parameters, when the expression is evaluated.

Report execution options can affect how parameters are processed. A report that runs as a snapshot cannot use parameters that are derived from a query unless the query includes default values for the parameters.

Parameters for a Subscription

You can define a subscription for an on demand or for a snapshot and specify parameter values to use during subscription processing.

- **On demand report.** For an on demand report, you can specify a different parameter value than the published value for each parameter listed for the report. For example, suppose you have a Call Service report that uses a *Time Period* parameter to return customer service requests for the current day, week, or month. If the default parameter value for the report is set to **today**, your subscription can use a different parameter value (such as **week** or **month**) to produce a report that contains weekly or monthly figures.
- **Snapshot.** For a snapshot, your subscription must use the parameter values defined for the snapshot. Your subscription cannot override a parameter value that is defined for a snapshot. For example, suppose you are subscribing to a Western regional sales report that runs as a report snapshot, and the snapshot specifies **Western** as a regional parameter value. In this case, if you create a subscription to this report, you must use the parameter value **Western** in your subscription. To provide a visual indication that the parameter is ignored, the parameter fields on the subscription page are set to read-only fields.

Report execution options can affect how parameters are processed. Parameterized reports that run as report snapshots use the parameter values defined for the report snapshot. Parameter values are defined in the parameter properties page of the report. A report that runs as a snapshot cannot use parameters that are derived from a query unless the query includes default values for the parameters.

If a parameter value changes in the report snapshot after the subscription is defined, the report server deactivates the subscription. Deactivating the subscription indicates that the report has been modified. To activate the subscription, open and then save the subscription.

NOTE

Data-driven subscriptions can use parameter values that are obtained from a subscriber data source. For more information, see [Use an External Data Source for Subscriber Data \(Data-Driven Subscription\)](#).

For more information, see [Subscriptions and Delivery \(Reporting Services\)](#).

Parameters and Securing Data

Use caution when distributing parameterized reports that contain confidential or sensitive information. A user can easily replace a report parameter with a different value, resulting in information disclosure that you did not intend.

A secure alternative to using parameters for employee or personal data is to select data based on expressions that include the **UserID** field from the Users collection. The Users collection provides a way to get the identity of the user running the report, and use that identity to retrieve user-specific data.

IMPORTANT

In any report that includes a parameter of type **String**, be sure to use an available values list (also known as a valid values list) and ensure that any user running the report has only the permissions necessary to view the data in the report. When you define a parameter of type **String**, the user is presented with a text box that can take any value. An available values list limits the values that can be entered. If the report parameter is tied to a dataset parameter and you do not use an available values list, it is possible for a report user to type SQL syntax into the text box, potentially opening the report and your server to a SQL injection attack. If the user has sufficient permissions to execute the new SQL statement, it may produce unwanted results on the server.

If a report parameter is not tied to a dataset parameter and the parameter values are included in the report, it is possible for a report user to type expression syntax or a URL into the parameter value, and render the report to Excel or HTML. If another user then views the report and clicks the rendered parameter contents, the user may inadvertently execute the malicious script or link.

To mitigate the risk of inadvertently running malicious scripts, open rendered reports only from trusted sources. For more information about securing reports, see [Secure Reports and Resources](#).

How-To Topics

This section lists procedures that show you, step by step, how to work with parameters and filters.

- [Add, Change, or Delete a Report Parameter \(Report Builder and SSRS\)](#)
- [Add, Change, or Delete Available Values for a Report Parameter \(Report Builder and SSRS\)](#)
- [Add, Change, or Delete Default Values for a Report Parameter \(Report Builder and SSRS\)](#)
- [Change the Order of a Report Parameter \(Report Builder and SSRS\)](#)
- [Add Cascading Parameters to a Report \(Report Builder and SSRS\)](#)
- [Add a Filter to a Dataset \(Report Builder and SSRS\)](#)
- [Add a Subreport and Parameters \(Report Builder and SSRS\)](#)
- [Customize the Parameters Pane in a Report \(Report Builder\)](#)

Related Sections

[Configuring SSRS Report Parameters \(quiz\)](#)

[Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)

[Report Parameters Concepts](#)

[Report Samples \(Report Builder and SSRS\)](#)

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Security \(Report Builder\)](#)

[Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#)

[Drillthrough, Drilldown, Subreports, and Nested Data Regions \(Report Builder and SSRS\)](#)

Add, Change, or Delete a Report Parameter (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

A report parameter provides a way to choose report data, connect related reports together, and vary the report presentation. You can provide a default value and a list of available values, and the user can change the selection.

After you publish a report, you can change the default values, the available values, and other properties for a report parameter on the report server. You can provide multiple sets of default parameter values by creating linked reports. For more information, see [Report Parameters \(Report Builder and Report Designer\)](#).

This article is about adding report parameters to a paginated report in Report Builder or Report Designer in SQL Server Data Tools (SSDT). You can also add report parameters to mobile reports in SQL Server Mobile Report Publisher. See [Create mobile reports with SQL Server Mobile Report Publisher](#) for more information.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add or edit a report parameter

1. In Report Builder or Report Designer in SQL Server Data Tools (SSDT), in the **Report Data** pane, right-click the **Parameters** node and click **Add Parameter**. The **Report Parameter Properties** dialog box opens.
2. In **Name**, type the name of the parameter or accept the default name.
3. In **Prompt**, type the text that appears next to the parameter text box when the user runs the report.
4. In **Data type**, select the data type for the parameter value.
5. If the parameter can contain a blank value, select **Allow blank value**.
6. If the parameter can contain a null value, select **Allow null value**.
7. To allow a user to select more than one value for the parameter, select **Allow multiple values**.
8. Set the visibility option.
 - To show the parameter on the toolbar at the top of the report, select **Visible**.
 - To hide the parameter so that it does not display on the toolbar, select **Hidden**.
 - To hide the parameter and protect it from being modified on the report server after the report is published, select **Internal**. The report parameter can then only be viewed in the report definition. For this option, you must set a default value or allow the parameter to accept a null value.
9. Click **OK**.

To delete a report parameter

1. In the **Report Data** pane, expand the **Parameters** node.
2. Right-click the report parameter and click **Delete**.

See Also

- [Add, Change, or Delete Available Values for a Report Parameter \(Report Builder and SSRS\)](#)
- [Add, Change, or Delete Default Values for a Report Parameter \(Report Builder and SSRS\)](#)
- [Change the Order of a Report Parameter \(Report Builder and SSRS\)](#)
- [Report Parameters \(Report Builder and Report Designer\)](#)
- [Add Cascading Parameters to a Report \(Report Builder and SSRS\)](#)
- [Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)
- [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)
- [Parameters Collection References \(Report Builder and SSRS\)](#)
- [Add a multi-value parameter to a Report](#)

Add, Change, or Delete Available Values for a Report Parameter

3/29/2017 • 3 min to read • [Edit Online](#)

After you create a report parameter, you can specify a list of available values to display to the user. An available values list limits the choices a user can make to only valid values for the parameter.

Available values appear in a drop-down list next to the report parameter on the toolbar when the report runs. Report parameters can represent one value or multiple values. For multiple values, the top of list begins with a **Select All** feature so the user can select or clear all values with a single click.

You can provide a static list of values or a list from a report dataset. You can optionally provide a friendly name for values by specifying a label field. For example, for a parameter based on a `ProductID` field, you can display the `ProductName` field in the parameter label. When the report runs, the user can choose from the product names, but the actual chosen value is the corresponding `ProductID`.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

After you publish a report, you can override the available values that you define in the report in the report authoring tool, by setting parameter property values on the report server. For more information, see [Report Parameters \(Report Builder and Report Designer\)](#).

To add or change the available values for a report parameter

1. In the Report Data pane, expand the Parameters node. Right-click the parameter and click **Parameter Properties**. The **Report Parameter Properties** dialog box opens.

NOTE

If the Report Data pane is not visible, click **View** and then click **Report Data**.

2. Click **Available Values**. Select an available values option:

- Click **Specify values** to manually provide a list of values, and optionally, friendly names (the labels) for the values.

Click **Add** and then enter the value in the **Value** text box, and optionally, the label in the **Label** text box. If you do not provide a label, the value is used. You can write an expression for a value. The data type must match the data type of the parameter. Field names cannot be used in an expression for a parameter. For examples, see [Commonly Used Filters \(Report Builder and SSRS\)](#).

Repeat this step for as many values as you want to provide. The order of items you see in this list determines the order that the user sees them in the drop-down list. To change the order of an item in the list, click a **Value** or **Label** text box to select the item, and then use the up and down arrow buttons to move the item higher or lower in the list.

- Click **Get values from a query** to provide the name of an existing dataset that retrieves the values, and optionally, the friendly names for this parameter.

IMPORTANT

If the same dataset contains the corresponding query parameter for the report parameter, the report will display an error message when you try to run it. You resolve this error by using a different dataset to retrieve the values.

In **Dataset**, choose the name of the dataset.

In **Value field**, choose the name of the field that provides parameter values.

In **Label field**, choose the name of the field that provides the friendly names for the parameter. If there is no separate field for friendly names, choose the same field as you chose for the **Value** field.

3. Click **OK**.

When you preview the report, you see a drop-down list of available values for the parameter.

To remove the available values for a report parameter

1. In the Report Data pane, expand the Parameters node. Right-click the parameter and click **Parameter Properties**. The **Report Parameters** dialog box opens.
2. Click **Available Values**.
3. In **Select from one of the following options**, click **None**.
4. Click **OK**.

When you preview the report, you the drop-down list of available values for the parameter no longer appears.

See Also

[Change the Order of a Report Parameter \(Report Builder and SSRS\)](#)

[Add, Change, or Delete a Report Parameter \(Report Builder and SSRS\)](#)

[Add Cascading Parameters to a Report \(Report Builder and SSRS\)](#)

[Add, Change, or Delete Default Values for a Report Parameter \(Report Builder and SSRS\)](#)

[Parameters Collection References \(Report Builder and SSRS\)](#)

[Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)

[Expressions \(Report Builder and SSRS\)](#)

Add, Change, or Delete Default Values for a Report Parameter

3/29/2017 • 2 min to read • [Edit Online](#)

After you create a report parameter, you can provide a list of default values. If all parameters have a valid default value, the report runs automatically when you first view or preview it.

Report parameters can represent one value or multiple values. For single values, you can provide a literal or expression. For multiple values, you can provide a static list or a list from a report dataset.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

After you publish a report, you can override the default values that you define in the report in the report authoring tool, by setting parameter property values on the report server. You can also provide multiple sets of default parameter values by creating linked reports. For more information, see [Report Parameters \(Report Builder and Report Designer\)](#)

To add or change the default values for a report parameter

1. In the Report Data pane, expand the **Parameters** node. Right-click the parameter and click **Edit**. The **Report Parameter Properties** dialog box opens.

NOTE

If the Report Data pane is not visible, click **View** and then click **Report Data**.

2. Click **Default Values**.

3. Select a default option:

- To manually provide a value or list of values, click **Specify values**. Click **Add** and then enter the value in the **Value** text box. You can write an expression for a value. The data type must match the data type of the parameter. Field names cannot be used in an expression for a parameter.

For multivalue parameters, repeat this step for as many values as you want to provide. The order of items you see in this list determines the order that the user sees them in the drop-down list. To change the order of an item in the list, click the **Value** text box to select the item, and then use the up and down arrow buttons to move the item higher or lower in the list.

- To provide the name of an existing dataset that retrieves the values, click **Get values from a query**. In **Dataset**, choose the name of the dataset.

In **Value field**, choose the name of the field that provides parameter values.

4. Click **OK**.

To remove the default values for a report parameter

1. In the Report Data pane, expand the **Parameters** node. Right-click the parameter and click **Edit**. The **Report Parameter Properties** dialog box opens.

2. Click **Default Values**.
3. In **Select from one of the following options**, click **No default value**.
4. Click **OK**.

See Also

- [Report Parameters \(Report Builder and Report Designer\)](#)
- [Add Cascading Parameters to a Report \(Report Builder and SSRS\)](#)
- [Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)
- [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)
- [Parameters Collection References \(Report Builder and SSRS\)](#)
- [Change the Order of a Report Parameter \(Report Builder and SSRS\)](#)
- [Add, Change, or Delete a Report Parameter \(Report Builder and SSRS\)](#)
- [Expressions \(Report Builder and SSRS\)](#)

Change the Order of a Report Parameter (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Change the order of report parameters when you have a dependent parameter that is listed before the parameter it is dependent on. Parameter order is important when you have cascading parameters, or when you want to show users the default value for one parameter before they choose values for other parameters. A dependent report parameter contains a reference, in either its default values query or valid values query, to a query parameter that points to a report parameter that is after it in the parameter list in the **Report Data** pane.

The order that you see parameters display on the report viewer toolbar when you run the report, is determined by the order of the parameters in the **Report Data** pane and the location of the parameters in the custom parameters pane. For more information, see [Customize the Parameters Pane in a Report \(Report Builder\)](#)

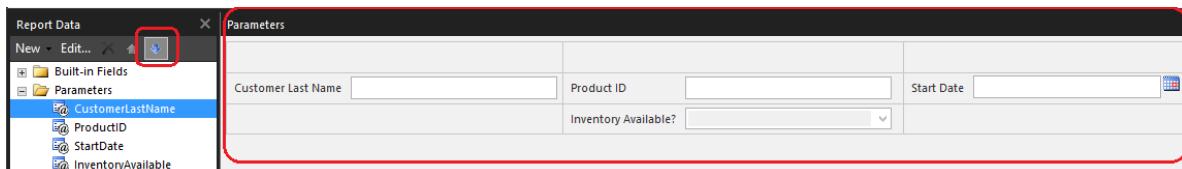
NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To change the order of report parameters

You can change the order of report parameters by doing either of the following:

- Click a parameter in the **Report Data** pane, and use the up and down arrow buttons to move the parameter higher or lower in the list, as shown in the following image. When you change the order of the parameter in the **Report Data** pane, the location of the parameter in the parameters pane is changed.



- In the parameters pane, drag the parameter to a new column or row in the pane. When you change the location of the parameter in the pane, the parameter order changes in the **Report Data** pane. For more information about moving parameters in the pane, see [Customize the Parameters Pane in a Report \(Report Builder\)](#).

See Also

[Report Parameters \(Report Builder and Report Designer\)](#)

[Report Builder Help for Dialog Boxes, Panes, and Wizards](#)

[Add Cascading Parameters to a Report \(Report Builder and SSRS\)](#)

[Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Parameters Collection References \(Report Builder and SSRS\)](#)

Add Cascading Parameters to a Report (Report Builder and SSRS)

3/24/2017 • 6 min to read • [Edit Online](#)

Cascading parameters provide a way of managing large amounts of report data. You can define a set of related parameters so that the list of values for one parameter depends on the value chosen in another parameter. For example, the first parameter is independent and might present a list of product categories. When the user selects a category, the second parameter is dependent on the value of the first parameter. Its values are updated with a list of subcategories within the chosen category. When the user views the report, the values for both the category and subcategory parameters are used to filter report data.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To create cascading parameters, you define the dataset query first and include a query parameter for each cascading parameter that you need. You must also create a separate dataset for each cascading parameter to provide available values. For more information, see [Add, Change, or Delete Available Values for a Report Parameter \(Report Builder and SSRS\)](#).

Order is important for cascading parameters because the dataset query for a parameter later in the list includes a reference to each parameter that is earlier in the list. At run time, the order of the parameters in the Report Data pane determines the order in which the parameter queries appear in the report, and therefore, the order in which a user chooses each successive parameter value.

For information about creating cascading parameters with multiple values and including the Select All feature, see [How to have a Select All Multivalue Cascading Parameter](#).

To create the main dataset with a query that includes multiple related parameters

1. In the Report Data pane, right-click a data source, and then click **Add Dataset**.
2. In **Name**, type the name of the dataset.
3. In **Data source**, choose the name of the data source or click **New** to create one.
4. In **Query type**, choose the type of query for the selected data source. In this topic, query type **Text** is assumed.
5. In **Query**, type the query to use to retrieve data for this report. The query must include the following parts:
 - a. A list of data source fields. For example, in a Transact-SQL statement, the SELECT statement specifies a list of database column names from a given table or view.
 - b. One query parameter for each cascading parameter. A query parameter limits the data retrieved from the data source by specifying certain values to include or exclude from the query. Typically, query parameters occur in a restriction clause in the query. For example, in a Transact-SQL SELECT statement, query parameters occur in the WHERE clause. For more information, see "Filtering Rows by Using WHERE and HAVING" in the Reporting Services documentation in [SQL Server Books](#)

[Online](#).

6. Click **Run (!)**. After you include query parameters and then run the query, report parameters that correspond to the query parameters are automatically created.

NOTE

The order of query parameters the first time you run a query determines the order that they are created in the report. To change the order, see [Change the Order of a Report Parameter \(Report Builder and SSRS\)](#)

7. Click **OK**.

Next, you will create a dataset that provides the values for the independent parameter.

To create a dataset to provide values for an independent parameter

1. In the Report Data pane, right-click a data source, and then click **Add Dataset**.
2. In **Name**, type the name of the dataset.
3. In **Data source**, verify the name is the name of the data source you chose in step 1.
4. In **Query type**, choose the type of query for the selected data source. In this topic, query type **Text** is assumed.
5. In **Query**, type the query to use to retrieve values for this parameter. Queries for independent parameters typically do not contain query parameters. For example, to create a query for a parameter that provides all category values, you might use a Transact-SQL statement similar to the following:

```
SELECT DISTINCT <column name> FROM <table>
```

The SELECT DISTINCT command removes duplicate values from the result set so that you get each unique value from the specified column in the specified table.

Click **Run (!)**. The result set shows the values that are available for this first parameter.

6. Click **OK**.

Next, you will set the properties of the first parameter to use this dataset to populate its available values at run-time.

To set available values for a report parameter

1. In the Report Data pane, in the Parameters folder, right-click the first parameter, and then click **Parameter Properties**.
2. In **Name**, verify that the name of the parameter is correct.
3. Click **Available Values**.
4. Click **Get values from a query**. Three fields appear.
5. In **Dataset**, from the drop-down list, click the name of the dataset you created in the previous procedure.
6. In **Value** field, click the name of the field that provides the parameter value.
7. In **Label** field, click the name of the field that provides the parameter label.
8. Click **OK**.

Next, you will create a dataset that provides the values for a dependent parameter.

To create a dataset to provide values for a dependent parameter

1. In the Report Data pane, right-click a data source, and then click **Add Dataset**.
2. In **Name**, type the name of the dataset.
3. In **Data source**, verify the name is the name of the data source you chose in step 1.
4. In **Query type**, choose the type of query for the selected data source. In this topic, query type **Text** is assumed.
5. In **Query**, type the query to use to retrieve values for this parameter. Queries for dependent parameters typically include query parameters for each parameter that this parameter is dependent on. For example, to create a query for a parameter that provides all subcategory (dependent parameter) values for a category (independent parameter), you might use a Transact-SQL statement similar to the following:

```
SELECT DISTINCT Subcategory FROM <table>
WHERE (Category = @Category)
```

In the WHERE clause, Category is the name of a field from <table> and @Category is a query parameter. This statement produces a list of subcategories for the category specified in @Category. At run time, this value will be filled in with the value that the user chooses for the report parameter that has the same name.

6. Click **OK**.

Next, you will set the properties of the second parameter to use this dataset to populate its available values at run time.

To set available values for a report parameter

1. In the Report Data pane, in the Parameters folder, right-click the first parameter, and then click **Parameter Properties**.
2. In **Name**, verify that the name of the parameter is correct.
3. Click **Available Values**.
4. Click **Get values from a query**.
5. In **Dataset**, from the drop-down list, click the name of the dataset you created in the previous procedure.
6. In **Value** field, click the name of the field that provides the parameter value.
7. In **Label** field, click the name of the field that provides the parameter label.
8. Click **OK**.

To test the cascading parameters

1. Click **Run**.
2. From the drop-down list for the first, independent parameter, choose a value.

The report processor runs the dataset query for the next parameter and passes it the value you chose for the first parameter. The drop-down list for the second parameter is populated with the available values based on the first parameter value.

3. From the drop-down list for the second, dependent parameter, choose a value.

The report does not run automatically after you choose the last parameter so that you can change your choice.

4. Click **View Report**. The report updates the display based on the parameters you have chosen.

See Also

[Add, Change, or Delete a Report Parameter \(Report Builder and SSRS\)](#)

[Report Parameters \(Report Builder and Report Designer\)](#)

[Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)

[Report Builder Tutorials](#)

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Report Embedded Datasets and Shared Datasets \(Report Builder and SSRS\)](#)

Add a multi-value parameter to a Report

3/24/2017 • 3 min to read • [Edit Online](#)

You can add a parameter to a report that allows the user to select more than one value for the parameter.

You can pass multiple parameter values to the report within the report URL. For a URL example includes a multi-value parameter, see [Pass a Report Parameter Within a URL](#).

For information on how to pass multiple parameter values to a stored procedure, see [Working With Multi-Select Parameters for SSRS Reports](#) on mssqltips.com.

To add a multi-value parameter

1. In Report Builder, open the report that you want to add the multi-value parameter to.
2. Right-click the report dataset, and then click **Dataset Properties**
3. Add a variable to the dataset query by either editing the query text in the **Query** box, or by adding a filter by using the query designer. For more information, see [Build a Query in the Relational Query Designer \(Report Builder and SSRS\)](#).

```
WHERE  
Production.ProductInventory.ProductID IN (@ProductID)
```

IMPORTANT

- The query text must not include the DECLARE statement for the query variable.
- The text for the query variable must include the **IN** operator, as shown in the example above.
- Be sure to include the parentheses around the variable as shown above. Otherwise, the report fails to render and the “must declare the scalar variable” error is displayed.

A dataset parameter for an embedded dataset or a shared dataset is created automatically for the query variable. A report parameter is created automatically for the dataset parameter.

4. In the **Report Data** pane, expand the **Parameters** node, right-click the report parameter that was automatically created for the dataset parameter, and then click **Parameter Properties**.
5. In the **General** tab, select **Allow multiple values** to allow a user to select more than one value for the parameter.
6. (Optional) In the **Available** values tab, specify a list of available values to display to the user.

An available values list limits the choices a user can make to only valid values for the parameter. For multiple values, the top of list begins with a **Select All** feature so the user can select or clear all values with a single click. If you choose to get the available values for the report parameter from a dataset query, be sure to select a dataset that does not contain the query variable that is associated with the same report parameter.

For more information, see [Add, Change, or Delete Available Values for a Report Parameter \(Report Builder and SSRS\)](#).

To add a multi-value parameter

1. In Report Builder, open the report that you want to add the multi-value parameter to.
2. Right-click the report dataset, and then click **Dataset Properties**
3. Add a variable to the dataset query by either editing the query text in the **Query** box, or by adding a filter by using the query designer. For more information, see [Build a Query in the Relational Query Designer \(Report Builder and SSRS\)](#).

```
WHERE  
Production.ProductInventory.ProductID IN (@ProductID)
```

IMPORTANT

- The query text must not include the DECLARE statement for the query variable.
- The text for the query variable must include the **IN** operator, as shown in the example above.
- Be sure to include the parentheses around the variable as shown above. Otherwise, the report fails to render and the “must declare the scalar variable” error is displayed

A dataset parameter for an embedded dataset or a shared dataset is created automatically for the query variable. A report parameter is created automatically for the dataset parameter.

4. In the **Report Data** pane, expand the **Parameters** node, right-click the report parameter that was automatically created for the dataset parameter, and then click **Parameter Properties**.
5. In the **General** tab, select **Allow multiple values** to allow a user to select more than one value for the parameter.
6. (Optional) In the **Available** values tab, specify a list of available values to display to the user.

An available values list limits the choices a user can make to only valid values for the parameter. For multiple values, the top of list begins with a **Select All** feature so the user can select or clear all values with a single click. If you choose to get the available values for the report parameter from a dataset query, be sure to select a dataset that does not contain the query variable that is associated with the same report parameter.

For more information, see [Add, Change, or Delete Available Values for a Report Parameter \(Report Builder and SSRS\)](#).

See Also

[Add Cascading Parameters to a Report \(Report Builder and SSRS\)](#)

[Add, Change, or Delete a Report Parameter \(Report Builder and SSRS\)](#)

Set Parameters on a Published Report - SharePoint Integrated Mode

3/29/2017 • 4 min to read • [Edit Online](#)

A parameterized report is a report that accepts input values that are used to filter data when you run the report. Parameters are defined when the report is created. Depending on how a report parameter is defined in the report definition, it can accept a single value, multiple values, or dynamic values, which change in response to a previous selection (for example, when you select product category, your next selection might be a specific product from that category). A parameter can have a default value, which can be used to run a filtered version of the report automatically or possibly be replaced with a different value.

These properties can be set in the report definition, or after the report is published. Although you can modify some parameter properties in a published to change the value and display properties, you cannot change a parameter name or the data type. The parameter name and data type can only be changed by the report author in the report definition.

To run a parameterized report, you typically must know which values to type, although a report might include drop-down lists of valid values from which to choose.

To set parameter properties on a published report, you must have Edit Items permission for the report. You can modify some or all of the parameter properties on a report that you run from a SharePoint site. You cannot personalize a report by saving a combination of parameter values that you want to use repeatedly. Any default values that you specify are used by all users who open the report.

If the report is embedded in a Report Viewer Web part that is configured to always show that report, set the properties on the Report Viewer Web Part. For more information, see [Add the Report Viewer Web Part to a Web Page \(Reporting Services in SharePoint Integrated Mode\)](#).

To run a parameterized report

1. Open the library that contains the report.
2. Click the report name. You must know in advance which reports have parameters. There is no visual identifier on the report to indicate that it is parameterized.
3. When the report opens, specify the parameters you want to use. The Parameters area might be visible, collapsed, or hidden depending on how properties are set on the Report Viewer Web Part.
 - a. If the Parameters area is visible, choose a value for each parameter.
 - b. If there is a thin strip of color that runs down the length of the report that has an arrow in the middle of it, the Parameters area is collapsed. You can click the arrow to expand it.
 - c. If the Parameters area is hidden, you cannot specify a value.
4. Click **Apply** at the bottom of the Parameters area to run the report.

It is possible to specify a combination of values that do not give you the results you expect. The report might need to be modified by the report author if you are not getting the information you require.

5. Click **OK**.

To set parameter properties

1. Open the library or folder that contains the report.

2. Point to the report and click the down arrow.
3. Click **Manage Parameters**. If the report contains parameters, each parameter will be listed on the page. The list shows the parameter name, data type, data value that is used by default, and whether it is visible in the parameter area when you open the report.
4. Click a parameter in the list to modify its settings.
5. In Default Value, enter a value for the parameter. The value will not be validated, so be sure that you are providing a value that works for the report.
 - a. Choose **Use value expression specified in report definition** if you want to use the default value that was defined when the report was created. If the report definition does not provide a default value, this option will be unavailable.
 - b. Choose **Override the report default value** if you want to specify a replacement for the report definition default value. Optionally, for report parameters that accept null values, you can select the **Null** check box to prevent filtering based on that parameter.
 - c. Choose **Parameter does not have a default value** if you want each user to specify a value before the report is processed. If you select this option, you should set display settings that prompt the user to specify a value.
- Note that if all parameter values have a default value, the report will automatically run with those values when the user opens the report. However, if the parameter area is displayed, users can override the default value and re-run the report.
6. Set display options to determine whether the parameter is visible.
 - a. Choose **Prompt User** to show the parameter on the page. You can specify prompt text that appears within a field to provide a brief statement about the format or type of data that the user must provide.
 - b. Choose **Hidden** if you are using a default value and you do not want the parameter to be visible in the Parameters area.
 - c. Choose **Internal** if you are using a default value and you do not want the parameter to be visible in the Parameters area or on subscription pages.
7. Click **Apply**.

See Also

[SharePoint Site and List Permission Reference for Report Server Items](#)

Customize the Parameters Pane in a Report (Report Builder)

3/24/2017 • 1 min to read • [Edit Online](#)

When creating paginated reports with parameters in Report Builder, you can customize the Parameters pane. In report design view, you can drag a parameter to a specific column and row in the Parameters pane. You can add and remove columns to change the layout of the pane.

When you drag a parameter to a new column and row in the pane, the parameter order changes in the **Report Data** pane. When you change the order of the parameter in the **Report Data** pane, the location of the parameter in the pane is changed. For more information about why parameter order is important, see [Change the Order of a Report Parameter \(Report Builder and SSRS\)](#).

To customize the parameters pane

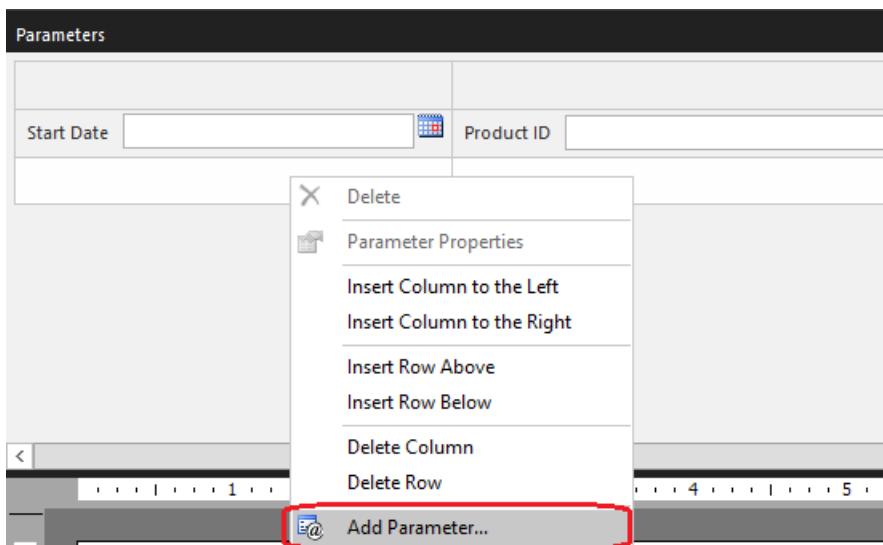
1. On the **View** tab, select the **Parameters** checkbox to display the parameters pane.



The pane appears at the top of the design surface.

2. To add a parameter to the pane, do one of the following.

- Right click an empty cell in the parameters pane, and then click **Add Parameter**.



- Right click **Parameters** in the **Report Data** pane, and then click **Add Parameter**.

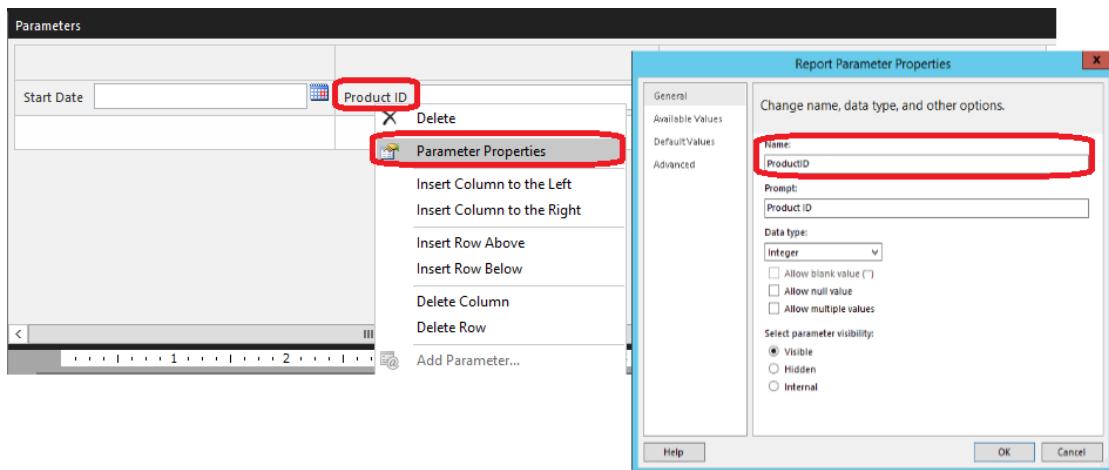
3. To move a parameter to a new location in the parameters pane, drag the parameter to a different cell in the pane.

When you change the location of the parameter in the pane, the order of the parameter in the **Parameters** list in the **Report Data** pane is automatically changed. For more information about the impact of the

parameter order, see [Change the Order of a Report Parameter \(Report Builder and SSRS\)](#)

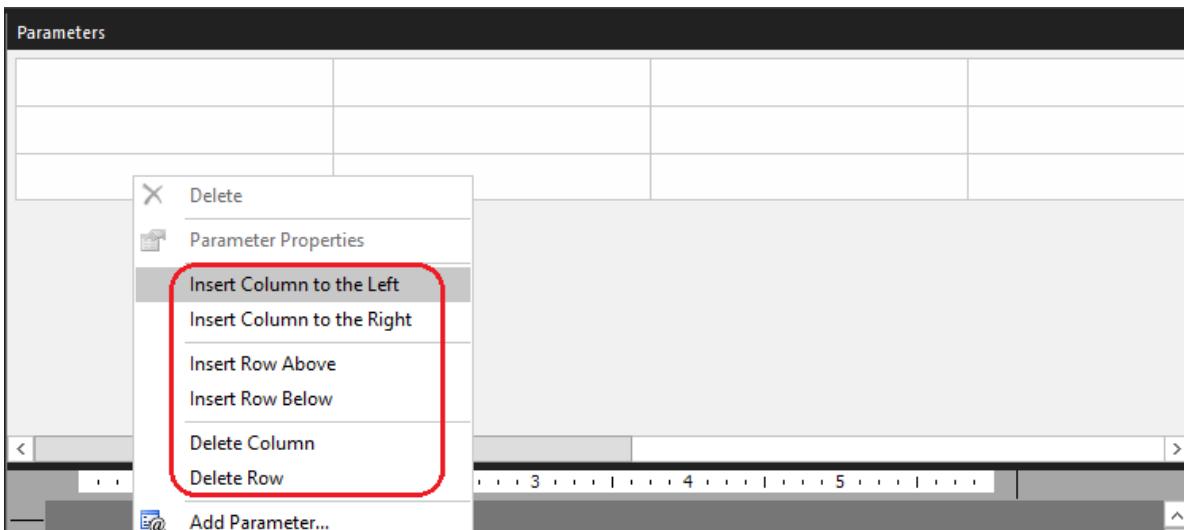
4. To access the properties for a parameter, do one of the following.

- Right click the parameter in the parameters pane, and then click **Parameter Properties**.



- Right click the parameter in the **Report Data** pane, and then click **Parameter Properties**.

5. To add new columns and rows to the pane, or delete existing rows and columns, right click anywhere in the parameters pane and then click a command on the menu that displays.

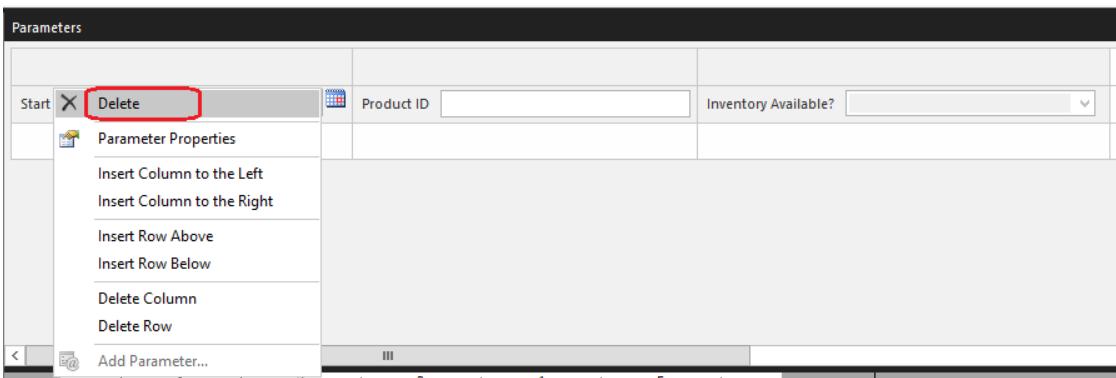


IMPORTANT

When you delete a column or row that contains parameters, the parameters are deleted from the report.

6. To delete a parameter from the pane and from the report, do one of the following.

- Right click the parameter in the parameters pane, and then click **Delete**.



- Right click parameter in the **Report Data** pane, and then click **Delete**.

See Also

[Report Parameters \(Report Builder and Report Designer\)](#)

Filter, Group, and Sort Data (Report Builder and SSRS)

3/24/2017 • 13 min to read • [Edit Online](#)

In a report, expressions are used to help control, organize, and sort report data. By default, as you create datasets and design the report layout, properties of report items are set automatically to expressions based on the dataset fields, parameters, and other items that appear in the Report Data pane. You can also add an interactive sort button to a table or matrix cell to enable a user to interactively change the row sort order for groups or rows within groups.

- **Filter expressions** A filter expression tests data for inclusion or exclusion based on a comparison that you specify. Filters are applied to data in a report after the data is retrieved from a data connection. You can add any combination of filters to the following items: a shared dataset definition on the report server; a shared dataset instance or embedded dataset in a report; a data region such as a table or a chart; or a data region group, such as a row group in a table or a category group in a chart.
- **Group expressions** A group expression organizes data based on a dataset field or other value. Group expressions are created automatically as you build the report layout. The report processor evaluates group expressions after filters are applied to the data, and as report data and data regions are combined. You can customize a group expression after it is created.
- **Sort expressions** A sort expression controls the order in which data appears in a data region. Sort expressions are created automatically as you build the report layout. By default, a sort expression for a group is set to the same value as the group expression. You can customize a sort expression after it is created.
- **Interactive sort** To enable a user to sort or reverse the sort order of a column, you can add an interactive sort button to a column header or group header cell in a table or matrix.

To help your users customize filter, group, or sort expressions, you can change an expression to add a reference to a report parameter. For more information, see [Report Parameters \(Report Builder and Report Designer\)](#).

For more information and examples, see the following topics:

- [Group Expression Examples \(Report Builder and SSRS\)](#)
- [Filter Equation Examples \(Report Builder and SSRS\)](#)
- [Report Builder Tutorials](#)
- [Reporting Services Tutorials \(SSRS\)](#)
- [Report Samples \(Report Builder and SSRS\)](#)

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Filtering Data in the Report

Filters are parts of a report that help control report data after it is retrieved from the data connection. Use filters when you cannot change a dataset query to filter data before it is retrieved from an external data source.

When it is possible, build dataset queries that return only the data that you need to display in the report. When you reduce the amount of the data that must be retrieved and processed, you are helping to improve report performance. For more information, see [Report Embedded Datasets and Shared Datasets \(Report Builder and SSRS\)](#).

After the data is retrieved from the external data source, you can add filters to datasets, data regions, and data region groups, including detail groups. Filters are applied at run time first on the dataset, and then on the data region, and then on the group, in top-down order for group hierarchies. In a table, matrix, or list, filters for row groups, column groups, and adjacent groups are applied independently. In a chart, filters for category groups and series groups are applied independently. For more information, see [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#).

For each filter, you specify a *filter equation*. A filter equation includes a dataset field or expression that specifies the data to filter, an operator, and a value to compare. Only those data values that match the filter condition are included when the item is processed.

To enable your users to help control the data in a report, you can include parameters in filter expressions. For more information, see [Parameters Collection References \(Report Builder and SSRS\)](#).

To customize a view for each user, you can include a reference to the built-in field UserID in a filter. For more information, see [Built-in Globals and Users References \(Report Builder and SSRS\)](#).

Grouping Data in the Report

Groups organize data in a report for display or for calculating aggregate values. Understanding how to define groups and use group features helps you to design reports that are more concise.

Group expressions are created automatically when you do the following:

- Arrange dataset fields in a Table, Matrix, Chart wizard or match fields in the Map wizard.
- In a table, matrix, or list, add a field to the Row Groups or Column Groups area in the Grouping pane.
- In a chart, add a field to the Category Groups or Series Groups area in the Chart data pane.
- In a map, specify a field to match map elements with analytical data in the Layer Data context menu item.

A group is a part of the report definition. Each group has a name. By default, the group name is the dataset field that it is based on.

In a table or matrix data region, you can create multiple row groups and column groups. You can display your data in a visual hierarchy by organizing nested groups, adjacent groups, and recursive hierarchy groups (such as an organizational chart).

The group name identifies an expression scope. You can specify the name of a group as a scope in which to calculate aggregates, to organize data hierarchically and toggle the display of child nodes from parent nodes in a drilldown report, to display different views of the same data on multiple data regions, and to visualize summary data in a table, matrix, chart, gauge, or map. For more information, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

To group on several dataset fields, add each field to the set of group expressions. You can also write your own group expressions in Microsoft Visual Basic. For example, you can group by a range of values, or by using a report parameter to enable your user to select how to group data in a data region. For

more information, see [Group Expression Examples \(Report Builder and SSRS\)](#).

For report presentation, you can add page breaks before and after each group, or each instance of a group, to reduce the amount of data on each page and help you manage report rendering performance. For more information, see [Add a Page Break \(Report Builder and SSRS\)](#).

Creating data region groups is one way to organize data in a report. There are several other ways to organize data, each with its own advantages. For more information, see [Drillthrough, Drilldown, Subreports, and Nested Data Regions \(Report Builder and SSRS\)](#).

Defining Group Variables

When you define a group, you can create a group variable to use in expressions that are scoped to the group and accessed from nested groups. A group variable is calculated once per group instance and can be accessed from expressions in child groups. For example, for data that is grouped by region and subregion, you can calculate a tax for each region and use that tax in calculations from the subregion group.

For more information, see [Report and Group Variables Collections References \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Groups and Scope in Data Regions

To provide multiple views of data from the same dataset, you can specify the same group expressions for each data region. For example, you can display categorized data in a table to show all detail data and in a pie chart to show aggregates and to help visualize each category in relation to the entire dataset. For more information, see [Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#).

When you nest a data region in a cell in a table, matrix, or list, you are automatically scoping the data to the innermost group memberships of the cell. For example, assume that you add a chart to a cell that is in both a row group and a column group. The data available to that chart is scoped to the innermost row group instance and innermost column group instance at run time. For more information, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Sorting Data in the Report

To control the sort order of data in your report, you can sort data in a dataset query, or define a sort expression for a data region or group. You can also add interactive sort buttons to tables and matrices to enable a user to change the sort order for rows.

All three types of sorts can be combined in the same report. By default, sort order is determined by the order in which data is returned by the dataset query. Sort expressions are applied in the data region and data region group. Interactive sorts are applied after sort expressions.

For expressions that contain aggregate functions, most results are not affected by sort order. Return values for the following aggregate functions are affected by sort order: First, Last, and Previous. For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

Sorting Data in a Dataset Query

Include sort order in the dataset query to pre-sort data before it is retrieved for a report. By sorting data in the query, the sorting work is done by the data source instead of by the report processor.

For a Microsoft SQL Server data source type, you can add an ORDER BY clause to the dataset query. For example, the following Transact-SQL query sorts the columns Sales and Region by Sales in descending order from the table SalesOrders: `SELECT Sales, Region FROM SalesOrders ORDER BY Sales DESC`. For more information, see "Sorting Rows with ORDER BY" in [SQL Server Books Online](#).

NOTE

Not all data sources support the ability to specify sort order in the query.

Sorting Data with Sort Expressions

To sort data in the report after it is retrieved from the data source, you can set sort expressions on a Tablix data region or a group, including the details group. The following list describes the effect of setting sort expressions on different items:

- **Tablix data region.** Set sort expressions on a table, matrix, or list data region to control the sort order of data in the data region, after dataset filters and data region filters are applied at run time.
- **Tablix data region group.** Set sort expressions for each group, including the details group, to control the sort order of group instances. For example, for the details group, you control the order of the detail rows. For a child group, you control the order of group instances for the child group within the parent group. By default, when you create a group, the sort expression is set to the group expression and to ascending order.

If you have only one details group, you can define a sort expression in the query, on the data region, or on the details group to the same effect.

- **Chart data region.** Set a sort expression for the category and series groups to control the sort order for data points. By default, the order of data points is also the order of the colors in the chart legend. For more information, see [Formatting Series Colors on a Chart \(Report Builder and SSRS\)](#).
- **Map report item.** You do not typically need to sort data for a map data region because the map groups data to display on map elements.
- **Gauge data region.** You do not typically need to sort data for a gauge data region because the gauge displays a single value relative to a range. If you do need sort data in a gauge, you must first define a group, and then set a sort expression for the group.

Sorting by a Different Value

You might want to sort the rows in a data region by a value other than the field value. For example, suppose that the field Size contains text values that correspond to small, medium, large, and extra large. By default, the sort expression for a row group based on Size is also [Size]. To have more control over the way that data is sorted, you can add a field to the dataset query that defines the sort order that you want.

Alternatively, you can define a dataset that includes only the sizes and a value that specifies the order that you want. You can change the sort expression to use the Lookup function for the sort order value.

For example, assume that the following Transact-SQL query defines a dataset named Sizes. The query uses a CASE statement to define a sort order value SizeSortOrder for each value of Size:

```
SELECT Size,
CASE Size
    WHEN 'S' THEN 1
    WHEN 'M' THEN 2
    WHEN 'L' THEN 3
    WHEN 'XL' THEN 4
    ELSE 0
END as SizeSortOrder
FROM Production.Product
```

In a table that has a row group based on `[Size]`, you can change the group sort expression to use a Lookup function to find the numeric field that corresponds to the size value. The expression would be similar to this:

```
=Lookup(Fields!Size.Value, Fields!Size.Value, Fields!SizeSortOrder.Value, "Sizes")
```

For more information, see [Sort Data in a Data Region \(Report Builder and SSRS\)](#) and [Lookup Function \(Report Builder and SSRS\)](#).

Adding Interactive Sorting for the User

To enable a user to change the sort order of report data in a table or matrix, you can add interactive sort buttons to column headers or group headers. Users can click the button to toggle the sort order. Interactive sort is supported in rendering formats that allow user interaction, such as HTML.

You add interactive sort buttons to a text box in a tablix data region cell. By default, every cell contains a text box. In the text box properties, you specify which part of a table or matrix data region to sort (the parent group values, the child group values, or the detail rows), what to sort by, and whether to apply the sort expression to other report items that have a peer relationship. For example, if a table and a chart that provide views on the same dataset are contained in a rectangle, they are peer data regions. When a user toggles the sort order in the table, the sort order for the chart also toggles. For more information, see [Interactive Sort \(Report Builder and SSRS\)](#).

How-To Topics

[Keep Headers Visible When Scrolling Through a Report \(Report Builder and SSRS\)](#)

[Display Headers and Footers with a Group \(Report Builder and SSRS\)](#)

[Add Interactive Sort to a Table or Matrix \(Report Builder and SSRS\)](#)

[Set a No Data Message for a Data Region \(Report Builder and SSRS\)](#)

[Create a Recursive Hierarchy Group \(Report Builder and SSRS\)](#)

[Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#)

[Display Headers and Footers with a Group \(Report Builder and SSRS\)](#)

[Add or Delete a Group in a Chart \(Report Builder and SSRS\)](#)

[Add a Total to a Group or Tablix Data Region \(Report Builder and SSRS\)](#)

In This Section

[Group Expression Examples \(Report Builder and SSRS\)](#)

[Filter Equation Examples \(Report Builder and SSRS\)](#)

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

Related Sections

[Understanding Groups \(Report Builder and SSRS\)](#)

[Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

[Report and Group Variables Collections References \(Report Builder and SSRS\)](#)

[Displaying a Series with Multiple Data Ranges on a Chart \(Report Builder and SSRS\)](#)

[Linking Multiple Data Regions to the Same Dataset \(Report Builder and SSRS\)](#)

See Also

- [Expressions \(Report Builder and SSRS\)](#)
- [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)
- [Charts \(Report Builder and SSRS\)](#)
- [Maps \(Report Builder and SSRS\)](#)
- [Sparklines and Data Bars \(Report Builder and SSRS\)](#)
- [Gauges \(Report Builder and SSRS\)](#)
- [Indicators \(Report Builder and SSRS\)](#)

Group Expression Examples (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In a data region, you can group data by a single field, or create more complex expressions that identify the data on which to group. Complex expressions include references to multiple fields or parameters, conditional statements, or custom code. When you define a group for a data region, you add these expressions to the **Group** properties. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#).

To merge two or more groups that are based on simple field expressions, add each field to the group expressions list in the group definition.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Examples of Group Expressions

The following table provides examples of group expressions that you can use to define a group.

DESCRIPTION	EXPRESSION
Group by the <code>Region</code> field.	<code>=Fields!Region.Value</code>
Group by last name and first name.	<code>=Fields!LastName.Value</code> <code>=Fields!FirstName.Value</code>
Group by the first letter of the last name.	<code>=Fields!LastName.Value.Substring(0,1)</code>
Group by parameter, based on user selection. In this example, the parameter <code>GroupBy</code> must be based on an available values list that provides a valid choice to group on.	<code>=Fields(Parameters!GroupBy.Value).Value</code>
Group by three separate age ranges: "Under 21", "Between 21 and 50", and "Over 50".	<code>=IIF(First(Fields!Age.Value)<21,"Under 21", (IIF(First(Fields!Age.Value)>=21 AND First(Fields!Age.Value)<=50,"Between 21 and 50","Over 50")))</code>

DESCRIPTION	EXPRESSION
Group by many age ranges. This example shows custom code, written in Visual Basic .NET, that returns a string for the following ranges:	<pre>=Code.GetRangeValueByAge(Fields!Age.Value)</pre>
25 or Under	Custom code:
26 to 50	<pre>Function GetRangeValueByAge(ByVal age As Integer) As String</pre>
51 to 75	<pre>Select Case age</pre>
Over 75	<pre>Case 0 To 25</pre>
	<pre>GetRangeValueByAge = "25 or Under"</pre>
	<pre>Case 26 To 50</pre>
	<pre>GetRangeValueByAge = "26 to 50"</pre>
	<pre>Case 51 to 75</pre>
	<pre>GetRangeValueByAge = "51 to 75"</pre>
	<pre>Case Else</pre>
	<pre>GetRangeValueByAge = "Over 75"</pre>
	<pre>End Select</pre>
	<pre>Return GetRangeValueByAge</pre>
	<pre>End Function</pre>

See Also

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Custom Code and Assembly References in Expressions in Report Designer \(SSRS\)](#)

Filter Equation Examples (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

To create a filter, you must specify one or more filter equations. A filter equation includes an expression, a data type, an operator, and a value. This topic provides examples of commonly used filters.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Filter Examples

The following table shows examples of filter equations that use different data types and different operators. The scope for the comparison is determined by report item for which a filter is defined. For example, for a filter defined on a dataset, **TOP % 10** is the top 10 percent of values in the dataset; for a filter defined on a group, **TOP % 10** is the top 10 percent of values in the group.

SIMPLE EXPRESSION	DATA TYPE	OPERATOR	VALUE	DESCRIPTION
[SUM(Quantity)]	Integer	>	7	Includes data values that are greater than 7.
[SUM(Quantity)]	Integer	TOP N	10	Includes the top 10 data values.
[SUM(Quantity)]	Integer	TOP %	20	Includes the top 20% of data values.
[Sales]	Text	>	=CDec(100)	Includes all values of type System.Decimal (SQL "money" data types) greater than \$100.
[OrderDate]	DateTime	>	2008-01-01	Includes all dates from January 1, 2008 to the present date.
[OrderDate]	DateTime	BETWEEN	2008-01-01 2008-02-01	Includes dates from January 1, 2008 up to and including February 1, 2008.
[Territory]	Text	LIKE	*east	All territory names that end in "east".

Simple Expression	Data Type	Operator	Value	Description
[Territory]	Text	LIKE	%o%th*	All territory names that include North and South at the beginning of the name.
=LEFT(Fields!Subcat.Va Text1)	Text	IN	B, C, T	All subcategory values that begin with the letters B, C, or T.

See Also

[Report Parameters \(Report Builder and Report Designer\)](#)

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Add Dataset Filters, Data Region Filters, and Group Filters

3/29/2017 • 6 min to read • [Edit Online](#)

In a report, a filter is a part of a dataset, a data region, or a data region group that you create to limit the data that is used in the report. Filters are a way to help you control report data if you cannot change the dataset query, for example, if you are using a shared dataset.

Filters help you control which data is displayed and processed in a report. You can specify filters for a dataset, a data region, or a group, in any combination.

For more information, see [Add a Filter to a Dataset \(Report Builder and SSRS\)](#) and [Filter Equation Examples \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Choosing When to Set a Filter

Specify filters for report items when you cannot filter data at the source. For example, use report filters when the data source does not support query parameters, or you must run stored procedures and cannot modify the query, or a parameterized report snapshot displays customized data for different users.

You can filter report data before or after it is retrieved for a report dataset. To filter data before it is retrieved, change the query for each dataset. When you filter data in the query, you filter data at the data source, which reduces the amount data that must be retrieved and processed in a report. To filter data after it is retrieved, create filter expressions in the report. You can set filter expressions for a dataset, a data region, or a group, including detail groups. You can also include parameters in filter expressions, providing a way to filter data for specific values or for specific users, for example, filtering on a value that identifies the user viewing the report.

Choosing Where to Set a Filter

Determine where you want to set a filter by the effect you want to achieve in your report. At run time, the report processor applies filters in the following order: on the dataset, and then on the data region, and then on groups from the top down in each group hierarchy. On a table, matrix, and list, filters for row groups, column groups, and adjacent groups are applied independently. On a chart, filters for category groups and series groups are applied independently. When the report processor applies the filter, all filter equations are applied in the order they are defined on the **Filter** page of the **Properties** dialog box for each report item, which is the equivalent of combining them with Boolean AND operations.

The following list compares the effect of setting filters on different report items:

- **On the dataset** Set a filter on the dataset when you want one or more data regions that are bound to a single dataset to be filtered in the same way. For example, set the filter on the dataset that is bound to both a table that displays sales data and a chart that displays the same data.
- **On the data region** Set a filter on the data region when you want one or more data regions that are bound to a single dataset to provide a different view of the dataset. For example, set the filter on one

Table data region to display the top ten stores for sales and a different Table data region to display the bottom ten stores for sales in the same report.

- **On the row or column groups in a Tablix data region** Set a filter on a group when you want to include or exclude certain values for a group expression to control which group values appear in the table, matrix, or list.
- **On the details group in a Tablix data region** Set a filter on the details group when you have multiple detail groups for a data region and want each detail group to display a different set of data from the dataset.
- **On the series or category groups in a Chart data region** Set a filter on a series or category group when you want to include or exclude certain values for a group expression to control which values appear in the chart.

[Back to Top](#)

Understanding a Filter Equation

At run time, the report processor converts the value to the specified data type, and then uses the specified operator to compare the expression and value. The following list describes each part of the filter equation:

- **Expression** Defines what you are filtering on. Typically, this is a dataset field.
- **Data Type** Specifies the data type to use when the filter equation is evaluated at run time by the report processor. The data type you select must be one of the data types supported by the report definition schema.
- **Operator** Defines how to compare the two parts of the filter equation.
- **Value** Defines the expression to use in the comparison.

The following sections describe each part of the filter equation.

Expression

When the filter equation is evaluated by the report processor at run time, the data types for the expression and the value must be the same. The data type of the field you select for **Expression** is determined by the data processing extension or data provider that is used to retrieve data from the data source. The data type of the expression that you enter for **Value** is determined by Reporting Services defaults. The choices for data type are determined by the data types supported for a report definition. Values from the database might be converted by the data provider to a CLR type.

Data Type

For the report processor to compare two values, the data types must be the same. The following table lists the mapping between CLR data types and report definition data types. Data that you retrieve from a data source might be converted to a data type that is different by the time it is report data.

REPORT DEFINITION SCHEMA DATA TYPE	CLR TYPE(S)
Boolean	Boolean
DateTime	DateTime, DateTimeOffset
Integer	Int16, Int32, UInt16, Byte, SByte
Float	Single, Double, Decimal

REPORT DEFINITION SCHEMA DATA TYPE	CLR TYPE(S)
Text	String, Char, GUID, Timespan

In cases where you must specify a data type, you can specify your own conversion in the Value part of the expression.

Operator

The following table lists the operators that you can use in a filter equation, and what the report processor uses to evaluate the filter equation.

OPERATOR	ACTION
Equal, Like, NotEqual, GreaterThan, GreaterThanOrEqual, LessThan, LessThanOrEqual	Compares the expression to one value.
TopN, BottomN	Compares the expression to one Integer value.
TopPercent, BottomPercent	Compares the expression to one Integer or Float value.
Between	Tests whether the expression is between two values, inclusive.
In	Tests whether the expression is contained in a set of values.

Value

The Value expression specifies the final part of the filter equation. The report processor converts the evaluated expression to the data type that you specified, and then evaluates the entire filter equation to determine if the data specified in Expression passes through the filter.

To convert to a data type that is not a standard CLR data type, you must modify the expression to explicitly convert to a data type. You can use the conversion functions listed in the **Expression** dialog box under **Common Functions, Conversion**. For example, for a field `ListPrice` that represents data that is stored as a **money** data type on a SQL Server data source, the data processing extension returns the field value as a **Decimal** data type. To set a filter to use only values greater than **\$50000.00** in the report currency, convert the value to Decimal by using the expression `=CDec(50000.00)`.

This value can also include a parameter reference to allow a user to interactively select a value on which to filter.

[Back to Top](#)

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Report Parameters \(Report Builder and Report Designer\)](#)

Add a Filter (Report Builder and SSRS)

3/24/2017 • 5 min to read • [Edit Online](#)

Add a filter to a dataset, data region, or group when you want to include or exclude specific values for calculations or display. Filters are applied at run time first on the dataset, and then on the data region, and then on the group, in top-down order for group hierarchies. In a table, matrix, or list, filters for row groups, column groups, and adjacent groups are applied independently. In a chart, filters for category groups and series groups are applied independently.

To add a filter, you must specify one or more filter equations. A filter equation consists of an expression that identifies the data that you want to filter, an operator, and the value to compare to. The data types of the filtered data and the value must match. Filtering on aggregate values for a dataset or data region is not supported.

To filter data points in a chart, you can set a filter on a category group or a series group. By default, the chart uses the built-in function Sum to aggregate values that belong to the same group into an individual data point in the series. If you change the aggregate function of a series, you must change the aggregate function in the filter expression.

For more information about filtering embedded and shared datasets, see [Add a Filter to a Dataset \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To set a filter on a data region

1. Open a report in **Design** view.
2. Select the data region on the design surface, and then right-click **<data region>Properties**. For a gauge, select **Gauge Panel Properties**. The **<data region>Properties** dialog box opens.

NOTE

On a Tablix data region, right-click the corner cell or a row or column handle, and then click **Tablix Properties**.

3. Click **Filters**. This displays the current list of filter equations. By default, the list is empty.
4. Click **Add**. A new blank filter equation appears.
5. In **Expression**, type or select the expression for the field to filter. To edit the expression, click the expression (fx) button.
6. From the drop-down box, select the data type that matches the type of data in the expression you created in step 5.
7. In the **Operator** box, select the operator that you want the filter to use to compare the values in the **Expression** box and the **Value** box. The operator you choose determines the number of values that are used from the next step.
8. In the **Value** box, type the expression or value against which you want the filter to evaluate the value in **Expression**.

For examples of filter equations, see [Filter Equation Examples \(Report Builder and SSRS\)](#).

9. Click **OK**.

To set a filter on a Tablix row or column group

1. Open a report in **Design** view.
2. Right-click the table, matrix, or list data region on the design surface to select it. The Grouping pane displays the groups for the selected item.
3. In the Grouping pane, right-click the group, and then click **Edit Group**. The **Tablix Group** dialog box opens.
4. Click **Filters**. This displays the current list of filter equations. By default, the list is empty.
5. Click **Add**. A new blank filter equation appears.
6. In **Expression**, type or select the expression for the field to filter. To edit the expression, click the expression (fx) button.
7. From the drop-down box, select the data type that matches the type of data in the expression you created in step 5.
8. In the **Operator** box, select the operator that you want the filter to use to compare the values in the **Expression** box and the **Value** box. The operator you choose determines the number of values that are used from the next step.
9. In the **Value** box, type the expression or value against which you want the filter to evaluate the value in **Expression**.

For examples of filter equations, see [Filter Equation Examples \(Report Builder and SSRS\)](#).

10. Click **OK**.

To set a filter on a Chart category group

1. Open a report in **Design** view.
2. On the design surface, click the chart twice to bring up data, series and category field drop zones.
3. Right-click on a field contained in the category field drop zone and select **Category Group Properties**.
4. Click **Filters**. This displays the current list of filter equations. By default, the list is empty.
5. Click **Add**. A new blank filter equation appears.
6. In **Expression**, type or select the expression for the field to filter. To edit the expression, click the expression (fx) button.
7. From the drop-down box, select the data type that matches the type of data in the expression you created in step 5.
8. In the **Operator** box, select the operator that you want the filter to use to compare the values in the **Expression** box and the **Value** box. The operator you choose determines the number of values that are used from the next step.
9. In the **Value** box, type the expression or value against which you want the filter to evaluate the value in **Expression**.

For examples of filter equations, see [Filter Equation Examples \(Report Builder and SSRS\)](#).

10. Click **OK**.

To set a filter on a Chart series group

1. Open a report in **Design** view.
2. On the design surface, click the chart twice to bring up data, series and category field drop zones.
3. Right-click on a field contained in the series field drop zone and select **Series Group Properties**.
4. Click **Filters**. This displays the current list of filter equations. By default, the list is empty.
5. Click **Add**. A new blank filter equation appears.
6. In **Expression**, type or select the expression for the field to filter. To edit the expression, click the expression (fx) button.
7. From the drop-down box, select the data type that matches the type of data in the expression you created in step 5.
8. In the **Operator** box, select the operator that you want the filter to use to compare the values in the **Expression** box and the **Value** box. The operator you choose determines the number of values that are used from the next step.
9. In the **Value** box, type the expression or value against which you want the filter to evaluate the value in **Expression**.

For examples of filter equations, see [Filter Equation Examples \(Report Builder and SSRS\)](#).

10. Click **OK**.

See Also

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Gauges \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

Sort Data in a Data Region (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

To change the sort order of data in a data region when a report first runs, you must set the sort expression on the data region or group. By default, the sort expression for a group is automatically set to the same value as the group expression.

- In a tablix data region, set the sort expression for the data region or for each group, including the details group. If you have only one details group in a tablix data region, you can define a sort expression in the query, on the data region, or on the details group and they all have the same effect.
- In a chart data region, set the sort expression for the Category and Series groups to control the sort order for each group. The order for colors in a chart legend is determined by the sort expression for the data points in the Category group.
- In a gauge data region, you do not typically need to sort data because the gauge displays a single value relative to a range. If you do need sort data in a gauge, you must first define a group, and then set the sort expression for the group.

For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).

For a tablix data region, you can also add an interactive sort button to the top of a column header to provide the user with the ability to change the sort order of groups or detail rows. For more information, see [Interactive Sort \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To sort data in a Tablix data region

1. On the design surface, right-click a row handle, and then click **Tablix Properties**.
2. Click **Sorting**.
3. For each sort expression, follow these steps:
 - a. Click **Add**.
 - b. Type or select an expression by which to sort the data.
 - c. From the **Order** column drop-down list, choose the sort direction for each expression. **A-Z** sorts the expression in ascending order. **Z-A** sorts the expression in descending order.
4. Click **OK**.

To sort values in a group, including the details group, for a Tablix

1. On the design surface, click in the tablix data region to select it. The Grouping pane displays the row groups and column groups for the Tablix data region.
2. In the Row Groups pane, right-click the group name, and then click **Edit Group**.

3. In the **Tablix Group** dialog box, click **Sort**.
4. For each sort expression, follow these steps:
 - a. Click **Add**.
 - b. Type or select an expression by which to sort the data.
 - c. From the **Order** column drop-down list, choose the sort direction for each expression. **A-Z** sorts the expression in ascending order. **Z-A** sorts the expression in descending order.
5. Click **OK**.

To sort x-axis labels in alphabetical order on a chart

1. Right-click a field in the Category Field drop-zone and click **Category GroupProperties**.
2. In the **Category Group Properties** dialog box, click **Sorting**.
3. For each sort expression, follow these steps:
 - a. Click **Add**.
 - b. Select the expression that matches your grouping field. You can verify the expression for the grouping field by clicking **Grouping**.
 - c. From the **Order** column drop-down list, choose the sort direction for each expression. **A-Z** sorts the expression in ascending alphabetical order. **Z-A** sorts the expression in descending alphabetical order.
4. Click **OK**.

To sort the data points in ascending or descending order on a chart

1. Right-click a field in the Category Field drop zone and click **Category GroupProperties**.
2. In the **Category Group Properties** dialog box, click **Sorting**.
3. For each sort expression, follow these steps:
 - a. Click **Add**.
 - b. Select the expression that matches your data field. In most cases, this is an aggregated value, such as `=Sum(Fields!Quantity.Value)`.
 - c. From the **Order** column drop-down list, choose the sort direction for each expression. **A-Z** sorts the expression in ascending order. **Z-A** sorts the expression in descending order.
4. Click **OK**.

To sort data in ascending or descending order for display on a gauge

1. Right-click the gauge and click **Add Data Group**.
2. In the **Gauge Panel GroupProperties** dialog box, click **General** if necessary.
3. In **Group expressions**, click **Add**.
4. In **Group on**, type or select an expression by which to group the data.
5. Repeat steps 3 and 4 until you have added all the group expressions you want to use.
6. Click **Sorting**.
7. For each sort expression, follow these steps:

- a. Click **Add**.
- b. Select the expression that matches your grouping field. You can verify the expression for the grouping field by clicking **Grouping**.
- c. From the **Order** column drop-down list, choose the sort direction for each expression. **A-Z** sorts the expression in ascending order. **Z-A** sorts the expression in descending order.

8. Click **OK**.

For more information about how data is grouped in a gauge, see [Gauges \(Report Builder and SSRS\)](#).

See Also

[Report Builder Help for Dialog Boxes, Panes, and Wizards](#)

[Charts \(Report Builder and SSRS\)](#)

[Formatting Axis Labels on a Chart \(Report Builder and SSRS\)](#)

[Specify Consistent Colors across Multiple Shape Charts \(Report Builder and SSRS\)](#)

Commonly Used Filters (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

To create a filter, you must specify one or more filter equations. A filter equation includes an expression, a data type, an operator, and a value. This topic provides examples of commonly used filters.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Filter Examples

The following table shows examples of filter equations that use different data types and different operators. The scope for the comparison is determined by report item for which a filter is defined. For example, for a filter defined on a dataset, **TOP % 10** is the top 10 percent of values in the dataset; for a filter defined on a group, **TOP % 10** is the top 10 percent of values in the group.

SIMPLE EXPRESSION	DATA TYPE	OPERATOR	VALUE	DESCRIPTION
[SUM(Quantity)]	Integer	>	7	Includes data values that are greater than 7.
[SUM(Quantity)]	Integer	TOP N	10	Includes the top 10 data values.
[SUM(Quantity)]	Integer	TOP %	20	Includes the top 20% of data values.
[Sales]	Text	>	=CDec(100)	Includes all values of type System.Decimal (SQL "money" data types) greater than \$100.
[OrderDate]	DateTime	>	2088-01-01	Includes all dates from January 1, 2008 to the present date.
[OrderDate]	DateTime	BETWEEN	2008-01-01 2008-02-01	Includes dates from January 1, 2008 up to and including February 1, 2008.
[Territory]	Text	LIKE	*east	All territory names that end in "east".

Simple Expression	Data Type	Operator	Value	Description
[Territory]	Text	LIKE	%o%th*	All territory names that include North and South at the beginning of the name.
=LEFT(Fields!Subcat.VaText)	Text	IN	B, C, T	All subcategory values that begin with the letters B, C, or T.

Examples with Report Parameters

The following table provides examples of filter expression that includes a single-value or multivalue parameter reference.

Parameter Type	(Filter) Expression	Operator	Value	Data Type
Single value	[EmployeeID]	=	[@EmployeeID]	Integer
Multivalue	[EmployeeID]	IN	[@EmployeeID]	Integer

See Also

[Report Parameters \(Report Builder and Report Designer\)](#)

[Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#)

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

Grouping Pane (Report Builder)

3/24/2017 • 5 min to read • [Edit Online](#)

The Grouping pane displays the row groups and column groups for the currently selected tablix data region. The Grouping pane is not available for the Chart or Gauge data regions. The Grouping pane contains a Row Groups pane and a Column Groups pane. The Grouping pane has two modes: default and Advanced. Default mode displays a hierarchical view of the dynamic members for row and column groups. Advanced mode displays both dynamic and static members for row and column groups. A group is a named set of data from a report dataset that is displayed on a data region. Groups are organized into hierarchies that include static and dynamic members. For more information, see [Understanding Groups \(Report Builder and SSRS\)](#).

NOTE

If you do not see the Grouping pane, on the **View** tab, in the **Show/Hide** group, click **Grouping**.

Cells in the row and column group areas can be static or dynamic members of a tablix row or column group. Static members repeat once per group and typically contain labels or totals. Dynamic members repeat once per group instance and typically contain the unique values of the group expression. As you select tablix cells in the row group area or column group area, the corresponding group member is selected in the Row Groups or Column Groups pane. Conversely, if you select groups in the Grouping pane, the corresponding cell associated with the group member is selected on the design surface. For more information about tablix row and column group areas, see [Tablix Data Region Areas \(Report Builder and SSRS\)](#).

The Grouping pane supports the following modes:

- **Default.** Use the default mode to add, edit, or delete groups. You can add parent, child, and detail groups by dragging fields from the Report Data pane and inserting them in the group hierarchy. To add an adjacent group, you must use the **Add Group** shortcut. For more information, see [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#).
- **Advanced.** Use the **Advanced mode** to view all members of row and column groups, and to set properties on static members. When you create groups or add totals, the properties that control how the tablix data region renders rows and columns on each report page are set automatically. To manually adjust these properties, you must set them on the tablix member. For more information, see [Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#).

Default Mode

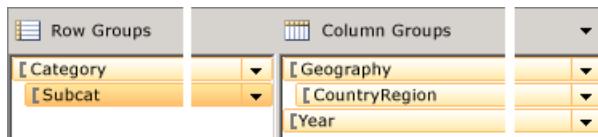
In default mode, the Row Groups pane and the Column Groups pane display a hierarchical view for all parent groups, child groups, and adjacent groups. A child group appears indented under its parent group. An adjacent group appears at the same indent level as its sibling groups. The following figure shows a tablix data region with nested row groups and nested and adjacent column groups.

The screenshot shows the Grouping pane with a tablix data region. The Row Groups pane on the left lists groups: Sales by [Area] and [Year], [Category], [Subcat], Subtotal, and Grand Total. The Column Groups pane at the top lists groups: [Geography] and [Year]. The main pane shows the tablix structure with three columns. The first column contains cells for [Area], [Category], and [Subcat]. The second column contains cells for [Geography] (with [CountryRegion] nested under it) and [Year]. The third column contains cells for [Year] and Total. The 'Subtotal' row is highlighted in yellow, indicating it is selected. The 'Sum(LineTotal)' expression is visible in several cells.

Sales by [Area] and [Year]	[Geography]	[Year]	Total
[Category]	[CountryRegion]		
Subtotal	[Sum(LineTotal)]	[Sum(LineTotal)]	[Sum(LineTotal)]
Grand Total	[Sum(LineTotal)]	[Sum(LineTotal)]	[Sum(LineTotal)]

The Grouping pane displays the corresponding row and column groups. In the following figure, the group based

on subcategory has been selected in the Row Groups pane, and the [Subcat] grouping cell is selected in the tablix data region:



In the Row Groups pane, the group based on subcategory is a child of the group based on category. In the Column Groups pane, the country/region group is a child of the geography group. The year group and the country/region groups are adjacent groups.

For more information, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder\)](#) and [SSRS](#).

Advanced Mode

In Advanced mode, the Row Groups pane and the Column Groups pane display a hierarchical view for all groups, including both static and dynamic members. When you select a member, the Properties pane displays properties for the currently selected tablix member.

NOTE

To toggle **Advanced mode**, right-click the down arrow at the side of the Column Groups pane, and then click **Advanced Mode**.

In most cases, properties that control the display of static and dynamic group rows and group columns are set automatically when you create a group or add totals. To edit the default values, you must select the group member in the Row or Column Groups pane, and change the property values in the Properties window. The following properties are available:

- **FixedData**. Boolean. For outer row and column headers. Freeze the row group area when scrolling vertically or the column group area when scrolling horizontally in a renderer such as HTML.
- **HideIfNoRows**. Boolean. For static members only. If set, Hidden and ToggleItem are ignored. Hide this member if the tablix data region contains no rows of data.
- **KeepTogether**. Boolean. Indicates that the entire tablix member and any nested members should be kept together on one page, if possible.
- **KeepWithGroup**. Boolean. For static row members only. Where possible, keep this row with the previous or following sibling dynamic member, if it is not hidden. To keep a row header with its associated group, set KeepWithGroup to **After**.
- **RepeatOnNewPage**. Boolean. For static row members only and where KeepWithGroup is not None. Where possible, repeat this static row on every page that has at least one instance of the dynamic member specified by KeepWithGroup. To keep a row header with its associated group, set RepeatOnNewPage to **True**.
- **Hidden**. Boolean. Indicates whether the row or column should be initially hidden.
- **ToggleItem**. String. The name of the text box to which to add the toggle image. The text box must be in the same group scope or a containing scope.

For more information, see [Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#), [Display Headers and Footers with a Group \(Report Builder and SSRS\)](#), and [Display Row and Column Headers on Multiple Pages \(Report Builder and SSRS\)](#).

Not every static member has a header that corresponds to a cell on the design surface. In the Grouping

pane, the following convention indicates whether a static member has no header:

- **Static** Indicates a static member with a header cell.
- **(Static)** Indicates a static member with no header cell, known as a hidden static.

See Also

[Report Builder Help for Dialog Boxes, Panes, and Wizards](#)

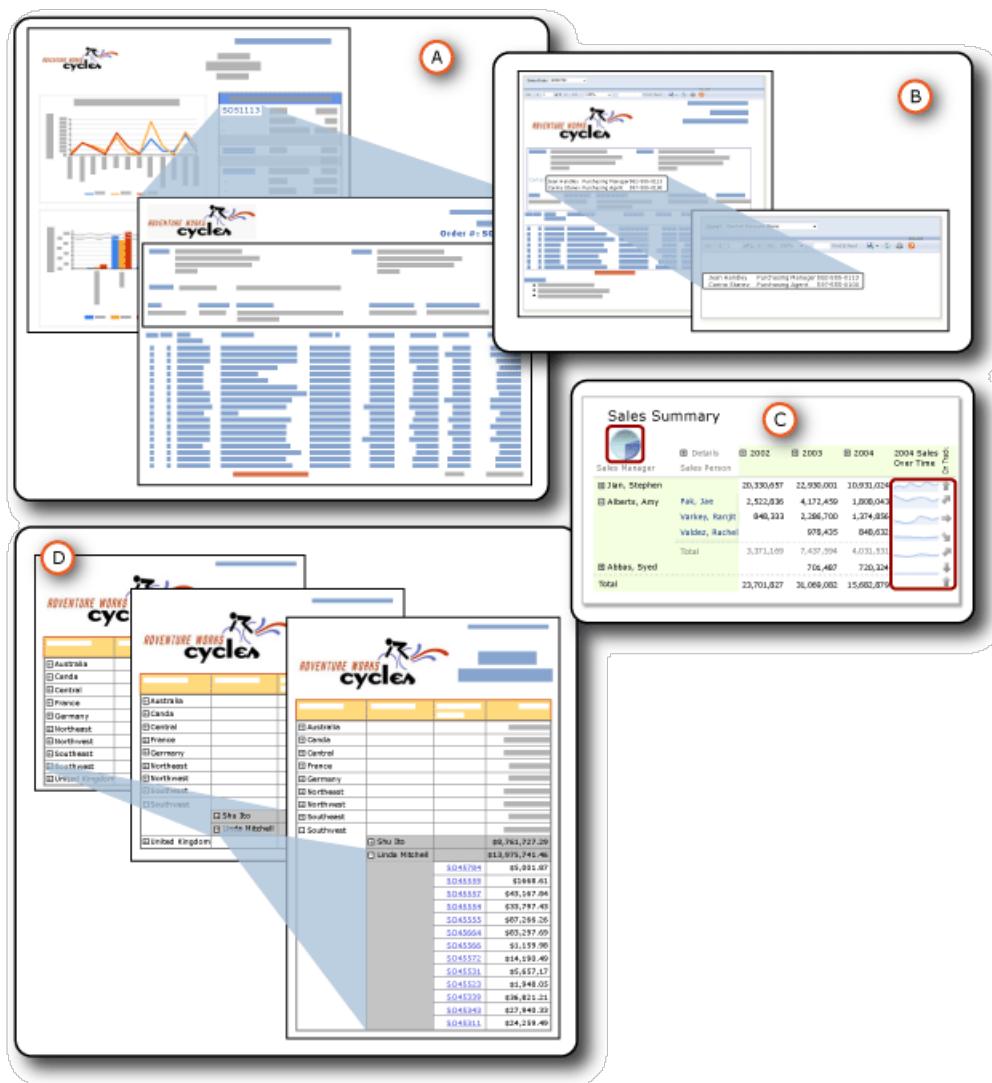
[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Drillthrough, Drilldown, Subreports, and Nested Data Regions

3/29/2017 • 5 min to read • [Edit Online](#)

You can organize data in a variety of ways to show the relationship of the general to the detailed. You can put all the data in the report, but set it to be hidden until a user clicks to reveal details; this is a *drilldown* action. You can display the data in a data region, such as a table or chart, which is *nested* inside another data region, such as a table or matrix. You can display the data in a *subreport* that is completely contained within a main report. Or, you can put the detail data in *drillthrough* reports, separate reports that are displayed when a user clicks a link.



A. Drillthrough report

B. Subreport

C. Nested data regions

D. Drilldown action

All of these have commonalities, but they serve different purposes and have different features. Two of them, drillthrough reports and subreports, are actually separate reports. Nesting is a means of putting one data region inside another data region. Drilldown is an action you can apply to any report item to hide and show other report items. They all are ways that you can organize and display data to help your users understand your report better.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Summary of Characteristics

This table summarizes these different traits. Details are in separate sections later in this topic. Drilldown isn't included in these comparisons because you can apply its showing and hiding action to any report item.

TRAIT	SUBREPORT	DRILLTHROUGH	NESTED
Uses dataset of main report	Same or different	Same or different	Same
Retrieves data	Data retrieved at the same time as main report	Data retrieved one drillthrough report at a time	Data retrieved all at the same time as main report
Is processed and rendered	With the main report	When link is clicked	With the main report.
Performs	Slower (but retrieves all data with main report)	Faster (but does not retrieve all data with main report)	Faster (and retrieves all data with main report)
Uses parameters	Yes	Yes	No
Can be reused	As report, or subreport or drillthrough report in other reports	As report, or subreport or drillthrough report in other reports	Cannot be reused.
Is located	External to main report, same or different report server	External to main report, same report server	Internal to main report
Is displayed	In the main report	In a different report	In the main report

Details of Characteristics

Datasets They Use

Subreports and drillthrough reports can use the same dataset at the main report, or they can use a different one. Nested data regions use the same dataset.

Retrieving Data

Subreports and nested data regions retrieve data at the same time as the main report. Drillthrough reports do not. Each drillthrough report retrieves data when a user clicks each link. This is significant if the data for the main report and the subordinate report must be retrieved at the same time.

Processing and Rendering

A subreport is processed as part of the main report. For example, if a subreport that displays order detail information is added to a table cell in the detail row, the subreport is processed once per row of the table and rendered as part of the main report. A drillthrough report is only processed and rendered when the user clicks the drillthrough link in the summary main report.

Performance

When deciding which to use, consider using a data region instead a subreport, particularly if the subreport is not

used by multiple reports. Because the report server processes each instance of a subreport as a separate report, performance can be impacted. Data regions provide much of the same functionality and flexibility as subreports, but with better performance. Drillthrough reports have better performance than subreports, too, because they don't retrieve all the data at the same time as the main report.

Use of Parameters

Drillthrough reports and subreports typically have report parameters that specify which report data to display. For example, when you click a sales order number in a main report, a drillthrough report opens, which accepts the sales order number as a parameter, and then displays all the data for that sales order. When you create the link in the main report, you specify values to pass as parameters to the drillthrough report.

To create a drillthrough report or subreport, you must design the target drillthrough report or subreport first and then create a drillthrough action or add the reference to the main report.

Reusability

Subreports and drillthrough reports are separate reports. Thus, they can be used in a number of reports, or displayed as standalone reports. Nested data regions are not reusable. You cannot save them as report parts because they are nested in a data region. You can save the data region that contains them as a report part, but not the nested data region.

Location

Subreports and drillthrough reports are both separate reports, so they're stored external to the main report. Subreports can be on the same or a different report server, but drillthrough reports must be on the same report server. Nested data regions are part of the main report.

Display

Subreports and nested data regions are displayed in the main report. Drillthrough reports are displayed on their own.

In This Section

[Drillthrough Reports \(Report Builder and SSRS\)](#)

Explains reports that open when a user clicks a link in a main report.

[Subreports \(Report Builder and SSRS\)](#)

Explains these reports that are displayed inside the body of a main report.

[Nested Data Regions \(Report Builder and SSRS\)](#)

Explains nesting one data region inside another, such as a chart nested inside a matrix.

[Drilldown Action \(Report Builder and SSRS\)](#)

Explains using the drilldown action to hide and show report items.

[Specifying Paths to External Items \(Report Builder and SSRS\)](#)

Explains how to refer to items that are external to the report definition file.

See Also

[Report Parameters \(Report Builder and Report Designer\)](#)

Drillthrough Reports (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

A drillthrough report is a report that a user opens by clicking a link within another report. Drillthrough reports commonly contain details about an item that is contained in an original summary report. For example, in this illustration, the sales summary report lists sales orders and totals. When a user clicks an order number in the summary list, another report opens that contains details about the order.



The data in the drillthrough report is not retrieved until the user clicks the link in the main report that opens the drillthrough report. If the data for the main report and the drillthrough report must be retrieved at the same time, consider using a subreport. For more information, see [Subreports \(Report Builder and SSRS\)](#).

NOTE

When you are working in Report Builder, you must be connected to a report server to view the drillthrough report that opens when you click the drillthrough link in the main report.

To get started quickly with drillthrough reports, see [Tutorial: Creating Drillthrough and Main Reports \(Report Builder\)](#).

Parameters in Drillthrough Reports

A drillthrough report typically contains parameters that are passed to it by the summary report. In the sales summary report example, the summary report includes the field [OrderNumber] in a text box in a table cell. The drillthrough report contains a parameter that takes the order number as a value. When you set the drillthrough report link on the text box for [OrderNumber], you also set the parameter for the target report to [OrderNumber]. When the user clicks the order number in the summary report, the target detail report opens and displays the information for that order number. To view instructions about customizing drillthrough reports based on parameter values, see [Report Parameters \(Report Builder and Report Designer\)](#) and the [InScope Function \(Report Builder and SSRS\)](#).

Designing the Drillthrough Report

To create a drillthrough report, you must design the drillthrough report first, before you create the drillthrough action in the main report.

A drillthrough report can be any report. Typically, the drillthrough report accepts one or more parameters to specify the data to show, based on the link from the main report. For example, if the link from the main report was defined for a sales order, then the sales order number is passed to the drillthrough report.

Creating a Drillthrough Action in the Main Report

You can add drillthrough links to text boxes (including text in the cells of a table or matrix), images, charts, gauges, and any other report item that has an Action property page. For more information, see [Add a Drillthrough Action on a Report \(Report Builder and SSRS\)](#).

You can create the drillthrough action in the main report as a report action or a URL action. For a report action, the drillthrough report must exist on the same report server as the main report. For a URL action, the report must exist at the fully qualified URL location. The way that you specify a report might differ for a report server or a SharePoint site that is integrated with a report server. If the report server is configured in SharePoint integrated mode, only URL actions are supported.

For more information, see [Add a Drillthrough Action on a Report \(Report Builder and SSRS\)](#) and [Specifying Paths to External Items \(Report Builder and SSRS\)](#).

Viewing a Drillthrough Report

To view a summary report with drillthrough links after it is published, you must ensure that the drillthrough reports reside on the same report server as the summary report. In all cases, the user must have permissions on the drillthrough report to view it.

See Also

[Drillthrough, Drilldown, Subreports, and Nested Data Regions \(Report Builder and SSRS\)](#)

Add a Drillthrough Action on a Report (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

The report that opens when you click the link in the main report is known as a *drillthrough report*. This drillthrough link enables a drillthrough action.

Drillthrough reports must be published to the same report server as the main report, but they can be in different folders. You can add a drillthrough link to any item that has an **Action** property, such as a text box, an image, or data points on a chart.

To add a drillthrough link to a report, you must first create the drillthrough report that the main report will link to. A drillthrough report commonly contains details about an item that is contained in the original summary report, and often contains parameters that filter the drillthrough report based on parameters passed to it from the main report. For more information on creating the drillthrough report, see [Drillthrough Reports \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a drillthrough action

1. In Design view, right-click the text box, image, or chart to which you want to add a link and then click **Properties**.
2. In the item's **Properties** dialog box, click **Action**.
3. Select **Go to report**. Additional sections appear in the dialog box for this option.
4. In **Specify a report**, click **Browse** to locate the report that you want to jump to, or type the name of the report. Alternatively, click the expression (**fx**) button to create an expression for the report name.

The format of the path to the drillthrough report differs for native and SharePoint integrated mode. If you browse to the report, a path of the correct format is provided. For more information, see [Specifying Paths to External Items \(Report Builder and SSRS\)](#).

If you have to specify parameters for the drillthrough report, follow the next step.

5. In **Use these parameters to run the report**, click **Add**. A new row is added to the parameter grid.
 - In the **Name** text box, click the drop-down list or type the name of the report parameter in the drillthrough report.

NOTE

The names in the parameter list must match the expected parameters in the target report exactly. For example, parameter names must match by case. If the names do not match, or if an expected parameter is not listed, the drillthrough report fails.

- In **Value**, type or select the value to pass to the parameter in the drillthrough report.

NOTE

Values can contain an expression that evaluates to a value to pass to the report parameter. The expressions in the value list include the field list for the current report.

For information on how to hide parameters in reports, see [Add, Change, or Delete a Report Parameter \(Report Builder and SSRS\)](#).

6. (Optional) For text boxes, it is helpful to indicate to the user that the text is a link by changing the color and effect of the text on the **Home** tab of the Ribbon.
7. To test the link, run the report and click the report item that you set this link on.

See Also

[Action Properties Dialog Box \(Report Builder and SSRS\)](#)

[Formatting Data Points on a Chart \(Report Builder and SSRS\)](#)

[Show ToolTips on a Series \(Report Builder and SSRS\)](#)

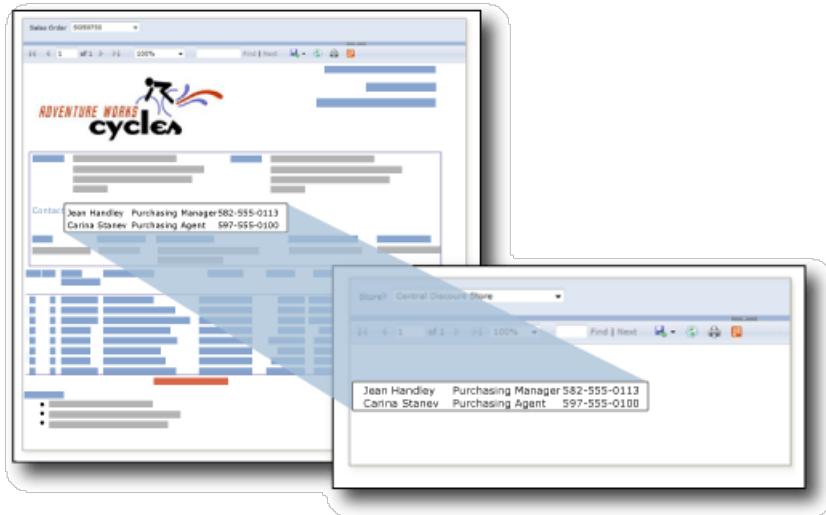
Subreports (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

A subreport is a report item that displays another report inside the body of a main report. Conceptually, a subreport in a report is similar to a frame in a Web page. It is used to embed a report within a report. Any report can be used as a subreport. The report that is displayed as the subreport is stored on a report server, usually in the same folder as the parent report. You can design the parent report to pass parameters to the subreport. A subreport can be repeated within data regions, using a parameter to filter data in each instance of the subreport.

NOTE

If you use a subreport in a tablix data region, the subreport and its parameters will be processed for every row. If there are many rows, consider whether a drillthrough report is more appropriate.



In this illustration, the contact information displayed in the main Sales Order report actually comes from a Contacts subreport.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Comparing Subreports and Nested Data Regions

If you're thinking of using subreports to display separate groups of data, consider using data regions, such as tables, matrices, and charts, instead. Reports with data regions only may perform better than reports that include subreports.

Use data regions to nest groups of data from the same data source within a single data region. Use subreports to nest groups of data from different data sources within a single data region, reuse a subreport in multiple parent reports, or display a standalone report inside of another report. For example, you can create a "briefing book" by placing multiple subreports inside the body of another report.

Data regions provide much of the same functionality and flexibility as subreports, but with better performance. Because the report server processes each instance of a subreport as a separate report, performance can be

impacted. For more information, see [Nested Data Regions \(Report Builder and SSRS\)](#).

Using Parameters in Subreports

To pass parameters from the parent report to the subreport, define a report parameter in the report that you use as the subreport. When you place the subreport in the parent report, you can select the report parameter and a value to pass from the parent report to the report parameter in the subreport.

NOTE

The parameter that you select from the subreport is a report parameter, not a query parameter.

You can place a subreport in the main body of the report, or in a data region. If you place a subreport in a data region, the subreport will repeat with each instance of the group or row in the data region. To pass a value from the group or row to the subreport, in the subreport value property, use a field expression for the field containing the value you want to pass to the subreport parameter.

For more information about working with subreports, see [Add a Subreport and Parameters \(Report Builder and SSRS\)](#).

Specifying Subreport Names and Locations

You can design a main report to specify a subreport in a different folder on the same report server.

The syntax you use to specify the subreport depends on whether the report server is in native mode or SharePoint integrated mode. For more information, see [Specifying Paths to External Items \(Report Builder and SSRS\)](#).

In Report Builder, to preview a subreport in a main report, both reports must be located in the same report server, or you must specify a full path to the subreport.

See Also

[Drillthrough, Drilldown, Subreports, and Nested Data Regions \(Report Builder and SSRS\)](#)

Add a Subreport and Parameters (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

Add subreports to a report when you want to create a main report that is a container for multiple related reports. A subreport is a reference to another report. To relate the reports through data values (for example, to have multiple reports show data for the same customer), you must design a parameterized report (for example, a report that shows the details for a specific customer) as the subreport. When you add a subreport to the main report, you can specify parameters to pass to the subreport.

You can also add subreports to dynamic rows or columns in a table or matrix. When the main report is processed, the subreport is processed for each row. In this case, consider whether you can achieve the desired effect by using data regions or nested data regions.

To add a subreport to a report, you must first create the report that will act as the subreport. For more information on creating the subreport, see [Subreports \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a subreport

1. On the **Insert** tab, click **Subreport**.
2. On the design surface, click a location on the report and then drag a box to the desired size of the subreport.
Alternatively, click the design surface to create a subreport of default size.
3. Right-click the subreport, and then click **Subreport Properties**.
4. In the **Subreport Properties** dialog box, type a name in the **Name** text box or accept the default. The name must be unique within the report. By default, a general name such as Subreport1 or Subreport2 is assigned.
5. In the **Use this report as a subreport** box, click **Browse**, or type the name of the report. Clicking **Browse** is preferred because the path to the subreport will be specified automatically. You can specify the report in the several ways. For more information, see [Specifying Paths to External Items \(Report Builder and SSRS\)](#).
6. (Optional) Click **Yes** for **Omit border on page break** to prevent a border from being rendered in the middle of the subreport if the subreport spans more than one page.
7. Click **OK**.

To specify parameters to pass to a subreport

1. In Design view, right-click the subreport and then click **Subreport Properties**.
2. In the **Subreport Properties** dialog box, click **Parameters**.
3. Click **Add**. A new row is added to the parameter grid.
4. In the **Name** text box, type the name of a parameter in the subreport or choose it from the list box. This name must match a report parameter, not a query parameter, in the subreport.
5. In the **Value** list box, type or select a value to pass to the subreport. This value can be static text or an

expression that references a field or other object in the main report.

NOTE

In Report Builder, if a parameter is missing from the **Parameters** list and the subreport has a default value defined, the subreport will be processed correctly.

In Report Designer, all parameters that are required by the subreport must be included in the **Parameters** list. If a required parameter is missing, the subreport is not displayed correctly in the main report.

6. Repeat steps 3-5 to specify a name and value for each subreport parameter.
7. To delete a subreport parameter, click the parameter in the parameter grid, and then click **Delete**.
8. To change the order of a subreport parameter, click the parameter, and then click the up button or the down button.

Changing the order of a subreport parameter does not affect the processing of the subreport.

See Also

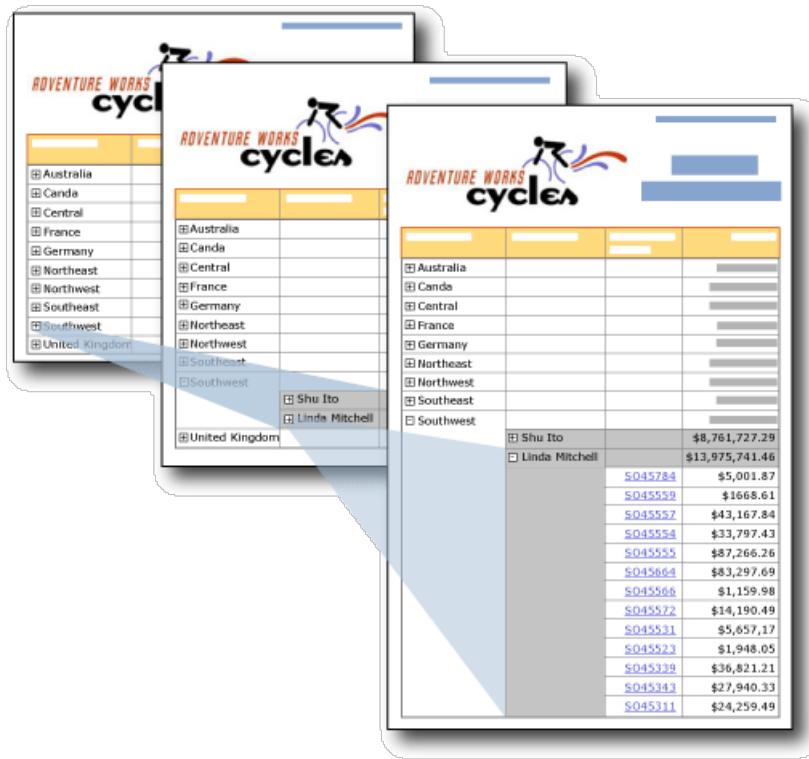
[Subreports \(Report Builder and SSRS\)](#)

[Rendering Behaviors \(Report Builder and SSRS\)](#)

Drilldown Action (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

By providing plus and minus icons on a text box, you can enable users to hide and display items interactively. This is called a *drilldown* action. For a table or matrix, you can show or hide static rows and columns, or rows and columns that are associated with groups.



In this illustration, the user clicks the plus signs (+) in the report to show detail data.

For example, you can initially hide all the rows except the outer group summary row for a table with row groups. For each inner group (including the details group), add an expand/collapse icon to the grouping cell of the containing group. When the report is rendered, the user can click the text box to expand and collapse the detail data. For more information, see [Tables \(Report Builder and SSRS\)](#).

To allow users to expand or collapse an item, you set the visibility properties for that item.

NOTE

When you create a report with a drilldown action, the visibility information must be set on the group, column, or row that you want to hide, not just a single text box in the row or column. In addition, the text box that you use for the toggle must be in a containing scope that controls the item that you want to show or hide.

For example, to hide a row associated with a nested group, the text box must be in a row associated with the parent group or higher in the containment hierarchy.

For information on setting visibility information on the group, column or row, see [Add an Expand or Collapse Action to an Item \(Report Builder and SSRS\)](#)

For more information about hiding report items, see [Hide an Item \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Comparing Drilldown and Drillthrough Reports

In a drilldown report, a user clicks a plus or minus button to expand or collapse a section of a report to show detail data in place. In a drillthrough report, the user clicks a link for a summary value, and this opens a separate, related report to show detail data. The detail data is only retrieved when the detail report runs. Drillthrough reports typically require fewer resources than drilldown reports. For more information, see [Drillthrough, Drilldown, Subreports, and Nested Data Regions \(Report Builder and SSRS\)](#).

Rendering Extension Support for Hidden Report Items

The show-and-hide toggle on report items is supported only by rendering extensions that support user interactivity, such as the HTML rendering extension that is used when you run a report in Report Builder and in Report Manager, for example. Other rendering extensions display hidden items. The following list describes support for report items with conditional visibility:

- In HTML, if items are hidden, they are not visible in the HTML source.
- The XML rendering extension displays all report items, regardless of whether they are hidden.
- The Excel rendering extension displays and expands hidden rows and columns for a table, matrix, or list. All rows and columns are visible.

For more information, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

See Also

[Drillthrough, Drilldown, Subreports, and Nested Data Regions \(Report Builder and SSRS\)](#)

[Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Add an Expand or Collapse Action to an Item (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

You can enable a user to interactively expand or collapse report items, or expand or collapse rows and columns associated with a group for a table or matrix. To allow users to expand or collapse an item, you set the visibility properties for that item. Setting visibility works in an HTML report viewer, and is sometimes called a *drilldown* action.

In report design view, you specify the name of the text box where you want to display the expand and collapse toggle icons. In the rendered report, the text box displays a plus (+) or minus (-) sign in addition to its contents. When the user clicks the toggle, the report display is refreshed to show or hide the report item, based on the current visibility settings for items in the report.

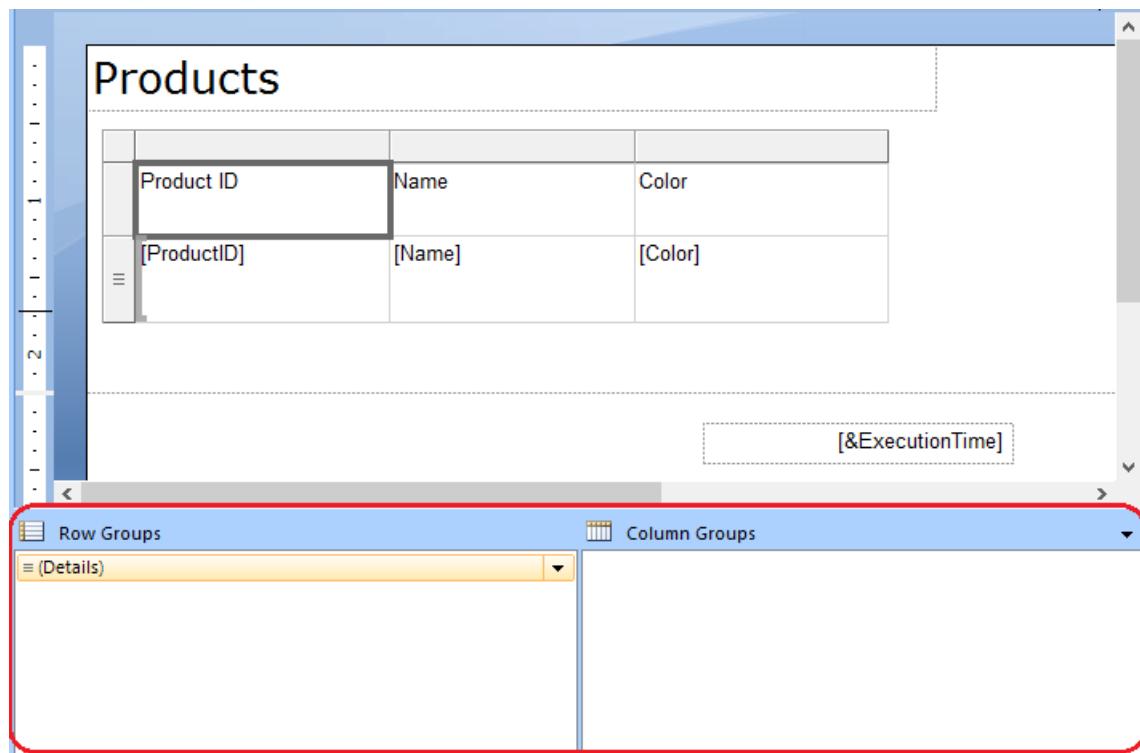
Typically, the expand and collapse action is used to initially display only summary data and to enable the user to click the plus sign to show detail data. For example, you can initially hide a table that displays values for a chart, or hide child groups for a table with nested row or column groups, as in a drilldown report.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add expand and collapse action to a group

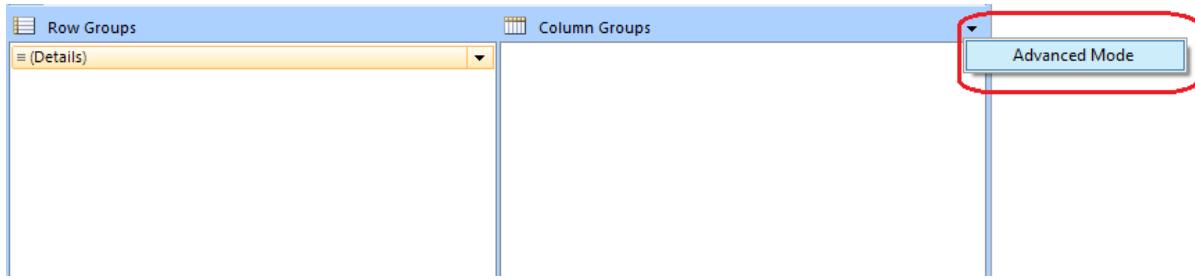
1. In report design view, click the table or matrix to select it. The Grouping pane displays the row and column groups.



If the Grouping pane does not appear, click the **View** menu and then click **Grouping**.

2. Right-click anywhere in the title bar of the Grouping pane, and then click **Advanced**. The Grouping pane

mode toggles to show the underlying display structure for rows and columns on the design surface.



3. In the appropriate group pane, click the name of the row group or column group for which you want to hide the associated rows or columns. The group is selected and the Properties pane shows the **Tablix Member** properties.

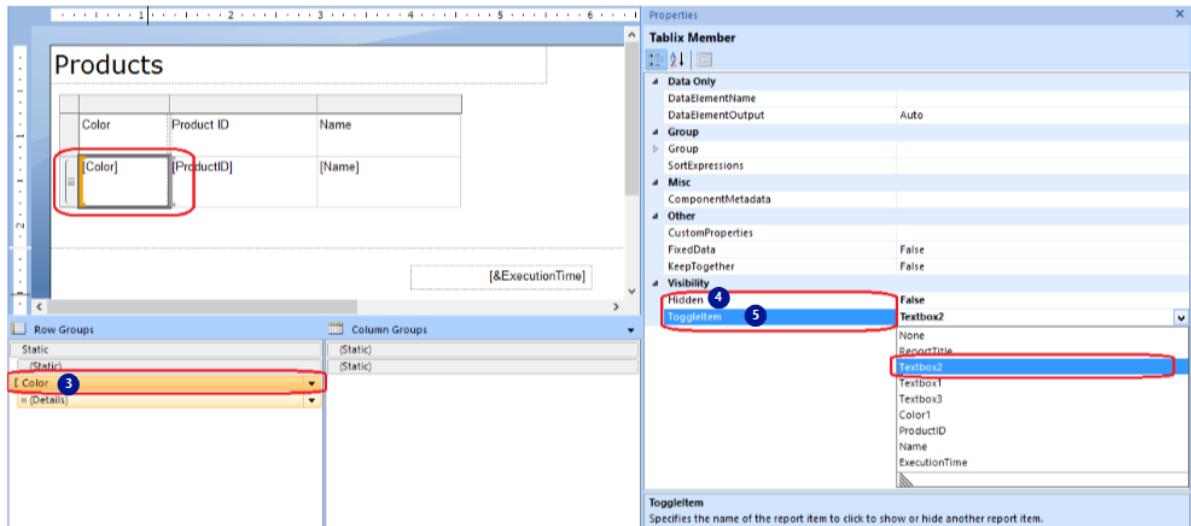
NOTE

If you do not see the Properties pane, click **View** on the Ribbon and then click **Properties**.

4. In **Hidden**, choose one of the following options to set the visibility of this report item the first time you run a report:
 - Select **False** to display the report item.
 - Select **True** to hide the report item.
 - Select **<Expression>** to open the **Expression** dialog box to create an expression that is evaluated at run time to determine the visibility.

5. In **ToggleItem**, from the drop-down box, select the name of a text box to which to add the toggle image.

In the following image, the Color row group is configured enable users to expand and collapse associated rows.



NOTE

The text box with the toggle image cannot be the row or column group for which you want to hide the associated rows or columns. It must either be in the same group as the item that is being hidden or in an ancestor group. For example, to toggle visibility of rows associated with a child group, select a text box in a row associated with the parent group.

6. To test the toggle, run the report and click the text box with the toggle image. The report display refreshes to

show row groups and column groups with their toggled visibility.

The screenshot shows the Microsoft Report Designer ribbon with the 'Run' tab selected. Below the ribbon is a table titled 'Products' with three rows. The first row has a red box around the 'Color' column header. The table data is as follows:

Color	Product ID	Name
	1	Adjustable Race
	2	Bearing Ball
	3	BB Ball Bearing

To add expand and collapse action to a report item

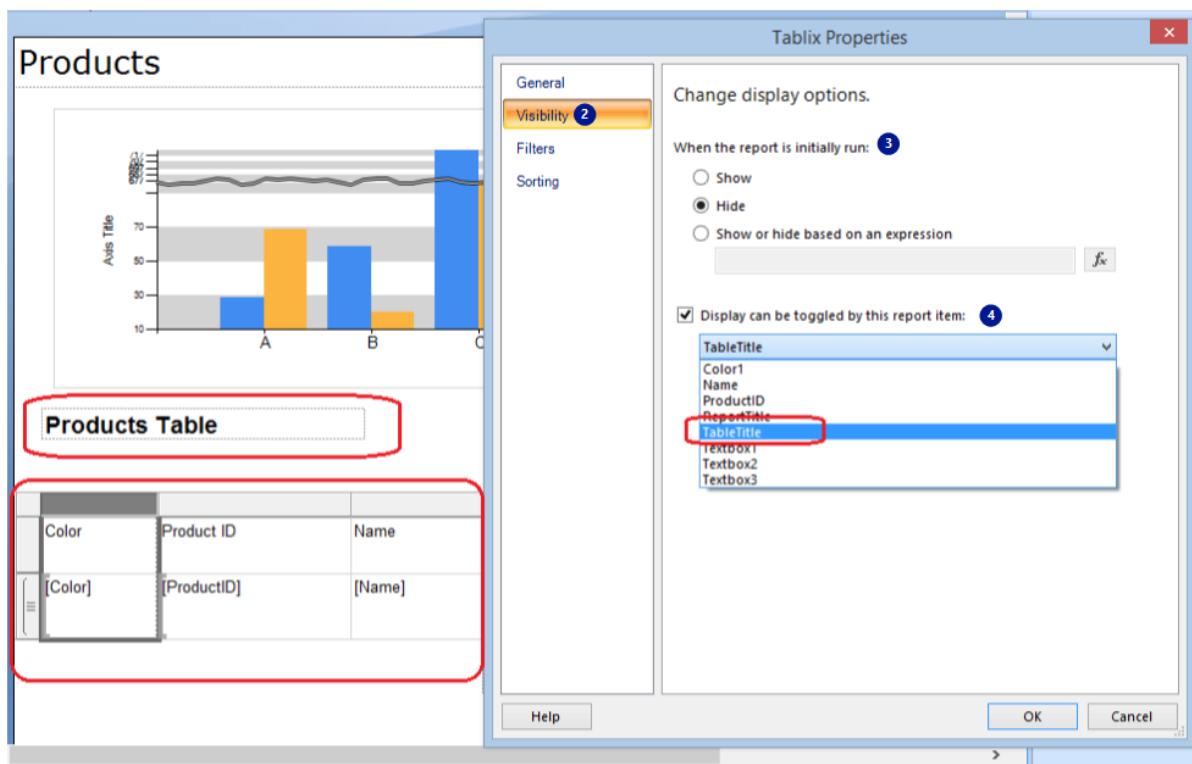
1. In report design view, right-click the report item to show or hide, and then click <report item> **Properties**.
The <report item> **Properties** dialog box for the report item opens.
2. Click **Visibility**.
3. In **When the report is initially run**, choose one of the following options to set the visibility of this report item the first time you run a report:
 - Select **Show** to display the report item.
 - Select **Hide** to hide the report item.
 - Select **Show or hide based on an expression** to use an expression evaluated at run time to determine the visibility. Click (fx) to open the **Expression** dialog box to create an expression.

NOTE

When you specify an expression for visibility, you are setting the Hidden property of the report item. The expression evaluates to a **Boolean** value of **True** to hide the item and **False** to show the item.

4. In **Display can be toggled by this report item**, from the drop-down box, type or select the name of a text box in the report in which to display a toggle image; for example, Textbox1.

In the following image, the table is configured to enable users to expand and collapse it. The display of the table is toggled by the Products Table text box.



NOTE

The text box that you choose must be in the current or containing scope for this report item (up to and including the report body). For example, to toggle visibility of a chart, select a text box that is in the same containing scope as the chart; for example, the report body or a rectangle. The text box must be in the same container hierarchy or higher.

- To test the toggle, run the report and click the text box with the toggle image. The report display refreshes to show report items with their toggled visibility.

See Also

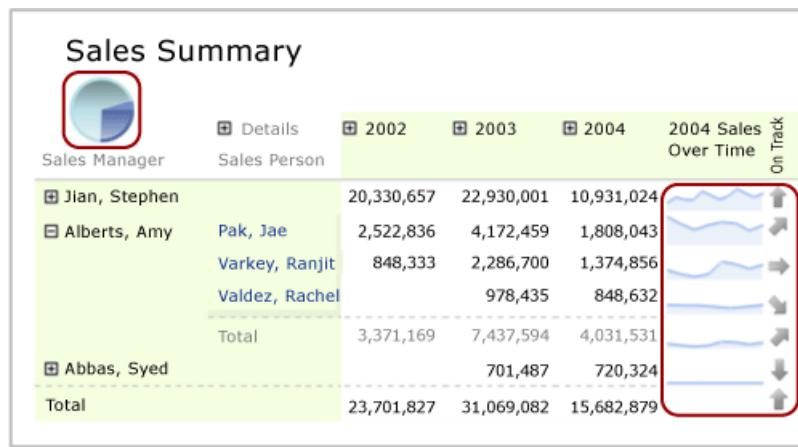
- [Drilldown Action \(Report Builder and SSRS\)](#)
- [Hide an Item \(Report Builder and SSRS\)](#)

Nested Data Regions (Report Builder and SSRS)

3/24/2017 • 6 min to read • [Edit Online](#)

You can nest one data region, such as a chart, inside another data region, such as a matrix, typically to display data summaries in a concise manner or to provide a visual display as well as a table or matrix display.

For example, for a matrix (also called a *tablix*) that contains sales orders grouped by Store on rows and by Quarter on columns, you can add a table or chart to the corner cell to summarize the sales for all stores, or add a chart to a matrix column header to show the sales contribution of the data in the column as a percentage of all sales.



In this illustration, the pie chart in the corner cell and the sparkline charts in the rows are nested data regions.

By definition, nested data regions are based on the same report dataset. You cannot nest data regions that are based on different datasets. To display data from different datasets, consider using drillthrough reports or subreports. For more information, see [Drillthrough, Drilldown, Subreports, and Nested Data Regions \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Scope for a Nested Data Region

The scope for data in a nested data region is automatically defined by its placement in the parent data region. For example, the scope for data for a chart nested in a tablix corner cell is the data from the dataset bound to the tablix data region after the filters are applied for the dataset, the tablix data region, and the chart data region. The scope for a tablix nested in a tablix cell is the same as the scope for the corner cell, but additionally scoped to the row and column group memberships of the cell in which it is nested, with the corresponding group filters applied. For more information about scope, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

The following list describes the scope for cells in the following tablix areas:

- **Tablix corner** The scope is the data in the data region linked to the tablix data region, after the filter and sort expressions for the dataset and the outer tablix are applied.
- **Tablix column group** The data in the innermost column group, after the filter and sort expressions for

the dataset, the outer tablix, and the column groups are applied.

- **Tablix row group** The data in the innermost row group, after the filter and sort expressions for the dataset, the outer tablix, and the row groups are applied.
- **Tablix body** The data in the innermost group represented by the intersection of row groups and column groups, after the filter and sort expressions for the dataset, the outer tablix, and the row and column groups are applied.

For more information, see [Tablix Data Region Areas \(Report Builder and SSRS\)](#).

Nesting a Chart, Sparkline, or Data Bar in a Tablix

When you add a chart (including a sparkline or data bar) to a tablix column group header or group footer row, or to a tablix body cell, the data passed to the chart is scoped to the subset of data for that cell. By default, when you add a chart to a tablix cell, the chart dimensions expand to fill the cell.

NOTE

To have more control over the size of a chart in a tablix cell, add the chart to a rectangle first, and then add the rectangle to the tablix cell.

By default, the chart legend colors are determined by the color of the data points in the chart series. To control colors so that nested chart data regions all use the same color for the same category of data, you must use custom colors and set sort expressions on the data. For more information, see [Specify Consistent Colors across Multiple Shape Charts \(Report Builder and SSRS\)](#) and [Sort Data in a Data Region \(Report Builder and SSRS\)](#).

Nesting a Gauge or an Indicator in a Tablix

You can nest a gauge or an indicator inside of a table, matrix, or list in order to show a key performance indicator (KPI). When you place a gauge or indicator inside of a table, it will be rendered for each row in the tablix. For more information about adding indicators to a tablix, see [Indicators \(Report Builder and SSRS\)](#).

Adding a gauge to a tablix

There are two ways to add a gauge to a tablix data region:

- Click inside the tablix cell and insert a gauge. The **Select Gauge Type** dialog box appears. Once you have selected a gauge type, the Gauge data region is placed inside of the selected tablix cell. You will likely need to resize the tablix in order to format the gauge.
- Click outside the table and insert a gauge. The **Select Gauge Type** dialog box appears. After you select a gauge type, the Gauge data region is placed in the top-left corner of the report. After you add data and format this gauge, drag and drop it inside of the tablix cell.

Like the chart, the dataset passed to the gauge is scoped to the subset of data for that cell. When a gauge is placed inside of a tablix cell, the gauge will always aggregate only one row of data.

When data in your tablix contains grouping, the Gauge data region that is nested inside the tablix does not automatically inherit this group. You must add a matching group expression to the gauge in order to show the same information that is shown on the tablix. For example, if data in your tablix is grouped by Product, you must add a Product group expression to the gauge to show the same data. For more information, see [Gauges \(Report Builder and SSRS\)](#) and [Add or Delete a Group in a Data Region \(Report Builder and SSRS\)](#).

You must set the minimum and maximum values that will be displayed on the gauge scale. To specify the maximum value of the gauge, you can use an expression, such as `=Max!MyField.Value`. However, because

this expression will be evaluated within the scope of the data in the cell only, the maximum of each gauge will not be the same for all rows in the tablix. This may make comparisons between gauges in the tablix more difficult to understand. Alternatively, you can specify a static value for the maximum. All rows inside of the tablix will show a gauge with this maximum value. For more information, see [Set a Minimum or Maximum on a Gauge \(Report Builder and SSRS\)](#).

If the data becomes too large on the gauge, consider using a scale multiplier to reduce the amount of digits displayed. To specify a multiplier, you can right-click on the scale and select **Scale Properties**.

When the **Scale Properties** dialog box opens, specify a value for **Multiplier**.

Nesting a Table or Matrix and a Chart in a List

To nest multiple data regions in a List add a rectangle first, and then add the data regions to the rectangle.

You can define a group for a List data region, and then add a tablix and a chart to provide different views of the same data. To achieve this effect, you must define identical group and sort expressions for the embedded tablix and chart. By definition, the tablix and chart use data from the dataset of the parent list data region.

NOTE

By default, when you add a List data region to the design surface, the list includes a detail row. You can change this default by adding a group row and removing the detail row. For more information, see [Exploring the Flexibility of a Tablix Data Region \(Report Builder and SSRS\)](#).

For more information, see [Understanding Groups \(Report Builder and SSRS\)](#) and [Add, Move, or Delete a Table, Matrix, or List \(Report Builder and SSRS\)](#).

See Also

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Gauges \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Formatting Report Items \(Report Builder and SSRS\)](#)

[Tutorial: Adding a KPI to Your Report \(Report Builder\)](#)

[Formatting Scales on a Gauge \(Report Builder and SSRS\)](#)

Specifying Paths to External Items (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

You specify paths in report item properties to reference items such as drillthrough reports, subreports, and image files that are external to the report definition file and are stored on a report server.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

NOTE

In Report Builder, paths to items must specify items on a report server. Paths to items on a file system are not supported. You can preview a report that uses these items only if you are connected to the report server where the items are located.

When you specify a path for an external item in a dialog box for actions, subreports, or images, you can browse directly to the report server and select the item. Browsing to an item and selecting it directly is the recommended way to specify a drillthrough report or subreport. That way the correct parameter names will be available in a drop-down list when you specify report or subreport parameters. When you change an item path to point to a different item, you must manually update the correct parameter names and values as needed.

On a report server configured in native mode, specify a drillthrough report name without the file extension .rdl.

On a report server configured in SharePoint integrated mode, you must include the file extension .rdl. The path can be one of the following:

- **A relative path to the item from the main report.** For example, ..\AllSubreports/Subreport1. In this example, the .. indicates the folder above the folder where the main report is located.

NOTE

Relative paths are not supported when the report is run inside Report Builder. To view a report that uses relative paths to external items, save the report to the report server, and run the report from there.

- **A full path to the item.**

- **On a report server:** The path starts from /, the Home folder. For example, /Reports/AllSubreports/Subreport1.
- **On a SharePoint site:** You must specify the report name in an expression, with the full URL of the item and the file extension .rdl. For example, `="http://server/site/library/folder/Report1.rdl"`.

See Also

[Add an External Image \(Report Builder and SSRS\)](#)

[Add a Subreport and Parameters \(Report Builder and SSRS\)](#)

[Add a Drillthrough Action on a Report \(Report Builder and SSRS\)](#)

Expressions (Report Builder and SSRS)

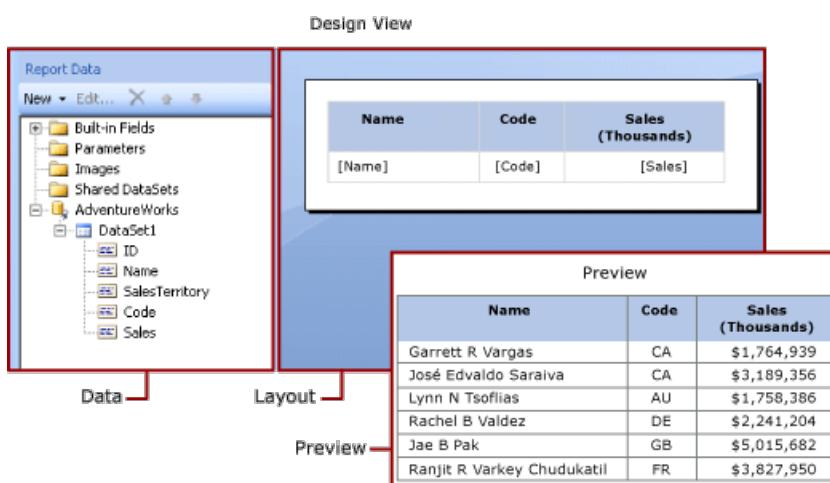
3/24/2017 • 6 min to read • [Edit Online](#)

Expressions are widely used throughout Reporting Services paginated reports to retrieve, calculate, display, group, sort, filter, parameterize, and format data.

Many report item properties can be set to an expression. Expressions help you control the content, design, and interactivity of your report. Expressions are written in Microsoft Visual Basic, saved in the report definition, and evaluated by the report processor when you run the report.

Unlike applications such as Microsoft Office Excel where you work with data directly in a worksheet, in a report, you work with expressions that are placeholders for data. To see the actual data from the evaluated expressions, you must preview the report. When you run the report, the report processor evaluates each expression as it combines report data and report layout elements such as tables and charts.

As you design a report, many expressions for report items are set for you. For example, when you drag a field from the data pane to a table cell on the report design surface, the text box value is set to a simple expression for the field. In the following figure, the Report Data pane displays the dataset fields ID, Name, SalesTerritory, Code, and Sales. Three fields have been added to the table: [Name], [Code], and [Sales]. The notation [Name] on the design surface represents the underlying expression `=Fields!Name.Value`.



When you preview the report, the report processor combines the table data region with the actual data from the data connection and displays a row in the table for every row in the result set.

To enter expressions manually, select an item on the design surface, and use shortcut menus and dialog boxes to set the properties of the item. When you see the **(fx)** button or the value `<Expression>` in a drop-down list, you know that you can set the property to an expression. For more information, see [Add an Expression \(Report Builder and SSRS\)](#).

To develop complex expressions or expressions that use custom code or custom assemblies, we recommend that you use Report Designer in SQL Server Data Tools (SSDT). For more information, see [Custom Code and Assembly References in Expressions in Report Designer \(SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Simple and Complex Expressions

Expressions begin with an equal sign (=) and are written in Microsoft Visual Basic. Expressions can include a combination of constants, operators, and references to built-in values (fields, collections, and functions), and to external or custom code.

You can use expressions to specify the value of many report item properties. The most common properties are values for text boxes and placeholder text. Typically, if a text box contains only one expression, the expression is the value of the text box property. If a text box contains multiple expressions, each expression is the value of placeholder text in the text box.

By default, expressions appear on the report design surface as *simple* or *complex expressions*.

- **Simple** A simple expression contains a reference to a single item in a built-in collection, for example, a dataset field, a parameter, or a built-in field. On the design surface, a simple expression appears in brackets. For example, `[FieldName]` corresponds to the underlying expression `=Fields!FieldName.Value`. Simple expressions are created for you automatically as you create the report layout and drag items from the Report Data pane to the design surface. For more information about the symbols that represent different built-in collections, see [Understanding Prefix Symbols for Simple Expressions](#).
- **Complex** A complex expression contains references to multiple built-in references, operators, and function calls. A complex expression appears as <>Expr>> when the expression value includes more than a simple reference. To view the expression, hover over it and use the tooltip. To edit the expression, open it in the **Expression** dialog box.

The following figure shows typical simple and complex expressions for both text boxes and placeholder text.

Sales			
[Category]		[Sum(Sales)]	«Expr»
Total		[Sum(Sales)]	[Sum(Qty)]

To display sample values instead of text for expressions, apply formatting to the text box or placeholder text. The following figure shows the report design surface toggled to show sample values:

Sales			
[Category]		[\$12,345.00]	[12%]
Total		[\$12,345.00]	[12345] [12%]

For more information, see [Formatting Text and Placeholders \(Report Builder and SSRS\)](#).

Report Model Formulas

When you are designing a query for a dataset that uses a report model as a data source, you can create *formulas*. Formulas are calculations performed on values in a report that are based on data from a report model.

For more information, see [Formulas in Report Model Queries \(Report Builder and SSRS\)](#).

Understanding Prefix Symbols in Simple Expressions

Simple expressions use symbols to indicate whether the reference is to a field, a parameter, a built-in

collection, or the ReportItems collection. The following table shows examples of display and expression text:

ITEM	DISPLAY TEXT EXAMPLE	EXPRESSION TEXT EXAMPLE
Dataset fields	[Sales]	=Fields!Sales.Value
	[SUM(Sales)]	=Sum(Fields!Sales.Value)
	[FIRST(Store)]	=First(Fields!Store.Value)
Report parameters	[@Param]	=Parameters!Param.Value
	[@Param.Label]	=Parameters!Param.Label
Built-in fields	[&ReportName]	=Globals!ReportName.Value
Literal characters used for display text	\[Sales\]	[Sales]

Writing Complex Expressions

Expressions can include references to functions, operators, constants, fields, parameters, items from built-in collections, and to embedded custom code or custom assemblies.

NOTE

To develop complex expressions or expressions that use custom code or custom assemblies, we recommend that you use Report Designer in SQL Server SQL Server Data Tools (SSDT). For more information, see [Custom Code and Assembly References in Expressions in Report Designer \(SSRS\)](#).

The following table lists the kinds of references you can include in an expression:

REFERENCES	DESCRIPTION	EXAMPLE
Constants	Describes the constants you can access interactively for properties that require constant values, such as font colors.	= "Blue"
Operators	Describes the operators you can use to combine references in an expression. For example, the & operator is used for concatenating strings.	= "The report ran at: " & Globals!ExecutionTime & "."
Built-in Collections	Describes the built-in collections that you can include in an expression, such as Fields, Parameters, and Variables.	=Fields!Sales.Value =Parameters!Store.Value =Variables!MyCalculation.Value
Built-in Report and Aggregate Functions	Describes the built-in functions, such as Sum or Previous, that you can access from an expression.	=Previous(Sum(Fields!Sales.Value))

REFERENCES	DESCRIPTION	EXAMPLE
Custom Code and Assembly References in Expressions in Report Designer (SSRS)	<p>Describes how you can access the built-in CLR classes Math and Convert, other CLR classes, Visual Basic run-time library functions, or methods from an external assembly.</p> <p>Describes how you can access custom code that is embedded in your report, or that you compile and install as a custom assembly on both the report client and the report server.</p>	<pre>=Sum(Fields!Sales.Value)</pre> <pre>=CDate(Fields!SalesDate.Value)</pre> <pre>=DateAdd("d",3,Fields!BirthDate.Value)</pre> <pre>=Code.ToUSD(Fields!StandardCost.Value)</pre>

Validating Expressions

When you create an expression for a specific report item property, the references that you can include in an expression depend on the values that the report item property can accept and the scope in which the property is evaluated. For example:

- By default, the expression [Sum] calculates the sum of data that is in scope at the time the expression is evaluated. For a table cell, the scope depends on row and column group memberships. For more information, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).
- For the value for a Font property, the value must evaluate to the name of a font.
- Expression syntax is validated at design time. Expression scope validation occurs when you publish the report. For validation that depends on the actual data, errors can only be detected at run-time. Some of these expressions produce #Error as an error message in the rendered report. To help determine the issues for this kind of error, you must use Report Designer in SQL Server Data Tools (SSDT). Report Designer provides an Output window that provides more information about these errors.

For more information, see [Expression Reference \(Report Builder and SSRS\)](#).

In This Section

[Add an Expression \(Report Builder and SSRS\)](#)

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

[Expression Reference \(Report Builder and SSRS\)](#)

See Also

For more information and examples, see the following topics:

- [Expression Uses in Reports \(Report Builder and SSRS\)](#)
- [Expression Examples \(Report Builder and SSRS\)](#)
- [Filter Equation Examples \(Report Builder and SSRS\)](#)
- [Group Expression Examples \(Report Builder and SSRS\)](#)
- [Tutorial: Introducing Expressions](#)
- [Report Samples \(Report Builder and SSRS\)](#)

Add an Expression (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Expressions are used throughout a report for defining report item properties, filters, groups, sort order, connection strings, and parameter values. Expressions begin with an equal sign (=) and are written in Microsoft Visual Basic. They are evaluated at run time by the report processor, which combines the evaluation result with report layout elements.

Expressions can be simple or complex. Simple expression refer to a single item in a built-in collection. Complex expressions can contain constants, operators, global collection items, and function calls. For more information, see [Expressions \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add an expression to a text box

- In **Design** view, click the text box on the design surface to which you want to add an expression.
 - For a simple expression, type the display text for the expression in the text box. For example, for the dataset field Sales, type `[Sales]`.
 - For a complex expression, right-click the text box, and select **Expression**. The **Expression** dialog box opens. Type or interactively create your expression after the '=' in the expression pane, and then click OK.

The expression appears on the design surface as `<>Expr>>`.

See Also

- [Formatting Text and Placeholders \(Report Builder and SSRS\)](#)
- [Text Boxes \(Report Builder and SSRS\)](#)
- [Expression Uses in Reports \(Report Builder and SSRS\)](#)
- [Filter Equation Examples \(Report Builder and SSRS\)](#)
- [Group Expression Examples \(Report Builder and SSRS\)](#)
- [Expression Dialog Box \(Report Builder\)](#)
- [Expression Examples \(Report Builder and SSRS\)](#)
- [Add Code to a Report \(SSRS\)](#)

Expression Examples (Report Builder and SSRS)

4/10/2017 • 20 min to read • [Edit Online](#)

Expressions are used frequently in Reporting Services paginated reports to control content and report appearance. Expressions are written in Microsoft Visual Basic, and can use built-in functions, custom code, report and group variables, and user-defined variables. Expressions begin with an equal sign (=). For more information about the expression editor and the types of references that you can include, see [Expression Uses in Reports \(Report Builder and SSRS\)](#), and [Add an Expression \(Report Builder and SSRS\)](#).

IMPORTANT

When RDL Sandboxing is enabled, only certain types and members can be used in expression text at report publish time. For more information, see [Enable and Disable RDL Sandboxing](#).

This topic provides examples of expressions that can be used for common tasks in a report.

- [Visual Basic Functions](#) Examples for date, string, conversion and conditional Visual Basic functions.
- [Report Functions](#) Examples for aggregates and other built-in report functions.
- [Appearance of Report Data](#) Examples for changing the appearance of a report.
- [Properties](#) Examples for setting report item properties to control format or visibility.
- [Parameters](#) Examples for using parameters in an expression.
- [Custom Code](#) Examples of embedded custom code.

For expression examples for specific uses, see the following topics:

- [Group Expression Examples \(Report Builder and SSRS\)](#)
- [Filter Equation Examples \(Report Builder and SSRS\)](#)
- [Commonly Used Filters \(Report Builder and SSRS\)](#)
- [Report and Group Variables Collections References \(Report Builder and SSRS\)](#)

For more information about simple and complex expressions, where you can use expressions, and the types of references that you can include in an expression, see topics under [Expressions \(Report Builder and SSRS\)](#). For more information about the context in which expressions are evaluated for calculating aggregates, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

To learn how to write expressions that use many of the functions and operators also used by expression examples in this topic, but in the context of writing a report, see [Tutorial: Introducing Expressions](#).

Functions

Many expressions in a report contain functions. You can format data, apply logic, and access report metadata using these functions. You can write expressions that use functions from the Microsoft Visual Basic run-time library, and from the [Convert](#) and [Math](#) namespaces. You can add references to functions from other assemblies or custom code. You can also use classes from the Microsoft .NET Framework, including [System.Text.RegularExpressions](#).

Visual Basic Functions

You can use Visual Basic functions to manipulate the data that is displayed in text boxes or that is used for parameters, properties, or other areas of the report. This section provides examples demonstrating some of these functions. For more information, see [Visual Basic Runtime Library Members](#) on MSDN.

The .NET Framework provides many custom format options, for example, for specific date formats. For more information, see [Formatting Types](#) on MSDN.

Math Functions

- The **Round** function is useful to round numbers to the nearest integer. The following expression rounds a 1.3 to 1:

```
= Round(1.3)
```

You can also write an expression to round a value to a multiple that you specify, similar to the **MRound** function in Excel. Multiply the value by a factor that creates an integer, round the number, and then divide by the same factor. For example, to round 1.3 to the nearest multiple of .2 (1.4), use the following expression:

```
= Round(1.3*5)/5
```

Date Functions

- The **Today** function provides the current date. This expression can be used in a text box to display the date on the report, or in a parameter to filter data based on the current date.

```
=Today()
```

- Use the **DateInterval** function to pull out a specific part of a date. Here are some valid **DateInterval** parameters:

- **DateInterval.Second**
- **DateInterval.Minute**

- DateInterval.Hour
- DateInterval.Weekday
- DateInterval.Day
- DateInterval.DayOfYear
- DateInterval.WeekOfYear
- DateInterval.Month
- DateInterval.Quarter
- DateInterval.Year

For example, this expression will show the number of the week in the current year for today's date:

```
=DatePart(DateInterval.WeekOfYear, today())
```

- The **DateAdd** function is useful for supplying a range of dates based on a single parameter. The following expression provides a date that is six months after the date from a parameter named *StartDate*.

```
=DateAdd(DateInterval.Month, 6, Parameters!StartDate.Value)
```

- The **Year** function displays the year for a particular date. You can use this to group dates together or to display the year as a label for a set of dates. This expression provides the year for a given group of sales order dates. The **Month** function and other functions can also be used to manipulate dates. For more information, see the Visual Basic documentation.

```
=Year(Fields!OrderDate.Value)
```

- You can combine functions in an expression to customize the format. The following expression changes the format of a date in the form month-day-year to month-week-week number. For example, 12/23/2009 to December Week 3:

```
=Format(Fields!MyDate.Value, "MMMM") & " Week " &  
(Int(DateDiff("d", DateSerial(Year(Fields!MyDate.Value),  
Month(Fields!MyDate.Value),1), Fields!FullDateAlternateKey.Value)/7)+1).ToString
```

When used as a calculated field in a dataset, you can use this expression on a chart to aggregate values by week within each month.

- The following expression formats the *SellStartDate* value as MMM-YY. *SellStartDate* field is a datetime data type.

```
=FORMAT(Fields!SellStartDate.Value, "MMM-yy")
```

- The following expression formats the *SellStartDate* value as dd/MM/yyyy. The *SellStartDate* field is a datetime data type.

```
=FORMAT(Fields!SellStartDate.Value, "dd/MM/yyyy")
```

- The **CDate** function converts the value to a date. The **Now** function returns a date value containing the current date and time according to your system. **DateDiff** returns a Long value specifying the number of time intervals between two Date values.

The following example displays the start date of the current year

```
=DateAdd(DateInterval.Year, DateDiff(DateInterval.Year, CDate("01/01/1900"), Now()), CDate("01/01/1900"))
```

- The following example displays the start date for the previous month based on the current month.

```
=DateAdd(DateInterval.Month, DateDiff(DateInterval.Month, CDate("01/01/1900"), Now())-1, CDate("01/01/1900"))
```

- The following expression generates the interval years between *SellStartDate* and *LastReceiptDate*. These fields are in two different datasets, *DataSet1* and *DataSet2*. The [First Function \(Report Builder and SSRS\)](#), which is an aggregate function, returns the first value of *SellStartDate* in *DataSet1* and the first value of *LastReceiptDate* in *DataSet2*.

```
=DATEDIFF("yyyy", First(Fields!SellStartDate.Value, "DataSet1"), First(Fields!LastReceiptDate.Value, "DataSet2"))
```

- The **DatePart** function returns an Integer value containing the specified component of a given Date value. The following expression returns the year for the first value of the *SellStartDate* in *DataSet1*. The dataset scope is specified because there are multiple datasets in the report.

```
=Datepart("yyyy", First(Fields!SellStartDate.Value, "DataSet1"))
```

- The **DateSerial** function returns a Date value representing a specified year, month, and day, with the time information set to midnight. The following example displays the ending date for the prior month, based on the current month.

```
=DateSerial(Year(Now()), Month(Now()), "1").AddDays(-1)
```

- The following expressions display various dates based on a date parameter value selected by the user.

EXAMPLE DESCRIPTION	EXAMPLE
Yesterday	=DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value)-1)
Two Days Ago	=DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value)-2)
One Month Ago	=DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value)-1,Day(Parameters!TodaysDate.Value))
Two Months Ago	=DateSerial(Year(Parameters!TodaysDate.Value),Month(Parameters!TodaysDate.Value)-2,Day(Parameters!TodaysDate.Value))
One Year Ago	=DateSerial(Year(Parameters!TodaysDate.Value)-1,Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value))
Two Years Ago	=DateSerial(Year(Parameters!TodaysDate.Value)-2,Month(Parameters!TodaysDate.Value),Day(Parameters!TodaysDate.Value))

String Functions

- Combine more than one field by using concatenation operators and Visual Basic constants. The following expression returns two fields, each on a separate line in the same text box:

```
=Fields!FirstName.Value & vbCrLf & Fields!LastName.Value
```

- Format dates and numbers in a string with the **Format** function. The following expression displays values of the *StartDate* and *EndDate* parameters in long date format:

```
=Format(Parameters!StartDate.Value, "D") & " through " & Format(Parameters!EndDate.Value, "D")
```

If the text box contains only a date or number, you should use the **Format** property of the text box to apply formatting instead of the **Format** function within the text box.

- The **Right**, **Len**, and **InStr** functions are useful for returning a substring, for example, trimming *DOMAIN\username* to just the user name. The following expression returns the part of the string to the right of a backslash (\) character from a parameter named *User*:

```
=Right(Parameters!User.Value, Len(Parameters!User.Value) - InStr(Parameters!User.Value, "\"))
```

The following expression results in the same value as the previous one, using members of the .NET Framework **String** class instead of Visual Basic functions:

```
=Parameters!User.Value.Substring(Parameters!User.Value.IndexOf("\") + 1, Parameters!User.Value.Length - Parameters!User.Value.IndexOf("\") - 1)
```

- Display the selected values from a multivalue parameter. The following example uses the **Join** function to concatenate the selected values of the parameter *MySelection* into a single string that can be set as an expression for the value of a text box in a report item:

```
= Join(Parameters!MySelection.Value)
```

The following example does the same as the above example, as well as displays a text string prior to the list of selected values.

```
="Report for " & JOIN(Parameters!MySelection.Value, " & ")
```

- The **Regex** functions from the .NET Framework **System.Text.RegularExpressions** are useful for changing the format of existing strings, for example, formatting a telephone number. The following expression uses the **Replace** function to change the format of a ten-digit telephone number in a field from "nnn-nnn-nnnn" to "(nnn) nnn-nnnn":

```
=System.Text.RegularExpressions.Regex.Replace(Fields!Phone.Value, "(\d{3})[ -.]*(\d{3})[ -.]*(\d{4})", "($1) $2-$3")
```

NOTE

Verify that the value for *Fields!Phone.Value* has no extra spaces and is of type **String**.

Lookup

- By specifying a key field, you can use the **Lookup** function to retrieve a value from a dataset for a one-to-one relationship, for example, a key-value pair. The following expression displays the product name from a dataset ("Product"), given the product identifier to match on:

```
=Lookup(Fields!PID.Value, Fields!ProductID.Value, Fields!ProductName.Value, "Product")
```

LookupSet

- By specifying a key field, you can use the **LookupSet** function to retrieve a set of values from a dataset for a one-to-many relationship. For example, a person can have multiple telephone numbers. In the following example, assume the dataset *PhoneList* contains a person identifier and a telephone number in each row. **LookupSet** returns an array of values. The following expression combines the return values into a single string and displays the list

of telephone numbers for the person specified by ContactID:

```
=Join(LookupSet(Fields!ContactID.Value, Fields!PersonID.Value, Fields!PhoneNumber.Value, "PhoneList"), ",")
```

Conversion Functions

You can use Visual Basic functions to convert a field from the one data type to a different data type. Conversion functions can be used to convert the default data type for a field to the data type needed for calculations or to combine text.

- The following expression converts the constant 500 to type Decimal in order to compare it to a Transact-SQL money data type in the Value field for a filter expression.

```
=CDec(500)
```

- The following expression displays the number of values selected for the multivalue parameter *MySelection*.

```
=CStr(Parameters!MySelection.Count)
```

Decision Functions

- The **If** function returns one of two values depending on whether the expression is true or not. The following expression uses the **If** function to return a Boolean value of **True** if the value of *LineTotal* exceeds 100. Otherwise it returns **False**:

```
=IIF(Fields!LineTotal.Value > 100, True, False)
```

- Use multiple **IIf** functions (also known as "nested IIfs") to return one of three values depending on the value of *PctComplete*. The following expression can be placed in the fill color of a text box to change the background color depending on the value in the text box.

```
=IIF(Fields!PctComplete.Value >= 10, "Green", IIF(Fields!PctComplete.Value >= 1, "Blue", "Red"))
```

Values greater than or equal to 10 display with a green background, between 1 and 9 display with a blue background, and less than 1 display with a red background.

- A different way to get the same functionality uses the **Switch** function. The **Switch** function is useful when you have three or more conditions to test. The **Switch** function returns the value associated with the first expression in a series that evaluates to true:

```
=Switch(Fields!PctComplete.Value >= 10, "Green", Fields!PctComplete.Value >= 1, "Blue", Fields!PctComplete.Value = 1, "Yellow", Fields!PctComplete.Value <= 0, "Red")
```

Values greater than or equal to 10 display with a green background, between 1 and 9 display with a blue background, equal to 1 display with a yellow background, and 0 or less display with a red background.

- Test the value of the *ImportantDate* field and return "Red" if it is more than a week old, and "Blue" otherwise. This expression can be used to control the Color property of a text box in a report item:

```
=IIF(DateDiff("d", Fields!ImportantDate.Value, Now())>7, "Red", "Blue")
```

- Test the value of the *PhoneNumber* field and return "No Value" if it is **null (Nothing)** in Visual Basic; otherwise return the phone number value. This expression can be used to control the value of a text box in a report item.

```
=IIF(Fields!PhoneNumber.Value Is Nothing, "No Value", Fields!PhoneNumber.Value)
```

- Test the value of the *Department* field and return either a subreport name or a **null (Nothing)** in Visual Basic. This expression can be used for conditional drillthrough subreports.

```
=IIF(Fields!Department.Value = "Development", "EmployeeReport", Nothing)
```

- Test if a field value is null. This expression can be used to control the **Hidden** property of an image report item. In the following example, the image specified by the field [LargePhoto] is displayed only if the value of the field is not null.

```
=IIF(IsNothing(Fields!LargePhoto.Value), True, False)
```

- The **MonthName** function returns a string value containing the name of the specified month. The following example displays NA in the Month field when the field contains the value of 0.

```
IIF(Fields!Month.Value=0,"NA",MonthName(IIF(Fields!Month.Value=0,1,Fields!Month.Value)))
```

Report Functions

In an expression, you can add a reference to additional report functions that manipulate data in a report. This section provides examples for two of these functions. For more information about report functions and examples, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

Sum

- The **Sum** function can total the values in a group or data region. This function can be useful in the header or footer of a group. The following expression

displays the sum of data in the Order group or data region:

```
=Sum(Fields!LineTotal.Value, "Order")
```

- You can also use the **Sum** function for conditional aggregate calculations. For example, if a dataset has a field that is named State with possible values Not Started, Started, Finished, the following expression, when placed in a group header, calculates the aggregate sum for only the value Finished:

```
=Sum(IIF(Fields!State.Value = "Finished", 1, 0))
```

RowNumber

- The **RowNumber** function, when used in a text box within a data region, displays the row number for each instance of the text box in which the expression appears. This function can be useful to number rows in a table. It can also be useful for more complex tasks, such as providing page breaks based on number of rows. For more information, see [Page Breaks](#) in this topic.

The scope you specify for **RowNumber** controls when renumbering begins. The **Nothing** keyword indicates that the function will start counting at the first row in the outermost data region. To start counting within nested data regions, use the name of the data region. To start counting within a group, use the name of the group.

```
=RowNumber(Nothing)
```

Average of Report Data

You can use expressions to manipulate how data appears on a report. For example, you can display the values of two fields in a single text box, display information about the report, or affect how page breaks are inserted in the report.

Page Headers and Footers

When designing a report, you may want to display the name of the report and page number in the report footer. To do this, you can use the following expressions:

- The following expression provides the name of the report and the time it was run. It can be placed in a text box in the report footer or in the body of the report. The time is formatted with the .NET Framework formatting string for short date:

```
=Globals.ReportName & ", dated " & Format(Globals.ExecutionTime, "d")
```

- The following expression, placed in a text box in the footer of a report, provides page number and total pages in the report:

```
=Globals.PageNumber & " of " & Globals.TotalPages
```

The following examples describe how to display the first and last values from a page in the page header, similar to what you might find in a directory listing. The example assumes a data region that contains a text box named `LastName`.

- The following expression, placed in a text box on the left side of the page header, provides the first value of the `LastName` text box on the page:

```
=First(ReportItems("LastName").Value)
```

- The following expression, placed in a text box on the right side of the page header, provides the last value of the `LastName` text box on the page:

```
=Last(ReportItems("LastName").Value)
```

The following example describes how to display a page total. The example assumes a data region that contains a text box named `Cost`.

- The following expression, placed in the page header or footer, provides the sum of the values in the `Cost` text box for the page:

```
=Sum(ReportItems("Cost").Value)
```

NOTE

You can refer to only one report item per expression in a page header or footer. Also, you can refer to the text box name, but not the actual data expression within the text box, in page header and footer expressions.

Page Breaks

In some reports, you may want to place a page break at the end of a specified number of rows instead of, or in addition to, on groups or report items. To do this, create a group that contains the groups or detail records you want, add a page break to the group, and then add a group expression to group by a specified number of rows.

- The following expression, when placed in the group expression, assigns a number to each set of 25 rows. When a page break is defined for the group, this expression results in a page break every 25 rows.

```
=Ceiling(RowNumber(Nothing)/25)
```

To allow the user to set a value for the number of rows per page, create a parameter named `RowsPerPage` and base the group expression on the parameter, as shown in the following expression:

```
=Ceiling(RowNumber(Nothing)/Parameters!RowsPerPage.Value)
```

For more information about setting page breaks for a group, see [Add a Page Break \(Report Builder and SSRS\)](#).

Properties

Expressions are not only used to display data in text boxes. They can also be used to change how properties are applied to report items. You can change style information for a report item, or change its visibility.

Formatting

- The following expression, when used in the Color property of a text box, changes the color of the text depending on the value of the `Profit` field:

```
=IIf(Fields!Profit.Value < 0, "Red", "Black")
```

You can also use the Visual Basic object variable `Me`. This variable is another way of referring to the value of a text box.

```
=IIf(Me.Value < 0, "Red", "Black")
```

- The following expression, when used in the BackgroundColor property of a report item in a data region, alternates the background color of each row between pale green and white:

```
=IIf(RowNumber(Nothing) Mod 2, "PaleGreen", "White")
```

If you are using an expression for a specified scope, you may have to indicate the dataset for the aggregate function:

```
=IIf(RowNumber("Employees") Mod 2, "PaleGreen", "White")
```

NOTE

Available colors come from the .NET Framework KnownColor enumeration.

Chart Colors

To specify colors for a Shape chart, you can use custom code to control the order that colors are mapped to data point values. This helps you use consistent colors for multiple charts that have the same category groups. For more information, see [Specify Consistent Colors across Multiple Shape Charts \(Report Builder and SSRS\)](#).

Visibility

You can show and hide items in a report using the visibility properties for the report item. In a data region such as a table, you can initially hide detail rows based on the value in an expression.

- The following expression, when used for initial visibility of detail rows in a group, shows the detail rows for all sales exceeding 90 percent in the `PctQuota` field:

```
=IIf(Fields!PctQuota.Value>.9, False, True)
```

- The following expression, when set in the Hidden property of a table, shows the table only if it has more than 12 rows:

```
=IIF(CountRows()>12,false,true)
```

- The following expression, when set in the **Hidden** property of a column, shows the column only if the field exists in the report dataset after the data is retrieved from the data source:

```
=IIF(Fields!Column_1.IsMissing, true, false)
```

URLs

You can customize URLs by using report data and also conditionally control whether URLs are added as an action for a text box.

- The following expression, when used as an action on a text box, generates a customized URL that specifies the dataset field `EmployeeID` as a URL parameter.

```
="http://adventure-works/MyInfo?ID=" & Fields!EmployeeID.Value
```

For more information, see [Add a Hyperlink to a URL \(Report Builder and SSRS\)](#).

- The following expression conditionally controls whether to add a URL in a text box. This expression depends on a parameter named `IncludeURLs` that allows a user to decide whether to include active URLs in a report. This expression is set as an action on a text box. By setting the parameter to False and then viewing the report, you can export the report Microsoft Excel without hyperlinks.

```
=IIF(Parameters!IncludeURLs.Value, "http://adventure-works.com/productcatalog", Nothing)
```

Report Data

Expressions can be used to manipulate the data that is used in the report. You can refer to parameters and other report information. You can even change the

query that is used to retrieve data for the report.

Parameters

You can use expressions in a parameter to vary the default value for the parameter. For example, you can use a parameter to filter data to a particular user based on the user ID that is used to run the report.

- The following expression, when used as the default value for a parameter, collects the user ID of the person running the report:

```
=User!UserID
```

- To refer to a parameter in a query parameter, filter expression, text box, or other area of the report, use the **Parameters** global collection. This example assumes that the parameter is named *Department*:

```
=Parameters!Department.Value
```

- Parameters can be created in a report but set to hidden. When the report runs on the report server, the parameter does not appear in the toolbar and the report reader cannot change the default value. You can use a hidden parameter set to a default value as custom constant. You can use this value in any expression, including a field expression. The following expression identifies the field specified by the default parameter value for the parameter named *ParameterField*:

```
=Fields(Parameters!ParameterField.Value).Value
```

Custom Code

You can use custom code in a report. Custom code is either embedded in a report or stored in a custom assembly which is used in the report. For more information about custom code, see [Custom Code and Assembly References in Expressions in Report Designer \(SSRS\)](#).

Using Group Variables for Custom Aggregation

You can initialize the value for a group variable that is local to a particular group scope and then include a reference to that variable in expressions. One of the ways that you can use a group variable with custom code is to implement a custom aggregate. For more information, see [Using Group Variables in Reporting Services 2008 for Custom Aggregation](#).

For more information about variables, see [Report and Group Variables Collections References \(Report Builder and SSRS\)](#).

Suppressing Null or Zero Values at Run Time

Some values in an expression can evaluate to null or undefined at report processing time. This can create run-time errors that result in #Error displaying in the text box instead of the evaluated expression. The IIF function is particularly sensitive to this behavior because, unlike an If-Then-Else statement, each part of the IIF statement is evaluated (including function calls) before being passed to the routine that tests for true or false. The statement

```
=IIF(Fields!Sales.Value is NOTHING, 0, Fields!Sales.Value)
```

To avoid this condition, use one of the following strategies:

- Set the numerator to 0 and the denominator to 1 if the value for field B is 0 or undefined; otherwise, set the numerator to the value for field A and the denominator to the value for field B.

```
=IIF(Field!B.Value=0, 0, Field!A.Value / IIF(Field!B.Value =0, 1, Field!B.Value))
```

- Use a custom code function to return the value for the expression. The following example returns the percentage difference between a current value and a previous value. This can be used to calculate the difference between any two successive values and it handles the edge case of the first comparison (when there is no previous value) and cases whether either the previous value or the current value is **null (Nothing in Visual Basic)**.

```
Public Function GetDeltaPercentage(ByVal PreviousValue, ByVal CurrentValue) As Object
    If IsNothing(PreviousValue) OR IsNothing(CurrentValue) Then
        Return Nothing
    Else if PreviousValue = 0 OR CurrentValue = 0 Then
        Return Nothing
    Else
        Return (CurrentValue - PreviousValue) / CurrentValue
    End If
End Function
```

The following expression shows how to call this custom code from a text box, for the "ColumnGroupByYear" container (group or data region).

```
=Code.GetDeltaPercentage(Previous(Sum(Fields!Sales.Value),"ColumnGroupByYear"), Sum(Fields!Sales.Value))
```

This helps to avoid run-time exceptions. You can now use an expression like `=IIF(Me.Value < 0, "red", "black")` in the **Color** property of the text box to conditionally display text based on whether the values are greater than or less than 0.

See Also

[Filter Equation Examples \(Report Builder and SSRS\)](#)

[Group Expression Examples \(Report Builder and SSRS\)](#)

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Commonly Used Filters \(Report Builder and SSRS\)](#)

Expression Uses in Reports (Report Builder and SSRS)

3/24/2017 • 6 min to read • [Edit Online](#)

In Reporting Services paginated reports, expressions are used throughout the report definition to specify or calculate values for parameters, queries, filters, report item properties, group and sort definitions, text box properties, bookmarks, document maps, dynamic page header and footer content, images, and dynamic data source definitions. This topic provides examples of the many places you can use expressions to vary the content or appearance of a report. This list is not comprehensive. You can set an expression for any property in a dialog box that displays the expression (fx) button or in a drop-down list that displays <Expression...>.

Expressions can be simple or complex. *Simple expressions* contain a reference to a single dataset field, parameter, or built-in field. Complex expressions can contain multiple built-in references, operators, and function calls. For example, a complex expression might include the Sum function applied to the Sales field.

Expressions are written in Microsoft Visual Basic. An expression begins with an equal sign (=) followed by a combination of references to built-in collections such as dataset fields and parameters, constants, functions, and operators.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Using Simple Expressions

Simple expressions appear on the design surface and in dialog boxes in brackets, for example, a dataset field appears as [ProductID]. Simple expressions are created for you automatically when you drag a field from a dataset onto a text box. A placeholder is created and the expression defines the underlying value. You can also type expressions directly into a data region cell or text box, both on the design surface or in a dialog box, (for example, [ProductID]).

The following table lists examples of the ways you can use simple expressions. The table describes the functionality, the property to set, the dialog box you typically use to set it, and the value for the property. You can type the simple expression directly on the design surface, in a dialog box, or in the Properties pane, or you can edit it in the Expression dialog box, just as you would with any expression.

FUNCTIONALITY	PROPERTY, CONTEXT, AND DIALOG BOX	PROPERTY VALUE
Specify a dataset field to display in a text box.	Value property for a placeholder inside a text box. Use Placeholder Properties Dialog Box, General .	[Sales]
Aggregate values for a group.	Value property for a placeholder inside a row associated with a tablix group. Use Textbox Properties Dialog Box .	[Sum(Sales)]
Include a page number.	Value property for a placeholder inside a text box that is placed in a page header. Use Textbox Properties Dialog Box, General .	[&PageNumber]
Display a selected parameter value.	Value property for a placeholder inside a text box on the design surface. Use Textbox Properties Dialog Box, General .	[@SalesThreshold]
Specify a group definition for a data region.	Group expression on the tablix group. Use Tablix Group Properties Dialog Box, General .	[Category]
Exclude a specific field value from a table.	Filter equation on the tablix. Use Tablix Properties Dialog Box, Filters .	For data type, select Integer . [Quantity] > 100
Include only a specific value for a group filter.	Filter equation on the tablix group. Use Tablix Group Properties Dialog Box, Filters .	[Category] = Clothing
Exclude specific values for more than one field from a dataset.	Filter equation for a group in a tablix. Use Tablix Properties Dialog Box, Filters .	= [Color] <> Red =[Color] <> Blue

FUNCTIONALITY	PROPERTY, CONTEXT, AND DIALOG BOX	PROPERTY VALUE
Specify sort order based on an existing field in a table.	Sort expression on the tablix. Use Tablix Properties Dialog Box, Sorting .	[SizeSortOrder]
Link a query parameter to a report parameter.	Parameters collection on the dataset. Use Dataset Properties Dialog Box, Parameters .	[@Category] [@Category]
Pass a parameter from a main report to a subreport.	Parameters collection on the subreport. Use Subreport Properties Dialog Box, Parameters .	[@Category] [@Category]

Using Complex Expressions

Complex expressions can contain multiple built-in references, operators, and function calls, and appear on the design surface as <>Expr><. To see or change the expression text, you must open the **Expression** dialog box or type directly in the Properties pane. The following table lists typical ways you can use a complex expression to display or organize data or change report appearance, including the property to set, the dialog box you typically use to set it, and the value for the property. You can type an expression directly into a dialog box, on the design surface, or in the Properties pane.

FUNCTIONALITY	PROPERTY, CONTEXT, AND DIALOG BOX	PROPERTY VALUE
Calculate aggregate values for a dataset.	Value property for a placeholder inside of a text box. Use Placeholder Properties Dialog Box, General .	=First(Fields!Sales.Value,"DataSet1")
Concatenate text and expressions in the same text box.	Value for a placeholder inside of a text box that is placed in a page header or page footer. Use Placeholder Properties Dialog Box, General .	="This report began processing at " & Globals!ExecutionTime
Calculate an aggregate value for a dataset in a different scope.	Value for a placeholder inside of a text box that is placed in a tablix group. Use Placeholder Properties Dialog Box, General .	=Max(Fields!Total.Value,"DataSet2")
Format data in a text box depending on value.	Color for a placeholder inside of a text box in the details row for a tablix. Use Text Box Properties Dialog Box, Font .	=IIF(Fields!TotalDue.Value < 10000, "Red", "Black")
Calculate a value once to refer to throughout the report.	Value for a report variable. Use Report Properties Dialog Box, Variables .	=Variables!MyCalculation.Value
Include specific values for more than one field from a dataset.	Filter equation for a group in a tablix. Use Tablix Properties Dialog Box, Filters .	For data type, select Boolean . =IIF(InStr(Fields!Subcat.Value,"Shorts")=0 AND (Fields!Size.Value="M" OR Fields!Size.Value="S"),TRUE, FALSE) = TRUE
Hide a text box on the design surface, that can be toggled by the user using a Boolean parameter named Show.	Hiddenproperty on a text box. Use Text Box Properties Dialog Box, Visibility .	=Not Parameters! Show<boolean parameter>.Value
Specify dynamic page header or footer content.	Value for a placeholder inside of a text box that is placed in the page header or footer.	="Page " & Globals!PageNumber & " of " & Globals!TotalPages
Specify a data source dynamically by using a parameter.	Connection string on the Data source. Use Data Source Properties Dialog Box, General .	=Data Source=" & Parameters!ServerName.Value & ";initial catalog=AdventureWorks2012"
Identify all the values for a multivalue parameter chosen by the user.	Value for a placeholder inside of a text box. Use Tablix Properties Dialog Box, Filters .	=Join(Parameters!MyMultivalueParameter.Value, ",")
Specify page breaks for every 20 rows in a tablix with no other groups.	Group expression for a group in a tablix. Use Group Properties Dialog Box, Page Breaks . Select the option Between each instance of a group .	=Ceiling(RowNumber(Nothing)/20)
Specify conditional visibility based on a parameter.	Hidden property for a tablix. Use Tablix Properties Dialog Box, Visibility .	=Not Parameters!< boolean parameter >.Value
Specify a date formatted for a specific culture.	Value for a placeholder inside of a text box in a data region. Use Textbox Properties Dialog Box, General .	=Fields!OrderDate.Value.ToString(System.Globalization.Cult DE))
Concatenate a string and a number formatted as a percentage to two decimal places.	Value for a placeholder inside of a text box in a data region. Use Textbox Properties Dialog Box, General .	="Growth Percent: " & Format(Fields!Growth.Value,"p2")

See Also

- [Expressions \(Report Builder and SSRS\)](#)
- [Expression Examples \(Report Builder and SSRS\)](#)

[Report Parameters \(Report Builder and Report Designer\)](#)
[Filter Equation Examples \(Report Builder and SSRS\)](#)
[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)
[Page Headers and Footers \(Report Builder and SSRS\)](#)
[Formatting Text and Placeholders \(Report Builder and SSRS\)](#)
[Hide an Item \(Report Builder and SSRS\)](#)

Expression Scope for Totals, Aggregates, and Built-in Collections

3/29/2017 • 12 min to read • [Edit Online](#)

When you write expressions, you will find that the term *scope* is used in multiple contexts. Scope can specify the data to use for evaluating an expression, the set of text boxes on a rendered page, the set of report items that can be shown or hidden based on a toggle. You will see the term *scope* in topics that relate to expression evaluation, aggregate function syntax, conditional visibility, and also in error messages related to these areas. Use the following descriptions to help differentiate which meaning of *scope* applies:

- **Data scope** Data scope is a hierarchy of scopes that the report processor uses as it combines report data and report layout, and builds out data regions such as tables and charts on which to display the data. Understanding data scope helps you to get the results that you want when you do the following:
 - **Write expressions that use aggregate functions** Specify which data to aggregate. The location of the expression in the report influences which data is in scope for aggregate calculations.
 - **Add sparklines to a table or matrix** Specify a minimum and maximum range for chart axes to align nested instances in a table or matrix.
 - **Add indicators to a table or matrix** Specify a minimum and maximum scale for the gauge to align nested instances in a table or matrix.
 - **Write sort expressions** Specify a containing scope that you can use to synchronize sort order among multiple related report items.
- **Cell scope** Cell scope is the set of row and column groups in a tablix data region to which a cell belongs. By default, each tablix cell contains a text box. The value of the text box is the expression. The location of the cell indirectly determines which data scopes you can specify for aggregate calculations in the expression.
- **Report item scope** Report item scope refers to the collection of items on a rendered report page. The report processor combines data and report layout elements to produce a compiled report definition. During this process, data regions such as tables and matrices expand as needed to display all of the report data. The compiled report is then processed by a report renderer. The report renderer determines which report items appear on each page. On a report server, each page is rendered as you view it. When you export a report, all pages are rendered. Understanding report item scope helps you get the results that you want when you do the following:
 - **Add toggle items** Specify a text box to add the toggle that controls the visibility of a report item. You can only add a toggle to text boxes that are in the scope of the report item that you want to toggle.
 - **Write expressions in page headers and footers** Specify values in expressions in text boxes or other report items that appear on the rendered page.

Understanding scopes helps you to successfully write expressions that give you the results that you want.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Data Scope and Data Hierarchy

Data scope specifies a set of report data. Data scope has a natural hierarchy with an inherent containment relationship. Scopes higher on the hierarchy contain scopes that are lower on the hierarchy. The following list of data scopes describes the hierarchy in order from most data to least data:

- **Datasets, after dataset filters are applied** Specifies the report dataset linked to the data region or to a report item in the report body. The data used for aggregation is from the report dataset after dataset filter expressions are applied. For shared datasets, this means both the filters in the shared dataset definition and the filters in the shared dataset instance in the report.
- **Data regions** Specifies data from the data region after data region filter and sort expressions are applied. Group filters are not used when calculating aggregates for data regions.
- **Data region groups, after group filters are applied** Specifies the data after the group expressions and group filters are applied for the parent group and child groups. For a table, this is the row and column groups. For a chart, this is the series and category groups. For the purposes of identifying scope containment, every parent group contains its child groups.
- **Nested data regions** Specifies the data for the nested data region in the context of the cell to which it has been added, and after the nested data region filter and sort expressions have been applied.
- **Row and column groups for the nested data regions** Specifies the data after the nested data region group expressions and group filters have been applied.

Understanding containing and contained scopes is important when you write expressions that include aggregate functions.

Cell Scope and Expressions

When you specify a scope, you are indicating to the report processor which data to use for an aggregate calculation. Depending on the expression and the location of the expression, valid scopes might be a *containing scopes*, also known as parent scopes, or a *contained scopes*, also known as child or nested scopes. In general, you cannot specify an individual group instance in an aggregate calculation. You can specify an aggregate across all group instances.

As the report processor combines data from a report dataset with the tablix data region, it evaluates group expressions and creates the rows and columns that are needed to represent the group instances. The value of expressions in a text box in each tablix cell is evaluated in the context of the cell scope. Depending on the tablix structure, a cell can belong to multiple row groups and column groups. For aggregate functions, you can specify which scope to use by using one of the following scopes:

- **Default scope** The data that is in scope for calculations when the report processor evaluates an expression. The default scope is the innermost set of groups to which the cell or data point belongs. For a tablix data region, the set can include row and column groups. For a chart data region, the set can

include category and series groups.

- **Named scope** The name of a dataset, a data region, or a data region group that is in scope for the expression. For aggregate calculations, you can specify a containing scope. You cannot specify a named scope for both a row group and a column group in a single expression. You cannot specify a contained scope unless the expression is for an aggregate of an aggregate.

The following expression generates the interval years between SellStartDate and LastReceiptDate. These fields are in two different datasets, DataSet1 and DataSet2. The [First Function \(Report Builder and SSRS\)](#), which is an aggregate function, returns the first value of SellStartDate in DataSet1 and the first value of LastReceiptDate in DataSet2.

```
=DATEDIFF("yyyy", First(Fields!SellStartDate.Value, "DataSet1"), First(Fields!LastReceiptDate.Value, "DataSet2"))
```

- **Domain scope** Also called synchronization scope. A type of data scope that applies to expression evaluation for nested data regions. Domain scope is used to specify aggregates across all instances of a group so that nested instances can be aligned and easily compared. For example, you can align the range and height for sparklines embedded in a table so that the values line up.

In some locations of a report, you must specify a scope. For example, for a text box on the design surface, you must specify the name of the dataset to use: `=Max(Fields!Sales.Value, "dataset1")`. In other locations, there is an implicit default scope. For example, if you do not specify an aggregate for a text box in a group scope, the default aggregate First is used.

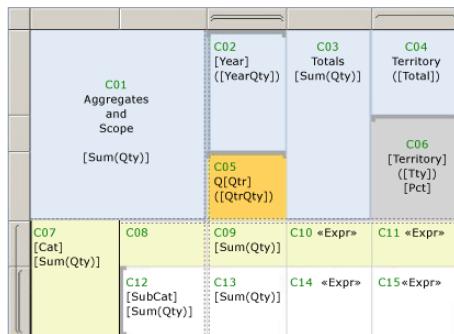
Each aggregate function topic lists the scopes that are valid for its use. For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

Example Aggregate Expressions for a Table Data Region

To write expressions that specify non-default scopes takes some practice. To help you understand different scopes, use the following figure and table. The figure labels each cell in a sales information table that displays quantity of items sold by year and quarter and also by sales territory. Note the visual cues on the row handles and column handles that display row and column group structure, indicating nested groups. The table has the following structure:

- A table header that contains the corner cell and three rows that include the column group headers.
- Two nested row groups based on category named Cat and subcategory named SubCat.
- Two nested column groups based on year named Year and quarter named Qtr.
- One static totals column labeled Totals.
- One adjacent column group based on sales territory named Territory.

The column header for the territory group has been split into two cells for display purposes. The first cell displays the territory name and totals, and the second cell has placeholder text that calculated the percentage contribution for each territory to all sales.



Assume the dataset is named DataSet1 and the table is named Tablix1. The following table lists the cell label, the default scope, and examples. The values for placeholder text are shown by in expression syntax.

CELL	DEFAULT SCOPE	PLACEHOLDER LABELS	TEXT OR PLACEHOLDER VALUES
C01	Tablix1	[Sum(Qty)]	Aggregates and Scope =Sum(Fields!Qty.Value)
C02	Outer column group "Year"	[Year] ([YearQty])	=Fields!Year.Value =Sum(Fields!Qty.Value)
C03	Tablix1	[Sum(Qty)]	Totals =Sum(Fields!Qty.Value)
C04	Peer column group "Territory"	([Total])	Territory =Sum(Fields!Qty.Value)
C05	Inner group "Qtr"	[Qtr] ([QtrQty])	Q =Fields!Qtr.Value =Sum(Fields!Qty.Value)

CELL	DEFAULT SCOPE	PLACEHOLDER LABELS	TEXT OR PLACEHOLDER VALUES
C06	Peer column group "Territory"	[Territory] ([Qty]) [Pct]	=Fields!Territory.Value =Sum(Fields!Qty.Value) =FormatPercent(Sum(Fields!Qty.Value,"Territory") & " of " & Sum(Fields!Qty.Value,"Tablix1"))
C07	Outer row group "Cat"	[Cat] [Sum(Qty)]	=Fields!Cat.Value =Sum(Fields!Qty.Value)
C08	Same as C07		
C09	Outer row group "Cat" and inner column group "Qtr"	[Sum(Qty)]	=Sum(Fields!Qty.Value)
C10	Same as C07	<<Expr>>	=Sum(Fields!Qty.Value) & ":" & FormatPercent(Sum(Fields!Qty.Value)/Sum(Fields!Qty.Value,"Territory") & " of " & Sum(Fields!Qty.Value,"Tablix1"))
C11	Outer row group "Cat" and column group "Territory"	<<Expr>>	=Sum(Fields!Qty.Value) & ":" & FormatPercent(Sum(Fields!Qty.Value)/Sum(Fields!Qty.Value,"Territory") & " of " & Sum(Fields!Qty.Value,"Territory"))
C12	Inner row group "Subcat"	[Subcat] [Sum(Qty)]	=Fields!SubCat.Value =Sum(Fields!Qty.Value)
C13	Inner row group "Subcat" and inner column group "Qtr"	[Sum(Qty)]	=Sum(Fields!Qty.Value)
C14	Inner row group "Subcat"	<<Expr>>	=Sum(Fields!Qty.Value) & ":" & FormatPercent(Sum(Fields!Qty.Value)/Sum(Fields!Qty.Value,"Cat") & " of " & Sum(Fields!Qty.Value,"Cat"))
C15	Inner row group "Subcat" and column group "Territory"	<<Expr>>	=Sum(Fields!Qty.Value) & ":" & FormatPercent(Code.CalcPercentage(Sum(Fields!Qty.Value,"Cat") & " of " & Sum(Fields!Qty.Value,"Cat")))

For more information about interpreting visual cues on tablix data regions, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder and SSRS\)](#). For more information about the tablix data region, see [Tablix Data Region Cells, Rows, and Columns \(Report Builder and SSRS\)](#). For more information about expressions and aggregates, see [Expression Uses in Reports \(Report Builder and SSRS\)](#) and [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

Synchronizing Scales for Sparklines

To compare values across time on the horizontal axis for a sparkline chart that is nested in a table or matrix, you can synchronize the category group values. This is called aligning axes. By selecting the option to align axes, the report automatically sets minimum and maximum values for an axis, and provides placeholders for aggregate values that do not exist in every category. This causes the values in the sparkline to line up across every category and enables you to compare values for each row of aggregated data. By selecting this option, you are changing the scope of the expression evaluation to the *domain scope*. Setting the domain scope for a nested chart also indirectly controls the color assignment for each category in the legend.

For example, in a sparkline that shows weekly trends, suppose that one city had sales data for 3 months and another city had sales data for 12 months. Without synchronized scales, the sparkline for first city would only have 3 bars and they would be much wider and occupy the same space as the 12 month set of bars for the second city.

For more information, see [Align the Data in a Chart in a Table or Matrix \(Report Builder and SSRS\)](#).

Synchronizing Ranges for Indicators

To specify the data values to use for a set of indicators, you must specify a scope. Depending on the layout of the data region that contains the indicator, you specify a scope or a containing scope. For example, in a group header row associated with category sales, a set of arrows (up, down, sideways) can indicate sales values relative to a threshold. The containing scope is the name of the table or matrix that contains the indicators.

For more information, see [Set Synchronization Scope \(Report Builder and SSRS\)](#).

Specifying Scopes from the Page Header or Page Footer

To display data that is different on each page of a report, you add expressions to a report item that must be on the rendered page. Because a report is split into pages while it is rendered, only during rendering can it be determined which items exist on a page. For example, a cell in a detail row has a text box that has many instances on a page.

For this purpose, there is a global collection called ReportItems. This is the set of text boxes on the current page.

For more information, see [Page Headers and Footers \(Report Builder and SSRS\)](#) and [ReportItems Collection References \(Report Builder and SSRS\)](#).

Specifying a Toggle Item for Drilldown and Conditional Visibility

Toggles are plus or minus sign images that are added to a text box and that a user can click to show or hide other report items. On the **Visibility** page for most report item properties, you can specify which report item to add the toggle to. The toggle item must be in a higher containment scope than the item to show or hide.

In a tablix data region, to create a drilldown effect where you click a text box to expand the table to show more data, you must set the **Visibility** property on the

group, and select as the toggle a text box in a group header that is associated with a containing group.

For more information, see [Add an Expand or Collapse Action to an Item \(Report Builder and SSRS\)](#).

Specifying a Sort Expression to Synchronize Sort Order

When you add an interactive sort button to a table column, you can synchronize sorting for multiple items that have a common containing scope. For example, you can add a sort button to a column header in a matrix, and specify the containing scope as the name of the dataset that is bound to the matrix. When a user clicks the sort button, not only are the matrix rows sorted, but also the chart series groups of charts that are bound to the same dataset are sorted. In this way, all data regions that depend on that dataset can be synchronized to show the same sort order.

For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).

Suppressing Null or Zero Values in a Cell

For many reports, calculations that are scoped to groups can create many cells that have zero (0) or null values. To reduce clutter in your report, add an expression to return blanks if the aggregate value is 0. For more information, see "Examples that Suppress Null or Zero Values" in [Expression Examples \(Report Builder and SSRS\)](#).

See Also

[Expression Examples \(Report Builder and SSRS\)](#)

[Group Expression Examples \(Report Builder and SSRS\)](#)

[Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Formatting Text and Placeholders \(Report Builder and SSRS\)](#)

Custom Code and Assembly References in Expressions in Report Designer (SSRS)

3/24/2017 • 9 min to read • [Edit Online](#)

You can add references to custom code embedded in a report or to custom assemblies that you build and save to your computer and deploy to the report server. Use embedded code for custom constants, complex functions or functions that are used multiple times in a single report. Use custom code assemblies to maintain code in a single place and share it for use by multiple reports. Custom code can include new custom constants, variables, functions, or subroutines. You can include read-only references to built-in collections such as the Parameters collection. However, you cannot pass sets of report data values to custom functions; specifically, custom aggregates are not supported.

IMPORTANT

For time-sensitive calculations that are evaluated once at run-time and that you want to remain the same value throughout report processing, consider whether to use a report variable or group variable. For more information, see [Report and Group Variables Collections References \(Report Builder and SSRS\)](#).

Report Designer is the preferred authoring environment to use to add custom code to a report. Report Builder supports processing reports that have valid expressions, or that include references to custom assemblies on a report server. Report Builder does not provide a way to add a reference to a custom assembly.

NOTE

Be aware that during an upgrade of a report server, reports that depend on custom assemblies might require additional steps to complete the upgrade.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Working with Custom Code in Report Builder

In Report Builder, you can open a report from a report server that includes references to custom assemblies. For example, you can edit reports that are created and deployed by using Report Designer in SQL Server Data Tools (SSDT). The custom assemblies must be deployed to the report server.

You cannot do the following:

1. Add references or class member instances to a report.
2. Preview a report with references to custom assemblies in local mode.

Including References to Commonly Used Functions

Use the **Expression** dialog box to view a categorized list of common functions built-in to Reporting Services.

When you expand **Common Functions** and click a category, the **Item** pane displays the list of functions that you

include in an expression. The common functions include classes from the .NET Framework [Math](#) and [Convert](#) namespaces and Visual Basic run-time library functions. For convenience, you can view the most commonly used functions in the **Expression** dialog box, where they are listed by category: Text, Date and Time, Math, Inspection, Program Flow, Aggregate, Financial, Conversion, and Miscellaneous. Less commonly used functions do not appear in the list but can still be used in an expression.

To use a built-in function, double-click the function name in the Item pane. A description of the function appears in the Description pane and an example of the function call appears in the Example pane. In the code pane, when you type the function name followed by a left parenthesis (, the IntelliSense help displays each valid syntax for the function call. For example, to calculate the maximum value for a field named `Quantity` in a table, add the simple expression `=Max(` to the Code pane, and then use the smart tags to view all possible valid syntaxes for the function call. To complete this example, type `=Max(Fields!Quantity.Value)`.

For more information about each function, see [Math](#), [Convert](#), and [Visual Basic Runtime Library Members](#) on MSDN.

Including References to Less Commonly Used Functions

To include a reference to other less commonly used CLR namespaces, you must use a fully qualified reference, for example, [StringBuilder](#). IntelliSense is not supported in the code pane of the **Expression** dialog box for these less commonly used functions.

For more information, see [Visual Basic Runtime Library Members](#) on MSDN.

Including References to External Assemblies

To include a reference to a class in an external assembly, you must identify the assembly for the report processor. Use the **References** page of the **Report Properties** dialog box to specify the fully qualified name of the assembly to add to the report. In your expression, you must use the fully qualified name for the class in the assembly. Classes in an external assembly do not appear in the **Expression** dialog box; you must provide the correct name for the class. A fully qualified name includes the namespace, the class name, and the member name.

Including Embedded Code

To add embedded code to a report, use the Code tab of the **Report Properties** dialog box. The code block you create can contain multiple methods. Methods in embedded code must be written in Microsoft Visual Basic and must be instance-based. The report processor automatically adds references for the System.Convert and System.Math namespaces. Use the **References** page of the **Report Properties** dialog box to add additional assembly references. For more information, see [Add an Assembly Reference to a Report \(SSRS\)](#).

Methods in embedded code are available through a globally defined **Code** member. You access these by referring to the **Code** member and the method name. The following example calls the method **ToUSD**, which converts the value in the `StandardCost` field to a dollar value:

```
=Code.ToUSD(Fields!StandardCost.Value)
```

To reference built-in collections in your custom code, include a reference to the built-in **Report** object:

```
=Report.Parameters!Param1.Value
```

The following examples show how to define some custom constants and variables.

```
Public Const MyNote = "Authored by Bob"  
Public Const NCopies As Int32 = 2  
Public Dim MyVersion As String = "123.456"  
Public Dim MyDoubleVersion As Double = 123.456
```

Although custom constants do not appear in the **Constants** category in the **Expression** dialog box (which only displays built-in constants), you can add references to them from any expression, as shown in the following examples. In an expression, a custom constant is treated as a **Variant**.

```
=Code.MyNote  
=Code.NCopies  
=Code.MyVersion  
=Code.MyDoubleVersion
```

The following example includes both the code reference and the code implementation of the function **FixSpelling**, which substitutes the text `"Bicycle"` for all occurrences of the text "Bike" in the `SubCategory` field.

```
=Code.FixSpelling(Fields!SubCategory.Value)
```

The following code, when embedded in a report definition code block, shows an implementation of the **FixSpelling** method. This example shows you how to use a fully qualified reference to the Microsoft .NET Framework **StringBuilder** class.

```
Public Function FixSpelling(ByVal s As String) As String  
    Dim strBuilder As New System.Text.StringBuilder(s)  
    If s.Contains("Bike") Then  
        strBuilder.Replace("Bike", "Bicycle")  
        Return strBuilder.ToString()  
    Else : Return s  
    End If  
End Function
```

For more information about built-in object collections and initialization, see [Built-in Globals and Users References \(Report Builder and SSRS\)](#) and [Initializing Custom Assembly Objects](#).

Including References to Parameters from Code

You can reference the global parameters collection via custom code in a Code block of the report definition or in a custom assembly that you provide. The parameters collection is read-only and has no public iterators. You cannot use a Visual Basic **For Each** construct to step through the collection. You need to know the name of the parameter defined in the report definition before you can reference it in your code. You can, however, iterate through all the values of a multi value parameter.

The following table includes examples of referencing the built-in collection `Parameters` from custom code:

Passing an entire global parameter collection to custom code. This function returns the value of a specific report parameter *MyParameter*.

Reference in Expression `=Code.DisplayAParameterValue(Parameters)`

Custom Code definition

```
Public Function DisplayAParameterValue(ByVal parameters as Parameters) as Object  
Return parameters("MyParameter").Value  
End Function
```

Passing an individual parameter to custom code.

Reference in Expression `=Code.ShowParameterValues(Parameters!DayOfWeek)`

This example returns the value of the parameter passed in. If the parameter is a multivalue parameter, the return string is a concatenation of all the values.

Custom Code definition

```
Public Function ShowParameterValues(ByVal parameter as Parameter)
    as String
    Dim s as String
    If parameter.IsMultiValue then
        s = "Multivalue: "
        For i as integer = 0 to parameter.Count-1
            s = s + CStr(parameter.Value(i)) + " "
        Next
    Else
        s = "Single value: " + CStr(parameter.Value)
    End If
    Return s
End Function
```

Including References to Code from Custom Assemblies

To use custom assemblies in a report, you must first create the assembly, make it available to Report Designer, add a reference to the assembly in the report, and then use an expression in the report to refer to the methods contained in that assembly. When the report is deployed to the report server, you must also deploy the custom assembly to the report server.

For information about creating a custom assembly and making it available to Reporting Services, see [Using Custom Assemblies with Reports](#).

To refer to custom code in an expression, you must call the member of a class within the assembly. How you do this depends on whether the method is static or instance-based. Static methods within a custom assembly are available globally within the report. You can access static methods in expressions by specifying the namespace, class, and method name. The following example calls the method **ToGBP**, which converts the value of the **StandardCost** value from dollar to pounds sterling:

```
=CurrencyConversion.DollarCurrencyConversion.ToGBP(Fields!StandardCost.Value)
```

Instance-based methods are available through a globally defined **Code** member. You access these by referring to the **Code** member, followed by the instance and method name. The following example calls the instance method **ToEUR**, which converts the value of **StandardCost** from dollar to euro:

```
=Code.m_myDollarConversion.ToEUR(Fields!StandardCost.Value)
```

NOTE

In Report Designer, a custom assembly is loaded once and is not unloaded until you close Visual Studio. If you preview a report, make changes to a custom assembly used in the report, and then preview the report again, the changes will not appear in the second preview. To reload the assembly, close and reopen Visual Studio and then preview the report.

For more information about accessing your code, see [Accessing Custom Assemblies Through Expressions](#).

Passing Built-in Collections into Custom Assemblies

If you want to pass built-in collections, such as the *Globals* or *Parameters* collection, into a custom assembly for processing, you must add an assembly reference in your code project to the assembly that defines the built-in collections and access the correct namespace. Depending on whether you are developing the custom assembly for a report that is run on a report server (server report) or a report that is run locally in a .NET application (local report), the assembly you need to reference is different. See below for details.

- **Namespace:** Microsoft.ReportingServices.ReportProcessing.ReportObjectModel
- **Assembly (local report):** Microsoft.ReportingServices.ProcessingObjectModel.dll
- **Assembly (server report):** Microsoft.ReportViewer.ProcessingObjectModel.dll

Since the content of the *Fields* and *ReportItems* collections can change dynamically at runtime, you should not hold onto them across calls into the custom assembly (for example, in a member variable). The same recommendation applies generally to all built-in collections.

See Also

- [Add Code to a Report \(SSRS\)](#)
- [Using Custom Assemblies with Reports](#)
- [Add an Assembly Reference to a Report \(SSRS\)](#)
- [Reporting Services Tutorials \(SSRS\)](#)
- [Expression Examples \(Report Builder and SSRS\)](#)
- [Report Samples \(Report Builder and SSRS\)](#)

Add an Assembly Reference to a Report (SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

When you embed custom code that contains references to Microsoft .NET Framework classes that are not in [Math](#) or [Convert](#), you must provide an assembly reference to the report so that the report processor can resolve the names. For more information, see [Add Code to a Report \(SSRS\)](#).

To add an assembly reference to a report

1. In **Design** view, right-click the design surface outside the border of the report and click **Report Properties**.
2. Click **References**.
3. In **Add or remove assemblies**, click **Add** and then click the ellipsis button to browse to the assembly.
4. In **Add or remove classes**, click **Add** and then type name of the class and provide an instance name to use within the report.

NOTE

Specify a class and instance name only for instance-based members. Do not specify static members in the **Classes** list. For more information, see [Custom Code and Assembly References in Expressions in Report Designer \(SSRS\)](#).

5. Click **OK**.

See Also

[Using Custom Assemblies with Reports](#)
[Report Properties Dialog Box, References](#)

Add Code to a Report (SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In any expression, you can call your own custom code. You can provide code in the following two ways:

- Embed code written in Visual Basic directly in your report. If your code refers to a Microsoft .NET Framework that is not [Math](#) or [Convert](#), you must add the reference to the report. For more information, see [Add an Assembly Reference to a Report \(SSRS\)](#). For more information about other references you can make from your code, see [Custom Code and Assembly References in Expressions in Report Designer \(SSRS\)](#).
- Provide a custom code assembly by using the .NET Framework. If you provide a custom assembly, you must install it on both the computer where you author the report and the report server where you view the report. For more information, see [Using Custom Assemblies with Reports](#).

To add embedded code to a report

1. In **Design** view, right-click the design surface outside the border of the report and click **Report Properties**.
2. Click **Code**.
3. In **Custom code**, type the code. Errors in the code produce warnings when the report runs. The following example creates a custom function named `ChangeWord` that replaces the word "Bike" with "Bicycle".

```
Public Function ChangeWord(ByVal s As String) As String
    Dim strBuilder As New System.Text.StringBuilder(s)
    If s.Contains("Bike") Then
        strBuilder.Replace("Bike", "Bicycle")
        Return strBuilder.ToString()
    Else : Return s
    End If
End Function
```

4. The following example shows how to pass a dataset field named Category to this function in an expression:

```
=Code.ChangeWord(Fields!Category.Value)
```

If you add this expression to a table cell that displays category values, whenever the word "Bike" is in the dataset field for that row, the table cell value displays the word "Bicycle" instead.

See Also

- [Report Properties Dialog Box, Code](#)
[Expression Examples \(Report Builder and SSRS\)](#)
[Parameters Collection References \(Report Builder and SSRS\)](#)

Expression Reference (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

Report expressions support a variety of references to built-in functions and built-in collections. Expressions must have valid Visual Basic syntax before a report can be published or processed.

To develop complex expressions or expressions that use custom code or custom assemblies, we recommend that you use Report Designer in SQL Server Data Tools (SSDT). For more information, see [Custom Code and Assembly References in Expressions in Report Designer \(SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Use the topics in this section to help write simple expressions in a report.

In This Section

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Constants in Expressions \(Report Builder and SSRS\)](#)

[Operators in Expressions \(Report Builder and SSRS\)](#)

[Built-in Collections in Expressions \(Report Builder and SSRS\)](#)

[Built-in Globals and Users References \(Report Builder and SSRS\)](#)

[Parameters Collection References \(Report Builder and SSRS\)](#)

[Dataset Fields Collection References \(Report Builder and SSRS\)](#)

[DataSources and DataSets Collection References \(Report Builder and SSRS\)](#)

[Report and Group Variables Collections References \(Report Builder and SSRS\)](#)

[ReportItems Collection References \(Report Builder and SSRS\)](#)

See Also

[Expressions \(Report Builder and SSRS\)](#)

Report Builder Functions - Aggregate Functions Reference

3/24/2017 • 8 min to read • [Edit Online](#)

To include aggregated values in your report, you can use built-in aggregate functions in expressions. The default aggregate function for numeric fields is SUM. You can edit the expression and use a different built-in aggregate function or specify a different scope. Scope identifies which set of data to use for the calculation.

As the report processor combines report data and the report layout, the expressions for each report item are evaluated. As you view each page of the report, you see the results for each expression in the rendered report items.

The following table lists categories of built-in functions that you can include in an expression:

- [Built-in Aggregate Functions](#)
- [Restrictions on Built-in Fields, Collections, and Aggregate Functions](#)
- [Restrictions on Nested Aggregates](#)
- [Calculating Running Values](#)
- [Retrieving Row Counts](#)
- [Looking Up Values from Another Dataset](#)
- [Retrieving Sort-Dependent Values](#)
- [Retrieving Server Aggregates](#)
- [Retrieving Recursive Level](#)
- [Testing for Scope](#)

To determine the valid scopes for a function, see the individual function reference topic. For more information and for examples, see [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Built-in Aggregate Functions

The following built-in functions calculate summary values for a set of non-null numeric data in the default scope or the named scope.

FUNCTION	DESCRIPTION
Avg	Returns the average of all non-null numeric values specified by the expression, evaluated in the given scope.

FUNCTION	DESCRIPTION
Count	Returns a count of non-null values specified by the expression, evaluated in the context of the given scope.
CountDistinct	Returns a count of all distinct non-null values specified by the expression, evaluated in the context of the given scope.
Max	Returns the maximum value of all non-null numeric values specified by the expression, in the context of the given scope. You can use this for specifying a chart axis maximum value to control the scale.
Min	Returns the minimum value of all non-null numeric values specified by the expression, in the context of the given scope. You can use this for specifying a chart axis minimum value to control the scale.
StDev	Returns the standard deviation of all non-null numeric values specified by the expression, evaluated in the given scope.
StDevP	Returns the population standard deviation of all non-null numeric values specified by the expression, evaluated in the context of the given scope.
Sum	Returns the sum of all the non-null numeric values specified by the expression, evaluated in the given scope.
Union	Returns the union of all the non-null spatial data values of type SqlGeometry or SqlGeography that are specified by the expression, evaluated in the given scope.
Var	Returns the variance of all non-null numeric values specified by the expression, evaluated in the given scope.
VarP	Returns the population variance of all non-null numeric values specified by the expression, evaluated in the context of the given scope.

 [Back to Top](#)

Restrictions on Built-in Fields, Collections, and Aggregate Functions

The following table summarizes restrictions in report locations on where you can add expressions that contain references to global built-in collections.

LOCATION IN REPORT	FIELDS	PARAMETERS	REPORTITEMS	PAGENUMBER	DATASOURCE	VARIABLES	RENDERFORMAT
				TOTALPAGES	DATASET		
Page Header	Yes	Yes	At most one	Yes	Yes	Yes	Yes
Page Footer			Note 1				
Body	Yes Note 2	Yes	Only items in the current scope or a containing scope	No	Yes	Yes	Yes
			Note 3				
Report Parameter	No	Only parameters earlier in the list Note 4	No	No	No	No	No
Field	Yes	Yes	No	No	No	No	No
Query Parameter	No	Yes	No	No	No	No	No
Group Expression	Yes	Yes	No	No	Yes	No	No
Sort Expression	Yes	Yes	No	No	Yes	Yes	No Note 5
Filter Expression	Yes	Yes	No	No	Yes	Yes	No Note 6
Code	No	Yes Note 7	No	No	No	No	No
Report.Language	No	Yes	No	No	No	No	No
Variables	Yes	Yes	No	No	Yes	Current or containing scope	No
Aggregates	Yes	Yes	Only in page header/page footer	Only in report item aggregates	Yes	No	No

LOCATION IN REPORT	FIELDS	PARAMETERS	REPORTITEMS	PAGENUMBER	DATASOURCE	DATASET	VARIABLES	RENDERFORMAT
				TOTALPAGES				
Lookup functions	Yes	Yes	Yes	No	Yes		No	No

- **Note 1.** ReportItems must exist in the rendered report page, or their value is Null. If the visibility of a report item depends on an expression that evaluates to False, the report item does not exist on the page.
- **Note 2.** If a field reference is used in a group scope, and the field reference is not included in the group expression, then the value for the field is undefined, unless there is only one value in the scope. To specify a value, use First or Last and the group scope.
- **Note 3.** Expressions that include a reference to ReportItems can specify values for other ReportItems in the same group scope or in a containing group scope.
- **Note 4.** Property values for earlier parameters might be null.
- **Note 5.** In Member sorts only. Cannot use in data region sort expressions.
- **Note 6.** In Member filters only. Cannot use in data region or dataset filter expressions.
- **Note 7.** The Parameters collection is not initialized until after the Code block is processed, so methods cannot be used to control parameters on initialization.
- **Note 8.** Data type for all aggregates except Count and CountDistinct must be the same data type, or null, for all values.

[Back to Top](#)

Restrictions on Nested Aggregates

The following table summarizes restrictions on which aggregates functions can specify other aggregate functions as nested aggregates.

CONTEXT	RUNNING VALUE	ROWNUMBER	FIRST LAST	PREVIOUS	SUM AND OTHER PRESORT FUNCTIONS	REPORTITEM AGGREGATES	LOOKUP FUNCTIONS	AGGREGATE FUNCTION
Running Value	No	No	No	No	Yes	No	Yes	No
First	No	No	No	No	Yes	No	No	No
Last								
Previous	Yes	Yes	Yes	No	Yes	No	Yes	No
Sum and other Presort functions	No	No	No	No	Yes	No	Yes	No

CONTEXT	RUNNING VALUE	ROWNUMBER	FIRST LAST	PREVIOUS	SUM AND OTHER PRESORT FUNCTIONS	REPORTITEM AGGREGATES	LOOKUP FUNCTIONS	AGGREGATE FUNCTION
ReportItem aggregates	No	No	No	No	No	No	No	No
Lookup functions	Yes	Yes Note 1	Yes Note 1	Yes Note 1	Yes Note 1	Yes Note 1	No	No
Aggregate Function	No	No	No	No	No	No	No	No

- **Note 1.** Aggregate functions are only allowed inside the *Source* expression of a Lookup function if the Lookup function is not contained in an aggregate. Aggregate functions are not allowed inside the *Destination* or *Result* expressions of a Lookup function.

[↑ Back to Top](#)

Calculating Running Values

The following built-in functions calculate running values for a set of data. **RowNumber** is like **RunningValue** in that it returns the running value of a count that increments for each row within the containing scope. The scope parameter for these functions must specify a containing scope, which controls when the count restarts.

FUNCTION	DESCRIPTION
RowNumber	Returns a running count of the number of rows for the specified scope. The RowNumber function restarts counting at 1, not 0.
RunningValue	Returns a running aggregate of all non-null numeric values specified by the expression, evaluated for the given scope.

[↑ Back to Top](#)

Retrieving Row Counts

The following built-in function calculates the number of rows in the given scope. Use this function to count all rows, including rows with null values.

FUNCTION	DESCRIPTION
CountRows	Returns the number of rows in the specified scope, including rows with null values.

[↑ Back to Top](#)

Looking Up Values from Another Dataset

The following lookup functions retrieve values from a specified dataset.

FUNCTION	DESCRIPTION
Lookup Function	Returns a value from a dataset for a specified expression.
LookupSet Function	Returns a set of values from a dataset for a specified expression.
Multilookup Function	Returns the set of first-match values for a set of names from a dataset that contains name/value pairs.

 [Back to Top](#)

Retrieving Sort-Dependent Values

The following built-in functions return the first, last, or previous value within a given scope. These functions depend on the sort order of the data values. Use these functions, for example, to find the first and last values on a page to create a dictionary-style page header. Use **Previous** to compare a value in one row to the previous row's value within a specific scope, for example, to find percentage year over year values in a table.

FUNCTION	DESCRIPTION
First	Returns the first value in the given scope of the specified expression.
Last	Returns the last value in the given scope of the specified expression.
Previous	Returns the value or the specified aggregate value for the previous instance of an item within the specified scope.

 [Back to Top](#)

Retrieving Server Aggregates

The following built-in function retrieves custom aggregates from the data provider. For example, using an Analysis Services data source type, you can retrieve aggregates calculated on the data source server for use in a group header.

FUNCTION	DESCRIPTION
Aggregate	Returns a custom aggregate of the specified expression, as defined by the data provider.

 [Back to Top](#)

Testing for Scope

The following built-in function tests the current context of a report item to see if it is a member of a specific scope.

FUNCTION	DESCRIPTION
InScope	Indicates whether the current instance of an item is within the specified scope.

 [Back to Top](#)

Retrieving Recursive Level

The following built-in function retrieves the current level when a recursive hierarchy is processed. Use the result of this function with the **Padding** property in a text box to control the indent level of a visual hierarchy for a recursive group. For more information, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

FUNCTION	DESCRIPTION
Level	Returns the current level of depth in a recursive hierarchy.

 [Back to Top](#)

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – Aggregate Function

3/24/2017 • 3 min to read • [Edit Online](#)

Returns a custom aggregate of the specified expression, as defined by the data provider.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Aggregate(expression, scope)
```

Parameters

expression

The expression on which to perform the aggregation. The expression must be a simple field reference.

scope

(**String**) The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. *Scope* must be a string constant and cannot be an expression. If *scope* is not specified, the current scope is used.

Return Type

Return type is determined by the data provider. Returns **Nothing** if the data provider does not support this function or data is not available.

Remarks

The **Aggregate** function provides a way to use aggregates that are calculated on the external data source. Support for this feature is determined by the data extension. For example, the SQL Server Analysis Services data processing extension retrieves flattened rowsets from an MDX query. Some rows in the result set can contain aggregate values calculated on the data source server. These are known as *server aggregates*. To view server aggregates in the graphical query designer for Analysis Services, you can use the **Show Aggregate** button on the toolbar. For more information, see [Analysis Services MDX Query Designer User Interface \(Report Builder\)](#).

When you display the combination of aggregate and detail dataset values on detail rows of a Tablix data region, server aggregates would not typically be included because they are not detail data. However, you may want to display all values retrieved for the dataset and customize the way aggregate data is calculated and displayed.

Reporting Services detects the use of the **Aggregate** function in expressions in your report in order to determine whether to display server aggregates on detail rows. If you include **Aggregate** in an expression in a data region, server aggregates can only appear on group total or grand total rows, not on detail rows. If you want to display server aggregates on detail rows, do not use the **Aggregate** function.

You can change this default behavior by changing the value of the **Interpret subtotals as details** option on the **Dataset Properties** dialog box. When this option is set to **True**, all data, including server aggregates, appears as detail data. When set to **False**, server aggregates appear as totals. The setting for this property affects all data

regions that are linked to this dataset.

NOTE

All containing groups for the report item that references **Aggregate** must have simple field references for their group expressions, for example, `[FieldName]`. You cannot use **Aggregate** in a data region that uses complex group expressions. For the SQL Server Analysis Services data processing extension, your query must include MDX fields of type **LevelProperty** (not **MemberProperty**) to support aggregation using the **Aggregate** function.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Comparing the Aggregate and Sum Functions

The **Aggregate** function differs from numeric aggregate functions like **Sum** in that the **Aggregate** function returns a value that is calculated by the data provider or data processing extension. Numeric aggregate functions like **Sum** return a value that is calculated by the report processor on a set of data from the dataset that is determined by the *scope* parameter. For more information, see the aggregate functions listed in [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

Example

The following code example shows an expression that retrieves a server aggregate for the field `LineTotal`. The expression is added to a cell in a row that belongs to the group `GroupbyOrder`.

```
=Aggregate(Fields!LineTotal.Value, "GroupbyOrder")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - Avg Function

3/24/2017 • 2 min to read • [Edit Online](#)

In Reporting Services paginated reports, returns the average of all non-null numeric values specified by the expression, evaluated in the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Avg(expression, scope, recursive)
```

Parameters

expression

(**Float**) The expression on which to perform the aggregation.

scope

(**String**) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns a **Decimal** for decimal expressions and a **Double** for all other expressions.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following two code examples provide an average of all values in the `Cost` field contained in a dataset named `Inventory`.

```
=Avg(Fields!Cost.Value, "Inventory")
OR
=Avg (Cdbl(Fields!Cost.Value), "Inventory")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – Count Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns a count of non-null values specified by the expression, evaluated in the context of the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Count(expression, scope, recursive)
```

Parameters

expression

(Variant or Binary) The expression on which to perform the aggregation, for example, `=Fields!FieldName.Value`.

scope

(String) The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(Enumerated Type) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns an **Integer**.

Remarks

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- *Scope* for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- *Scope* for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder\)](#)

and SSRS).

Example

Description

The following code example shows an expression that calculates the number of non-null values of `Size` for the default scope and for a parent group scope. The expression is added to a cell in a row that belongs to the child group `GroupbySubcategory`. The parent group is `GroupbyCategory`. The expression displays the results for `GroupbySubcategory` (the default scope) and then for `GroupbyCategory` (the parent group scope).

NOTE

Expressions should not contain actual carriage returns and line breaks; these are included in the example to support documentation renderers. If you copy the following example, remove carriage returns from each line.

Code

```
= "Count (Subcategory): " & Count(Fields!Size.Value) &  
"Count (Category): " & Count(Fields!Size.Value,"GroupbyCategory")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – CountDistinct Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns a count of all distinct non-null values specified by the expression, evaluated in the context of the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
CountDistinct(expression, scope, recursive)
```

Parameters

expression

(Variant) The expression on which to perform the aggregation.

scope

(String) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(Enumerated Type) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns an **Integer**.

Remarks

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- *Scope* for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- *Scope* for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example shows an expression that calculates the number of unique non-null values of `size` for the default scope and for a parent group scope. The expression is added to a cell in a row that belongs to the child group `GroupbySubcategory`. The parent group is `GroupbyCategory`. The expression displays the results for `GroupbySubcategory` (the default scope) and then for `GroupbyCategory` (the parent group scope).

NOTE

Expressions should not contain actual carriage returns and line breaks; these are included in the example code to support documentation renderers. If you copy the following example, remove carriage returns from each line.

```
= "Distinct count (Subcategory): " & CountDistinct(Fields!Size.Value) &  
"Distinct count (Category): " & CountDistinct(Fields!Size.Value, "GroupbyCategory")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – CountRows Function

3/24/2017 • 1 min to read • [Edit Online](#)

Returns the number of rows in the specified scope, including rows with null values.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
CountRows(scope, recursive)
```

Parameters

scope

(**String**) The name of a dataset, data region, or group that contains the report items to count.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns an **Integer**.

Remarks

CountRows counts all rows in the specified scope, including rows that have null values.

The value of *scope* cannot be an expression and must refer to the current scope or a containing scope.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example shows an expression that calculates the number of rows in a row group named `GroupbyCategory` (based on the expression `[Category]`).

```
= "Number of rows: " & CountRows("GroupbyCategory")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - First Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the first value in the given scope of the specified expression.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
First(expression, scope)
```

Parameters

expression

(**Variant** or **Binary**) The expression on which to perform the aggregation, for example, `=Fields!FieldName.Value`.

scope

(**String**) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

Return Type

Determined by the type of expression.

Remarks

The **First** function returns the first value in a set of data after all sorting and filtering have been applied at the specified scope.

The **First** function cannot be used in group filter expressions with anything except the current (default) scope.

You can also use **First** in a page header to return the first value from the **ReportItems** collection for a page in order to produce dictionary-style headings that display the first and last entries on a page.

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example returns the first product number in the `Category` group of a data region:

```
=First(Fields!ProductNumber.Value, "Category")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – InScope Function

3/24/2017 • 1 min to read • [Edit Online](#)

Indicates whether the current instance of an item is in the specified scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
InScope(scope)
```

Parameters

scope

(**String**) The name of a dataset, data region, or group that specifies a scope.

Return Type

Returns a **Boolean**.

Remarks

The **InScope** function tests the scope of the current instance of a report item for membership in the scope specified by the *scope* parameter.

Scope cannot be an expression.

A typical use for the **InScope** function is in data regions that have dynamic scoping. For example, **InScope** can be used in a drillthrough link in a data region cells to provide a different report name and different sets of parameters depending on which cell is clicked. An example of this is as follows:

- The following expression, used as the report name in a drillthrough link, opens the `ProductDetail` report if the clicked cell is in the `Month` group, and the `ProductSummary` report if it is not.

```
=Iif(InScope("Month"), "ProductDetail", "ProductSummary")
```

- The following expression, used in the **Omit** property of a drillthrough report parameter, will pass the parameter to the target report only if the clicked cell is in the `Product` group.

```
=Not(InScope("Product"))
```

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Example

The following code example indicates whether the current instance of the item is in the `Product` dataset, data region, or group scope.

```
=InScope("Product")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - Last Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the last value in the given scope of the specified expression.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Last(expression, scope)
```

Parameters

expression

(**Variant** or **Binary**) The expression on which to perform the aggregation, for example, `=Fields!Fieldname.Value`.

scope

(**String**) (Optional) The name of a dataset, data region, or group that contains the report items to which to apply the function. If *scope* is not specified, the current scope is used.

Return Type

Determined by the type of expression.

Remarks

The **Last** function returns the final value in a set of data after all sorting and filtering have been applied at the specified scope.

The **Last** function cannot be used in group filter expressions with anything except the current (default) scope.

You can also use **Last** in a page header to return the last value from the **ReportItems** collection for a page in order to produce dictionary-style headings that display the first and last entries on a page.

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- *Scope* for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- *Scope* for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example returns the last product number in the `Category` group of a data region.

```
=Last(Fields!ProductNumber.Value, "Category")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – Level Function

3/24/2017 • 1 min to read • [Edit Online](#)

Returns the current level of depth in a recursive hierarchy.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Level(scope)
```

Parameters

scope

(**String**) (Optional). The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

Return Type

Returns an **Integer**. If *scope* specifies a dataset or data region, or specifies a nonrecursive grouping (that is, a grouping with no **Parent** element), **Level** returns 0. If *scope* is omitted, it returns the level of the current scope.

Remarks

The value returned by the **Level** function is zero based; that is, the first level in a hierarchy is 0.

The **Level** function can be used to provide indentation in a recursive hierarchy, such as an employee list.

For more information about recursive hierarchies, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example provides the level of row in the Employees group:

```
=Level("Employees")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - Lookup Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the first matching value for the specified name from a dataset that contains name/value pairs.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Lookup(source_expression, destination_expression, result_expression, dataset)
```

Parameters

source_expression

(**Variant**) An expression that is evaluated in the current scope and that specifies the name or key to look up. For example, `=Fields!ProdID.Value`.

destination_expression

(**Variant**) An expression that is evaluated for each row in a dataset and that specifies the name or key to match on. For example, `=Fields!ProductID.Value`.

result_expression

(**Variant**) An expression that is evaluated for the row in the dataset where *source_expression* = *destination_expression*, and that specifies the value to retrieve. For example, `=Fields!ProductName.Value`.

dataset

A constant that specifies the name of a dataset in the report. For example, "Products".

Return

Returns a **Variant**, or **Nothing** if there is no match.

Remarks

Use **Lookup** to retrieve the value from the specified dataset for a name/value pair where there is a 1-to-1 relationship. For example, for an ID field in a table, you can use **Lookup** to retrieve the corresponding Name field from a dataset that is not bound to the data region.

Lookup does the following:

- Evaluates the source expression in the current scope.
- Evaluates the destination expression for each row of the specified dataset after filters have been applied, based on the collation of the specified dataset.
- On the first match of source expression and destination expression, evaluates the result expression for that row in the dataset.
- Returns the result expression value.

To retrieve multiple values for a single name or key field where there is a 1-to-many relationship, use [LookupSet Function \(Report Builder and SSRS\)](#). To call **Lookup** for a set of values, use [Multilookup Function \(Report Builder and SSRS\)](#).

The following restrictions apply:

- **Lookup** is evaluated after all filter expressions are applied.
- Only one level of lookup is supported. A source, destination, or result expression cannot include a reference to a lookup function.
- Source and destination expressions must evaluate to the same data type. The return type is the same as the data type of the evaluated result expression.
- Source, destination, and result expressions cannot include references to report or group variables.
- **Lookup** cannot be used as an expression for the following report items:
 - Dynamic connection strings for a data source.
 - Calculated fields in a dataset.
 - Query parameters in a dataset.
 - Filters in a dataset.
 - Report parameters.
 - The Report.Language property.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Example

In the following example, assume that a table is bound to a dataset that includes a field for the product identifier ProductID. A separate dataset called "Product" contains the corresponding product identifier ID and the product name Name.

In the following expression, **Lookup** compares the value of ProductID to ID in each row of the dataset called "Product" and, when a match is found, returns the value of the Name field for that row.

```
=Lookup(Fields!ProductID.Value, Fields!ID.Value, Fields!Name.Value, "Product")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - LookupSet Function

3/24/2017 • 4 min to read • [Edit Online](#)

Returns the set of matching values for the specified name from a dataset that contains name/value pairs.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
LookupSet(source_expression, destination_expression, result_expression, dataset)
```

Parameters

source_expression

(Variant) An expression that is evaluated in the current scope and that specifies the name or key to look up. For example, `=Fields!ID.Value`.

destination_expression

(Variant) An expression that is evaluated for each row in a dataset and that specifies the name or key to match on. For example, `=Fields!CustomerID.Value`.

result_expression

(Variant) An expression that is evaluated for the row in the dataset where *source_expression* = *destination_expression*, and that specifies the value to retrieve. For example, `=Fields!PhoneNumber.Value`.

dataset

A constant that specifies the name of a dataset in the report. For example, "ContactInformation".

Return

Returns a **VariantArray**, or **Nothing** if there is no match.

Remarks

Use **LookupSet** to retrieve a set of values from the specified dataset for a name/value pair where there is a 1-to-many relationship. For example, for a customer identifier in a table, you can use **LookupSet** to retrieve all the associated phone numbers for that customer from a dataset that is not bound to the data region.

LookupSet does the following:

- Evaluates the source expression in the current scope.
- Evaluates the destination expression for each row of the specified dataset after filters have been applied, based on the collation of the specified dataset.
- For each match of source expression and destination expression, evaluates the result expression for that row in the dataset.
- Returns the set of result expression values.

To retrieve a single value from a dataset with name/value pairs for a specified name where there is a 1-to-1 relationship, use [Lookup Function \(Report Builder and SSRS\)](#). To call **Lookup** for a set of values, use [Multilookup Function \(Report Builder and SSRS\)](#).

The following restrictions apply:

- **LookupSet** is evaluated after all filter expressions are applied.
- Only one level of lookup is supported. A source, destination, or result expression cannot include a reference to a lookup function.
- Source and destination expressions must evaluate to the same data type.
- Source, destination, and result expressions cannot include references to report or group variables.
- **LookupSet** cannot be used as an expression for the following report items:
 - Dynamic connection strings for a data source.
 - Calculated fields in a dataset.
 - Query parameters in a dataset.
 - Filters in a dataset.
 - Report parameters.
 - The Report.Language property.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Example

In the following example, assume the table is bound to a dataset that includes a sales territory identifier TerritoryGroupID. A separate dataset called "Stores" contains the list of all stores in a territory and includes the territory identifier ID and the name of the store StoreName.

In the following expression, **LookupSet** compares the value TerritoryGroupID to ID for each row in the dataset called "Stores". For each match, the value of the StoreName field for that row is added to the result set.

```
=LookupSet(Fields!TerritoryGroupID.Value, Fields!ID.Value, Fields!StoreName.Value, "Stores")
```

Example

Because **LookupSet** returns a collection of objects, you cannot display the result expression directly in a text box. You can concatenate the value of each object in the collection as a string.

Use the Visual Basic function **Join** create a delimited string from a set of objects. Use a comma as a separator to combine the objects in a single line. In some renderers, you might use a Visual Basic line feed (`vbcrlf`) as a separator to list each value on a new line.

The following expression, when it is used as the Value property for a text box, uses **Join** to create a list.

```
=Join(LookupSet(Fields!TerritoryGroupID.Value, Fields!ID.Value, Fields!StoreName.Value, "Stores"), ",")
```

Example

For text boxes that only render a few times, you might choose to add custom code to generate HTML to display values in a text box. HTML in a text box requires extra processing, so this would not be a good choice for a text box that is rendered thousands of times.

Copy the following Visual Basic functions to a Code block in a report definition. **MakeList** takes the object array that is returned in *result_expression* and builds an unordered list by using HTML tags. **Length** returns the number of items in the object array.

```
Function MakeList(ByVal items As Object()) As String
    If items Is Nothing Then
        Return Nothing
    End If

    Dim builder As System.Text.StringBuilder =
        New System.Text.StringBuilder()
    builder.Append("<ul>")

    For Each item As Object In items
        builder.Append("<li>")
        builder.Append(item)
    Next
    builder.Append("</ul>")

    Return builder.ToString()
End Function

Function Length(ByVal items as Object()) as Integer
    If items is Nothing Then
        Return 0
    End If
    Return items.Length
End Function
```

Example

To generate the HTML, you must call the function. Paste the following expression in the Value property for the text box and set the markup type for text to HTML. For more information, see [Add HTML into a Report \(Report Builder and SSRS\)](#).

```
=Code.MakeList(LookupSet(Fields!TerritoryGroupID.Value, Fields!ID.Value, Fields!StoreName.Value, "Stores"))
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – Max Function

3/24/2017 • 1 min to read • [Edit Online](#)

Returns the maximum value of all non-null numeric values specified by the expression, in the context of the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Max(expression, scope, recursive)
```

Parameters

expression

(Variant) The expression on which to perform the aggregation.

scope

(String) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(Enumerated Type) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Determined by the type of the expression.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example provides the highest total in the `Year` group or data region.

```
=Max(Fields!OrderTotal.Value, "Year")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – Min Function

3/24/2017 • 1 min to read • [Edit Online](#)

Returns the minimum value of all non-null numeric values specified by the expression, in the context of the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Min(expression, scope, recursive)
```

Parameters

expression

(Variant) The expression on which to perform the aggregation.

scope

(String) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(Enumerated Type) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Determined by the type of the expression.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example provides the lowest total in the current scope.

```
=Min(Fields!OrderTotal.Value)
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - Multilookup Function

3/24/2017 • 3 min to read • [Edit Online](#)

Returns the set of first-match values for the specified set of names from a dataset that contains name/value pairs.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Multilookup(source_expression, destination_expression, result_expression, dataset)
```

Parameters

source_expression

(VariantArray) An expression that is evaluated in the current scope and that specifies the set of names or keys to look up. For example, for a multivalue parameter, `=Parameters!IDs.value`.

destination_expression

(Variant) An expression that is evaluated for each row in a dataset and that specifies the name or key to match on. For example, `=Fields!ID.Value`.

result_expression

(Variant) An expression that is evaluated for the row in the dataset where *source_expression* = *destination_expression*, and that specifies the value to retrieve. For example, `=Fields!Name.Value`.

dataset

A constant that specifies the name of a dataset in the report. For example, "Colors".

Return

Returns a **VariantArray**, or **Nothing** if there is no match.

Remarks

Use **Multilookup** to retrieve a set of values from a dataset for name-value pairs where each pair has a 1-to-1 relationship. **MultiLookup** is the equivalent of calling **Lookup** for a set of names or keys. For example, for a multivalue parameter that is based on primary key identifiers, you can use **Multilookup** in an expression in a text box in a table to retrieve associated values from a dataset that is not bound to the parameter or to the table.

Multilookup does the following:

- Evaluates the source expression in the current scope and generates an array of variant objects.
- For each object in the array, calls [Lookup Function \(Report Builder and SSRS\)](#) and adds the result to the return array.
- Returns the set of results.

To retrieve a single value from a dataset with name-value pairs for a specified name where there is a 1-to-1

relationship, use [Lookup Function \(Report Builder and SSRS\)](#). To retrieve multiple values from a dataset with name-value pairs for a name where there is a 1-to-many relationship, use [LookupSet Function \(Report Builder and SSRS\)](#).

The following restrictions apply:

- **Multilookup** is evaluated after all filter expressions are applied
- Only one level of look-up is supported. A source, destination, or result expression cannot include a reference to a lookup function.
- Source and destination expressions must evaluate to the same data type.
- Source, destination, and result expressions cannot include references to report or group variables.
- **Multilookup** cannot be used as an expression for the following report items:
 - Dynamic connection strings for a data source.
 - Calculated fields in a dataset.
 - Query parameters in a dataset.
 - Filters in a dataset.
 - Report parameters.
 - The Report.Language property.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Example

Assume a dataset called "Category" contains the field CategoryList, which is a field that contains a comma-separated list of category identifiers, for example, "2, 4, 2, 1".

The dataset CategoryNames contains the category identifier and category name, as shown in the following table.

ID	NAME
1	Accessories
2	Bikes
3	Clothing
4	Components

To look up the names that correspond to the list of identifiers, use **Multilookup**. You must first split the list into a string array, call **Multilookup** to retrieve the category names, and concatenate the results into a string.

The following expression, when placed in a text box in a data region bound to the Category dataset, displays "Bikes, Components, Bikes, Accessories":

```
=Join(MultiLookup(Split(Fields!CategoryList.Value,""),  
Fields!CategoryID.Value,Fields!CategoryName.Value,"Category")),  
", ")
```

Example

Assume a dataset ProductColors contains a color identifier field ColorID and a color value field Color, as shown in the following table.

COLORID	COLOR
1	Red
2	Blue
3	Green

Assume the multivalue parameter *MyColors* is not bound to a dataset for its available values. The default values for the parameter are set to 2 and 3. The following expression, when placed in a text box in a table, concatenates the multiple selected values for the parameter into a comma-separated list and displays "Blue, Green".

```
=Join(MultiLookup(Parameters!MyColors.Value,Fields!ColorID.Value,Fields!Color.Value,"ProductColors"),", ")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - Previous Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the value or the specified aggregate value for the previous instance of an item within the specified scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Previous(expression, scope)
```

Parameters

expression

(**Variant** or **Binary**) The expression to use to identify the data and for which to retrieve the previous value, for example, `Fields!Fieldname.Value` or `Sum(Fields!Fieldname.Value)`.

scope

(**String**) Optional. The name of a group or data region, or null (**Nothing** in Visual Basic), that specifies the scope from which to retrieve the previous value specified by *expression*.

Return Type

Returns a **Variant** or **Binary**.

Remarks

The **Previous** function returns the previous value for the expression evaluated in the specified scope after all sorting and filtering have been applied.

If *expression* does not contain an aggregate, the **Previous** function defaults to the current scope for the report item.

In a details group, use **Previous** to specify the value of a field reference in the previous instance of the detail row.

NOTE

The **Previous** function only supports field references in the details group. For example, in a text box in the details group, `=Previous(Fields!Quantity.Value)` returns the data for the field `Quantity` from the previous row. In the first row, this expression returns a null (**Nothing** in Visual Basic).

If *expression* contains an aggregate function that uses a default scope, **Previous** aggregates the data within the previous instance of the scope specified in the aggregate function call.

If *expression* contains an aggregate function that specifies a scope other than the default, the *scope* parameter for the **Previous** function must be a containing scope for the scope specified in the aggregate function call.

The functions **Level**, **InScope**, **Aggregate** and **Previous** cannot be used in the *expression* parameter. Specifying the

recursive parameter for any aggregate function is not supported.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Examples

Description

The following code example, when placed in the default data row of a data region, provides the value for the field `LineTotal1` in the previous row.

Code

```
=Previous(Fields!LineTotal.Value)
```

Description

The following example shows an expression that calculates the sum of sales on a specific day of the month and the previous value for that day of the month in a previous year. The expression is added to a cell in a row that belongs to the child group `GroupbyDay`. Its parent group is `GroupbyMonth`, which has a parent group `GroupbyYear`. The expression displays the results for `GroupbyDay` (the default scope) and then for `GroupbyYear` (the parent of the parent group `GroupbyMonth`).

For example, for a data region with a parent group named `Year`, its child group named `Month`, and its child group named `Day` (3 nested levels). The expression `=Previous(Sum(Fields!Sales.Value, "Day"), "Year")` in a row associated with the group `Day` returns the sales value for the same day and month for the previous year.

Code

```
=Sum(Fields!Sales.Value) & " " & Previous(Sum(Fields!Sales.Value, "GroupbyDay"), "GroupbyYear")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – RowNumber Function

3/24/2017 • 1 min to read • [Edit Online](#)

Returns a running count of the number of rows for the specified scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
RowNumber(scope)
```

Parameters

scope

(**String**) The name of a dataset, data region, or group, or null (**Nothing** in Visual Basic), that specifies the context in which to evaluate the number of rows. **Nothing** specifies the outermost context, usually the report dataset.

Remarks

RowNumber returns a running value of the count of rows within the specified scope, just as **RunningValue** returns the running value of an aggregate function. When you specify a scope, you specify when to reset the row count to 1.

scope cannot be an expression. *scope* must be a containing scope. Typical scopes, from the outermost to the innermost containment, are report dataset, data region, row groups or column groups.

To increment values across columns, specify a scope that is the name of a column group. To increment numbers down rows, specify a scope that is the name of a row group.

NOTE

Including aggregates that specify both a row group and a column group in a single expression is not supported.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

Code Example

The following is an expression that you can use for the **BackgroundColor** property of a Tablix data region detail row to alternate the color of detail rows for each group, always beginning with White.

```
=IIF(RowNumber("GroupbyCategory") Mod 2, "White", "PaleGreen")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – RunningValue Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns a running aggregate of all non-null numeric values specified by the expression, evaluated for the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
RunningValue(expression, function, scope)
```

Parameters

expression

The expression on which to perform the aggregation, for example, `[Quantity]`.

function

(**Enum**) The name of the aggregate function to apply to the expression, for example, **Sum**. This function cannot be **RunningValue**, **RowNumber**, or **Aggregate**.

scope

(**String**) A string constant that is the name of a dataset, data region, or group, or null (**Nothing** in Visual Basic), that specifies the context in which to evaluate the aggregation. **Nothing** specifies the outermost context, usually the report dataset.

Return Type

Determined by the aggregate function that is specified in the *function* parameter.

Remarks

The value for **RunningValue** resets to 0 for each new instance of the scope. If a group is specified, the running value is reset when the group expression changes. If a data region is specified, the running value is reset for each new instance of the data region. If a dataset is specified, the running value is not reset throughout the entire dataset.

RunningValue cannot be used in a filter or sort expression.

The set of data for which the running value is calculated must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

Scope cannot be an expression.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all

distinct scopes in the expression, one scope must be in a child relationship to all other scopes.

- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

To calculate the running value of the number of rows, use **RowNumber**. For more information, see [RowNumber Function \(Report Builder and SSRS\)](#).

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Examples

The following code example provides a running sum of the field named `cost` in the outermost scope, which is the dataset.

```
=RunningValue(Fields!Cost.Value, Sum, Nothing)
```

The following code example provides a running sum of the field named `Score` in the dataset named `DataSet1`.

```
=RunningValue(Fields!Score.Value,sum,"DataSet1")
```

The following code example provides a running sum of the field named `Traffic Charges` in the outermost scope.

```
=RunningValue(Fields!Traffic Charges.Value, Sum, Nothing)
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – StDev Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the standard deviation of all non-null numeric values specified by the expression, evaluated in the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
StDev(expression, scope, recursive)
```

Parameters

expression

(**Integer** or **Float**) The expression on which to perform the aggregation.

scope

(**String**) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns a **Decimal** for decimal expressions and a **Double** for all other expressions.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example provides the standard deviation of line item totals in the `Order` group or data region.

```
=StDev(Fields!LineTotal.Value, "Order")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – StDevP Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the population standard deviation of all non-null numeric values specified by the expression, evaluated in the context of the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
StDevP(expression, scope, recursive)
```

Parameters

expression

(**Integer** or **Float**) The expression on which to perform the aggregation.

scope

(**String**) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns a **Decimal** for decimal expressions and a **Double** for all other expressions.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example provides the population standard deviation of line item totals in the `order` group or data region.

```
=StDevP(Fields!LineTotal.Value, "Order")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - Sum Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the sum of all the non-null numeric values specified by the expression, evaluated in the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Sum(expression, scope, recursive)
```

Parameters

expression

(**Integer** or **Float**) The expression on which to perform the aggregation.

scope

(**String**) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns a **Decimal** for decimal expressions and a **Double** for all other expressions.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- *Scope* for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- *Scope* for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following two code examples provides a sum of line item totals in the `Order` group or data region.

```
=Sum(Fields!LineTotal.Value, "Order")
' or
=Sum(CDbl(Fields!LineTotal.Value), "Order")
```

Example

In a matrix data region with nested row groups Category and Subcategory, and nested column groups Year and Quarter, in a cell that belongs to the innermost row and column groups, the following expression evaluates to the maximum value from all quarters for all subcategories.

```
=Max(Sum(Fields!Sales.Value))
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions - Union Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the union of all the non-null numeric values specified by the expression, evaluated in the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Union(expression, scope, recursive)
```

Parameters

expression

(**SqlGeometry** or **SqlGeography**) The expression on which to perform the aggregation.

scope

(**String**) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return

Returns a spatial object, either **SqlGeometry** or **SqlGeography**, based on the expression type. For more information about **SqlGeometry** and **SqlGeography** spatial data types, see [Spatial Data Types Overview](#).

Remarks

The set of data specified in the expression must have the same data type.

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. Dataset scopes are not supported. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- *Scope* for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- *Scope* for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope](#)

for Totals, Aggregates, and Built-in Collections (Report Builder and SSRS).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following table shows examples of **SqlGeometry** expressions and **Union** result expression, shown in WKT (Well Known Text) format for spatial data.

FIELD WITH SPATIAL DATA	EXAMPLE	UNION RESULT
[PointLocation]	POINT(1 2)	MULTIPOINT((1 2), (3 4))
	POINT(3 4)	
[PathDefinition]	LINESTRING(1 2, 3 4)	MULTILINESTRING((7 8, 5 6), (3 4, 1 2))
	LINESTRING(5 6, 7 8)	
[PolygonDefinition]	POLYGON((1 2, 3 4, 5 2, 1 2))	MULTIPOLYGON(((1 2, 5 2, 3 4, 1 2)), ((-5 2, -1 2, -3 4, -5 2)))
	POLYGON((-1 2, -3 4, -5 2, -1 2))	

```
=Union(Fields!PointLocation.Value)
=Union(Fields!PathDefinition.Value)
=Union(Fields!PolygonDefinition.Value, "Group1")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – Var Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the variance of all non-null numeric values specified by the expression, evaluated in the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
Var(expression, scope, recursive)
```

Parameters

expression

(**Integer** or **Float**) The expression on which to perform the aggregation.

scope

(**String**) Optional. A constant that is the name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns a **Decimal** for decimal expressions and a **Double** for all other expressions.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- *Scope* for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- *Scope* for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example provides the variance of line item totals in the `Order` group or data region:

```
=Var(Fields!LineTotal.Value, "Order")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Report Builder Functions – VarP Function

3/24/2017 • 2 min to read • [Edit Online](#)

Returns the population variance of all non-null numeric values specified by the expression, evaluated in the context of the given scope.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Syntax

```
VarP(expression, scope, recursive)
```

Parameters

expression

(**Integer** or **Float**) The expression on which to perform the aggregation.

scope

(**String**) Optional. The name of a dataset, group, or data region that contains the report items to which to apply the aggregate function. If *scope* is not specified, the current scope is used.

recursive

(**Enumerated Type**) Optional. **Simple** (default) or **RdlRecursive**. Specifies whether to perform the aggregation recursively.

Return Type

Returns a **Decimal** for decimal expressions and a **Double** for all other expressions.

Remarks

The set of data specified in the expression must have the same data type. To convert data that has multiple numeric data types to the same data type, use conversion functions like **CInt**, **CDbl** or **CDec**. For more information, see [Type Conversion Functions](#).

The value of *scope* must be a string constant and cannot be an expression. For outer aggregates or aggregates that do not specify other aggregates, *scope* must refer to the current scope or a containing scope. For aggregates of aggregates, nested aggregates can specify a child scope.

Expression can contain calls to nested aggregate functions with the following exceptions and conditions:

- Scope for nested aggregates must be the same as, or contained by, the scope of the outer aggregate. For all distinct scopes in the expression, one scope must be in a child relationship to all other scopes.
- Scope for nested aggregates cannot be the name of a dataset.
- *Expression* must not contain **First**, **Last**, **Previous**, or **RunningValue** functions.
- *Expression* must not contain nested aggregates that specify *recursive*.

For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#) and [Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#).

For more information about recursive aggregates, see [Creating Recursive Hierarchy Groups \(Report Builder and SSRS\)](#).

Example

The following code example provides a population variance of line item totals in the `Order` group or data region.

```
=VarP(Fields!LineTotal.Value, "Order")
```

See Also

[Expression Uses in Reports \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Scope for Totals, Aggregates, and Built-in Collections \(Report Builder and SSRS\)](#)

Data Types in Expressions (Report Builder and SSRS)

3/24/2017 • 8 min to read • [Edit Online](#)

Data types represent different kinds of data so that it can be stored and processed efficiently. Typical data types include text (also known as strings), numbers with and without decimal places, dates and times, and images. Values in a report must be an Report Definition Language (RDL) data type. You can format a value according to your preference when you display it in a report. For example, a field that represents currency is stored in the report definition as a floating point number, but can be displayed in a variety of formats depending on the format property you choose.

For more information about display formats, see [Formatting Report Items \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Report Definition Language (RDL) Data Types and Common Language Runtime (CLR) Data Types

Values that are specified in an RDL file must be an RDL data type. When the report is compiled and processed, RDL data types are converted to CLR data types. The following table displays the conversion, which is marked Default:

RDL TYPE	CLR TYPES
String	Default: String Chart, GUID, Timespan
Boolean	Default: Boolean
Integer	Default: Int64 Int16, Int32, UInt16, UInt64, Byte, Sbyte
DateTime	Default: DateTime DateTimeOffset
Float	Default: Double Single, Decimal
Binary	Default: Byte[]
Variant	Any of the above except Byte[]
VariantArray	Array of Variant
Serializable	Variant or types marked with Serializable or that implement ISerializable.

Understanding Data Types and Writing Expressions

It is important to understand data types when you write expressions to compare or combine values, for example, when you define group or filter expressions, or calculate aggregates. Comparisons and calculations are valid only between items of the same data type. If the data types do not match, you must explicitly convert the data type in the report item by using an expression.

The following list describes cases when you may need to convert data to a different data type:

- Comparing the value of a report parameter of one data type to a dataset field of a different data type.
- Writing filter expressions that compare values of different data types.
- Writing sort expressions that combine fields of different data types.
- Writing group expressions that combine fields of different data types.
- Converting a value retrieved from the data source from one data type to a different data type.

Determining the Data Type of Report Data

To determine the data type of a report item, you can write an expression that returns its data type. For example, to show the data type for the field `MyField`, add the following expression to a table cell:

`=Fields!MyField.Value.GetType().ToString()`. The result displays the CLR data type used to represent `MyField`, for example, **System.String** or **System.DateTime**.

Converting Dataset Fields to a Different Data Type

You can also convert dataset fields before you use them in a report. The following list describes ways that you can convert an existing dataset field:

- Modify the dataset query to add a new query field with the converted data. For relational or multidimensional data sources, this uses data source resources to perform the conversion.
- Create a calculated field based on an existing report dataset field by writing an expression that converts all the data in one result set column to a new column with a different data type. For example, the following expression converts the field Year from an integer value to a string value: `=CStr(Fields!Year.Value)`. For more information, see [Add, Edit, Refresh Fields in the Report Data Pane \(Report Builder and SSRS\)](#).
- Check whether the data processing extension you are using includes metadata for retrieving preformatted data. For example, a SQL Server Analysis Services MDX query includes a FORMATTED_VALUE extended property for cube values that have already been formatted when processing the cube. For more information, see [Extended Field Properties for an Analysis Services Database \(SSRS\)](#).

Understanding Parameter Data Types

Report parameters must be one of five data types: Boolean, DateTime, Integer, Float, or Text (also known as String). When a dataset query includes query parameters, report parameters are automatically created and linked to the query parameters. The default data type for a report parameter is String. To change the default data type of a report parameter, select the correct value from the **Data type** drop-down list on the **General** page of the **Report Parameter Properties** dialog box.

NOTE

Report parameters that are DateTime data types do not support milliseconds. Although you can create a parameter based on values that include milliseconds, you cannot select a value from an available values drop-down list that includes Date or Time values that include milliseconds.

Writing Expressions that Convert Data Types or Extract Parts of Data

When you combine text and dataset fields using the concatenation operator (&), the common language runtime

(CLR) generally provides default formats. When you need to explicitly convert a dataset field or parameter to a specific data type, you must use a CLR method or a Visual Basic runtime library function to convert the data.

The following table shows examples of converting data types.

Type of Conversion	Example
DateTime to String	=CStr(Fields!Date.Value)
String to DateTime	=DateTime.Parse(Fields!DateTimeinStringFormat.Value)
String to DateTimeOffset	=DateTimeOffset.Parse(Fields!DateTimeOffsetinStringFormat.Val
Extracting the Year	<pre>=Year(Fields!TimeinStringFormat.Value)</pre> <p>-- or --</p> <pre>=Year(Fields!TimeinDateTimeFormat.Value)</pre>
Boolean to Integer	<pre>=CInt(Parameters!BooleanField.Value)</pre> <p>-1 is True and 0 is False.</p>
Boolean to Integer	<pre>=System.Convert.ToInt32(Fields!BooleanFormat.Value)</pre> <p>1 is True and 0 is False.</p>
Just the DateTime part of a DateTimeOffset value	=Fields!MyDatetimeOffset.Value.DateTime
Just the Offset part of a DateTimeOffset value	=Fields!MyDatetimeOffset.Value.Offset

You can also use the Format function to control the display format for value. For more information, see [Functions \(Visual Basic\)](#).

Advanced Examples

When you connect to a data source with a data provider that does not provide conversion support for all the data types on the data source, the default data type for unsupported data source types is String. The following examples provide solutions to specific data types that are returned as a string.

Concatenating a String and a CLR DateTimeOffset Data Type

For most data types, the CLR provides default conversions so that you can concatenate values that are different data types into one string by using the & operator. For example, the following expression concatenates the text "The date and time are:" with a dataset field StartDate, which is a [DateTime](#) value:

```
=The date and time are: & Fields!StartDate.Value .
```

For some data types, you may need to include the ToString function. For example, the following expression shows the same example using the CLR data type [DateTimeOffset](#), which include the date, the time, and a time-zone offset relative to the UTC time zone: `=The time is: & Fields!StartDate.Value.ToString()`.

Converting a String Data Type to a CLR DateTime Data Type

If a data processing extension does not support all data types defined on a data source, the data may be retrieved as text. For example, a **datetimeoffset(7)** data type value may be retrieved as a String data type. In Perth, Australia, the string value for July 1, 2008, at 6:05:07.9999999 A.M. would resemble:

```
2008-07-01 06:05:07.999999 +08:00
```

This example shows the date (July 1, 2008), followed by the time to a 7-digit precision (6:05:07.9999999 A.M.), followed by a UTC time zone offset in hours and minutes (plus 8 hours, 0 minutes). For the following examples, this

value has been placed in a **String** field called `MyDateTime.Value`.

You can use one of the following strategies to convert this data to one or more CLR values:

- In a text box, use an expression to extract parts of the string. For example:
 - The following expression extracts just the hour part of the UTC time zone offset and converts it to minutes: `=CInt(Fields!MyDateTime.Value.Substring(Fields!MyDateTime.Value.Length-5,2)) * 60`

The result is `480`.

- The following expression converts the string to a date and time value:
`=DateTime.Parse(Fields!MyDateTime.Value)`

If the `MyDateTime.Value` string has a UTC offset, the `DateTime.Parse` function first adjusts for the UTC offset (7 A.M. - `[+08:00]`) to the UTC time of 11 P.M. the night before). The `DateTime.Parse` function then applies the local report server UTC offset and, if necessary, adjusts the time again for Daylight Saving Time. For example, in Redmond, Washington, the local time offset adjusted for Daylight Saving Time is `[-07:00]`, or 7 hours earlier than 11 PM. The result is the following **DateTime** value:
`2007-07-06 04:07:07 PM` (July 6, 2007 at 4:07 P.M.).

For more information about converting strings to **DateTime** data types, see [Parsing Date and Time Strings](#), [Formatting Date and Time for a Specific Culture](#), and [Choosing Between DateTime, DateTimeOffset, and TimeZoneInfo](#) on MSDN.

- Add a new calculated field to the report dataset that uses an expression to extract parts of the string. For more information, see [Add, Edit, Refresh Fields in the Report Data Pane \(Report Builder and SSRS\)](#).
- Change the report dataset query to use Transact-SQL functions to extract the date and time values independently to create separate columns. The following example shows how to use the function **DatePart** to add a column for the year and a column for the UTC time zone converted to minutes:

```
SELECT  
    MyDateTime,  
    DATEPART(year, MyDateTime) AS Year,  
    DATEPART(tz, MyDateTime) AS OffsetinMinutes  
FROM MyDates
```

The result set has three columns. The first column is the date and time, the second column is the year, and the third column is the UTC offset in minutes. The following row shows example data:

```
2008-07-01 06:05:07 2008 480
```

For more information about SQL Server database data types, see [Data Types \(Transact-SQL\)](#), and [Date and Time Data Types and Functions \(Transact-SQL\)](#) in [SQL Server Books Online](#).

For more information about SQL Server Analysis Services data types, see [Data Types in Analysis Services](#) in [SQL Server Books Online](#).

See Also

[Formatting Report Items \(Report Builder and SSRS\)](#)

Constants in Expressions (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A constant consists of literal text or predefined text. The report processor has access to predefined constants so that when you include them in an expression, the values they represent are substituted in the expression before it is evaluated.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Literal Text

In an expression, literal text is text that is in double quotation marks. You can also type text directly into a text box without double quotation marks if it is not part of an expression. If the text box value does not begin with an equal sign (=), the text is treated as literal text. The following table shows several examples of literal text in an expression.

CONSTANT	DISPLAY TEXT	EXPRESSION TEXT
Report run at:	<<Expr>>	= "Report run at: " & Globals!ExecutionTime
Adventure Works Cycles	Adventure Works Cycles	Adventure Works Cycles
[Bracketed display text]	\[Bracketed display text\]	[Bracketed display text]

RDL Constants

You can use constants defined in Report Definition Language (RDL) in an expression. In the **Expression** dialog box, constants appear when you create an expression for a report property that only accepts certain valid values, also known as enumerated types. The following table shows two examples.

PROPERTY	DESCRIPTION	VALUES
TextAlign	Valid values for aligning text in a text box.	General, Left, Center, Right
BorderStyle	Valid values for a line added to a report.	Default, None, Dotted, Dashed, Solid, Double, DashDot, DashDotdot

Visual Basic Constants

You can use constants defined in the Visual Basic run-time library in an expression. For example, you can use the constant **DateInterval.Day**. The following expression for the date January 10, 2008 returns the number 10:

```
=DatePart("d",Globals!ExecutionTime)
```

CLR Constants

You can use constants defined in .NET Framework common language run-time (CLR) classes in an expression. The following table shows an example of a system-defined color.

CONSTANT	DESCRIPTION
MistyRose	When you create an expression for a report property that is based on background color, you can specify a color by name. Valid names are listed in the Expression dialog box.

See Also

[Expression Dialog Box](#)

[Expressions \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Dialog Box \(Report Builder\)](#)

Operators in Expressions (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

An operator is a symbol that represents actions applied to one or more terms in an expression. The following categories of operators are supported in an expression: arithmetic, comparison, concatenation, logical or bitwise, and bit shift.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Arithmetic

Arithmetic operators perform mathematical operations on two numeric terms in an expression.

OPERATOR	DESCRIPTION
$^$	Raises a number to the power of another number.
*	Multiplies two numbers.
/	Divides two numbers and returns a floating-point result.
	Divides two numbers and returns an integer result.
Mod	Returns the integer remainder of a division. For example, 7 Mod 5 = 2 because the remainder of 7 divided by 5 is 2.
+	Adds two numbers together.
-	Returns the difference between two numbers or indicates the negative value of a numeric term.

Comparison

Comparison operators test whether two expressions are the same.

OPERATOR	DESCRIPTION
<	Less than.
<=	Less than or equal to.
>	Greater than.
>=	Greater than or equal to.
=	Equal to.
<>	Not equal to.

OPERATOR	DESCRIPTION
Like	<p>Determines whether a specific character string matches a specified pattern. A pattern can include regular characters and wildcard characters. During pattern matching, regular characters must exactly match the characters specified in the character string. However, wildcard characters can be matched with arbitrary fragments of the character string. Using wildcard characters makes the LIKE operator more flexible than using the = and != string comparison operators.</p> <p>The following table lists characters that can be used as wildcards:</p> <ul style="list-style-type: none"> %: Any string of zero or more characters. _: Any single character. []: Any single character within the specified range (for example, [a-f]) or set (for example, [aeiou]). [^]: Any single character not within the specified range (for example, [^a-f]) or set (for example, [^aeiou])
Is	Compares two object references.

String Concatenation

String concatenation appends the second string to the first string in an expression. For other string operations, use built-in functions.

OPERATOR	DESCRIPTION
&	Concatenates two strings
+	Concatenates two strings

Logical and Bitwise

Logical and bitwise operators perform logical manipulations between two integer terms in an expression.

OPERATOR	DESCRIPTION
And	Performs a logical conjunction on two Boolean expressions, or bitwise conjunction on two numeric expressions.
Not	Performs logical negation on a Boolean expression, or bitwise negation on a numeric expression.
Or	Performs a logical disjunction on two Boolean expressions, or bitwise disjunction on two numeric values.
Xor	Performs a logical exclusion operation on two Boolean expressions, or a bitwise exclusion on two numeric expressions.
AndAlso	Performs logical conjunction on two expressions.

OPERATOR	DESCRIPTION
OrElse	Performs logical disjunction on two expressions.

Bit Shift

Bitwise operators perform bit manipulations between two integer terms in an expression.

OPERATOR	DESCRIPTION
<<	Performs an arithmetic left-shift on a bit pattern.
>>	Performs an arithmetic right-shift on a bit pattern.

See Also

[Expression Dialog Box](#)

[Expressions \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Expression Dialog Box \(Report Builder\)](#)

Built-in Collections in Expressions (Report Builder)

3/24/2017 • 2 min to read • [Edit Online](#)

In an expression in a report, you can include references to the following built-in collections: ReportItems, Parameters, Fields, DataSets, DataSources, Variables, and built-in fields for global information such as the report name. Not all collections appear in the **Expression** dialog box. The DataSets and DataSources collections are available only at run-time for published reports on a report server. The ReportItems collection is collection of text boxes in a report region, for example, the text boxes on a page or in a page header.

For more information, see [Expressions \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Understanding Built-in Collections

The following table lists the built-in collections available when you write an expression. Each row includes the case-sensitive programmatic name for the collection, whether you can use the Expression dialog box to interactively add a reference to the collection, an example, and a description that includes when the collection values are initialized and available for use.

BUILT-IN COLLECTION	CATEGORY IN THE EXPRESSION DIALOG BOX	EXAMPLE	DESCRIPTION
Globals	Built-in Fields	<code>=Globals.ReportName</code> - or - <code>=Globals.PageNumber</code>	Represents global variables useful for reports, such as the report name or page number. Always available. For more information, see Built-in Globals and Users References (Report Builder and SSRS) .
User	Built-in Fields	<code>=User.UserID</code> - or - <code>=User.Language</code>	Represents a collection of data about the user running the report, such as the language setting or the user ID. Always available. For more information, see Built-in Globals and Users References (Report Builder and SSRS) .

BUILT-IN COLLECTION	CATEGORY IN THE EXPRESSION DIALOG BOX	EXAMPLE	DESCRIPTION
Parameters	Parameters	=Parameters("ReportMonth").Value - or - =Parameters!ReportYear.Value	Represents the collection of report parameters, each of which can be single-value or multivalue. Not available until processing initialization is complete. For more information, see Parameters Collection References (Report Builder and SSRS) .
Fields(<Dataset>)	Fields	=Fields!Sales.Value	Represents the collection of fields of the dataset that are available to the report. Available after data is retrieved from a data source into a dataset. For more information, see Dataset Fields Collection References (Report Builder and SSRS) .
DataSets	Not Displayed	=DataSets("TopEmployees").Count	Represents the collection of datasets referenced from the body of a report definition. Does not include data sources used only in page headers or page footers. Not available in local preview. For more information, see DataSources and DataSets Collection References (Report Builder and SSRS) .
DataSources	Not Displayed	=DataSources("AdventureWorksLT").Count	Represents the collection of data sources referenced from within the body of a report. Does not include data sources used only in page headers or page footers. Not available in local preview. For more information, see DataSources and DataSets Collection References (Report Builder and SSRS) .
Variables	Variables	=Variables!CustomTimeStamp.Value	Represents the collection of report variables and group variables. For more information, see Report and Group Variables Collections References (Report Builder and SSRS) .

BUILT-IN COLLECTION	CATEGORY IN THE EXPRESSION DIALOG BOX	EXAMPLE	DESCRIPTION
ReportItems	Not Displayed	=ReportItems("Textbox1").Value	Represents the collection of text boxes for a report item. This collection can be used to summarize items on the page for including in a page header or page footer. For more information, see ReportItems Collection References (Report Builder and SSRS) .

Using Collection Syntax in an Expression

To refer to a collection from an expression, use standard Microsoft Visual Basic syntax for an item in a collection. The following table shows examples of collection syntax.

SYNTAX	EXAMPLE
<i>Collection!ObjectName.Property</i>	=Fields!Sales.Value
<i>Collection!ObjectName("Property")</i>	=Fields!Sales("Value")
<i>Collection("ObjectName").Property</i>	=Fields("Sales").Value
<i>Collection("Member")</i>	=User("Language")
<i>Collection.Member</i>	=User.Language

See Also

- [Add an Expression \(Report Builder and SSRS\)](#)
- [Expression Examples \(Report Builder and SSRS\)](#)

Built-in Collections - Built-in Globals and Users References (Report Builder)

3/24/2017 • 6 min to read • [Edit Online](#)

The Built-in fields collection, which includes both the **Globals** and the **User** collections, represent global values provided by Reporting Services when a report is processed. The **Globals** collection provides values such as the name of the report, the time when report processing began, and current page numbers for the report header or footer. The **User** collection provides the user identifier and language settings. These values can be used in expressions to filter results in a report.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Using the Globals Collection

The **Globals** collection contains the global variables for the report. On the design surface, these variables appear prefixed by an & (ampersand), for example, `[&ReportName]`. The following table describes the members of the **Globals** collection.

MEMBER	TYPE	DESCRIPTION
ExecutionTime	DateTime	The date and time that the report began to run.
PageNumber	Integer	<p>The current page number relative to page breaks that reset the page number. At the beginning of report processing, the initial value is set to 1. The page number increments for each rendered page.</p> <p>To number pages within page breaks for a rectangle, a data region, a data region group, or a map, on the PageBreak property, set the ResetPageNumber property to True. Not supported on tablix column hierarchy groups.</p> <p>PageNumber can only be used in an expression in a page header or page footer.</p>
ReportFolder	String	The full path to the folder containing the report. This does not include the report server URL.
ReportName	String	The name of the report as it is stored in the report server database.

MEMBER	TYPE	DESCRIPTION
ReportServerUrl	String	The URL of the report server on which the report is being run.
TotalPages	Integer	<p>The total number of pages relative to page breaks that reset PageNumber. If no page breaks are set, this value is the same as OverallTotalPages.</p> <p>TotalPages can only be used in an expression in a page header or page footer.</p>
PageName	String	<p>The name of the page. At the beginning of report processing, the initial value is set from InitialPageName, a report property. As each report item is processed, this value is replaced by the corresponding value of PageName from a rectangle, a data region, a data region group, or a map. Not supported on tablix column hierarchy groups.</p> <p>PageName can only be used in an expression in a page header or page footer.</p>
OverallPageNumber	Integer	<p>The page number of the current page for the entire report. This value is not affected by ResetPageNumber.</p> <p>OverallPageNumber can only be used in an expression in a page header or page footer.</p>
OverallTotalPages	Integer	<p>The total number pages for the entire report. This value is not affected by ResetPageNumber.</p> <p>OverallTotalPages can only be used in an expression in a page header or page footer.</p>
RenderFormat	RenderFormat	<p>Information about the current rendering request.</p> <p>For more information, see "RenderFormat" in the next section.</p>

Members of the **Globals** collection return a variant. If you want to use a member of this collection in an expression that requires a specific data type, you must first cast the variable. For example, to convert the execution time variant into a Date format, use `=CDate(Globals!ExecutionTime)`. For more information, see [Data Types in Expressions \(Report Builder and SSRS\)](#).

RenderFormat

The following table describes the members for **RenderFormat**.

MEMBER	TYPE	DESCRIPTION
Name	String	The name of the renderer as registered in the RSReportServer configuration file. Available during specific parts of the report processing/rendering cycle.
IsInteractive	Boolean	Whether the current rendering request uses an interactive rendering format.
DeviceInfo	Read-only name/value collection	Key/value pairs for deviceinfo parameters for the current rendering request. String values can be specified by using either the key or an index into the collection.

Examples

The following examples show how to use a reference to the **Globals** collection in an expression:

- This expression, placed in a text box in the footer of a report, provides the page number and total pages in the report:

```
=Globals.PageNumber & " of " & Globals.TotalPages
```

- This expression provides the name of the report and the time it was run. The time is formatted with the Microsoft .NET Framework formatting string for short date:

```
=Globals.ReportName & ", dated " & Format(Globals.ExecutionTime, "d")
```

- This expression, placed in the **Column Visibility** dialog box for a selected column, displays the column only when the report is exported to Excel. Otherwise, the column is hidden.

`EXCELOPENXML` refers to the format of Excel that is included in Office 2007. `EXCEL` refers to the format of Excel that is included in Office 2003.

```
=IIF(Globals!RenderFormat.Name = "EXCELOPENXML" OR Globals!RenderFormat.Name = "EXCEL", false, true)
```

Using the User Collection

The **User** collection contains data about the user who is running the report. You can use this collection to filter the data that appears in a report, for example, showing only the data of the current user, or to display the UserID, for example, in a report title. On the design surface, these variables appear prefixed by an & (ampersand), for example, `[&UserID]`.

The following table describes the members of the **User** collection.

MEMBER	TYPE	DESCRIPTION
Language	String	The language of the user running the report. For example, <code>en-US</code> .

MEMBER	TYPE	DESCRIPTION
UserID	String	The ID of the user running the report. If you are using Windows Authentication, this value is the domain account of the current user. The value is determined by the Reporting Services security extension, which can use Windows Authentication or custom authentication.

For more information about supporting multiple languages in a report, see "Solution Design Considerations for Multi-Lingual or Global Deployments" in the Reporting Services documentation in [SQL Server Books Online](#).

Using Locale Settings

You can use expressions to refer to the locale settings on a client computer through the **User.Language** value to determine how a report appears to the user. For example, you can create a report that uses a different query expression based on the locale value. The query may change to retrieve localized information from a different column depending on the language returned. You can also use an expression in the language settings of the report or report items based on this variable.

NOTE

While you can change the language settings of a report, you must be careful about any display issues this may cause. For example, changing the locale setting of the report can change the date format in the report, but it can also change the currency format. Unless there is a conversion process for the currency, this may cause the incorrect currency symbol to be displayed in the report. To avoid this, set the language information about the individual items that you want to change, or set the item with the currency data to a specific language.

Identifying UserID for Snapshot or History Reports

In some cases, reports that include the *User!UserID* variable will fail to show report data that is specific to the current user who is viewing the report.

See Also

[Expressions \(Report Builder and SSRS\)](#)

[Expression Dialog Box \(Report Builder\)](#)

[Data Types in Expressions \(Report Builder and SSRS\)](#)

[Formatting Numbers and Dates \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Built-in Collections – Parameters Collection References (Report Builder)

3/24/2017 • 3 min to read • [Edit Online](#)

Report parameters are one of the built-in collections you can reference from an expression. By including parameters in an expression, you can customize report data and appearance based on choices a user makes. Expressions can be used for any report item property or text box property that provides the *(Fx)* or **<Expression>** option. Expressions are also used to control report content and appearance in other ways. For more information, see [Expression Examples \(Report Builder and SSRS\)](#).

When you compare parameter values with dataset field values at run time, the data types for the two items you are comparing must be the same. Report parameters can be one of the following types: Boolean, DateTime, Integer, Float, or Text, which represents the underlying data type String. If necessary, you might have to convert the data type of the parameter value to match the dataset value. For more information, see [Data Types in Expressions \(Report Builder and SSRS\)](#).

In order to include a parameter reference in an expression, you must understand how to specify the correct syntax for the parameter reference, which varies depending on whether the parameter is a single-value or multivalue parameter.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Using a Single-Valued Parameter in an Expression

The following table shows examples of the syntax to use when you include a reference to a single-value parameter of any data type in an expression.

EXAMPLE	DESCRIPTION
=Parameters!<ParameterName>.IsMultiValue	Returns False . Checks if a parameter is multivalue. If True , the parameter is multivalue and it is a collection of objects. If False , the parameter is single-value and is a single object.
=Parameters!<ParameterName>.Count	Returns the integer value 1. For a single-value parameter, the count is always 1.
=Parameters!<ParameterName>.Label	Returns the parameter label, frequently used as the display name in a drop-down list of available values.
=Parameters!<ParameterName>.Value	Returns the parameter value. If the Label property has not been set, this value appears in the drop-down list of available values.
=CStr(Parameters!<ParameterName>.Value)	Returns the parameter value as a string.

EXAMPLE	DESCRIPTION
=Fields(Parameters! <ParameterName>.Value).Value	Returns the value for the field that has the same name as the parameter.

For more information about using parameters in a filter, see [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#).

Using a Multivalue Parameter in an Expression

The following table shows examples of the syntax to use when you include a reference to a multivalue parameter of any data type in an expression.

EXAMPLE	DESCRIPTION
=Parameters! <MultivalueParameterName>.IsMultiValue	Returns True or False . Checks if a parameter is multivalue. If True , the parameter is multivalue and is a collection of objects. If False , the parameter is single-valued and is a single object.
=Parameters! <MultivalueParameterName>.Count	Returns an integer value. Refers to the number of values. For a single-value parameter, the count is always 1. For a multivalue parameter, the count is 0 or more.
=Parameters! <MultivalueParameterName>.Value(0)	Returns the first value in a multivalue parameter.
=Parameters! <MultivalueParameterName>.Value(Parameters! <MultivalueParameterName>.Count - 1)	Returns the last value in a multivalue parameter.
=Split("Value1,Value2,Value3", ",")	Returns an array of values. Create an array of values for a multivalue String parameter. You can use any delimiter in the second parameter to Split. This expression can be used to set defaults for a multivalue parameter or to create a multivalue parameter to send to a subreport or drillthrough report.
=Join(Parameters! <MultivalueParameterName>.Value, ", ")	Returns a String that consists of a comma-delimited list of values in a multivalue parameter. You can use any delimiter in the second parameter to Join.

For more information about using parameters in a filter, see [Report Parameters \(Report Builder and Report Designer\)](#).

See Also

[Expressions \(Report Builder and SSRS\)](#)

[Commonly Used Filters \(Report Builder and SSRS\)](#)

[Add, Change, or Delete a Report Parameter \(Report Builder and SSRS\)](#)

[Tutorial: Add a Parameter to Your Report \(Report Builder\)](#)

[Report Builder Tutorials](#)

[Built-in Collections in Expressions \(Report Builder and SSRS\)](#)

Built-in Collections - Dataset Fields Collection References (Report Builder)

3/24/2017 • 5 min to read • [Edit Online](#)

Each dataset in a report contains one Fields collection. The Fields collection is the set of fields specified by the dataset query plus any additional calculated fields that you create. After you create a dataset, the field collection appears in the **Report Data** pane.

A simple field reference in an expression displays on the design surface as a simple expression. For example, when you drag the field `Sales` from the Report Data pane to a table cell on the design surface, `[Sales]` is displayed. This represents the underlying expression `=Fields!Sales.Value` that is set on the text box Value property. When the report runs, the report processor evaluates this expression and displays the actual data from the data source in the text box in the table cell. For more, see [Expressions \(Report Builder and SSRS\)](#) and [Dataset Fields Collection \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Displaying the Field Collection for a Dataset

To display the individual values for a field collection, drag each field to a table detail row and run the report. References from the detail row of a table or list data region display a value for each row in the dataset.

To display summarized values for a field, drag each numeric field to the data area of a matrix. The default aggregate function for the total row is Sum, for example, `=Sum(Fields!Sales.Value)`. You can change the default function in order to calculate different totals. For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

To display summarized values for a field collection in a text box directly on the design surface (not part of a data region), you must specify the dataset name as a scope for the aggregate function. For example, for a dataset named `SalesData`, the following expression specifies the total of all values for the field `Sales` :
`=Sum(Fields!Sales, "SalesData")`.

When you use the **Expression** dialog box to define a simple field reference, you can select the Fields collection in the Category pane and see the list of available fields in the **Field** pane. Each field has several properties, including Value and IsMissing. The remaining properties are predefined extended field properties that may be available to the dataset depending on the data source type.

Detecting Nulls for a Dataset Field

To detect a field value that is null (**Nothing** in Visual Basic), you can use the function **IsNothing**. When placed in a text box in a table details row, the following expression tests the field `MiddleName` and substitutes the text "No Middle Name" when the value is null, and the field value itself when the value is not null:

```
=IIF(IsNothing(Fields!MiddleName.Value),"No Middle Name",Fields!MiddleName.Value)
```

Detecting Missing Fields for Dynamic Queries at Run Time

By default, items in the Fields collection have two properties: Value and IsMissing. The IsMissing property indicates whether a field that is defined for a dataset at design time is contained in the fields retrieved at run time. For

example, your query might call a stored procedure in which the result set varies with an input parameter, or your query might be `SELECT * FROM <table>` where the table definition changed.

NOTE

`IsMissing` detects changes in the dataset schema between design time and run time for any type of data source. `IsMissing` cannot be used to detect empty members in a multidimensional cube and is not related to the MDX query language concepts of **EMPTY** and **NON EMPTY**.

You can test the `IsMissing` property in custom code to determine if a field is present in the result set. You cannot test for its presence using an expression with a Visual Basic function call such as **IIF** or **SWITCH**, because Visual Basic evaluates all parameters in the call to the function, which results in an error when the reference to the missing is evaluated.

Example for Controlling the Visibility of a Dynamic Column for a Missing Field

To set an expression that controls the visibility of a column that displays a field in a dataset, you must first define a custom code function that returns a Boolean value based on whether the field is missing. For example, the following custom code function returns true if the field is missing and false if the field exists.

```
Public Function IsFieldMissing(field as Field) as Boolean
    If (field.IsMissing) Then
        Return True
    Else
        Return False
    End If
End Function
```

To use this function to control visibility of a column, set the `Hidden` property of the column to the following expression:

```
=Code.IsFieldMissing(Fields!FieldName)
```

The column is hidden when the field does not exist.

Example for Controlling the Text Box Value for a Missing Field

To substitute text that you write in place of the value of a missing field, you must write custom code that returns text you can use in place of a field value when the field is missing. For example, the following custom code function returns the value of the field if the field exists, and the message that you specify as the second parameter if the field does not exist:

```
Public Function IsFieldMissingThenString(field as Field, strMessage as String) as String
    If (field.IsMissing) Then
        Return strMessage
    Else
        Return field.Value
    End If
End Function
```

To use this function in a text box, add the following expression to the `Value` property:

```
=Code.IsFieldMissingThenString(Fields!FieldName, "Missing")
```

The text box displays either the field value or the text that you specified.

Using Extended Field Properties

Extended field properties are additional properties defined on a field by the data processing extension, which is determined by the data source type for the dataset. Extended field properties are predefined or specific to a data

source type. For more information, see [Extended Field Properties for an Analysis Services Database \(SSRS\)](#).

If you specify a property that is not supported for that field, the expression evaluates to **null (Nothing)** in Visual Basic). If a data provider does not support extended field properties, or if the field is not found when the query is executed, the value for the property is **null (Nothing)** in Visual Basic) for properties of type **String** and **Object**, and zero (0) for properties of type **Integer**. A data processing extension may take advantage of predefined properties by optimizing queries that include this syntax.

See Also

[Expression Examples \(Report Builder and SSRS\)](#)

[Report Datasets \(SSRS\)](#)

Built-in Collections - DataSources and DataSets References (Report Builder)

3/24/2017 • 2 min to read • [Edit Online](#)

The **DataSources** collection represents all the data sources used in a report. Similarly, the **DataSets** collection represents all the datasets for all the data sources in a report. Use the **Report Data** pane for a hierarchical view of report datasets organized under the data source they reference. If you include references to these collections, you will not see values when previewing your report. These collections are only available after the report has been published to a report server.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

DataSources

The **DataSources** collection represents the data sources referenced in a published report definition. You may choose to include this information in your report to document the source of the report data. This collection is not available in **Preview** mode. The following table describes the variables within the **DataSources** collection.

VARIABLE	TYPE	DESCRIPTION
DataSourceReference	String	The full path of the data source definition on the report server. For example, you might include a list of all the data sources a report used as part of a report history. The following example shows the full path for the data source named AdventureWorks2012: /DataSources/AdventureWorks2012 .
Type	String	The type of data provider for the data source. For example, <code>SQL</code> .

DataSets

The **DataSets** collection represents the datasets referenced in a report definition. You may choose to include the query in the report in a text box, so a user interested in exactly which data is in the report can see the original command text. This collection is not available in **Preview** mode. The following table describes the members of the **DataSets** collection.

MEMBER	TYPE	DESCRIPTION
--------	------	-------------

MEMBER	TYPE	DESCRIPTION
CommandText	String	For database data sources, this is the query used to retrieve data from the data source. If the query is an expression, this is the evaluated expression.
RewrittenCommandText	String	The data provider's expanded CommandText value. This is typically used for reports with query parameters that are mapped to report parameters. The data provider sets this property when expanding the command text parameter references into the constant values selected for the mapped report parameters.

Using Query Expressions

You can use expressions to define the query that is contained in a dataset. You can use this feature to design reports in which the query changes based on input from the user, data in other datasets, or other variables. For more information about queries, see [Report Embedded Datasets and Shared Datasets \(Report Builder and SSRS\)](#).

See Also

[Expressions \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Built-in Collections - Report and Group Variables References (Report Builder)

3/24/2017 • 3 min to read • [Edit Online](#)

When you have a complex calculation that is used more than once in expressions in a report, you might want to create a variable. You can create a report variable or a group variable. Variable names must be unique in a report.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Report Variables

Use a report variable to hold a value for time-dependent calculations, such as currency rates or time stamps, or for a complex calculation that is referenced multiple times. By default, a report variable is calculated once and can be used in expressions throughout a report. Report variables are read-only by default. You can change the default to enable a report variable as read-write. The value in a report variable is preserved throughout a session, until the report is processed again.

To add a report variable, open the **ReportProperties** dialog box, click **Variables**, and provide a name and a value. Names are case-sensitive strings that begin with a letter and have no spaces. A name can include letters, numbers, or underscores (_).

To refer to the variable in an expression, use the global collection syntax, for example,

`=Variables!CustomTimeStamp.Value`. On the design surface, the value appears in a text box as `<<Expr>>`.

You can use report variables in the following ways:

- **Read-only use** Set a value once to create a constant for the report session, for example, to create a time stamp.

Because expressions in text boxes are evaluated on-demand as a user pages through a report, dynamic values (for example, an expression that includes the `Now()` function, which returns the time of day) can return different values if you page forward and backward by using the **Back** button. By setting a the value of a report variable to the expression `=Now()`, and then adding the variable to your expression, you ensure the same value is used throughout report processing.

- **Read-write use** Set a value once and serialize the value within a report session. The read-write option for variables provides a better alternative than using a static variable in the Code block in the report definition.

When you clear the **Read-Only** option for a variable, the **Writable** property for the variable is set to **true**.

To update the value from an expression, use the **SetValue** method, for example,

`=Variables!MyVariable.SetValue("123")`.

NOTE

You cannot control when the report processor initializes a variable or evaluates an expression that updates a variable. The order of execution for variable initialization is undefined.

For more information about sessions, see [Previewing Reports in Report Builder](#).

Group Variables

Use a group variable to calculate a complex expression once in the scope of a group. A group variable is valid only in the scope of the group and its child groups.

For example, suppose a data region displays inventory data for items that are in different tax categories and you want to apply different tax rates for each category. You would group the data on Category and define a *Tax* variable on the parent group. Then you would define a group variable for *ItemTax* for each tax category and assign each of the different Category subgroups to the correct group variable. For example:

- For the parent group based on `[Category]`, define the variable *Tax* with a value `[Tax]`. Assume the category values are Food and Clothing.
- For the child group based on `[Subcategory]`, define the variable *ItemsTax* as
`=Variables!Tax.Value * Sum(Fields!Price.Value)`. Assume the subcategory values for the category Food are Beverages and Bread. Assume the subcategory values for Clothing are Shirts and Hats.
- For a text box in a row in the child group, add the expression `=Variables!ItemsTax.Value`.

The text box displays the total tax for Beverages and Bread using the Food tax and for Shirts and Hats using the Clothing tax.

To add a group variable, open the **Tablix Group Properties** dialog box, click **Variables**, and provide a name and a value. The group variable is calculated once per unique group value.

To refer to the variable in an expression, use the global collection syntax, for example,
`=Variables!GroupDescription.Value`. On the design surface, the value appears in a text box as `<<Expr>>`.

See Also

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Built-in Collections in Expressions \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

Built-in Collections - ReportItems Collection References (Report Builder)

3/24/2017 • 2 min to read • [Edit Online](#)

The **ReportItems** built-in collection is the set of text boxes from report items such as rows of a data region or text boxes on the report design surface. The **ReportItems** collection includes text boxes that are in the current scope of a page header, page footer, or report body. This collection is determined at run time by the report processor and the report renderer. The current scope changes as the report processor successively combines report data and the report item layout elements as the user views pages of a report. You can use the **ReportItems** built-in collection to produce dictionary-style page headers that show the first and last items on each page.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Using the ReportItems Value Property

Items within the **ReportItems** collection have only one property: Value. The value for a **ReportItems** item can be used to display or calculate data from another field in the report. To access the value of the current text box, you can use the Visual Basic built-in global Me.Value or simply Value. In report functions such as First and aggregate functions, use the fully qualified syntax.

For example:

- This expression, placed in a text box, displays the value of a **ReportItem** text box named `Textbox1` :
`=ReportItems!Textbox1.Value`
- This expression, placed in a **ReportItem** text box Color property, displays the text in black when the value is > 0 ; otherwise, the value is displayed in red:
`=IIF(Me.Value > 0, "Black", "Red")`
- This expression, placed in a text box in the page header or page footer, displays the first value per page of the rendered report, for a text box named `LastName` :
`=First(ReportItems("LastName").Value)`

Dictionary-Style Page Header Expressions

You can create a page header to display the first customer on the page and the last customer on the page. Because a text box in the page header can only refer to the **ReportItems** built-in collection once in an expression, you need to add two text boxes to the page header: one for the first customer name (`=First(ReportItems!textboxLastName.Value)`) and one for the last customer name (`=Last(ReportItems!textboxLastName.Value)`).

In a page header or page footer section, only text boxes on the current page are available as a member of the **ReportItems** collection. For example, if `ReportItems!textboxLastName.Value` refers to a text box that only appears on the first page for a multipage data region, you see a value for the first page, but all other pages display `#Error` to show the expression could not be evaluated as written.

Scope for the ReportItems Collection

As the report is processed, each text box in the report body or in a data region is evaluated in the context of its dataset, data region, and group associations. The scope for a reference to the **ReportItems** collection is the current scope or any point higher than the current scope.

For example, a text box in a row that is in a parent group must not contain an expression that refers to the name of a text box in a child group row. Such an expression does not resolve to a value in the report because the child row text box is out of scope. For more information, see [Aggregate Functions Reference \(Report Builder and SSRS\)](#).

See Also

[Built-in Collections in Expressions \(Report Builder and SSRS\)](#)

[Expression Examples \(Report Builder and SSRS\)](#)

[Pagination in Reporting Services \(Report Builder and SSRS\)](#)

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

Formulas in Report Model Queries (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

Formulas are calculations performed on values in a report that use a report model as a data source. You define formulas in the **Define Formula Dialog Box** in the Report Model Query Designer when you define a query for a report model data source. A formula can contain functions, operators, constants, and references to fields or entities. Formulas allow you to combine, aggregate, filter, and evaluate both numeric and text data. You can create formulas and save them as new fields or you can modify the formulas of existing fields.

Formulas are not RDL expressions and do not begin with an equals sign (=). For more information about RDL expressions, see [Expressions \(Report Builder and SSRS\)](#).

Formulas can look similar to any of the following:

- **Sum Line Total**
- $6+12$
- **SUM(IF(Finished Goods Flag, "Finished", "Unfinished"))**

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

References

A reference is a field name. This can be an existing field name within the entity, or a calculated field name that you have created and added to the Fields list. The reference tells Report Builder where to look for the values, or data, you want to use within a formula. You can refer to fields within your context entity and to fields in other entities within one formula or use the value from one field in several formulas.

When you use references, the report processor runs the formula against each value within the field. For example, suppose a field contains the yearly sales total for the past five years. This field contains five values, each representing the sales total for a given year. If your formula contains a reference to this field, the formula calculates the new value using each individual value.

Operators

Operators specify the type of calculation that you want to perform on the values of a formula. There are three different types of calculation operators: arithmetic, comparison, and text. Operators are indicated using symbols, such as the plus sign (+).

Arithmetic Operators. Arithmetic operators perform basic mathematical operations such as addition, subtraction or multiplication, combine numbers, and produce numeric results.

Comparison Operators. You can compare two values using comparison operators. When two values are compared by using these operators, the result is a logical value, either TRUE or FALSE.

Text Concatenation Operator. Use the ampersand (&) to join, or concatenate, one or more text strings to

produce a single piece of text.

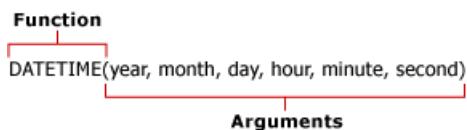
Constants

A constant is a value that is not calculated and, therefore, does not change. Report Builder uses the following constants: **True**, **False**, and **Empty**. These constants are used to evaluate Boolean fields. For example, suppose you have a field called IsDiscontinued. The only valid values for this field are True, False, or Empty (" ").

Functions

Functions are predefined formulas that perform calculations by using specific values, called *arguments*, specified in a particular order. Arguments can be literal values or fields, or combinations of both. When fields are used in formulas, the field name represents each instance of the field. If the argument is a literal value, you might need to indicate that the argument is a literal value using specific characters.

Functions can be used to perform simple or complex calculations. The structure of a function begins with the function name, followed by an opening parenthesis, the arguments for the function separated by commas, and a closing parenthesis.



Arguments can be field references, numbers, text, and logical values such as **TRUE** or **FALSE**. Arguments can also be constants, formulas, or other functions. The arguments that you enter must produce a valid value for that argument. For example, if the formula is multiplying two integers, the result cannot be a text string.

Report Builder comes with the following nine categories of commonly used functions:

Aggregate functions	AVG, COUNT, COUNTDISTINCT, MAX, MIN, STDEV, STDEVP, SUM, VAR, VARP
Conditional functions	IF, IN, SWITCH
Conversion functions	INT, DECIMAL, FLOAT, TEXT
Date and time functions	DATE, DATEADD, DATEDIFF, DATETIME, DATEONLY, DAY, DAYOFWEEK, DAYOFYEAR, HOUR, MINUTE, MONTH, NOW, QUARTER, SECOND, TIMEONLY, TODAY, WEEK, YEAR
Information functions	GETUSERCULTURE, GETUSERID
Logical functions	AND, NOT, OR
Math functions	MOD, ROUND, TRUNC
Operators	Add (+), Divide (/), Equal to (=), Exponentiation (^), Greater than (>), Greater than or equal to (>=), Less than (<), Less than or equal to (<=), Multiply (*), Negate (-), Not equal to (<>), Subtract (-)
Text functions	CONCAT, FIND, LEFT, LENGTH, LOWER, LTRIM, REPLACE, RIGHT, RTRIM, SUBSTRING, UPPER

Interactive Sort, Document Maps, and Links (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

In Web-based environments, you can add a number of features that let your users interact with reports. Your users can change the sort order of values in your report, show or hide items in the report, or click links that go to other reports or Web pages. You can also add a table of contents or document map. Your report users can click items in the table of contents or document map to jump to areas within a report.

Report Builder and Report Designer support three types of links with the following actions:

- **Bookmark links** Jump to other areas within the report.
- **Hyperlinks** Jump to URLs that specify the address of Web pages or reports on a report server by using URL access.
- **Drillthrough report links** Jump to other reports on the same report server. For more information, see [Drillthrough Reports \(Report Builder and SSRS\)](#).

NOTE

Links that are bound to dataset fields can be vulnerable to tampering for malicious purposes. For more information, see [Secure Reports and Resources](#) in SQL Server Books Online on msdn.microsoft.com.

You can also let your users control report display and content by designing expressions that include parameter references for sort, filter, and visibility. For more information, see [Report Parameters \(Report Builder and Report Designer\)](#), [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#), and [Add Dataset Filters, Data Region Filters, and Group Filters \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

In This Section

[Interactive Sort \(Report Builder and SSRS\)](#)

Explains how to add interactive sort buttons to column headers.

[Create a Document Map \(Report Builder and SSRS\)](#)

Explains how to add a table of contents to support navigation in a large report.

[Add a Bookmark to a Report \(Report Builder and SSRS\)](#)

Explains how to add bookmarks to create links within a report.

[Add a Hyperlink to a URL \(Report Builder and SSRS\)](#)

Explains how to add a link from your report to a URL

See Also

Drillthrough, Drilldown, Subreports, and Nested Data Regions (Report Builder and SSRS)

Interactive Sort (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

You can add interactive sort buttons to enable a user to toggle between ascending and descending order for rows in a table or for rows and columns in a matrix. The most common use of interactive sort is to add a sort button to every column header. The user can then choose which column to sort by.

However, you can add an interactive sort button to any text box, not just column headers. For example, for a text box in a row outside a row group, you can specify a sort for the parent group rows or columns, for child group rows or columns, or for the detail rows or columns. You can also combine fields into a single group expression, and then sort by multiple fields.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

When you add an interactive sort, you must specify the following items:

- **What to sort:** Rows or columns?
- **What to sort by:** A field that is displayed in a table column? A field that is not displayed?
- **What context to sort in:** For example, you can sort on rows associated with row groups; columns associated with column groups; detail rows; child groups within a parent group; or parent and child group together.
- **Which text box to add the sort button to:** In the column header or in the group row header?
- **Whether to synchronize the sort for multiple data regions:** You can design a report so that when the user toggles the sort order, other data regions with the same ancestor also sort.

For step-by-step instructions, see [Add Interactive Sort to a Table or Matrix \(Report Builder and SSRS\)](#).

The following table summarizes the effects you can achieve by using interactive sort buttons.

ACTION	WHAT TO SORT	WHERE TO ADD THE SORT BUTTON	WHAT TO SORT ON	SORT SCOPE
Sort detail rows for a table with no groups	Details	Column header	Dataset field bound to this column	Data region
Sort top-level group instances for a matrix	Groups	Column header	Group expression for parent group	Data region
Sort detail rows for a child group in a table	Details	Child group header row	Dataset field to sort by	Child group
Sort rows for multiple row groups and detail rows in a table	Groups, but you must redefine the group expression	Column header	Aggregate of dataset field to sort by	Data region

Action	What to Sort	Where to Add the Sort Button	What to Sort On	Sort Scope
Synchronize the sort order for multiple data regions	Groups	Typically, column header	Group expression	Dataset

The report processor applies interactive sort after all data region and group sort expressions are applied. For more information, see [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#).

Adding Interactive Sort for Multiple Groups

In a table with nested row groups each based on a single dataset field, you can add an interactive sort button that sorts parent group values, child group values, or detail rows. However, you might want to provide the user with the ability to sort the table by both the parent and child group values without having to click multiple times.

To do this, you must redesign the table to group on an expression that combines multiple fields. For example, for a dataset with inventory counts, if the original table grouped by size and then by color, you can specify a single group with a group expression that is a combination of size and color. For more information, see [Add Interactive Sort to a Table or Matrix \(Report Builder and SSRS\)](#).

See Also

[Sort Data in a Data Region \(Report Builder and SSRS\)](#)

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

[Add Interactive Sort to a Table or Matrix \(Report Builder and SSRS\)](#)

Add Interactive Sort to a Table or Matrix (Report Builder and SSRS)

3/24/2017 • 10 min to read • [Edit Online](#)

Add interactive sort buttons to enable users to change the sort order of rows and columns in tables and matrices. This feature is supported only in rendering formats that support user interaction, such as HTML.

When you create an interactive sort button, you must specify what to sort, what to sort by, and the scope to which to apply the sort. For example, you can sort detail rows by customer last name, subcategory group values within a category group by sales, or category and subcategory group values combined by totals.

When you view the report, columns that support interactive sorting have arrow icons that change to indicate the sort order. The first time you click an interactive sort button, items are sorted in ascending order. Subsequent clicks toggle the sort order between ascending and descending order.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

In this Article

[Sorting Detail Rows for a Table with No Groups](#)

[Sorting a Top Level Parent Row Group for a Table or Matrix](#)

[Sorting Child Groups or Detail Rows for a Group](#)

[Sorting Rows Based on a Complex Group Expression](#)

[Synchronizing Sort Order for Multiple Data Regions](#)

Sorting Detail Rows for a Table with No Groups

Add an interactive sort button to a column header to enable a user to click the column header and sort the details rows in a table by the value displayed in that column.

To add an interactive sort button to a column header to sort the table by value

1. In report design view, in a table with no groups, right-click the text box in the column header to which you want to add an interactive sort button, and then click **Text Box Properties**.
2. Click **Interactive Sorting**.
3. Select **Enable interactive sorting on this text box**.
4. In **Choose what to sort**, click **Detail rows**.
5. In **Sort by**, specify a sort expression. From the drop-down list, select the field that corresponds to the column for which you are defining a sort action (for example, for a column heading named "Title", choose `[Title]`). Specifying a sort expression is required.
6. Click **OK**.

7. Repeat steps 1-6 for every column to which you want to add an interactive sort button.

To verify the sort action, click **Run** to preview the report, and then click the interactive sort buttons.

 [Back to Top](#)

Sorting a Top-Level Parent Row Group for a Table or Matrix

Add an interactive sort button to a column header to enable a user to click the column header and sort the parent group rows in a table or matrix by the value displayed in that column. The order of child groups remains unchanged.

To add an interactive sort button to a column header to sort groups

1. In a table or matrix in report design view, right-click the text box in the column header for the group to which you want to add an interactive sort button, and then click **Text Box Properties**.

2. Click **Interactive Sorting**.

3. Select **Enable interactive sorting on this text box**.

4. In **Choose what to sort**, click **Groups**.

5. From the drop-down list, select the name of the group that you are sorting. For groups based on simple group expressions, the **Sort by** value is populated with group expression.

NOTE

For complex group expressions, manually set the **Sort by** expression to the same value as the group expression.

6. Click **OK**.

To verify the sort action, click **Run** to preview the report, and then click the interactive sort buttons.

 [Back to Top](#)

Sorting Child Groups or Detail Rows for a Group

Add an interactive sort button to a group header row to enable the user to sort the values of a child group from a parent group or to sort the detail rows for the innermost child group.

To add an interactive sort button to a text box in a group row header to sort child groups or detail rows

1. In report design view, right-click the text box in the group header row to which you want to add an interactive sort button, and then click **Text Box Properties**.

2. Click **Interactive Sorting**.

3. Select **Enable interactive sorting on this text box**.

4. In **Choose what to sort**, click one of the following options:

- **Details** Click **Details** to sort the detail rows. From the drop-down list, select the field to sort by. For this option, you must specify the value to sort by.
- **Groups** Click **Groups** to sort the child group values. For this option, the **Sort by** expression is automatically filled in from the group expression.

5. Click **OK**.

To verify the sort action, click **Run** to preview the report, and then click the interactive sort buttons.

Sorting Rows Based on a Complex Group Expression

Add an interactive sort button to a column header to enable a user to click the column header and sort the combined parent and child groups. To achieve this affect, you must change the group expression to be a composite of both groups. For example, suppose a matrix displays inventory totals for a store for items grouped by both color and size. To sort the rows based on the combination of color and size, instead of having a separate group for color and a separate group for size, you can define a group based on the combination of color and size. For more information about defining group expressions, see [Group Expression Examples \(Report Builder and SSRS\)](#).

In the following procedure, terms specify tablix data region areas. For more information, see [Tablix Data Region Areas \(Report Builder and SSRS\)](#).

Typically, when you sort rows based on multiple groups, you want to see totals for the sorted rows, regardless of column groups. In this procedure, no column groups are used. You start by adding a matrix and removing the default column group. Alternatively, you could start by adding a table and removing the details group.

To add an interactive sort button to a column header to sort multiple groups

1. In report design view, add a matrix.
2. Drag a numeric field to the data cell to link the dataset to the matrix.

Next, you will create a group with a group expression that specifies multiple fields, and a group header to use to display the group values.

3. Verify that the matrix is selected on the design surface. The Grouping pane displays a default row and column group.
4. In the Row Groups pane, right-click the default row group, and then click **Edit Group**. The **Group Properties** dialog box opens.
5. In **Name**, replace the default name with a name that specifies the multiple groups that you want to group by.
6. In **Group expressions**, in **Group on**, click the Expression (fx) button to open the **Expression** dialog box.
7. Type the expression that specifies all fields that you want to group by. For example, the following group expression combines a field named Color and a field named Size: `=Fields!Color.Value & Fields!Size.Value`.
8. Click **OK**.

You have now defined the group. Next, drag the fields to display to the tablix body area of the matrix. Add the fields that you chose to group by in step 7 to the tablix body area, each in its own column.

For this scenario, the first column in the tablix row groups area is not needed. To delete the column, right-click the column header, and then click **Delete Columns**. A dialog box asks whether to delete the associated groups. Click **No**. The row group area is deleted and only the tablix body area remains.

Next, you will remove the default column group.

9. In the Column Groups pane, right-click the default column group, and then click **Delete Group**. A dialog box asks whether to delete the group and related rows and columns or the group only. Click **Delete group only**. The column group is deleted, and the column group area is deleted. Only the tablix body area remains.

Next, you will add an interactive sort button to the text box that spans the matrix.

10. Click in the text box in the first row and then click **Text Box Properties**.

11. Click **Interactive Sorting**.
12. Select **Enable interactive sorting on this text box**.
13. In **Choose what to sort**, click **Groups**.
14. From the drop-down list, select the name of the group you created in step 5. The group expression is automatically copied to the **Sort by** text box.
15. Click **OK**.

You have added the sort button to the text box.

16. (Optional) You can suppress duplicate values in the columns that display group values. On the report design surface, click the text box that displays the value for which you want to hide repeating values. In the Properties pane, scroll to **HideDuplicates**, and from the drop-down list, select the name of the dataset that is linked to this matrix.

To verify the sort action, click **Run** to preview the report, and then click the interactive sort button. The matrix sorts by the combined values of the group expression, although each individual value displays in its own column.

 [Back to Top](#)

Synchronizing Sort Order for Multiple Data Regions

Add an interactive sort button that enables a user to click one sort button and sort multiple data regions. When you create an interactive sort button, you can specify whether to synchronize the sort for multiple data regions based on the same report dataset. For example, a report might include a matrix and a chart that graphically displays the data. When a user changes the sort order of the rows in the matrix, the chart automatically displays the same sort order.

To synchronize the sort order, you must use identical sort expressions for the data regions or groups to sort, and define the scope for the sort to be a mutual ancestor of both data regions. The mutual ancestor can be either the dataset to which both data regions are linked or a containing data region within which both data regions appear. For example, assume a report has both a matrix and a chart that display data from the same dataset and that are contained in a list. To synchronize the sort action, you must specify the interactive sort on a column in the matrix and set the scope to the list. When the user sorts the matrix, the chart is also sorted.

To synchronize sort order with a chart for an interactive sort button on a matrix data region

1. In report design view, add a matrix to the report.
2. Add a numeric dataset field to the matrix data cell, for example, a field representing quantity or sales.
3. Define a row group. By default, the sort order for the group is set to the same expression as the group expression.
4. Add a chart to the report, for example, a pie chart.
5. Drag the field you chose in step 2 to the **Value** area of the **Chart Data** pane.
6. Drag the field you chose to group by to the **Category Groups** area.

The group expression for the matrix row group and the chart category group must be identical.

7. Right-click the category group, and then click **Category Group Properties**.
8. Click **Sorting**.
9. Click **Add**. A new sort row is added to the sorting options grid.

10. In Sort by, from the drop-down list, choose the same field that you chose in step 6 to group by.
11. Click **OK**.
12. In the matrix, right-click the text box in the column header to which you want to add an interactive sort button, and then click **Text Box Properties**.
13. Click **Interactive Sorting**.
14. Select **Enable interactive sorting on this text box**.
15. In **Choose what to sort**, click **Groups**.
16. From the drop-down list under **Groups**, select the name of the group that you are sorting. The group expression for this group is automatically set for the **Sort by** value.
17. Select **Also apply this sort to other groups and data regions within**. In the text box, type the name of the dataset, for example, "SalesData".
18. Click **OK**.

To verify the sort action, click **Run** to preview the report, and then click the interactive sort button. The matrix sorts by the combined values of the group expression, although each individual value displays in its own column.

 [Back to Top](#)

See Also

- [Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)
- [Interactive Sort \(Report Builder and SSRS\)](#)
- [Sort Data in a Data Region \(Report Builder and SSRS\)](#)
- [Exploring the Flexibility of a Tablix Data Region \(Report Builder and SSRS\)](#)

Add a Hyperlink to a URL (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

Learn how to add hyperlink actions to text boxes, images, charts, and gauges in Reporting Services paginated reports. Links can go to other reports, to bookmarks in a report, or to static or dynamic URLs.

You can add a hyperlink action to any item that has an **Action** property, for example, a text box, an image, or a calculated series in a chart. When the user clicks that report item, the action that you define takes place.

- You can **add a hyperlink that will open a browser with a URL** that you specify. The hyperlink can be a static URL or an expression that evaluates to a URL. If you have a field in a database that contains URLs, the expression can contain that field, resulting in a dynamic list of hyperlinks in the report. Make sure your report readers have access to the URL that you provide.
- You can also **specify URLs to create drillthroughs** to reports on any report server that you and your users have permission to view using URL requests to the report server. For example, you can specify a report and hide the document map for the user when they first view the report. For more information, see [URL Access \(SSRS\)](#) and [Specifying Paths to External Items \(Report Builder and SSRS\)](#).
 - And you can **add a bookmark to a specific place** in the same report.

Try adding hyperlinks with sample data in [Tutorial: Format Text \(Report Builder\)](#).

NOTE

Links that are bound to dataset fields can be vulnerable to tampering for malicious purposes. For more information, see [Secure Reports and Resources](#).

To add a hyperlink and...

1. In report design view, right-click the text box, image, or chart to which you want to add a link and then click **Properties**.
2. In the Properties dialog box, click the **Action** tab. Read on for information about your options.

... add drillthrough to another report

1. On the **Action** tab, select **Go to report**.
2. Specify the target report and parameters you want to use. The parameter names must match the parameters defined for the target report.
3. Use the **Add** and **Delete** buttons to add and remove parameters, and the up and down arrows to order the list of parameters.
4. Type or select a **Value** to pass for the named parameter in the drillthrough report. Click the Expression (fx) button to edit the expression.
5. Select **Omit** to prevent the parameter from running. By default, this check box is cleared and not active. To select the check box, click the Expression (fx) button and either type True or create an expression. The check box is selected when you click **OK** in the Expression dialog box.

See [Add a Drillthrough Action on a Report](#) for more information.

6. Click **OK**.

... add a bookmark

You can link to bookmarks to a location in the current report. To link to a bookmark, you must first set the **Bookmark** property of a report item. To set the **Bookmark** property, select a report item, and in the Properties pane, type a value or expression for the bookmark ID; for example, SalesChart or 5TopSales.

1. On the **Action** tab, select **Go to bookmark**.
2. Type or select the bookmark ID for the report to jump to. Click the Expression (fx) button to change the expression.

The bookmark ID can be either a static ID or an expression that evaluates to a bookmark ID. The expression can include a field that contains a bookmark ID.

See [Add a Bookmark to a Report](#) for more information.

3. Click **OK**.

... add a hyperlink

1. On the **Action** tab, select **Go to URL**. An additional section appears in the dialog box for this option.
2. In **Select URL**, type or select a URL or an expression that evaluates to a URL, or click the drop-down arrow and click the name of a field that contains a URL.

For an item published to a report server configured for native mode, use a full or relative path. For example, `http://<servername>/images/image1.jpg`.

For an item published to a report server configured in SharePoint integrated mode, use a fully qualified URL. For example, `http://<SharePointservername>/<site>/Documents/images/image1.jpg`.

3. Click **OK**.

After you add a hyperlink

1. (Optional) The text is not automatically formatted as a link. For text, it is helpful to change the color and effect of the text to indicate that the text is a link. For example, change the color to blue and the effect to underline in the **Font** section in the Home tab of the Ribbon.
2. To test the link, click **Run** to preview the report, and then click the report item that you set this link on.

See Also

[Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#)

[Create a Document Map \(Report Builder and SSRS\)](#)

Add a Bookmark to a Report (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

Add bookmarks and bookmark links to a report when you want to provide a customized table of contents or to provide customized internal navigation links in the report. Typically, you add bookmarks to locations in the report to which you want to direct users, such as to each table or chart or to the unique group values displayed in a table or matrix. You can create your own strings to use as bookmarks, or, for groups, you can set the bookmark to the group expression.

After you create bookmarks, you can add report items that the user can click to go to each bookmark. These items are typically text boxes or images.

For example, if your report displays a table grouped by color, you would add a bookmark based on the group expression to the group header. Then you would add a table with a single text box at the beginning of the report that displayed the color values, and set the bookmark link on that text box. When you click the color, the report jumps to the page that displays the group header row for that color.

You can add a bookmark to any report item and add a bookmark link to any item that has an **Action** property, for example, a text box, an image, or a calculated series in a chart. For more information, see the [Action Properties Dialog Box \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a bookmark

1. In report design view, select the text box, image, chart, or other report item to which you want to add a bookmark. The properties for the selected item appear in the Properties pane.
2. In the text box next to **Bookmark**, type a string that is the label for this bookmark. For example, you could type **BikePhoto** as the bookmark for an image in your report. Alternatively, click the Expression (**fx**) button to open the **Expression** dialog box to specify an expression that evaluates to a label. For a group, the expression you type should be the group expression.

NOTE

The bookmark can be any string, but it must be unique in the report. If the bookmark is not unique, a link to the bookmark finds the first matching bookmark.

To add a bookmark link

1. In Design view, right-click the text box, image, chart, to which you want to add a link and then click **Properties**.
2. In The **Properties** dialog box for that report item, click **Action**.
3. Select **Go to bookmark**. An additional section appears in the dialog box for this option.
4. In the **Select bookmark** box, type or select a bookmark or an expression that evaluates to a bookmark.

Using the previous example, type **BikePhoto** to create a link to the image in your report.

5. Click **OK**.
6. (Optional) The text is not automatically formatted like a link. For text, it is helpful to change the color and effect of the text to indicate that the text is a link. For example, change the color to blue and the effect to underline in the **Font** section in the Home tab of the Ribbon.
7. To test the link, click **Run** to preview the report, and then click the report item that you set this link on..

See Also

[Interactive Sort, Document Maps, and Links \(Report Builder and SSRS\)](#)

[Expressions \(Report Builder and SSRS\)](#)

[Filter, Group, and Sort Data \(Report Builder and SSRS\)](#)

Create a Document Map (Report Builder and SSRS)

3/24/2017 • 4 min to read • [Edit Online](#)

A document map provides a set of navigational links to report items in a rendered report. When you view a report that includes a document map, a separate side pane appears next to the report. A user can click links in the document map to jump to the report page that displays that item. Report sections and groups are arranged in a hierarchy of links. Clicking items in the document map refreshes the report and displays the area of the report that corresponds to the item in the document map.

To add links to the document map, you set the **DocumentMapLabel** property of the report item to text that you create or to an expression that evaluates to the text that you want display in the document map. You can also add the unique values for a table or matrix group to the document map. For example, for a group based on color, each unique color is a link to the report page that displays the group instance for that color.

You can also create a URL to a report that overrides the display of the document map, so that you can run the report without displaying the document map, and then click the **Show/Hide Document Map** button on the report viewer toolbar to toggle the display.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Document Maps and Rendering Extensions

The document map is intended for use in the HTML rendering extension—for example, in Preview and the Report Viewer. Other rendering extensions have different ways of articulating a document map:

- PDF renders a document map as the Bookmarks pane.
- Excel renders a document map as a named worksheet that includes a hierarchy of links. Report sections are rendered in separate worksheets that are included with the document map in the same workbook.
- Word includes a document map as the table of contents.
- Atom, TIFF, XML, and CSV ignore document maps.

For more information, see [Interactive Functionality for Different Report Rendering Extensions \(Report Builder and SSRS\)](#).

To add a report item to a document map

1. In Design view, select the report item such as a table, matrix, or gauge that you want to add to the document map. The report item properties appear in the Properties pane.

NOTE

To select a tablix data region, click in any cell to display the row and column handles, and then click the corner handle.

2. In the Properties pane, type the text that you want to appear in the document map in the **DocumentMapLabel** property, or enter an expression that evaluates to a label. For example, type **Sales**

Chart.

NOTE

If you do not see the Properties pane, on the **View** tab, in the **Show/Hide** group, select **Properties**.

3. Repeat steps 1 and 2 for every report item that you want to appear in the document map.
4. Click **Run**. The report runs and the document map displays the labels you created. Click any link to jump to the report page with that item.

To add unique group values to a document map

1. In Design view, select the table, matrix, or list that contains the group that you want to display in the document map. The Grouping pane displays the row and column groups.
2. In the Row Groups pane, right-click the group, and then click **Edit Group**. The **General** page of the **Tablix Group Properties** dialog box opens.
3. Click **Advanced**.
4. In the **Document map** list box, type or select an expression that matches the group expression.
5. Click **OK**.
6. Repeat steps 1-4 for every group that you want to appear in the document map.
7. Click **Run**. The report runs and the document map displays the group values. Click any link to jump to the report page with that item.

To hide the document map when you view a report

1. In Report Manager, browse to the report that has the document map.

For example, for the **AdventureWorks2012** sample reports, the following URL specifies the report named Product Catalog.

```
http://localhost/Reports/Pages/Report.aspx?  
ItemPath=%2fAdventureWorks2012+Sample+Reports%2fProduct+Catalog
```

2. Copy the report path on the server. In the example, the report path is

```
%2fAdventureWorks2012+Sample+Reports%2fProduct+Catalog .
```

3. Create a new URL with the following three components:

- The report viewer on the report server: `http://localhost/ReportServer/Pages/ReportViewer.aspx?`
- The name of the report you copied in step 1, for example:
`%2fAdventureWorks2012+Sample+Reports%2fProduct+Catalog`
- The device information parameters that specify hiding the document map:
`&rs%3aCommand=Render&rc%3aFormat=HTML4.0&rc%3aDocMap=False`

The following URL consists of these three components appended in the order they are listed.

```
http://localhost/ReportServer/Pages/ReportViewer.aspx?  
%2fAdventureWorks2012+Sample+Reports%2fProduct+Catalog  
&rs%3aCommand=Render&rc%3aFormat=HTML4.0&rc%3aDocMap=False
```

To use this URL, copy it and remove all line breaks.

4. Paste the URL in Report Manager, and then press ENTER. The report runs, and the document map is hidden.

NOTE

For more information about downloading sample reports, see SQL Server 2016[Report Builder and Report Designer sample reports](#).

For more information, see "URL Access" in the [Reporting Services documentation](#) in SQL Server Books Online.

Page Layout and Rendering (Report Builder and SSRS)

3/24/2017 • 7 min to read • [Edit Online](#)

Read about Reporting Services rendering extensions for paginated reports so you're sure your report looks the way you want, including page layout, page breaks, and paper size.

When you view reports in a Reporting Services report server or the preview pane of Report Builder or Report Designer, the report is first rendered by the HTML renderer. You can then export the report to different formats such as Excel or comma-delimited (CSV) files. The exported report can then be used for further analysis in Excel or as a data source for applications that can import and use CSV files.

Reporting Services includes a set of renderers for exporting reports to different formats. Each renderer has applies rules when rendering reports. When you export a report to a different file format, especially for renderers such as the Adobe Acrobat (PDF) renderer that uses pagination based on the physical page size, you might need to change the layout of your report to have the exported report look and print correctly after the rendering rules are applied.

Getting the best results for exported reports is often an iterative process; you author and preview the report in Report Builder or Report Designer, export the report to the preferred format, review the exported report, and then make changes to the report.

Report Items

Report items are layout elements that are associated with different types of report data.

- Table, Matrix, List, Chart, and Gauge are data region report items that each link to a report dataset. When the report is processed, the data region expands across and down the report page to display data.

Other report items link to and display a single item.

- An **Image** report item links to a picture.
- A **Text Box** report item contains either simple text like a title or an expression that can include references to built-in fields, report parameters, or dataset fields.
- The **Line** and **Rectangle** report items provide simple graphical elements on the report page. The **Rectangle** can also be a container for other report items.

A report can also contain subreports.

Page Layout

With Reporting Services, you can place report items anywhere on the design surface. You can interactively position, expand, and contract the initial shape of the report item using snap lines and resizing handles. You can place data regions with different sets of data, or even the same data in different formats, side-by-side. When you place a report item on the design surface, it has a default size and shape and an initial relationship to all other report items.

You can place report items inside other report items to create more complex report designs. For example, charts or images in table cells, tables in table cells, and multiple images in a rectangle. In addition to providing the organization and look you want in the report, placing report items in containers such as rectangles help control the way the report items are displayed on the report page.

A report can span multiple pages, with a page header and page footer that are repeated on each page. A report can

contain graphical elements such as images and lines, and it can have multiple fonts, colors, and styles, which can be based on expressions.

Report Sections

A report consists of three main sections: an optional *page* header, an optional *page* footer, and a report body. The *report* header and footer are not separate sections of the report, but rather comprise the report items that are placed at the top and bottom of the report body. The page header and page footer repeat the same content at the top and bottom of each page of the report. You can place images, text boxes, and lines in headers and footers. You can place all types of report items in the report body.

You can set properties on report items to initially hide or show them on the page. You can set visibility properties on rows or columns or groups for data regions and provide toggle buttons to allow the user to interactively show or hide report data. You can set visibility or initial visibility by using expressions, including expressions based on report parameters.

When a report is processed, report data is combined with the report layout elements and the combined data is sent to a report renderer. The renderer follows predefined rules for report item expansion and determines how much data fits on each page. To design an easy-to-read report that is optimized for the renderer that you plan to use, you should understand the rules used to control pagination in Reporting Services. For more information, see [Pagination in Reporting Services \(Report Builder and SSRS\)](#).

Renderers

Reporting Services includes a set of renderers, also referred to as rendering extensions, that you can use to export reports to different formats. There are three types of renderers:

- **Data renderers** Data renderers strip all formatting and layout information from the report and display only the data. The resulting file can be used to import the raw report data into another file type, such as Excel, or another database, an XML data message, or a custom application. The available data renders are CSV and XML.

NOTE

Although it does not provide direct export to a different format, Atom rendering generates data files from reports.

- **Soft page-break renderers** Soft page-break renderers maintain the report layout and formatting. The resulting file is optimized for screen-based viewing and delivery, such as on a Web page. The available soft page-break renderers are Microsoft Excel, Microsoft Word, Web archive (MHTML), and HTML.
- **Hard page-break renderers** Hard page-break renderers maintain the report layout and formatting. The resulting file is optimized for a consistent printing experience, or for viewing the report online in a book format. The available hard page-break renderers are TIFF and PDF.

When you preview a report in Report Builder or Report Designer or run a report on a Reporting Services report server, the report is always first rendered in HTML. After you run the report, you can export it to different file formats. For more information, see [Export Reports \(Report Builder and SSRS\)](#).

Rendering Behaviors

Depending on the renderer you select, certain rules are applied when rendering the report. How report items fit together on a page is determined by the combination of these factors:

- Rendering rules.
- The width and height of report items.

- The size of the report body.
- The width and height of the page.
- Renderer-specific support for paging.

For example, reports rendered to HTML and MHTML formats are optimized for a computer screen-based experience where pages can be various lengths.

For more information, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

Pagination

Pagination refers to the number of pages within a report and how report items are arranged on these pages. Pagination in Reporting Services varies depending on the rendering extension you use to view and deliver the report and the page break and keep-together options you configure the report to use.

To successfully design an easy-to-read report for your users that is optimized for the renderer that you plan to use to deliver your report, you need to understand the rules used to control pagination in Reporting Services. Reports exported by using the **data** and **soft page** rendering extensions are typically not affected by pagination. When you use a data rendering extension the report is rendered as tabular rowset in an XML or CSV format. To ensure the exported report data is usable you should understand the rules applied to rendered a flattened tabular rowset from a report.

When you use a **soft page** rendering extension such as the HTML rendering extension, you might want to know how the report looks printed and also how well it renders using a hard page renderer such as PDF. During the creation or updating of a report you can preview and export it in Report Builder and Report Designer.

Hard page renderers have the most impact on report layout and physical page size. To learn more, see [Pagination in Reporting Services \(Report Builder and SSRS\)](#).

How-To Topics

This section lists procedures that show you, step by step, how to work with pagination in reports.

- [Add a Page Break \(Report Builder and SSRS\)](#)
- [Display Row and Column Headers on Multiple Pages \(Report Builder and SSRS\)](#)
- [Add or Remove a Page Header or Footer \(Report Builder and SSRS\)](#)
- [Keep Headers Visible When Scrolling Through a Report \(Report Builder and SSRS\)](#)
- [Display Page Numbers or Other Report Properties \(Report Builder and SSRS\)](#)
- [Hide a Page Header or Footer on the First or Last Page \(Report Builder and SSRS\)](#)

In This Section

The following topics provide additional information about page layout and rendering.

[Page Headers and Footers \(Report Builder and SSRS\)](#)

Provides information about using headers and footers in reports and how to control pagination using them.

[Controlling Page Breaks, Headings, Columns, and Rows \(Report Builder and SSRS\)](#)

Provides information about using page breaks.

See Also

[Interactive Functionality for Different Report Rendering Extensions \(Report Builder and SSRS\)](#)

Export Reports (Report Builder and SSRS)

Rendering Behaviors (Report Builder and SSRS)

3/24/2017 • 7 min to read • [Edit Online](#)

Depending on the renderer you select, certain rules are applied to the report body and its contents when rendering a report. How report items fit together on a page is determined by the combination of these factors:

- Rendering rules.
- The width and height of report items.
- The size of the report body.
- The width and height of the page.
- Renderer-specific support for paging.

This topic discusses the general rules that are applied by Reporting Services. For more information, see [Rendering Report Items \(Report Builder and SSRS\)](#), [Rendering Data Regions \(Report Builder and SSRS\)](#), and [Rendering Data \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

General Behaviors for HTML, MHTML, Word, and Excel (Soft Page-Break Renderers)

Reports exported using HTML and MHTML formats are optimized for a computer screen-based experience where pages can be various lengths. Page breaks are inserted vertically only at approximate locations within the report body. These approximate locations are determined by the interactive height setting in the Properties pane. For example, suppose the interactive height is set to 5 inches. When the report is rendered, the page height is approximately 5 inches in length. Word and Excel paginate based on logical page breaks and ignore the interactive height setting.

NOTE

To determine how a report will appear in a soft page-break renderer, use Report Preview. The report appears as it would in a HTML, MHTML, Word, or Excel format.

When exporting a report to HTML, MHTML, Word, or Excel, the following general rules are followed:

- Logical page breaks, the page breaks that you explicitly insert, are applied to report items. For example, if you insert a page break between each group, they are applied when the report is rendered.
- An approximate layout is created using the page height and the number of times that the report item appears. For example, if a text box is .5 inches in height and repeats five times in the report, 2.5 inches are reserved.
- Multiple soft page breaks are inserted based on the interactive height setting. To suppress in HTML and the ReportViewer controls, and control pagination only with explicit page breaks, set the **interactive height** value to 0 or an extremely large number.

NOTE

The interactive width setting is not used in the soft page break renderers.

- Report pages can grow to accommodate widows, orphans and report items that need to be kept together. This means that the report can extend beyond the screen width and can be viewed by using slider bars.
- Pagination is applied to reports vertically only.
- Page margins are not applied.

General Behaviors for PDF, Image, and Print (Hard Page-Break Renderers)

Reports exported using PDF and Image are optimized for a book-like or printed experience where pages are a consistent size. Page breaks are inserted vertically and horizontally at specific locations within the report body. These specific locations are determined by the page width and page height settings.

NOTE

To determine how a report will appear in a hard page-break renderer, use Print Preview. The report appears as it would in a PDF or Image format.

- Pages are numbered sequentially from left to right, then top to bottom.
- Logical page breaks, the page breaks that you explicitly insert, are applied to report items. These page breaks can cause report items to push other items to the next page.
- If a physical page break occurs through report items that must be kept together, the items that must be kept together are moved to the next page.
- Because of page size constraints, it may not be possible to keep all the items together or to repeat items. If that occurs, the renderer might ignore certain rules for repeating with another item in order to allow the report item to fit on the page.
- If an item cannot be kept together, for example a text box that grows too large to fit within the vertical usable page area, then the item will be clipped at the physical page boundary and will continue on the next page.
- Pagination is applied to reports vertically and horizontally.

NOTE

The interactive width setting is not used in the hard page break renderers.

Minimum Spacing Between Report Items

Report items grow within the report body to accommodate their contents. For example, a matrix data region typically expands across and down the page when the report is rendered, and the height of a text box adjusts depending on the data returned from an expression.

Renderers maintain the minimum space between report items that you define in the report layout. When you place a report item adjacent to another on the report layout, the distance between the report items becomes the

minimum distance that must be maintained as the report grows horizontally or vertically. For example, if you add a matrix data region to a report and then add a rectangle .25 inches to the right of the matrix, that space is maintained as the matrix grows. Each item moves to the right to maintain the minimum distance from items that end to the left of it.

Page Headers and Footers

Page headers and footers appear at the top and bottom of each rendered page. You can format the page header and footer so that there is a border color, border style, and border width. You can also add a background color or background image. These formatting options are all rendered, depending on the format that you choose.

The following rules apply to page headers and footers when rendered in the HTML or MHTML rendering format:

NOTE

For information about how Excel renders headers and footers, see [Exporting to Microsoft Excel \(Report Builder and SSRS\)](#).

For information about how Word renders headers and footers, see [Exporting to Microsoft Word \(Report Builder and SSRS\)](#).

- When present, the header and footer is rendered at the top and bottom of every page within the usable page area.
- On pages where the header or footer is hidden, the height of the header or footer is still reserved within the usable page area, even though the header or footer is not rendered.
- If the contents of the header or footer grows beyond the boundaries of the header or footer, the header or footer increases in size to accommodate the contents.

The following rules apply to page headers and footers when rendered in the PDF or Image rendering format:

- The header or footer is rendered at the top and bottom of every page within the usable page area.
- On pages where the header or footer is hidden, the height of the header or footer is still reserved within the usable page area, even though the header or footer is not rendered.
- The header and footer do not grow or shrink. They are rendered on every page at the height specified when you created the header or footer.
- Regardless of the number of columns within the report, there is only one header and footer per page.
- If the contents of the header or footer grow beyond the boundaries of the header or footer, the contents are clipped.
- Headers and footers that are defined in the original RDL file are not rendered when the report is rendered as a subreport.

Logical Page Breaks

Logical page breaks are page breaks that you insert before or after report items or groups. Page breaks help to determine how the content is fitted to a report page for optimal viewing when rendering or exporting the report.

The following rules apply when rendering logical page breaks:

- Logical page breaks are ignored for report items that are constantly hidden and for report items where the visibility is controlled by clicking another report item.

- Logical page breaks are applied on conditionally visible items if they are currently visible at the time the report is rendered.
- Space is preserved between the report item with the logical page break and its peer report items.
- Logical page breaks that are inserted before a report item push the report item down to the next page. The report item is rendered at the top of the next page.
- Logical page breaks defined on items in table or matrix cells are not kept. This does not apply to items in lists.

See Also

[Interactive Functionality for Different Report Rendering Extensions \(Report Builder and SSRS\)](#)

[Rendering to HTML \(Report Builder and SSRS\)](#)

[Page Layout and Rendering \(Report Builder and SSRS\)](#)

Rendering Data Regions (Report Builder and SSRS)

3/24/2017 • 3 min to read • [Edit Online](#)

In addition to the general rendering behaviors that apply to all report items, data regions have additional pagination and rendering behaviors that they follow. Data region-specific rendering rules include how a data region grows, how special cells such as the corner cell or header cells are rendered, and how a data region for right-to-left reading is rendered. This topic discusses how the various parts of a data region are rendered.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Tablix Data Regions

The tablix data region, which enables you to create tables, matrices, and lists, is rendered as a grid comprised of columns and rows. The intersection of a row and a column is a cell. When rendered, this cell can contain data or other report items, such as images, rectangles, text boxes, or subreports. A tablix data region can grow vertically and/or horizontally. In addition, the corner cell, the data region header cells, and the data region body cells may grow based on their contents. If the data region spans multiple pages, report items that are set to repeat with the data region are rendered on every page on which the data region is displayed. For more information, see [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#).

Right to Left

A tablix data region set to display from right to left is rendered with its structure as a mirror image of the data region if it were rendered left to right. The corner of the data region appears in the upper right corner. If dynamic columns exist in the report, they expand to the left. Right-to-left settings do not affect the order of the data in the data region; your columns are simply ordered differently.

Tablix Headers

Tablix headers are rendered as a row header or a column header depending on where the header cell appears in the row group hierarchy or the column group hierarchy. If a logical page break exists within the cell contents of a header, it is ignored. Logical page breaks on column groups are ignored.

Logical page breaks on groups do not cause outer group headers to break. For example, suppose your report has an outer group of country and an inner group of country region. If there is a logical page break between instances of the country region group, the outer group, the country, will appear on both pages of the report.

Repeated Tablix Headers

When the RepeatWith property is set in the **Properties** pane, items that do not change within the data region, such as column headers, repeat on each page where that part of the data region is rendered. For example, if a row of data appears on the next page and the Repeat With property is set, the column headers appear on the rendered page as well.

Tablix Corner

The upper left corner is called the tablix corner. The Tablix corner can contain other report items within it but, if logical page breaks are inserted in the corner, they are ignored when the Tablix data region is rendered.

Tablix Body

The Tablix body is made up of Tablix cells. The Tablix body is rendered based on pagination rules and the rendering

behaviors of report items. For more information, see [Rendering Report Items \(Report Builder and SSRS\)](#).

Chart, Gauge, and Map Data Regions

Chart, Gauge, and Map data regions behave like images when they are rendered and displayed in the report body. Values within the data region can have associated actions, such as linking to another report or going to a bookmark, and these actions can be rendered as well, if the renderer supports it.

See Also

[Pagination in Reporting Services \(Report Builder and SSRS\)](#)

[Rendering Behaviors \(Report Builder and SSRS\)](#)

[Interactive Functionality for Different Report Rendering Extensions \(Report Builder and SSRS\)](#)

[Rendering Report Items \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Charts \(Report Builder and SSRS\)](#)

[Gauges \(Report Builder and SSRS\)](#)

Rendering Report Items (Report Builder and SSRS)

3/24/2017 • 5 min to read • [Edit Online](#)

The number, size, and location of report items affect how the renderers paginate the report body. Below is a description of how various report items are rendered.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Overlapping Report Items

Overlapping report items are not supported in HTML, MHTML, Word, Excel, in Preview, or the Report Viewer. If overlapping items exist, they are moved. The following rules are applied to overlapping report items:

- If the vertical overlap of report items is greater, one of the overlapping items is moved to the right. The left-most item remains where it is positioned.
- If the horizontal overlap of report items is greater, one of the overlapping items is moved down. The top-most item remains where it is positioned.
- If the vertical and horizontal overlap is equal, one of the overlapping items is moved to the right. The left-most item remains where it is positioned.
- If an item must be moved to correct overlapping, adjacent report items move down and/or to the right to maintain a minimum amount of spacing between the item and the report items that end above it and/or to the left of it. For example, suppose two report items overlap vertically and a third report item is 2 inches to the right of them. When the overlapping report item is moved to the right, the third report item moves to the right as well in order to maintain the 2 inches between itself and the report item to its left.

Overlapping report items are supported in hard page-break formats, including print.

Visibility and Report Items

Report items can be hidden or displayed by default, or hidden or displayed conditionally using expressions. Optionally, the visibility can be switched by clicking another report item.

The following visibility rules apply when rendering report items:

- If a report item and its contents are always hidden (it is not hidden based on an expression or its visibility cannot be switched by clicking another report item), then other report items to the right or below it do not move to fill the empty space. For example, if a rectangle and the image contained within it are hidden, the report item that starts to the right of the rectangle does not move to the left to fill what appears to be empty space. The space occupied by the rectangle is preserved.
- If a report item and its contents are hidden conditionally (it is hidden based on an expression or its visibility is switched by clicking another report item), then report items to the right or below it move to the left to fill in the space when the item is hidden.
- If the visibility of a report item and its contents can be switched by clicking another report item, then pagination changes to accommodate the report item and its contents only when it is initially displayed.

Keeping Report Items Together on a Single Page

Many report items within a report can be kept together on a single page implicitly or explicitly by setting the keep with group or keep together properties. Report items are always rendered on the same page if the report item does not have any logical page breaks and is smaller in size than the usable page area. If a report item does not fit completely on the page on which it would usually start, a hard page break is inserted before the report item, forcing it to the next page. For soft page-break renderers, the page grows to accommodate the report item.

When the report item is always hidden, the rules for keeping items together are ignored.

The following items are always kept together:

- Images.
- Lines.
- Charts, gauges, and maps.
- A single row in a data region that appears separately on another page, by selecting the keep with group option. This will implicitly keep together the single row with at least one instance of the group so that the row is not orphaned. You can set this option on a data region or a group.
- Header area of a data region.
- Header area of a data region and the first row of data.
- Report items that can be toggled in a tablix data region.

Priority Order

Due to page size limitations, conflicts can arise between the rules for keeping report items together. When conflicts occur, the following priority order is used to keep items together when rendering:

- Lines, charts, and images.
- Widow and orphan control.
- Repeated column headers and row headers.

Headers take precedence over footers. Inner repeated groups have priority over outer groups. Items where the **RepeatWith** property is set that are closer to the target data region have priority over items that are farther away from the data region.

- Small report items, such as text boxes or rectangles, with an explicit KeepTogether property set to **true**.
- Large report items, such as subreports or a non-innermost tablix member, with an explicit KeepTogether property set to **true**.
- Tablix data regions with an explicit KeepTogether property set to **true**.

Subreports

A subreport renders as a rectangle that contains another report that is defined in a separate report .rdl file. The subreport file must be published to a report server before it can be accessed by the parent report.

The following rules apply when rendering subreports:

- Subreports can grow to the size of the body defined in the .rdl file that defines the subreport. For example, if the RDL for the subreport states that the subreport body is 5 inches wide, then the subreport will be 5 inches wide within the parent report.
- Subreports inherit column settings from the parent report. Column settings that are defined in the original RDL are always ignored.

- Only the body of the subreport is rendered. Header and footer sections that are defined in the subreport's .rdl file are not rendered when the subreport is rendered in the parent report.
- Subreports have an explicit KeepTogether property. When it is set to **true**, all the items in the subreport are kept together on one page when possible.
- If a subreport cannot run, it is displayed in the report as a text box with an error message. The style properties applied to the subreport are applied to the text box instead.
- If the subreport is split by a page break, the **Omit border on page break** setting controls whether or not the borders on the subreport are closed or open.

For more information about subreports, see [Subreports \(Report Builder and SSRS\)](#).

See Also

[Pagination in Reporting Services \(Report Builder and SSRS\)](#)

[Rendering Behaviors \(Report Builder and SSRS\)](#)

[Interactive Functionality for Different Report Rendering Extensions \(Report Builder and SSRS\)](#)

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

Rendering Data (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

When using layout renderers, such as HTML, MHTML, Word, Excel, PDF or Image, the data and its organization remains unchanged. When exporting using a data renderer format, such as Comma-Separated Value (CSV) or XML, no visual layout elements are rendered. CSV and XML apply certain rules to the report body and its contents when rendering the report. These rules determine how the data is rendered in these formats.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

You can use data renderers to:

- Import to a database. CSV is a common format that is easily importable by many database applications including SQL Server and Microsoft Access.
- Import to Excel. Use the CSV renderer to export your data to Excel without the visual layout. After your data is in Excel, you can take advantage of standard Excel tools such as charts, formulas, and pivot tables.
- XSLT transformations. An XSLT can be applied to the output of the XML renderer. This server-side transformation is a powerful technique to transform your data to virtually any format.
- Data exchange/EDI. An external process can request a CSV or XML rendering of a report and consume that data.

Data renderer formats are controlled by a different set of properties than layout renderers. The following is a list of properties set in the **Properties** pane that apply only to data renderers:

- The **DataElementOutput** property controls whether or not a specific item is present in the data when exported.
- The **DataElementName** property controls the name of the data element. In CSV, this controls the name of the CSV column header. In XML, this becomes the name of the XML element or attribute for the item.
- The **DataElementStyle** property controls, in XML, whether or not the report item is rendered as an element or an attribute.

The CSV export option saves report data as comma-delimited plain text files, without any formatting. By default, the file uses a comma (,) to delimit fields and rows but this setting is configurable using the Device Information Settings. The resulting file can be opened in a spreadsheet program like Office SharePoint Server or used as an import format for other programs. The .csv file opens in a text editor such as Notepad. If accessed as a URL, the .csv file returns a MIME type of **text/csv**. The files are MIME version 1.0. For more information about rendering your report in the CSV file type, see [Exporting to a CSV File \(Report Builder and SSRS\)](#).

The XML file with report data export option saves a report as an XML file. The XML schema for the report is specific to the report. The report layout information is not saved by the XML export option. The XML generated using this option can be imported into a database, used as an XML data message, or sent to a custom application. For more information about rendering your report in the XML file type, see [Exporting to XML \(Report Builder and SSRS\)](#).

See Also

- [Pagination in Reporting Services \(Report Builder and SSRS\)](#)
- [Rendering Behaviors \(Report Builder and SSRS\)](#)
- [Interactive Functionality for Different Report Rendering Extensions \(Report Builder and SSRS\)](#)
- [Rendering Report Items \(Report Builder and SSRS\)](#)
- [Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)
- [Reporting Services Device Information Settings](#)

Pagination in Reporting Services (Report Builder and SSRS)

3/24/2017 • 8 min to read • [Edit Online](#)

Pagination refers to the number of pages within a report and how report items are arranged on these pages. Pagination in Reporting Services varies depending on the rendering extension you use to view and deliver the report. When you run a report on the report server, the report uses the HTML renderer. HTML follows a specific set of pagination rules. If you export the same report to PDF, for example, the PDF renderer is used and a different set of rules are applied; therefore, the report paginates differently. To successfully design an easy-to-read report for your users that is optimized for the renderer that you plan to use to deliver your report, you need to understand the rules used to control pagination in Reporting Services.

This topic discusses the impact of the physical page size and the report layout on how hard page break renderers render the report. You can set properties to modify the physical page size and margins, and divide the report into columns, by using the **Report Properties** pane, the **Properties** pane, or the **Page Setup** dialog box. You access the **Report Properties** pane by clicking the blue area outside the report body. You access the **Page Setup** dialog box by clicking **Run** on the Home tab, and then clicking **Page Setup** on the Run tab.

NOTE

If you have designed a report to be one page wide, but it renders across multiple pages, check that the width of the report body, including margins, is not larger than the physical page size width. To prevent empty pages from being added to your report, you can reduce the container size by dragging the container corner to the left.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

The Report Body

The report body is a rectangular container displayed as white space on the design surface. It can grow or shrink to accommodate the report items contained within it. The report body does not reflect the physical page size and, in fact, the report body can grow beyond the boundaries of the physical page size to span multiple report pages. Some renderers, such as Microsoft Excel, Word, HTML and MHTML, render reports that grow or shrink depending on the contents of the page. Reports rendered in these formats are optimized for screen-based viewing, such as in a Web browser. These renderers add vertical page breaks when required.

You can format the report body so that there is a border color, border style and border width. You can also add a background color and background image.

The Physical Page

The physical page size is the paper size. The paper size that you specify for the report controls how the report is rendered. Reports rendered in hard page break formats insert page breaks horizontally and vertically based on the physical page size to provide an optimized reading experience when printed or viewed in a hard page break file format. Reports rendered in soft page break formats insert page breaks horizontally based on the physical size to provide an optimized reading experience when viewed in a Web browser.

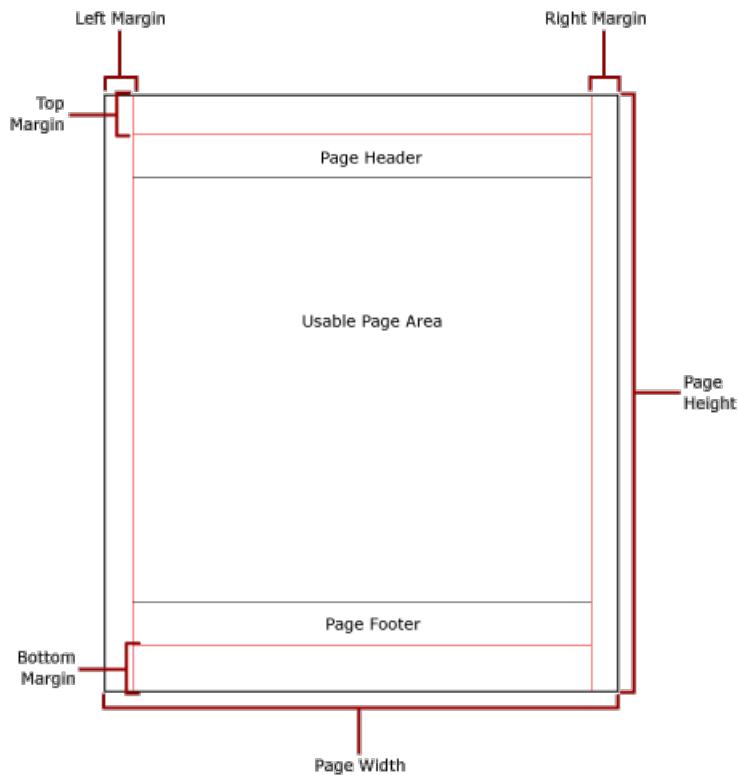
By default, the page size is 8.5 x 11 inches but you can change this size by using the **Report Properties** pane, **Page Setup** dialog box or by changing the PageHeight and PageWidth properties in the **Properties** pane. The page size does not grow or shrink to accommodate the contents of the report body. If you want the report to appear on a single page, all the content within the report body must fit on the physical page. If it does not fit and you use the hard page break format, then the report will require additional pages. If the report body grows past the right edge of the physical page, then a page break is inserted horizontally. If the report body grows past the bottom edge of the physical page, then a page break is inserted vertically.

If you want to override the physical page size that is defined in the report, you can specify the physical page size using the Device Information settings for the specific renderer that you are using to export the report. For more information, see [Reporting Services Device Information Settings](#).

Margins

Margins are drawn from the edge of the physical page dimensions inward to the specified margin setting. If a report item extends into the margin area, it is clipped so that the overlapping area is not rendered. If you specify margin sizes that cause the horizontal or vertical width of the page to equal zero, the margin settings default to zero. Margins are specified using the **Report Properties** pane, **Page Setup** dialog box or by changing the TopMargin, BottomMargin, LeftMargin and RightMargin properties in the **Properties** pane. If you want to override the margin size that is defined in the report, you can specify the margin size using the Device Information settings for the specific renderer that you are using to export the report.

The area of the physical page that remains after space is allocated for margins, column spacing, and the page header and footer, is called the *usable page area*. Margins are only applied when you render and print reports in hard page break renderer formats. The following image indicates the margin and usable page area of a physical page.

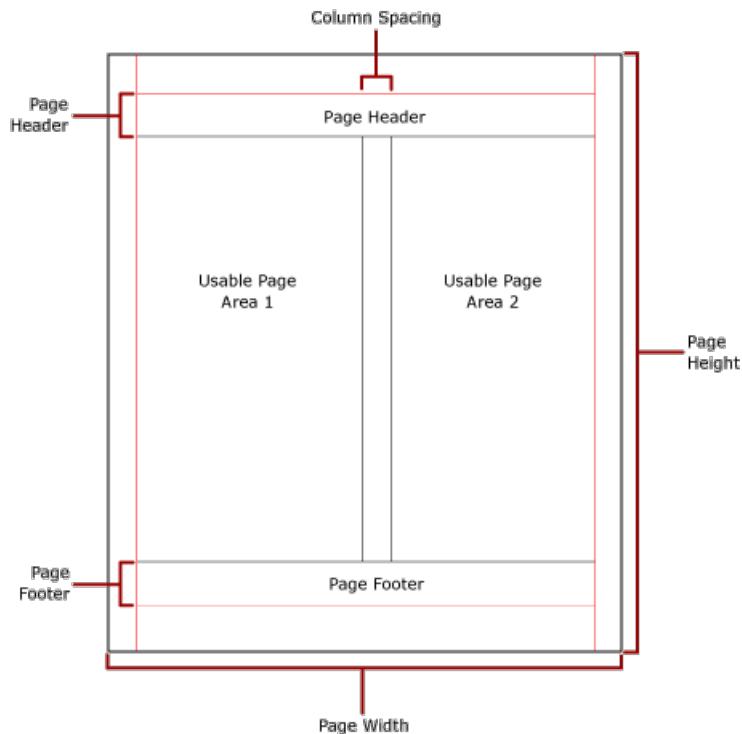


Newsletter-Style Columns

Your report can be divided into columns, such as columns in a newspaper, that are treated as logical pages rendered on the same physical page. They are arranged from left to right, top to bottom, and are separated by white space between each column. If the report is divided into more than one column, each physical page is divided vertically into columns, each of which is considered a logical page. For example, suppose you have two columns on a physical page. The content of your report fills the first column and then the second column. If the report does not fit entirely within the first two columns, the report fills the first column and then the second

column on the next page. Columns continue to be filled, from left to right, top to bottom until all report items are rendered. If you specify column sizes that cause the horizontal width or vertical width to equal zero, the column spacing defaults to zero.

Columns are specified using the **Report Properties** pane, **Page Setup** dialog box or by changing the TopMargin, BottomMargin, LeftMargin and RightMargin properties in the **Properties** pane. If you want to use a margin size that is not defined, you can specify the margin size using the Device Information settings for the specific renderer that you are using to export the report. Columns are only applied when you render and print reports in PDF or Image formats. The following image indicates the usable page area of a page containing columns.



Page Breaks and Page Names

A report might be more readable and its data easier to audit and export when the report has page names. Reporting Services provides properties for reports and tablix data regions (table, matrix, and list), groups, and rectangles in the report to control pagination, reset page numbers, and provide new report page names on page breaks. These features can enhance reports regardless of the format in which reports are rendered, but are especially useful when exporting reports to Excel workbooks.

The InitialPageName property provides the initial page name of the report. If your report does not include page names for page breaks, then the initial page name is used for all the new pages created by page breaks. It is not required to use an initial page name.

A rendered report can provide a new page name for the new page that a page break causes. To provide the page name, you set the PageName property of a table, matrix, list, group, or rectangle. It is not required that you specify page names on breaks. If you do not, the value of InitialPageName is used instead. If InitialPageName is also blank, the new page has no name.

Tablix data regions (table, matrix, and list), groups, and rectangles support page breaks.

The page break includes the following properties:

- BreakLocation provides the location of the break for the page break enabled report element: at the start, end, or start and end. On groups, BreakLocation can be located between groups.
- Disabled indicates whether a page break is applied to the report element. If this property evaluates to

True, the page break is ignored. This property is used to dynamically disable page breaks based on expressions when the report is run.

- ResetPageNumber indicates whether the page number should be reset to 1 when a page break occurs. If this property evaluates to True, the page number is reset.

You can set the BreakLocation property in the **Tablix Properties**, **Rectangle Properties**, or **Group Properties** dialog boxes, but you must set the Disabled, ResetPageNumber, and PageName properties in the Report Builder Properties pane. If the properties in the Properties pane are organized by category, you will find the properties in the **PageBreak** category. For groups, the **PageBreak** category is inside the **Group** category.

You can use constants and simple or complex expressions to set the value of the Disabled and ResetPageNumber properties. However, you cannot use expression with the BreakLocation property. For more information about writing and using expressions, see [Expressions \(Report Builder and SSRS\)](#).

In your report you can write expressions that reference the current page names or page numbers by using the **Globals** collection. For more information, see [Built-in Globals and Users References \(Report Builder and SSRS\)](#).

Naming Excel Worksheet Tabs

These properties are useful when you export reports to Excel workbooks. Use the InitialPage property to specify a default name for the worksheet tab name when you export the report, and use page breaks and the PageName property to provide different names for each worksheet. Each new report page, defined by a page break, is exported to a different worksheet named by the value of the PageName property. If PageName is blank, but the report has an initial page name, then all worksheets in the Excel workbook use the same name, the initial page name.

For more information about how these properties work when reports are exported to Excel, see [Exporting to Microsoft Excel \(Report Builder and SSRS\)](#).

See Also

[Page Layout and Rendering \(Report Builder and SSRS\)](#)

Page Headers and Footers (Report Builder and SSRS)

3/24/2017 • 8 min to read • [Edit Online](#)

A report can contain a header and footer that run along the top and bottom of each page, respectively. Headers and footers can contain static text, images, lines, rectangles, borders, background color, background images, and expressions. Expressions include dataset field references for reports with exactly one dataset and aggregate function calls that include the dataset as a scope.

NOTE

Each rendering extension processes pages differently. For more information about report pagination and rendering extensions, see [Pagination in Reporting Services \(Report Builder and SSRS\)](#).

By default, reports have page footers, but not page headers. For more information about how to add or remove them, see [Add or Remove a Page Header or Footer \(Report Builder and SSRS\)](#).

Headers and footers commonly contain page numbers, report titles, and other report properties. For more information about how to add these items to your report header or footer, see [Display Page Numbers or Other Report Properties \(Report Builder and SSRS\)](#).

After you create a page header or footer, it is displayed on each report page. For more information about how to suppress page headers and footers on the first and last pages, see [Hide a Page Header or Footer on the First or Last Page \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

Report Headers and Footers

Page headers and footers are not the same as report headers and footers. Reports do not have a special report header or report footer area. A report header consists of the report items that are placed at the top of the report body on the report design surface. They appear only once as the first content in the report. A report footer consists of report items that are placed at the bottom of the report body. They appear only once as the last content in the report.

Displaying Variable Data in a Page Header or Footer

Page headers and footers can contain static content, but they are more commonly used to display varying content like page numbers or information about the contents of a page. To display variable data that is different on each page, you must use an expression.

If there is only one dataset defined in the report, you can add simple expressions such as `[FieldName]` to a page header or footer. Drag the field from the Report Data pane dataset field collection or the Built-in Fields collection to the page header or page footer. A text box with the appropriate expression is automatically added for you.

To calculate sums or other aggregates for values on the page, you can use aggregate expressions that specify

ReportItems or the name of a dataset. The ReportItems collection is the collection of text boxes on each page after report rendering occurs. The dataset name must exist in the report definition. The following table displays which items are supported in each type of aggregate expression:

SUPPORTED IN EXPRESSION	REPORTITEMS AGGREGATES	DATASET AGGREGATES (SCOPE MUST BE NAME OF DATASET)
Text boxes in body of report	Yes	No
&PageNumber	Yes	No
&TotalPages	Yes	No
Aggregate function	Yes. For example, <code>=First(ReportItems!TXT_LastName.Value) =Max(Quantity.Value,"DataSet1")</code>	Yes. For example,
Fields collection for items on the page	Indirectly. For example, <code>=Sum(ReportItems!Textbox1.Value)</code>	Yes. For example, <code>=Sum(Fields!Quantity.Value,"DataSet1")</code>
Data-bound image	Indirectly. For example, <code>=ReportItems!TXT_Photo.Value</code>	Yes. For example, <code>=First(Fields!Photo.Value,"DataSet1")</code>

The following sections in this topic show ready-to-use expressions that get variable data commonly used in headers and footers. There is also a section on how the Excel rendering extension processes headers and footers. For more information about expressions, see [Expressions \(Report Builder and SSRS\)](#).

Adding Calculated Page Totals to a Header or Footer

For some reports, it is useful to include a calculated value in the header or footer of each report; for example, a per-page sum total if the page includes numeric values. Because you cannot reference the fields directly, the expression that you put in the header or footer must reference the name of the report item (for example, a text box) rather than the data field:

```
=Sum(ReportItems!Textbox1.Value)
```

If the text box is in a table or list that contains repeated rows of data, the value that appears in the header or footer at run time is a sum of all values of all `TextBox1` instance data in the table or list for the current page.

When calculating page totals, you can expect to see differences in the totals when you use different rendering extensions to view the report. Paginated output is calculated differently for each rendering extension. The same page that you view in HTML might show different totals when viewed in PDF if the amount of data on the PDF page is different. For more information, see [Rendering Behaviors \(Report Builder and SSRS\)](#).

For Reports with Multiple Datasets

For reports with more than one dataset, you cannot add fields or data-bound images directly to a header or footer. However, you can write an expression that indirectly references a field or data-bound image that you want to use in a header or footer.

To put variable data in a header or footer:

- Add a text box to the header or footer.
- In the text box, write an expression that produces the variable data that you want to appear.
- In the expression, include references to report items on the page; for example, you can reference a text box

that contains data from a particular field. Do not include a direct reference to fields in a dataset. For example, you cannot use the expression `[LastName]`. You can use the following expression to display the contents of the first instance of a text box named `TXT_LastName`:

```
=First(ReportItems!TXT_LastName.Value)
```

You cannot use aggregate functions on fields in the page header or footer. You can only use an aggregate function on report items in the report body. For common expressions in page headers and footers, see [Expression Examples \(Report Builder and SSRS\)](#).

Adding a Data-Bound Image to a Header or Footer

You can use image data stored in a database in a header or footer. However, you cannot reference database fields from the Image report item directly. Instead, you must add a text box in the body of the report and then set the text box to the data field that contains the image (note that the value must be base64 encoded). You can hide the text box in the body of the report to avoid showing the base64-encoded image. Then, you can reference the value of the hidden text box from the Image report item in the page header or footer.

For example, suppose you have a report that consists of product information pages. In the header of each page, you want to display a photograph of the product. To print a stored image in the report header, define a hidden text box named `TXT_Photo` in the body of the report that retrieves the image from the database and use an expression to give it a value:

```
=Convert.ToString(Fields!Photo.Value)
```

In the header, add an Image report item which uses the `TXT_Photo` text box, decoded to show the image:

```
=Convert.FromBase64String(ReportItems!TXT_Photo.Value)
```

Using Headers and Footers to Position Text

You can use headers and footers to position text on a page. For example, suppose you are creating a report that you want to mail out to customers. You can use a header or footer to position the customer address so that it appears in an envelope window when folded.

If you are only using the text box to populate a header or footer, you can hide the text box in the report body. Placement of the text box in the report body can have an effect on whether the value appears on the header or footer of the first or last page of a report. For example, if you have tables, matrices, or lists that cause the report to span multiple pages, the hidden text box value appears on the last page. If you want it to appear on the first page, place the hidden text box at the top of the report body.

Designing Reports with Page Headers and Footers for Specific Renderers

When a report is processed, data and layout information are combined. When you view a report, the combined information is passed to a renderer that determines how much report data fits on each report page.

If you view a report on the report server using a browser, the HTML renderer controls the content on the report pages that you see. If you plan to deliver reports in a different format than you use for viewing, or if you plan to print reports in a specific format, you may want to optimize the report layout for the renderer you plan to use for the final report format. For more information about report pagination, see [Pagination in Reporting Services \(Report Builder and SSRS\)](#).

Working with Page Headers and Footers in Excel

When defining page headers and footers for reports that target the Excel rendering extension, follow these guidelines to achieve best results:

- Use page footers to display page numbers.
- Use page headers to display images, titles, or other text. Do not put page numbers in the header.

In Excel, page footers have a limited layout. If you define a report that includes complex report items in the page footer, the page footer won't process as you expect when the report is viewed in Excel.

The Excel rendering extension can accommodate images and absolute positioning of simple or complex report items in the page header. A side effect of supporting a richer page header layout is reduced support for calculating page numbers in the header. In the Excel rendering extension, default settings cause page numbers to be calculated based on the number of worksheets. Depending on how you define the report, this might produce erroneous page numbers. For example, suppose you have a report that renders as a single large worksheet that prints on four pages. If you include page number information in the header, each printed page will show "Page 1 of 1" in the header.

A more accurate page count is based on logical pages that correlate to the dimensions of a printed page. In Excel, the page footer uses logical page numbers automatically. To put the logical page count in the page header, you must configure the device information settings to use simple headers. Be aware that when you use simple headers, you remove the capability of handling complex report layout in the header region.

For more information, see [Exporting to Microsoft Excel \(Report Builder and SSRS\)](#).

See Also

[Embed an Image in a Report \(Report Builder and SSRS\)](#)

[Rectangles and Lines \(Report Builder and SSRS\)](#)

Controlling Page Breaks, Headings, Columns, and Rows (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

A page break divides a report into separate pages for viewing and printing. Page breaks determine how the content is fitted to a report page for optimal viewing when you preview a report or export it to a different file format.

Adding page breaks also improves the performance of large reports when they are processed. A rendered page is displayed while the rest of the pages are rendered in the background. This allows you to begin viewing the initial pages of the report while waiting for additional pages to become available.

Page breaks can be added to report items such as a table, matrix, list, chart, gauge, or image. You can also add page breaks to groups in a table, matrix, or list. Page breaks can be added before, after, and between groups. Page breaks between groups are not added to the report by default.

For more information, see [Display Row and Column Headers on Multiple Pages \(Report Builder and SSRS\)](#) and [Keep Headers Visible When Scrolling Through a Report \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

See Also

[Tables, Matrices, and Lists \(Report Builder and SSRS\)](#)

[Controlling the Tablix Data Region Display on a Report Page \(Report Builder and SSRS\)](#)

[Grouping Pane \(Report Builder\)](#)

[Display Headers and Footers with a Group \(Report Builder and SSRS\)](#)

Add a Page Break (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can add a page break to rectangles, data regions, or groups within data regions to control the amount of information on each page. Adding page breaks can improve the performance of published reports because only the items on each page have to be processed as you view the report. When the whole report is a single page, all items must be processed before you can view the report.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a page break to a data region

1. On the design surface, right-click the corner handle of the data region and then click **Tablix Properties**.
2. On the **General** tab, under **Page break options**, select one of the following options:
 - **Add a page break before**. Select this option when you want to add a page break before the table.
 - **Add a page break after**. Select this option when you want to add a page break after the table.
 - **Fit table on one page if possible**. Select this option when you want the data to stay on one page.

To add a page break to a rectangle

1. On the design surface, right-click the rectangle where you want to add a page break, and then click **Rectangle Properties**.
2. On the **General** tab, under **Page break options**, select one of the following options:
 - **Add a page break before**. Select this option when you want to add a page break before the rectangle.
 - **Add a page break after**. Select this option when you want to add a page break after the rectangle.
 - **Omit border on page break**. Select this option when you do not want a border on the page break.
 - **Keep contents together on a single page, if possible**. Select this option when you want contents inside the rectangle to stay on one page.

To add a page break to a row group in a table, matrix, or list

1. In the Grouping pane, right-click a row group, and then click **Group Properties**.
2. On the **Page Breaks** tab, select **Between each instance of a group** to add a page break between each instance of a group in the table.
3. Optionally, select **Also at the start of a group** or **Also at the end of a group** to specify that a page break be added when a group starts or ends in the table.

NOTE

Page breaks are ignored on column groups.

See Also

[Pagination in Reporting Services \(Report Builder and SSRS\)](#)

[Rendering Behaviors \(Report Builder and SSRS\)](#)

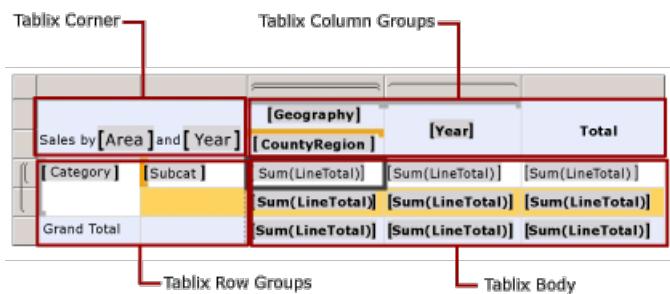
[Page Headers and Footers \(Report Builder and SSRS\)](#)

Display Row and Column Headers on Multiple Pages (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

You can control whether to repeat row and column headers on every page of a Reporting Services paginated report for a tablix data region (a table, matrix, or list) that spans multiple pages.

How you control the rows and columns depends on whether the tablix data region has group headers. When you click in a tablix data region that has group headers, a dotted line shows the tablix areas, as shown in the following figure:



Row and column group headers are created automatically when you add groups by using the New Table or Matrix wizard or the the New Chart wizard, by adding fields to the Grouping pane, or by using context menus. If the tablix data region has only a tablix body area and no group headers, the rows and columns are tablix members.

For static members, you can display the top adjacent rows or the side adjacent columns on multiple pages.

To display row headers on multiple pages

1. Right-click the row, column, or corner handle of a tablix data region, and then click **Tablix Properties**.
2. In **Row Headers**, select **Repeat header rows on each page**.
3. Click **OK**.

To display column headers on multiple pages

1. Right-click the row, column, or corner handle of a tablix data region, and then click **Tablix Properties**.
2. In **Column Headers**, select **Repeat header columns on each page**.
3. Click **OK**.

To display a static row or column on multiple pages

1. On the design surface, click the row or column handle of the tablix data region to select it. The Grouping pane displays the row and column groups.
2. On the right side of the Grouping pane, click the down arrow, and then click **Advanced Mode**. The Row Groups pane displays the hierarchical static and dynamic members for the row groups hierarchy and the Column groups pane shows a similar display for the column groups hierarchy.
3. Click the static member that corresponds to the static member (row or column) that you want to remain

visible while scrolling. The Properties pane displays the **Tablix Member** properties.

If you don't see the Properties pane, click the **View** tab at the top of the Report Builder window and then click **Properties**.

4. In the Properties pane, set **RepeatOnNewPage** to True.
5. Set **KeepWithGroup** to After.
6. Repeat this for as many adjacent members as you want to repeat.
7. Preview the report.

As you view each page of the report that the tablix data region spans, the static tablix members repeat on each page.

See Also

[Finding, Viewing, and Managing Reports \(Report Builder and SSRS \)](#)

[Export Reports \(Report Builder and SSRS\)](#)

[Controlling Page Breaks, Headings, Columns, and Rows \(Report Builder and SSRS\)](#)

[Display Headers and Footers with a Group \(Report Builder and SSRS\)](#)

[Keep Headers Visible When Scrolling Through a Report \(Report Builder and SSRS\)](#)

Add or Remove a Page Header or Footer (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

You can add static text, images, lines, rectangles, and borders to page headers or footers. You can place expressions and data-bound images in a textbox if you want variable or computed data in a header or footer.

NOTE

Each rendering extension processes page headers and footers differently. For more information, see [Page Headers and Footers \(Report Builder and SSRS\)](#) and [Pagination in Reporting Services \(Report Builder and SSRS\)](#).

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a page header or footer

1. Open a report.
2. On the design surface, right-click the report, point to **Insert**, and then click **Header or Footer**.

NOTE

The **Header** and **Footer** options appear only when a header or footer is not already part of the report.

To configure a page header or footer

1. On the design surface, right-click the page header or footer.
2. Point to **Insert**, and then click one of the following items to add it to the header or footer area:
 - **Textbox**
 - **Line**
 - **Rectangle**
 - **Image**
3. Right-click the page header, and then click **Header Properties** to add borders, background images, or colors, or to adjust the width of the header. Then click **OK**.
4. Right-click the page footer, and then click **Footer Properties** to add borders, background images, or colors, or to adjust the width of the footer. Then click **OK**.

To remove a page header or footer

1. Open a report.
2. On the design surface, right-click the page header or footer, and then click **Delete**.

NOTE

When you remove a page header or footer, you delete it from the report. Any items that you previously added to the page header or footer will not reappear if you subsequently add the header or footer again.

See Also

[Page Headers and Footers \(Report Builder and SSRS\)](#)

Keep Headers Visible When Scrolling Through a Report (Report Builder and SSRS)

3/24/2017 • 2 min to read • [Edit Online](#)

To prevent row and column labels from scrolling out of view after rendering a report, you can freeze the row or column headings.

How you control the rows and columns depends on whether you have a table or a matrix. If you have a table, you configure static members (row and column headings) to remain visible. If you have a matrix, you configure row and column group headers to remain visible.

If you export the report to Excel, the header will not be frozen automatically. You can freeze the pane in Excel. For more information see the **Page Headers and Footers** section of [Exporting to Microsoft Excel \(Report Builder and SSRS\)](#).

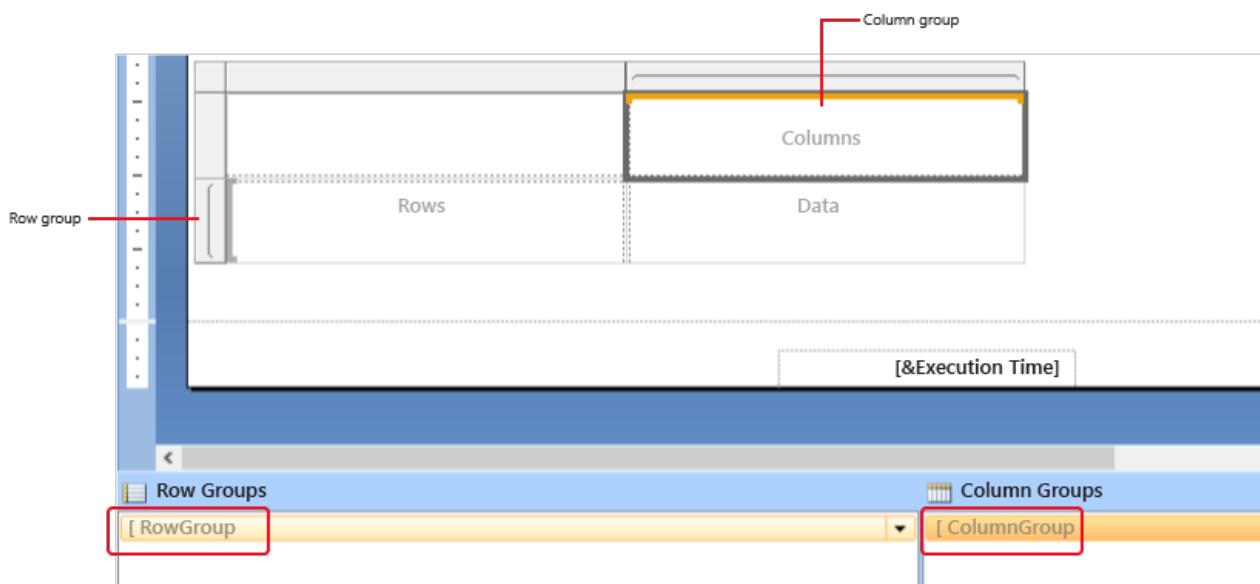
NOTE

Even if a table has row and column groups, you cannot keep those group headers visible while scrolling

The following image shows a table.

	Header	
	Data	

The following image shows a matrix.



NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To keep matrix group headers visible while scrolling

1. Right-click the row, column, or corner handle of a tablix data region, and then click **Tablix Properties**.
2. On the **General** tab, under **Row Headers** or **Column Headers**, select **Header should remain visible while scrolling**.
3. Click **OK**.

To keep a static tablix member (row or column) visible while scrolling

1. On the design surface, click anywhere in the table to display static members, as well as groups, in the grouping pane.

The screenshot shows the SSRS designer interface with a tablix data region. A specific cell in the first row and first column is selected, indicated by a blue circle with the number '1'. The grouping panes at the bottom show the hierarchy of row and column groups. The Row Groups pane lists '(Static)', '(Group1)', and '(Details)'. The Column Groups pane lists '(Static)', '(Static)', and '(Static)'. The Properties pane on the right is not visible in this screenshot.

The Row Groups pane displays the hierarchical static and dynamic members for the row groups hierarchy, and the Column groups pane shows a similar display for the column groups hierarchy.

2. On the right side of the grouping pane, click the down arrow, and then click **Advanced Mode**.
3. Click the static member (row or column) that you want to remain visible while scrolling. The Properties pane displays the **Tablix Member** properties.

The screenshot shows the SSRS designer with the grouping panes and a tablix data region. The Row Groups pane is highlighted with a red box, showing '(Static)' selected. The Properties pane on the right is open, with the 'Tablix Member' section selected and highlighted with a red box. The 'FixedData' property is expanded and set to 'True', which is also highlighted with a red box. The 'ComponentMetadata' pane at the bottom is also visible.

4. In the Properties pane, set **FixedData** to **True**.
5. Repeat this for as many adjacent members as you want to keep visible while scrolling.
6. Preview the report.

As you page down or across the report, the static tablix members remain in view.

See Also

[Tablix Data Region \(Report Builder and SSRS\)](#)

[Finding, Viewing, and Managing Reports \(Report Builder and SSRS \)](#)

[Export Reports \(Report Builder and SSRS\)](#)

[Display Headers and Footers with a Group \(Report Builder and SSRS\)](#)

[Display Row and Column Headers on Multiple Pages \(Report Builder and SSRS\)](#)

[Grouping Pane \(Report Builder\)](#)

Display Page Numbers or Other Report Properties (Report Builder and SSRS)

3/24/2017 • 1 min to read • [Edit Online](#)

It's easy to add page numbers, a report title, file name, and other report properties to the page headers or footers of your report. These properties are stored as fields in the Built-in Fields folder in the Report Data pane:

- Execution time
- Page number
- Report folder
- Report name
- Report server URL
- Total pages
- User ID
- Language

For a page number, you may want to add the word "Page" before the number. You may also want to show the total number of pages.

NOTE

Adding the total number of pages to the footer may slow performance when you run or preview your report.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To add a page number or other report properties

1. In the Report Data pane, expand the Built-in Fields folder.

NOTE

If you don't see the Report Data pane, on the View tab, check **Report Data**.

2. Drag the **Page Number** field from the Report Data pane to the report header or footer.

NOTE

The page footer is added to the report automatically. To add a page header, on the **Insert** tab, click **Header**, and then click **Add Header**.

A text box that contains the simple expression `[&PageNumber]` is added.

To add the word "Page" before the page number

1. Right-click the text box that contains [&PageNumber] and click **Expressions**.

The **Set Expression for: Value** text box contains the expression =Globals!PageNumber.

2. Place the cursor after the = sign and type "**Page** " &.

The expression is now ="Page "&Globals!PageNumber

3. Click **OK**.

To add total number of pages after the page number

1. Right click the text box with the expression and click **Expressions**.

2. Type **&" of "&** at the end of the expression.

3. In the Category pane, expand **Built-in Fields** and double-click **TotalPages**.

The expression is now ="Page "&Globals!PageNumber &" of "&Globals!TotalPages

4. Click **OK**.

See Also

[Page Headers and Footers \(Report Builder and SSRS\)](#)

[Format Text in a Text Box \(Report Builder and SSRS\)](#)

Hide a Page Header or Footer on the First or Last Page (Report Builder and SSRS)

3/29/2017 • 1 min to read • [Edit Online](#)

A report can contain a page header and page footer that run along the top and bottom of each page, respectively. After you add a header or footer, you can selectively hide it on the first and last pages of a report.

NOTE

You can create and modify paginated report definition (.rdl) files in Report Builder and in Report Designer in SQL Server Data Tools. Each authoring environment provides different ways to create, open, and save reports and related items.

To hide a page header on the first or last page

1. Open a report in Design view.
2. Right-click the page header, and then click **Header Properties**. The **Report Header Properties** dialog box opens.
3. Verify that **Display header for this report** is not selected.
4. In the **Print options** section, clear the check box for each option to hide the display on the first or last page of the report.
5. Click **OK**.

To hide a page footer on the first or last page

1. Open a report in Design view.
2. Right-click the page footer, and then click **Footer Properties**. The **Report Footer Properties** dialog box opens.
3. Verify that **Display footer for this report** is not selected.
4. In the **Print options** section, clear the check box for each option to hide the display on the first or last page of the report.
5. Click **OK**.

See Also

- [Page Headers and Footers \(Report Builder and SSRS\)](#)
- [Pagination in Reporting Services \(Report Builder and SSRS\)](#)
- [Add or Remove a Page Header or Footer \(Report Builder and SSRS\)](#)

Report Parts in Report Designer (SSRS)

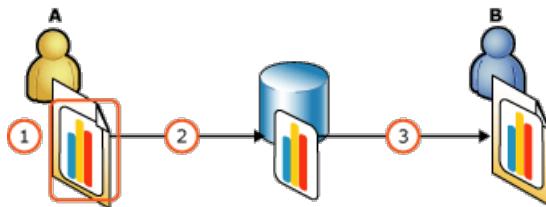
3/24/2017 • 4 min to read • [Edit Online](#)

In Report Designer, after you create tables, charts, and other paginated report items in a project, you can publish them as *report parts* to a report server or SharePoint site integrated with a report server so that you and others can reuse them in other reports.

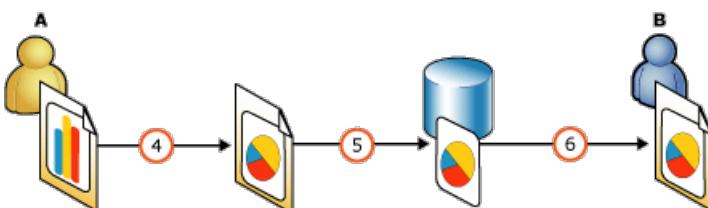
In general, report parts function the same way in Report Designer and in Report Builder. To read about basic functionality, see [Report Parts \(Report Builder and SSRS\)](#).

There are fundamental differences in the way report parts work in Report Designer. A main difference is the workflow. Report Builder enables collaborative authoring: I create a report part and publish it. You can reuse, modify, and republish it. In Report Designer, publishing is one-way: I can publish a report part from Report Designer, and you can reuse it. But I cannot reuse an existing report part in a report in Report Designer. This topic elaborates on these differences, after a quick overview of report parts.

Life Cycle of Report Part Publishing



1. In Report Designer, Person A creates a project that contains a report with a chart that depends on an embedded dataset.
2. Person A flags the chart with its embedded dataset for publishing. Report Designer assigns it a unique ID. Person A then deploys the report to the report server. Report Designer publishes the chart.
3. Person B creates a blank report in Report Builder and adds the chart to it. The chart is now part of Person B's report, along with the embedded dataset. Person B can modify the instances of the chart and dataset that are in the report. This will have no effect on the instances of the chart and dataset on the report server, nor will it break the relationship between the instances in the report and on the report server.



4. In Report Designer, Person A modifies the chart in the original report.
5. Person A redeploys the report, which republishes the chart to the server, thus updating the chart on the server.
6. In Report Builder, Person B accepts the updated chart from the server. This overwrites the changes that Person B had made to the chart in Person B's report.

Publishing Report Parts

When you publish a report part, Report Designer assigns it a unique ID. From then on, it maintains that ID, no

matter what else you change about it. The ID links the original report item in your report to the report part. When other report authors reuse the report part in Report Builder, the ID also links the report part in their report to the report part.

These are the report items you can publish as report parts:

- Charts
- Gauges
- Images and embedded images
- Maps
- Parameters
- Rectangles
- Tables
- Matrices
- Lists

If you are publishing a report part that displays data, such as a table, matrix, or chart, you can base it on a shared dataset; otherwise, when you publish the report part, the dataset that it depends on is saved as an embedded dataset. Embedded datasets can be based on embedded data sources, but credentials are not stored in embedded data sources. Thus, if your report part depends on an embedded dataset that uses an embedded data source, anyone who reuses this report part will need to provide the credentials for the embedded data source. To avoid this, base your embedded and shared datasets on shared data sources with stored credentials. For more information, see [Report Parts and Datasets in Report Builder](#).

Publishing a report part in Report Designer is a two-step process:

1. Flag the report items that you want to publish in the **Publish Report Parts** dialog box.
2. Deploy the report.

When you deploy the report, the report part is published to a SharePoint site or report server, and others can reuse it. To publish a report part, you must have a connection to and sufficient permissions on a SQL Server 2016 report server when you deploy the report.

Reusing Report Parts

Unlike in Report Builder, you cannot search for and reuse a report part in a project other than the one in which it was created.

Report authors working in Report Builder can search for and reuse report parts that you publish in reports that they create.

Republishing Report Parts

In Report Designer, you should update an existing report part from within the report in which you created it. In Report Builder, report authors can reuse the report part, and publish it as a new report part without replacing the report part that you published. If they have sufficient permissions they can also update the report part that you published. Anyone with sufficient permissions for a folder on a site or server can update the report parts stored there. The last update overwrites previous updates.

You can modify and then republish the report part to the site or server. Report Builder report authors who have added that report part to a report are informed of the change the next time they open that report. They can choose

to accept your changes or not.

You can also choose to publish as new a report that you have already published. In the Publish Report Parts dialog box, click the Publish as a new report part. This new report part has a new unique ID and no relationship to the old report part.

See Also

[Managing Report Parts](#)

Managing Report Parts

3/24/2017 • 6 min to read • [Edit Online](#)

Report parts can be reused in paginated reports, by multiple users and in multiple reports. Users can search for report parts on the server and add them to a report. Users can also be informed of updates to the report part on the server, and republish new versions of a report part. Those report authoring actions can be affected by and controlled by reporting services security permissions. This topic reviews report part properties and behavior after they are on the server.

Managing Report Parts

To manage report parts, you can use the Reporting Services web portal for a report server in native mode, or application pages for a report server in SharePoint integrated mode.

Server-side interaction and search

Report parts can be published to a report server in either native mode or SharePoint integrated mode. Users can use the report part gallery feature in a report authoring application such as Microsoft SQL Server Report Builder to find and add report parts to their reports. When a user searches for a report part, the search looks at the report server catalog regardless of which mode the server was installed for.

When report parts are published from a report authoring application such as Report Builder to a report server in SharePoint integrated mode, the report server catalog is also updated, and searches from the gallery to accurately reflect the new or update report part.

Directly uploading report parts to a SharePoint folder

If a report part is uploaded directly to a SharePoint document folder instead of being published from a report authoring application, the report server catalog is not updated. Searches from the report part gallery will not find the uploaded report part. To help keep your SharePoint folders and report server catalog synchronized, you can activate the Reporting Services file sync feature on the SharePoint server. For more information, see [Activate the Report Server File Sync Feature in SharePoint Central Administration](#).

The files can also be synchronized by calling some of the reporting services management APIs such as GetProperties and SetProperties.

Organizing and moving report parts

You should consider and plan ahead for how your team will use and organize report parts, shared datasets, and shared data sources. Although you can move them at a later time, there can be problems.

Native mode report server

If you move a report part on a native mode report server to any other folder on the same server, it does not affect the ability of report authoring applications to search for or download updates of the report part. This is because the server relies on the unique ComponentID. However, if the report part is moved to a folder that a user does not have permissions to, their searches will not find the report part, and their reports will not be notified when there are updates to the report part.

Report server in SharePoint integrated mode

Moving report parts to a different document library or folder has the same effect as uploading them directly to a SharePoint server: the report server catalog will not be synchronized. To avoid this, activate the Report Server File Sync Feature on the SharePoint server.

The exception is subfolders. The search will always search subfolders, so if you manually move a report part to a subfolder, a search from the report gallery will find the report part. It will still be found in a search from the report

part gallery.

Report server catalog properties

The following table describes how existing report server catalog fields relate to a report part, and to new fields that are added to the catalog for report parts. These are exposed in the Reporting Services web portal and SharePoint libraries, and in report authoring applications such as Report Builder.

(*) indicates it is new for this release.

PROPERTY	DESCRIPTION	REPORT PART
		GALLERY SEARCH CRITERIA
Name	This is one of the criteria a user can search for in the Report Part Gallery.	Yes
Description	You might want to organize report part names in a way that will make it easier for users to find in the gallery. For example, you can search for the description starting with "Sales>>" to find all report parts that involve sales related data and presentation.	Yes
CreatedBy	The ID of the user that added the report part to the report server database. The exact format depends on the authentication method. For example, some authentication methods result in showing the full domain\user name in the CreatedBy and ModifiedBy fields.	Yes
CreationDate	The date the report part was originally created. This is one of the criteria a user can search for in the Report Part Gallery.	Yes
ModifiedBy	ModifiedBy is the ID of the user who last modified the report part.	Yes
ModifiedDate	The date the report part was last modified on the server. This field is used as part of the logic to determine when there are server-side updates to a report part. For more information, see the description of the ComponentID later in this table.	Yes
SubType (*)	SubType is a string that indicates the kind of report part to search for, such as "Tablix" or "Chart".	Yes

PROPERTY	DESCRIPTION	REPORT PART GALLERY SEARCH CRITERIA
ComponentID (*)	<p>ComponentID is a unique identifier for the report part. This is a new field added to the catalog, and is visible on both the server-side and report authoring applications such as Report Builder.</p> <p>This field is used by client applications when checking the server for updates of a report part. The client application searches the server for ComponentIDs that are in the current client-side report. When there is a ComponentID match, the ModifiedDate is then compared to the client side SyncDate of the report item.</p>	NO

Controlling access to report parts

The following tables describe the default role assignments and how they let you perform various operations. The role assignment names are different depending on the type of report server that is being used.

Server in Native mode

ACTIONS	ROLES
Add, delete, edit item properties, manage security, and download report parts	Content Manager My Reports
Add, delete, and download report parts	Publisher
Search for and re-use	Browser Report Builder

Server in SharePoint integrated mode

ACTIONS	ROLE
Add, delete, edit item properties, manage security, and download report parts	Full Control
Add, delete, edit item properties, and download report parts	Design Contribute
Search for and re-use	Read View Only

Security considerations

- When report part definitions are re-used in a report, they are copied into the report definition in their entirety, along with the identifying ComponentID. If a report part is updated on the server, users can choose

to download the updated report parts to their report. The updates are also complete copies of the report part, replacing the existing version of the report part that was in their report.

IMPORTANT

In each of these steps, ensure that the report parts are being reused in reports from trusted locations and users.

- Report parts use the same permission policies as the existing "Resource" item type. Within a folder, there is no differentiation between traditional resource items and report parts from a security inheritance perspective. The report part will inherit the same permission policy as the images in the same folder. When this distinction is needed, item level security can be configured for the desired report parts. Or, you can put report parts should be in separate folders that have the correct permissions configured.

See Also

[Report Parts and Datasets in Report Builder](#)

[Report Server Content Management \(SSRS Native Mode\)](#)

[Troubleshoot Report Parts \(Report Builder and SSRS\)](#)

[Report Parts in Report Designer \(SSRS\)](#)