

MODULE 4: DESIGN CONSIDERATIONS

Module Overview

This module contains a more detailed analysis of some advanced report features; such as dashboard reports and green bar reports. Additionally, the module covers the different export possibilities and the rendering differences.

The lessons require that you are familiar with the techniques taught in previous modules.

Objectives

The objectives are:

- Apply green bar effects and create dashboard reports.
- Understand how a report will be rendered when it is exported to an Excel spreadsheet or a PDF file.
- Apply some helpful information that can be useful when you design reports.

Report Patterns and Recipes

This lesson describes some typical challenges that you can encounter when you develop reports. It shows how to combine the techniques from previous modules to build more advanced reports.

Green Bar Matrix

The term "green bar report" refers to a continuous paper form that is typically used to print reports. The paper form has pre-printed white and light green areas to increase its readability. A green bar report prints rows or row groups with alternating background colors.

The "Adding Code to a Report" module explains how you can define alternating background colors in table controls by setting the **BackgroundColor** property to an expression.

Code Example

```
=IIF(LineNumber("table1") Mod 2 = 0, "LimeGreen", "Transparent")
```

The expression uses the **LineNumber()** function and the **Mod** operator to calculate the correct background color.

If you want to create a green bar report by using a Matrix control, the technique illustrated in the "Adding Code to a Report" module cannot be used. This is because every row in a matrix control must be a group. There is currently no **GroupNumber()** function on which to base a green bar calculation. But a GroupNumber function can be simulated by using the RunningValue function to receive a running distinct count of group expression values.

Code example

```
=IIF(RunningValue(Fields!Country.Value,CountDistinct,Nothing) Mod 2, "AliceBlue", "White")
```

Because the matrix generates column cells dynamically, you cannot specify a different expression for each column. Additionally, some cells in the matrix can contain no data at all, and this causes the group number calculation to be wrong for the empty cells.

To work around this, you must effectively calculate the group number in the row header and then use that value inside the data cells. You can do this by using a small custom code function or by using the **RunningValue** function.

Method 1: The Custom Code Function

The first method uses a small custom code function:

Code example

```
Private bOddRow As Boolean

Public Function AlternateColor(ByVal OddColor As String, ByVal EvenColor as
String, ByVal Toggle As Boolean) As String

    If Toggle Then

        bOddRow = Not bOddRow

    End If

    If bOddRow Then

        Return OddColor

    Else

        Return EvenColor

    End If

End Function
```

The function has three parameters: a color for odd rows, a color for even rows and a Boolean toggle parameter. The toggle parameter is used to indicate the start of a new row. Typically it will be set to **True** on the first cell and set to **False** on all other cells of the matrix row. With each new value that is printed in the column (after each row group), the Toggle parameter changes and the function returns another color. Therefore, all cells in a row group will have the same color.

Method 2: The RunningValue() Function

The second method uses the **RunningValue()** function:

Code example

```
=IIF(RunningValue(Fields!Item_No_.Value,CountDistinct,Nothing) Mod 2,
"PaleGreen", "White")
```

The **RunningValue()** function returns a running aggregate of the specified expression. The function has three parameters: a group expression, an aggregate function and a scope.

The first parameter contains a group expression that you want to perform the aggregate function on. The expression cannot contain aggregate functions.

The second parameter is the aggregate function that you want to apply to the group expression. This function cannot be **RunningValue**, **RowNumber**, or **Aggregate**.

The third parameter, scope, refers to the dataset, grouping, or data region that contains the report items to apply the aggregate function to. If a dataset is specified, the running value is not reset throughout the whole dataset. If a grouping 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.

In this second method, the result of the previous expression is stored in an invisible text box named **Color**. Next, the **BackgroundColor** property of the other cells will be set to refer to the invisible text box by using the following expression:

Code example

```
=ReportItems!Color.Value
```

The following demonstrations show how to use both methods.

Demonstration: Create a Green Bar Matrix with a Custom Code Function

Ellen asks Mort to improve the readability of the report. Instead of having the item number displayed on an orange background, she asks Mort to have the inventory amounts printed with alternating background colors. She also prefers a lighter pastel background color (**PaleGreen**).

Demonstration Steps

1. Add the GreenBar effect to the report.
 - a. Open the CRONUS International Ltd. demonstration company.
 - b. Select Tools, Object Designer to open the Object Designer.
 - c. Select File, Import.
 - d. Browse to report R123456701.fob.
 - e. Click **OK** to start the import.
 - f. In the dialog box, select to open the **Import Worksheet** window.
 - g. In the Action column, verify that the action is set to Create.
 - h. Click **OK** to import the object in the database.
 - i. Open report 123456701 in Microsoft Visual Studio® Report Designer.

- j. Select **Report, Report Properties**.
- k. On the **Code** tab, add the following code if it's not already present and review the code.

Code example

```
Private bOddRow As Boolean

Public Function AlternateColor(ByVal OddColor As String, ByVal EvenColor as
String, ByVal Toggle As Boolean) As String

    If Toggle Then

        bOddRow = Not bOddRow

    End If

    If bOddRow Then

        Return OddColor

    Else

        Return EvenColor

    End If

End Function
```

- l. Click **OK** to close the **Report Properties** window.
- m. Select the cell that contains the **Item No.** and **Description** field. It's on the second row, first column.
- n. In the **Properties** window, change the **BackgroundColor** property to the following expression:
=Code.AlternateColor("PaleGreen","White", True)
- o. Select the cell that contains the Inventory amounts. This is the textbox on the second row, last column.
- p. In the **Properties** window, change the **BackgroundColor** property to the following expression:
=Code.AlternateColor("White", "PaleGreen", False)
- q. Select the cell on the first row, first column.
- r. In the **Properties** window, change the **BackgroundColor** property to **No Color**.
- s. Select the rectangle **Rectangle1** in the properties window.
- t. In the **Properties** window, change the **BackgroundColor** property to **No Color**.
- u. In the properties window, select textbox **Color**.

- v. Set the **Width** property to **0cm**.
- w. Exit Visual Studio, import the new RDLC layout in the object, save the object and run it.
- x. Click the Print Layout button in the toolbar of the Report Viewer window.

The result will resemble this:

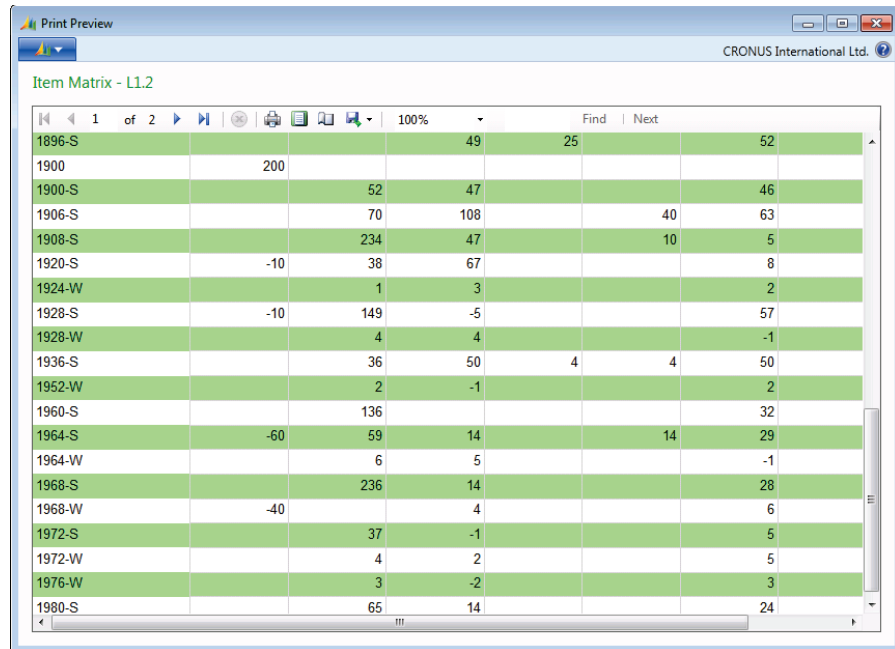
	BLUE	GREEN	OUT. LOG.	OWN LOG.	RED	WHITE	YELLOW
1896-S ATHENS Desk	49	25			52		160
1900-S PARIS Guest Chair, black	52	47			46		160
1906-S ATHENS Mobile Pedestal	70	108		40	63		
1908-S LONDON Swivel Chair, blue	234	47		10	5		
1920-S ANTWERP Conference Table	38	67			8		
1924-W CHAMONIX Base Storage Unit	1	3			2		15
1928-S AMSTERDAM Lamp	149	-5			57		97
1928-W ST MORITZ Storage Unit/Drawers	4	4			-1		41
1936-S BERLIN Guest Chair, yellow	36	50	4	4	50		
1952-W OSLO Storage Unit/Shelf	2	-1			2		
1960-S ROME Guest Chair, green	136				32		
1964-S TOKYO Guest Chair, blue	59	14		14	29		
1964-W INNSBRUCK Storage Unit/G Door	6	5			-1		8
1968-S MEXICO Swivel Chair, black	236	14			28		
1968-W GRENOBLE Whiteboard, red		4			6		10
1972-S MUNICH Swivel Chair, yellow	37	-1			5		90
1972-W SAPPORO Whiteboard, black	4	2			5		
1976-W INNSBRUCK Storage Unit/W Door	3	-2			3		3
1980-S MOSCOW Swivel Chair, red	65	14			24		
1988-S SEOUL Guest Chair, red	41	83					43

FIGURE 4.1: THE GREEN BAR MATRIX WINDOW

2. In the previous figure, Mort notices that the item number is also printed with the alternating background colors. To change this, he inserts a hidden column.
 - a. Open report 123456701 in Visual Studio Report Designer.
 - b. In the **Row Groups** window, select the option **Add Group, Child Group** on the row that contains the **Item No.** field.
 - c. In the **Tablix group** window, set the **Group by** field to the same expression as the cell that contains the **Item No.** In this case, select [Item_No_].
 - d. Click **OK** to close the window.
 - e. Select the newly added row group cell. (on the second row, second column)
 - f. In the Properties window, set the BackgroundColor property to the following expression:
=Code.AlternateColor("PaleGreen","White", True)
 - g. Select the cell that contains the Group1.
 - h. In the **Properties** window, set the **BackgroundColor** property to **No Color**.

- i. Resize the column width to a **Width of 4cm**.
- j. Right click the first Column in the Tablix and click Delete. When asked click Ok to delete the associated groups.
- k. In the textbox on the first row, first column type **Item No.**
- l. Exit Visual Studio, import the new RDLC layout in the object, save the object and run it.

The result will resemble this:



Item No.						
1896-S			49	25		52
1900	200					
1900-S		52	47			46
1906-S		70	108		40	63
1908-S		234	47		10	5
1920-S	-10	38	67			8
1924-W		1	3			2
1928-S	-10	149	-5			57
1928-W		4	4			-1
1936-S		36	50	4	4	50
1952-W		2	-1			2
1960-S		136				32
1964-S	-60	59	14		14	29
1964-W		6	5			-1
1968-S		236	14			28
1968-W	-40		4			6
1972-S		37	-1			5
1972-W		4	2			5
1976-W		3	-2			3
1980-S		65	14			24

FIGURE 4.2: THE GREEN BAR MATRIX WITH A HIDDEN COLUMN WINDOW

Demonstration: Create a Green Bar Matrix with RunningValue()

In this demonstration, you will use the RunningValue function to create a green bar matrix.

Demonstration Steps

1. To add the GreenBar effect to the report that uses the RunningValue function follow these steps.
 - a. Open the CRONUS International Ltd. demonstration company.
 - b. Select Tools, Object Designer to open the Object Designer.
 - c. Select File, Import.
 - d. Browse to report R123456701.fob.
 - e. Click **OK** to start the import.

- f. In the dialog box, select to open the **Import Worksheet** window.
- g. In the Action column, verify that the action is set to Create.
- h. Click **OK** to import the object in the database.
- i. Open report 123456701 in Visual Studio Report Designer.
- j. Select the row header cell (the cell that contains the **Item No.** field).
- k. Press **Delete** to clear the cell.
- l. Right-click the **Item_No__** text box and select Expression.
- m. In the Expression window, verify the following expression is correct:
=Fields!Item_No__.Value & Fields!Item_Description.Value
- n. In the **Properties** window, select the **Textbox Color**.
- o. In the **Properties** window, verify that the **Hidden** property of the Color text box is True.
- p. In the Properties window, verify the Value property of the Color textbox is set to the following expression:
=IIF(RunningValue(Fields!Item_No__.Value,CountDistinct,Nothing) Mod 2, "White", "PaleGreen")
- q. Select the rectangle **Rectangle1** control in the properties window.
- r. In the Properties window, verify that the BackgroundColor property is set to the following expression:
=ReportItems!Color.Value
- s. Select the Item_Ledger_Entry_Quantity matrix cell.
- t. In the Properties window, verify that the BackgroundColor property is set to the following expression:
=ReportItems!Color.Value
- u. In the textbox on the first row, first column, type: **Item No.**
- v. In the properties window, select textbox **Color**.
- w. Set the **Width** property to **0cm**.
- x. Exit Visual Studio, import the RDLC report layout, save and compile the report.

The result will resemble this:

Item No	BLUE	GREEN	OUT. LOG.	OWN LOG.	RED	WHITE	YELLOW
1896-SATHENS Desk		49	25			52	16
1900-SPARIS Guest Chair, black	52	47				46	16
1906-SATHENS Mobile Pedestal	70	108		40		63	
1908-SLONDON Swivel Chair, blue	234	47		10		5	
1920-SANTWERP Conference Table	38	67				8	
1924-WCHAMONIX Base Storage Unit	1	3				2	1
1928-SAMSTERDAM Lamp	149	-5				57	9
1928-WST.MORITZ Storage Unit/Drawers	4	4				-1	4
1936-SBERLIN Guest Chair, yellow	36	50	4	4		50	
1952-WOSLO Storage Unit/Shelf	2	-1				2	
1960-SROME Guest Chair, green	136					32	
1964-STOKYO Guest Chair, blue	59	14		14		29	
1964-WINNSBRUCK Storage Unit/G Door	6	5				-1	
1968-SMEXICO Swivel Chair, black	236	14				28	
1968-WGRENOBLE Whiteboard, red		4				6	1
1972-SMUNICH Swivel Chair, yellow	37	-1				5	9
1972-WSAPPORO Whiteboard, black	4	2				5	
1976-WINNSBRUCK Storage Unit/W Door	3	-2				3	
1980-SMOSCOW Swivel Chair, red	65	14				24	
1984-WSARAJEVO Whiteboard, blue	3	3				4	
1988-SSEOUL Guest Chair, red	41	83					4
1988-WCALGARY Whiteboard, yellow		9				5	1

FIGURE 4.3: THE GREEN BAR MATRIX WITH THE RUNNINGVALUE FUNCTION WINDOW

Dashboards

Essentially, a dashboard report is a way to visually present important data in summary form so that you can make quick and effective decisions.

With the new reporting solution, you can build your own dashboard reports by combining information from different areas in a single (report) interface. Also, features such as interactive analysis and views, charts and indicators can be added.

Frequently, a dashboard presents information from different tables by using charts, and is also filtered so that it only shows a subset of data. Instead of having the full details display on all records in different tables, only the Top X for specific areas will display. The Grouping And Sorting Properties Window figure shows a typical example of a sales dashboard.

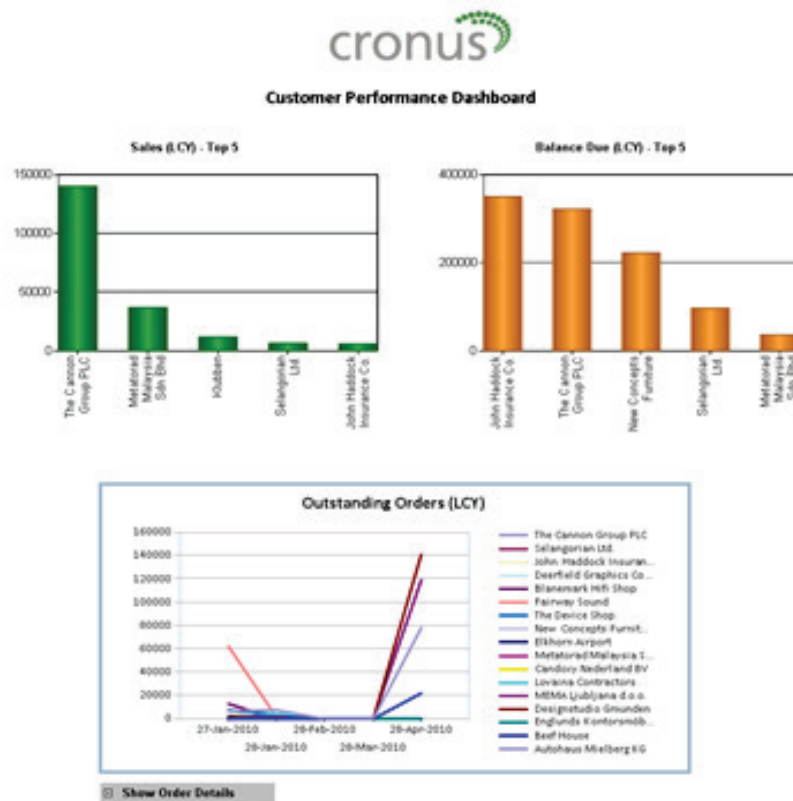


FIGURE 4.4: SALES DASHBOARD REPORT DIAGRAM

The dashboard contains two charts that give instant access to the top five (5) customers who are based on **Total Sales** and **Balance Due** in local currency (LCY). Additionally, it contains a chart with the total amount of outstanding orders in LCY. At the bottom of the report, a toggle button is used to show or hide the order details.

Dashboard Report Structure

The basic structure of a dashboard report typically includes multiple data items without any indentation.

In some cases, indented data items can be added to perform additional data processing or grouping.

The report dataset designer will also contain the necessary fields.

Dashboard Report Design

Dashboard reports frequently use charts and images to present data. You can add a chart in a report by inserting chart controls in the body section. When the chart control is added to the report, you can continue to design it. To do this, right-click the chart control and select **Properties** to open the **Chart Properties** window.

Chart Elements

The following illustration shows many of the different elements used in a chart.

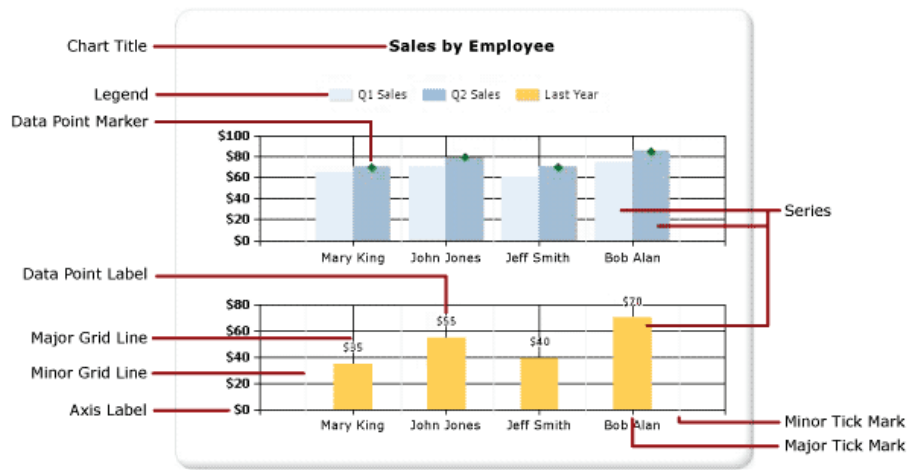


FIGURE 4.5: ELEMENTS OF A CHART DIAGRAM

A Chart data region has many properties. When you add a chart to the report body, you first select the type of chart to use.

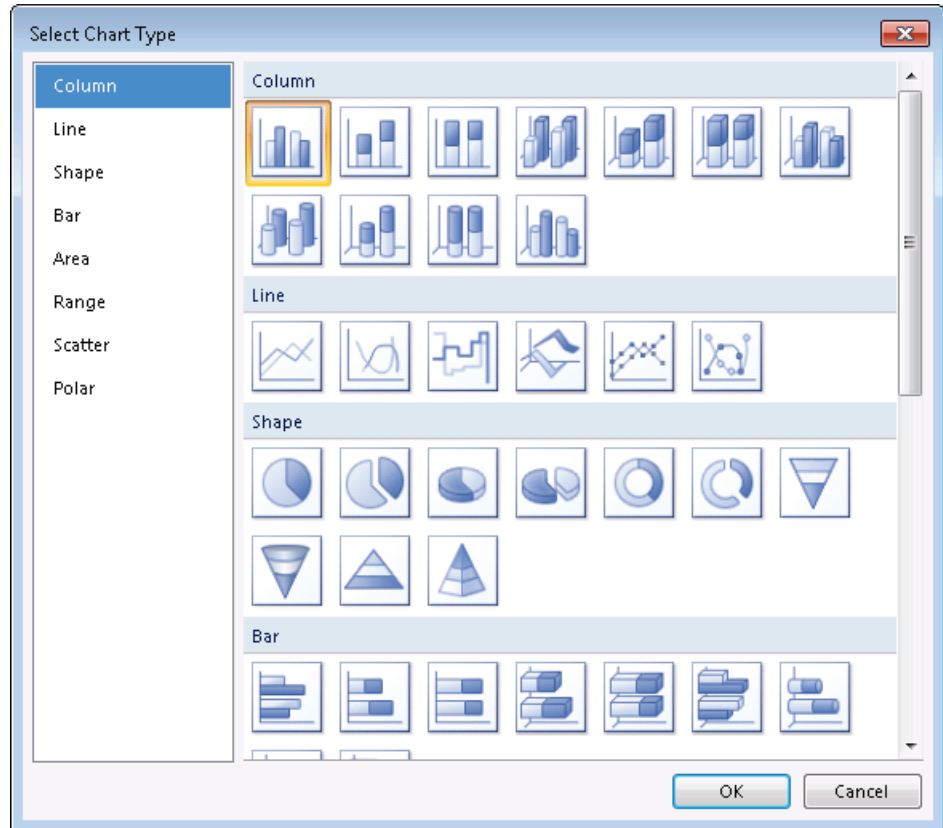



FIGURE 4.6: SELECT CHART TYPE POPUP WINDOW

Each chart type has unique characteristics to help you visualize your dataset. You can use any chart type to display your data. However, the 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.

 **Additional Reading:** For more information about chart types please read: <http://go.microsoft.com/fwlink/?LinkId=267281>

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 drop zones of the chart. When you select the chart on the design surface, three drop zones Series, Category and Data will appear. The fields from your dataset appear in the **Report Data** pane. Drag your fields from the dataset into the appropriate drop zone. By default, when a field is added to one of the field drop zones of the chart, an aggregate is calculated for the field. You can also use series grouping to dynamically generate series. The chart is also closely related to the matrix.

The chart behaves identically to the matrix template of the tablix control:

- The **Columns** field drop zone on the matrix is identical to the category group drop zone on the chart.
- The **Rows** field drop zone on the matrix is identical to the series group drop zone on the chart.
- The **Data** field drop zone on the matrix is identical to the data field drop zone on the chart.

The different elements of a chart, Area, Title, Legend, and Series have their own property list. When you select a chart, depending on where you rest the pointer, you can open the corresponding **Property** window.

For example, in the **Chart Properties** window, in the **General** tab, you can select the **Color palette**.

All properties that relate to the chart are located in the **Properties** pane. However, many of these properties can also be set from a dialog box. If you are formatting the series, you can specify series-specific properties by using custom attributes. These can only be found in the CustomAttributes category of properties, located in the **Properties** pane.

To effectively format the chart by using a minimal number of steps, you must 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. They define the drawing style of data points.

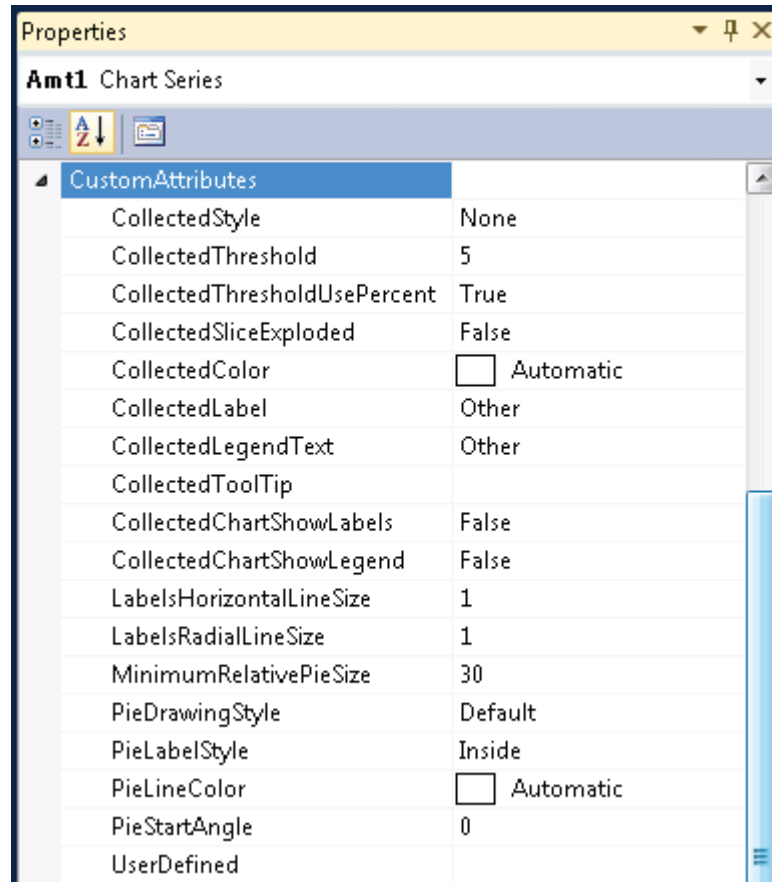


FIGURE 4.7: CHART CUSTOM ATTRIBUTES PROPERTIES

Data values, categories and series can be added to the chart by dragging fields from the dataset to the chart.

The Top X Filter

You can add a filter to a dataset, data region, or group, when you include or exclude specific values for calculations or display.

Dataset filters cannot be added in Visual Studio Report Designer, because the dataset is provided by Microsoft Dynamics NAV 2013. However, data region and group filters can be added in Visual Studio Report Designer.

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.

Setting Filters on a Data Region Control

To add a filter to a data region control, follow these steps.

1. Open a report in Visual Studio Report Designer.
2. Right-click the data region on the design surface, and then click **Properties** to open the *<data region>* **Properties** window.
3. Click the **Filters** tab. This displays the current list of filter equations. By default, the list is empty.
4. Click an empty line. 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. 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 select determines the number of values that are used from the next step.
7. In the **Value** box, type the expression or value against which you want the filter to evaluate the value in **Expression**.
8. Click **OK**.

Setting Filters on a Chart Category Group

To add a filter to a chart category group, follow these steps.

1. Open a report in Visual Studio Report Designer.
2. Right-click the data region control and select **Properties**.
3. Click the **Data** tab.
4. In the **Category group** field, select the group to filter.
5. Click the **AssistEdit (...)** button.
6. Click the **Filters** tab. This displays the current list of filter equations. By default, the list is empty.
7. Click an empty line. A new blank filter equation appears.
8. In **Expression**, type or select the expression for the field to filter. To edit the expression, click the expression **(fx)** button.
9. 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 select determines the number of values that are used from the next step.
10. In the **Value** box, type the expression or value against which you want the filter to evaluate the value in **Expression**.
11. Click **OK**.

Setting Filters on a Chart Series Group

To add a filter to a chart series group, follow these steps.

1. Open a report in Visual Studio Report Designer.
2. Right-click the data region control and select Properties.
3. Click the **Data** tab.
4. In the **Series group** field, select the group to filter.
5. Click the **AssistEdit** button.
6. Click the **Filters** tab. This displays the current list of filter equations. By default, the list is empty.
7. Click an empty line. A new blank filter equation appears.
8. In **Expression**, type or select the expression for the field to filter. To edit the expression, click the expression (**fx**) button.
9. 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 select determines the number of values that are used from the next step.
10. In the **Value** box, type the expression or value against which you want the filter to evaluate the value in **Expression**.
11. Click **OK**.

The following table shows some examples of filter equations.

Simple Expression	Data Type	Operator	Value	Description
[SUM(Quantity)]	Integer	>	7	Includes data values that are greater than seven.
[SUM(Quantity)]	Integer	TOP N	10	Includes the top 10 data values.
[SUM(Quantity)]	Integer	TOP %	20	Includes the top 20 percent of data values.
[Sales]	Text	>	=CDec(100)	Includes all values of type System.Decimal (SQL "money" data types) greater than \$100 U.S. dollars (USD).
[OrderDate]	DateTime	>	2088-01-01	Includes all dates from January 1, 2008 to the present date.

Module 4: Design Considerations

Simple Expression	Data Type	Operator	Value	Description
[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".
[Territory]	Text	LIKE	%o%th*	All territory names that include North and South at the beginning of the name.
=LEFT(Fields!Subcat.Value,1)	Text	IN	B, C, T	All subcategory values that begin with the letters B, C, or T.

Demonstration: Adding a Chart to a Report

Demonstration Steps

1. To add a chart to a report follow these steps.
 - a. Open the CRONUS International Ltd. demonstration company.
 - b. Select Tools, Object Designer to open the Object Designer.
 - c. Select File, Import.
 - d. Browse to report R123456701.fob.
 - e. Click OK to start the import.
 - f. In the dialog box, select to open the Import Worksheet window.
 - g. In the Action column, verify that the action is set to Create.
 - h. Click OK to import the object in the database.
 - i. Open report 123456701 in Visual Studio Report Designer.
 - j. In Visual Studio Report Designer on the **View** menu, click **Toolbox** to display a list of report items that can be added to the report.
 - k. Double-click the **Chart object**, or drag the Chart object to the design surface. The **Select a Chart Type** dialog box appears.
 - l. Select the type of chart that you want to add. Click **OK**.
 - m. Click the **chart** to display the drop-zones.
 - n. Add one or more fields to the data field drop-zone. This information will be plotted on the value (y) axis.

- o. On this zone usually numerical fields are used. By default they will be summarized with the **Sum** function.
- p. Add a field to the category field drop-zone by dragging it from the Dataset_Result onto the “Drop Series here” zone of the Chart. When you add this field to the series drop zone, a grouping field is automatically created for you. Each group represents a data point in your series.
- q. To summarize the data by category, drag a field from the Dataset_Result onto the “Drop Categories here” zone and then right-click the newly added field and select **Category Properties**. In the **Category** box, select the category field from the drop-down list. Click **OK**.

On charts with axes, such as bar and column charts, the category (x) axis might not display all the category labels.

Demonstration: Creating a Tablix Report

A Tablix is a flexible report item that can be used to display data in a grid format, with layout possibilities ranging from simple tables to advanced matrices. This demonstration will show how to create a tablix with adjacent groups, totals and subtotals.

Demonstration Steps

1. Open the Microsoft Dynamics NAV 2013 demonstration database.
 - a. Open the CRONUS International Ltd. demonstration company.
 - b. Select Tools, Object Designer to open the Object Designer.
 - c. Select File, Import.
 - d. Browse to report R123456708.fob.
 - e. Click **OK** to start the import.
 - f. In the dialog box, select to open the **Import Worksheet** window.
 - g. In the Action column, verify that the action is set to Create.
 - h. Click **OK** to import the object in the database.
2. Open report 123456708 in Visual Studio Report Designer.
 - a. Open the Toolbox and add a table to the report body.
 - b. In the detail row add the following field:
Amount_CustLedgerEntry.
 - c. Select the second and third columns by using the corresponding column handle and then click **Delete**.
 - d. Select the first text box and set the **FontWeight** property to **Bold**.
 - e. Select the tablix.

Module 4: Design Considerations

- f. In the **Row Groups** window, click the **Details Row** and select **Add Group, Parent Group**.
 - g. In the **Group by** field, select [**SalesPersonName**].
 - h. Click **OK** to close the **Tablix group** window.
 - i. For the first row, delete the contents of the first text box.
3. Add a group on [**CustomerNo_CustLedgerEntry**].
 - a. Select the tablix.
 - b. In the **Row Groups** window, click the **Details Row** and select **Add Group, Parent Group**.
 - c. In the **Group by** field, select [**CustomerNo_CustLedgerEntry**].
 - d. Click **OK** to close the **Tablix group** window.
 - e. For the first row, delete the contents of the second text box.
 4. Create an expression for Salesperson.
 - a. Right-click the first text box for the second row and select **Expression**.
 - b. In the Expression text box, enter the following expression:

```
=Fields!SalesPersonName.Value & " (" &  
Fields!SalespersonCode_CustLedgerEntry.Value & ")"
```

5. Create an expression for CustomerName.
 - a. Right-click the second text box of the second row and select **Expression**.
 - b. In the Expression text box, enter the following expression:

```
=Fields!CustomerNo_CustLedgerEntry.Value & " " &  
Fields!CustomerName.Value
```

6. Set the width of the first two text boxes.
 - a. Set the **Width** property of the first text box to **3.4925cm**.
 - b. Set the **Width** property of the second text box to **6.37647cm**.
7. Create a group on the Year of the posting date.
 - a. From the **Report Data** window, select the field **PostingDate_CustLedgerEntry** and drag it into the **Column Groups** window.
 - b. Open the **Group Properties** of the newly added column group.
 - c. In the **Group Expression**, change the **Group On** expression to the following:

```
=Year(Fields!PostingDate_CustLedgerEntry.Value)
```

- d. Right-click the third text box for the first row and select **Expression**.
- e. In the **Expression** text box enter the following expression:

```
=Year(Fields!PostingDate_CustLedgerEntry.Value)
```

8. Create a group on Document Type.
 - a. In the **Column Groups** window, click **Add Group, Adjacent After**.
 - b. In the **Tablix group** window, set the **Group by** field to the following: [**DocumentType_CustLedgerEntry**].
 - c. Click **OK** to close the **Tablix Groups** window.
 - d. For the third row, in the fourth text box select: [**Amount_CustLedgerEntry**]
9. Create a header row for column titles.
 - a. Right-click the row handle for the second row (the middle row) and select: **Delete Rows**.
 - b. Right-click the row handle for the first row and select: **Insert Row, Outside Group, Above**.
 - c. In the third text box of the newly added first row type: **Year**.
 - d. In the fourth text box of the newly added first row type: **Type**.
 - e. Set the property **TextAlign** of the text boxes in the first row to **Center**.
10. Create a total for Amount.
 - a. In the **Column Groups** window, select group 3 and click **Add Group, Adjacent After**.
 - b. In the **Tablix group** window, set the **Group by** field to **ALL**. To do this type ALL in the Group by field.
 - c. As an alternative, select the **Fx** button. The Expression editor opens. Type in the following expression:

```
="ALL"
```

- d. Click **OK** to close the **Column Groups** window.
- e. In the last text box of the first row, right-click and select: **Split Cells**.
- f. In the fifth text box of the first row type: **Grand Total**.
- g. Select the fifth text box of the second row and then click **Delete**.
- h. In the fifth text box of the third row, enter the expression:

```
=Sum(Fields!Amount_CustLedgerEntry.Value)
```

11. Create a group total row.
 - a. In the **Row Groups** window, on the first line, select **Add Total, After**.
 - b. In the newly added row, in the first text box, change the value to **Grand Total**.
 - c. Select the row handle for the last row in the tablix and then click the **B** button in the **Report Formatting toolbar**.
 - d. In the **Value** Expression of the third and fourth text box, on the last row, type the following expression:

```
=Sum(Fields!Amount_CustLedgerEntry.Value)
```

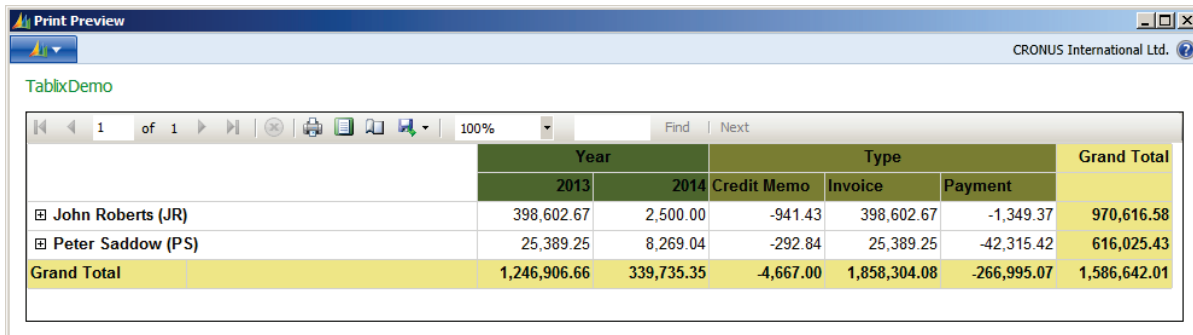
- e. On the second row, delete the contents of the last text box.
12. Set the **BackColor**.
 - a. Set the **BackColor** property of the text boxes of the last row and the last column to **Khaki**.
 - b. Set the **BackColor** property of the text boxes of the third column, first and second row to **DarkOliveGreen**.
 - c. Set the **BackColor** property of the text boxes of the fourth column, first and second row to **Olive**.
 - d. Set the **Format** property of the six text boxes that contain Amounts to the following expression:

```
Fields!Amount_CustLedgerEntryFormat.Value
```

13. Change the visibility settings for Group3.
 - a. In the **Row Groups** window, open the properties of **Group 2**.
 - b. In the **Visibility** tab, set the option **When the report is initially run** to **Hide**.
 - c. Set the **Display can be toggled by this report item** to **Group1**.
 - d. Click **OK** to close the **Group Properties** window.

14. Save and Run the report.
 - a. Exit Visual Studio, import the RDLC report layout, and then save and compile the report.

The result will resemble this:



The screenshot shows a 'Print Preview' window for a report named 'TablixDemo'. The report is a matrix with columns for 'Year' (2013, 2014), 'Type' (Credit Memo, Invoice, Payment), and 'Grand Total'. The rows represent individual customers: John Roberts (JR) and Peter Sadow (PS). The Grand Total row is highlighted in yellow.

	Year		Type			Grand Total
	2013	2014	Credit Memo	Invoice	Payment	
John Roberts (JR)	398,602.67	2,500.00	-941.43	398,602.67	-1,349.37	970,616.58
Peter Sadow (PS)	25,389.25	8,269.04	-292.84	25,389.25	-42,315.42	616,025.43
Grand Total	1,246,906.66	339,735.35	-4,667.00	1,858,304.08	-266,995.07	1,586,642.01

FIGURE 4.8: MATRIX REPORT WITH ADJACENT GROUPS AND TOTALS SCREEN SHOT

Demonstration: Simulating Transheader and Transfooter

In earlier versions, you could use a transheader and transfooter in reports. Transfooters show the current subtotals through the end of the current page or the sum for each page.

These sections are not available in RDLC reports in Microsoft Dynamics NAV 2013. However, this demonstration will show how to achieve the same functionality in RDLC reports.

Demonstration Steps

1. Import the demonstration report.
 - a. Open the Microsoft Dynamics NAV 2013 demonstration database.
 - b. Open the CRONUS International Ltd. demonstration company.
 - c. Select **Tools**, and **Object Designer** to open the Object Designer.
 - d. Select File, Import.
 - e. Browse to report R123456707.fob.
 - f. Click **OK** to start the import.
 - g. In the dialog box, select to open the **Import Worksheet** window.
 - h. In the Action column, verify that the action is set to Create.
 - i. Click **OK** to import the object in the database.
 - j. Open report 123456707 in Visual Studio Report Designer.

Module 4: Design Considerations

2. Add the transheader and transfooter to the report.
 - a. Open the Toolbox and add a table to the report body.
 - b. In the detail row, add the following fields: **Customer No, Customer Name, Debit Amount**.
 - c. Add a total row to the table by selecting the Details in the **Row Groups** section and then by clicking the option **Add Total, After**.
 - d. In the total row, in the text box on the third column enter the following expression:

```
= "Grand total: " & Sum(Fields!DebitAmount_Customer.Value)
```

- e. Add a Page Header to the report.
- f. Add a Page Footer to the report.
- g. In the Page Header add a text box from the toolbox.
- h. Move this textbox to the bottom of the Page Header and make it the same Width as the Width of the first column in the table in the body of the report.
- i. In the Page Footer add a text box from the toolbox.
- j. Move this textbox to the top of the Page Footer and make it the same Width as the Width of the first column in the table in the body of the report.
- k. In the text box of the Page Header enter the following expression:

```
= "Transheader subtotal = " &  
Code.GetRunningTotal(Globals!PageNumber-1)
```

- l. In the text box of the Page Footer enter the following expression:

```
= "Transfooter subtotal = " & Code.SetRunningTotal(  
Sum(ReportItems!DebitAmount_Customer.Value),  
Globals!PageNumber)
```

- m. In the **Hidden** property of the text box in the Page Header enter the following expression:

```
= IIF(Globals!PageNumber > 1, False, True)
```

- n. In the Hidden property of the text box in the Page Footer enter the following expression:

```
= IIF(Globals!PageNumber < Globals!TotalPages, False, True)
```

- o. Set the **BackColor** of the text box in the Page Header to **DarkOliveGreen**.
- p. Set the **Color** of the text box in the Page Header to **White**.

- q. Set the **BackColor** of the text box in the Page Footer to **SteelBlue**.
 - r. Set the **Color** of the text box in the Page Footer to **White**.
 - s. Set the **Color** of the text box in the Page Header to **White**.
3. Save and Run the report.
 - a. Exit Visual Studio, import the RDL report layout, and then save and compile the report.
 - b. Run the report.
 - c. Select the **Print Layout** button in the Report Viewer window on the toolbar.
 - d. Use the < and > buttons in the report viewer toolbar to view the different pages.

The result will resemble this:

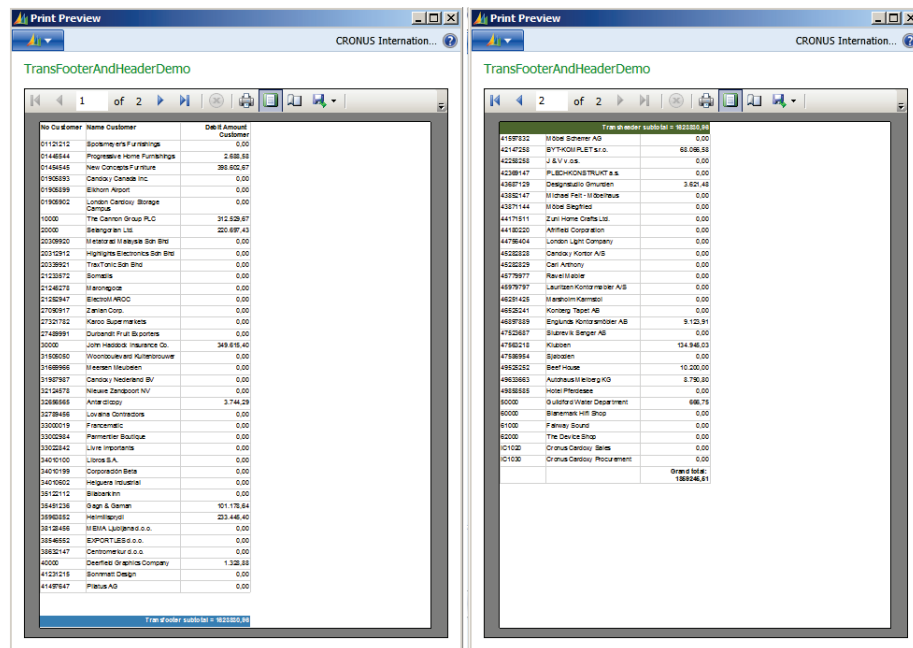


FIGURE 4.9: RUNNING VALUE MIMIC TRANSHEADER AND TRANSFOOTER SECTIONS WINDOW

Report Rendering Considerations

Report Viewer provides interactive reporting features that you can use when you work with a report at run time. However, not all the report export formats support the full range of interactive features.

Lesson Objectives

Examine how reported layouts will be rendered when you export to Excel, PDF and Word.

Excel Rendering

Report Viewer supports the rendering of reports for Microsoft Office Excel. The Excel rendering extension includes some unique attributes. These include the following:

- Each page in the report becomes an Excel worksheet. Excel does not support the concept of page height and width. Therefore, only explicitly defined page breaks will occur.
- Specifying worksheet names is not supported.
- The rendering extension builds a tabular structure out of the report.
- Excel does not support background images for individual cells.
- Excel does not support nested containers other than lists.
- Charts are rendered as pictures, not as Excel charts.
- Rectangles are converted to groups of cells. If rectangles contain other items, the rectangle becomes a region of cells, and the border and background color of the rectangle are applied to the region of cells.
- Documents created by the Excel renderer are not in the Office 2007 XLSX file format. However, Excel 2007 can use the created document because Excel 2007 provides full support for the XLS format.

Using Tables Instead of Lists for Excel-specific Reports

A table uses a fixed column width. This matches well with the tabular format of Excel reports. The items in the report table will line up as they are expected to when they are rendered in Excel.

In contrast, a list is a freeform style. Items in the list are positioned in the worksheet in relation to their location in the report. This could lead to unexpected results. If your report uses a list, you must check the rendering to Excel to determine whether the results are acceptable.

If you are using tables, and you have a header that spans multiple columns in a report, the Excel rendering extension might have to merge cells or introduce new columns. This can affect the ability to sort and manipulate data in the Excel spreadsheet. If you plan to render to Excel, make sure that the left or right edges of the report item line up to minimize cell merging.

Maximum Number of Pages in Long Reports

To prevent Excel from generating an error, keep track of the number of pages in lengthy reports. Specifically, each page in a report becomes a worksheet in Excel. However, Excel can only support a maximum number of worksheets for each workbook, limited by available memory. If the report pages exceed that limit, Excel generates an error.

Color Differences in Rendering to Excel

Excel supports a predefined set of colors. When you render a report, the Excel rendering extension maps the report colors to the best match in the natively supported colors in Excel.

PDF Rendering

The portable document format (PDF) rendering extension creates reports that can be viewed by using Adobe Acrobat readers.

The rendering extension does not require Adobe software to render the report. However, PDF viewers such as Adobe Acrobat are required for viewing or printing PDF reports. When you design PDF reports output, consider the following:

- Document maps are rendered as PDF bookmarks. In the document maps, the hierarchy is ignored. (All bookmarks will be on the same level.)
- Specify page width, page height and margins of the PDF, if this is necessary.



Additional Reading: A good explanation of a report with a custom barcode font can be found here: <http://go.microsoft.com/fwlink/?LinkId=267282>

Word Rendering

Reports exported to Microsoft Office Word are displayed as a nested table that represents the report body. A Tablix data region is rendered as a nested table that reflects the structure of the data region in the report. Text boxes and rectangles are each rendered as a cell within the table. The text box value is displayed inside the cell.

Images, charts, and gauges are each rendered as a static image in a table cell. Hyperlinks and drillthrough links on these report items are rendered. Maps and areas that can be clicked in a chart are not supported.

Newsletter-style column reports are not rendered in Word. Report body and page background images and colors are not rendered.

Word limitations

The following limitations are applied by Microsoft Word:

- Word tables support no more than 63 columns. If your report has more than 63 columns and you try to render it, Word splits the table. The additional columns are positioned next to the 63 columns that are displayed in the report body. Therefore, the report columns might not line up as expected.
- Word supports a maximum page width of 22 inches wide and 22 inches high. If your content is wider than 22 inches, some data might not be displayed in Print Layout view.
- Word ignores page header and footer height settings.
- Documents created by the Word renderer are not in the Microsoft Office 2007 DOCX file format. However, Microsoft Office Word 2007 can use the created document because Word 2007 provides full support for the DOC format.
- Reports can be viewed in Microsoft Word 1997 but the layout will not appear correctly. Word 97 does not support nested tables, 24-bit colors, cell padding and possibly other features that are used by the Word renderer.
- After the report is exported, Word paginates the report again. This can cause additional page breaks to be added to the rendered report.
- Text boxes grow when they contain nonbreaking spaces.
- When text is exported to Word, text with font decoration in certain fonts can generate unexpected or missing glyphs in the rendered report.

SAVEASEXCEL, SAVEASPDF and SAVEASWORD

Microsoft Dynamics NAV 2013 expands the C/AL functions SAVEASEXCEL and SAVEASPDF that were introduced in the previous version with a new function: SAVEASWORD.

SAVEASEXCEL

You can use the **SAVEASEXCEL** function on the Report type and on Report variables to save a report as a Microsoft Office Excel Worksheet. It has the following syntax:

Code Example

```
[Ok :=] REPORT.SAVEASEXCEL(Number, FileName, [, Record])  
[Ok :=] REPORT.SAVEASEXCEL(FileName)
```

When SAVEASEXCEL is called, the report is generated and saved to "FileName". The **Saving to Excel** window shows the status of the process. If you click **Cancel**, the report will not be generated or saved as an Excel Worksheet. Also, the Request Page will not be shown.

The return value informs you whether the report is saved successfully. If the report cannot be saved for a specific reason, the function will return False.

SAVEASPDF

You can use the **SAVEASPDF** function to save a report as a PDF file. It has the same syntax as the SAVEASEXCEL function. No PDF software is required to create PDF files. It has the following syntax:

Code Example

Code Example

```
[Ok :=] REPORT.SAVEASPDF(Number, FileName[, Record])  
[Ok :=] REPORT.SAVEASPDF(FileName)
```

You can use the **SAVEASPDF** function on the global REPORT object or on Report variables. If, at design time, you do not know the specific report that you want to run, then use the global REPORT object and specify the report number in the Number parameter. If you do know which report that you want to run, then create a Report variable, set the Subtype of the variable to a specific report, and use this variable when you call the SAVEASPDF function.

When you call SaveAsPDF, the report is generated and saved to "FileName". The **Saving to PDF** window shows the status of the process. Also, the request page will not be shown.

The function can also be used in codeunits that are exposed as web services. Other applications can call the web services to have reports generated.

SAVEASWORD

You can use the **SAVEASWORD** function on the Report type and on Report variables to save a report as a Microsoft Office Word file. It has the following syntax:

Code example

```
[Ok :=] REPORT. SAVEASWORD (Number, FileName, [, Record])
```

```
[Ok :=] REPORT. SAVEASWORD (FileName)
```

When **SAVEASWORD** is called, the report is generated and saved to "FileName". The **Saving to Word** window shows the status of the process. If you click **Cancel**, the report will not be generated or saved as a Word file. Also, the Request Page will not be shown.

The return value informs you whether the report is saved successfully. If the report cannot be saved for a specific reason, the function will return **False**.

Pagination

Pagination refers to the number of pages in a report and how report items are arranged on these pages. Pagination in reports varies, depending on the rendering extension that you use to view and deliver the report.

Understanding Pagination

When you run a report in the report viewer, 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 may paginate differently. To successfully design an easy-to-read report that is better for the renderer, then how you plan to deliver the report, you must understand the rules that are used to control pagination. This topic will explain the effects of the physical page size and the report layout on how hard page break renderers must render the report.

The Report Body

The report body is a rectangular container displayed as white space on the design surface. It can grow or decrease to accommodate the report items that are contained in 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 and HTML, render reports that grow or decrease, depending on the contents of the page. Reports rendered in these formats are better for screen-based viewing, such as in a web browser. These renderers add vertical page breaks when they are 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.



Note: *If you have designed a report to be one page wide, and it renders across multiple pages, check that the width of the report body (and this includes 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.*

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. This is based on the physical page size and provides a better reading experience when the reports are printed or viewed in a hard page break file format. Reports rendered in soft page break formats insert page breaks horizontally. This is based on the physical size and provides a better reading experience when the reports are 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, Page Setup** dialog box or by changing the **PageHeight** and **PageWidth** properties in the **Properties** pane.

The page size does not grow or decrease 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 side 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.

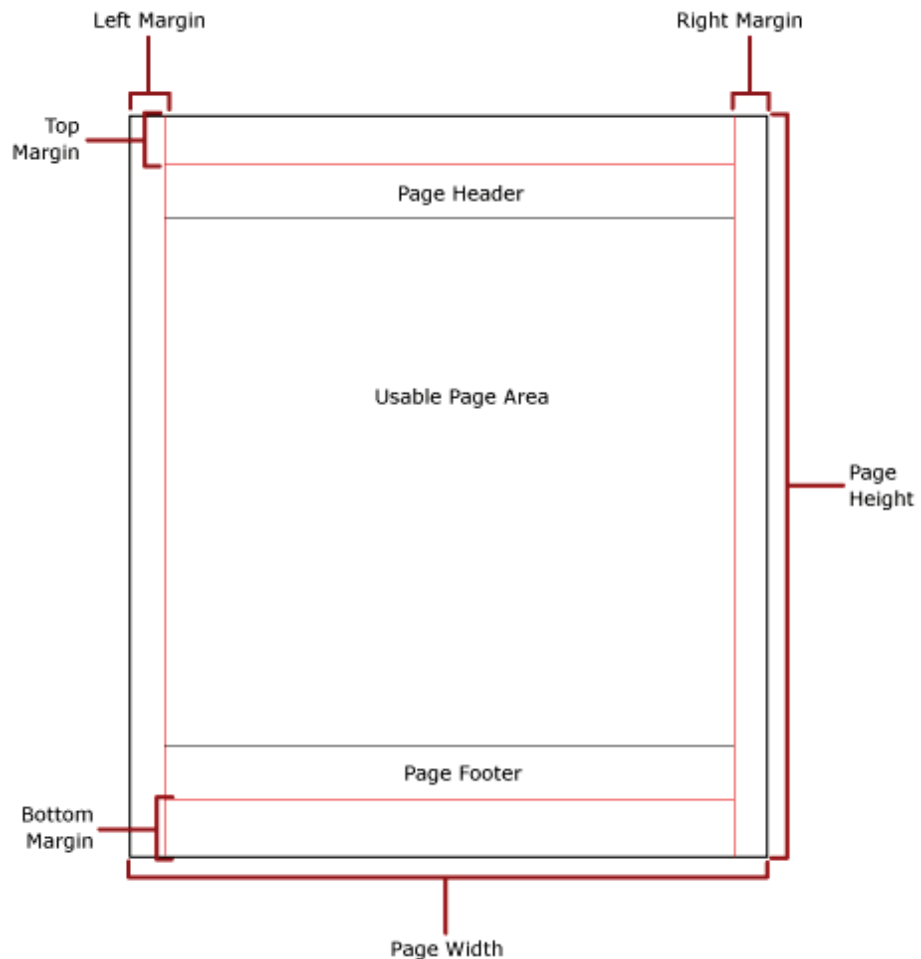
Margins


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 will default to zero. Margins are specified by using the **Report Properties, 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 by using the Device Information settings for the specific renderer that you are using to export the report.

Module 4: Design Considerations

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 Source Image File Instructions figure indicates the margin and usable page area of a physical page.



 **Note:** Note that once page the page header and page footer have been set for the document the space defined for them is ALWAYS allocated and never released to accommodate more rows from the “usable page area”. This can have an effect on the development of document reports were it can result in blank pages.

Rendering in Excel

This renderer supports only logical page breaks. A new Excel worksheet tab is created for each explicitly defined logical page break.



Reader Aid: *A soft (or logical) page break is a place in a document or text that automatically separates pages. Soft page breaks are inserted automatically by word processing programs such as Microsoft Word and Excel, as opposed to hard page breaks that are inserted manually by the user to separate pages.*

Page Naming

If there is only one worksheet in the workbook, excluding the Document Map, the name of the worksheet is the report name.

If there are multiple worksheets in the workbook, excluding the Document Map, the worksheets are named SheetX, where X is the number of the sheet in the workbook, for example, the fifth worksheet is then called Sheet5.

Page Sizing

The Excel rendering extension uses the page height and width settings to determine what paper setting to define in the Excel worksheet. Excel tries to match the PageHeight and PageWidth property settings to one of the most common paper sizes.

If no matches are found, Excel uses the default page size for the printer. Orientation is set to Portrait if the page width is less than the page height; otherwise, orientation is set to Landscape.

Rendering in Word

When the report is opened in Word, the whole report repaginates again based on the page size. Repagination can cause page breaks to be inserted in locations where you did not intend to add them. Additionally, in some instances, repagination can cause the exported report to have two successive page breaks in a row or add blank pages. You can try to change Word's pagination by adjusting the page margins.

This renderer supports only logical page breaks.

Page Sizing

When the report is rendered, the Word page height and width are set by the following RDL properties: paper size height and width, left and right page margins, and the top and bottom page margins.

Page Width

Word supports page widths that are up to 22 inches wide. If the report is wider than 22 inches, the renderer will still render the report. However, Word will not display the report contents while in print layout view or reading layout view. To view the data, switch to normal view or web layout view. In these views, Word reduces the white space. So, more of your report content is displayed.

When it is rendered, the report grows as wide as required, up to 22 inches, to display the contents. The minimum width of the report is based on the RDL Width property in the **Properties** pane.

Useful Information

This lesson describes several techniques that can be useful to create reports. It assumes that you are familiar with the content from the previous modules.

CanGrow and CanShrink

When you design reports, it can be a challenging when you design the column size. This is especially true if there are multiple columns and all the information must be available on one line.

The width and height of a text box can be defined at design time by setting the **Width** and **Height** properties. Because both properties are not available in run-time, you cannot use a complex expression to make a text box fit the widest or longest value. The width must be defined by using a single expression in design-time and it will be respected in run-time.

However, this does not apply to the height of a column. Although the **Height** property cannot be bound to a complex expression either, you can control the height of a text box using the **CanGrow** and **CanShrink** properties.

The **CanGrow** and **CanShrink** properties indicate whether the size of a text box can increase or decrease *vertically* according to its content. If both properties are set to **False**, the text box will not grow. The static size of the text box as defined in Visual Studio Report Designer will be used, regardless of the length of its value. If you set **CanGrow** to **True**, the text box will expand vertically if the text box is not wide enough to show the value.

For the text box to decrease based on its contents, you can change the **CanShrink** property for the text box.

If you set **CanGrow** to **True** for one text box in a table row, all other cells on the same row might expand vertically too (even if **CanGrow** for these text boxes is set to **False**).

However, **CanGrow** and **CanShrink** behave differently for each rendering extension.

Demonstration: Using CanGrow and CanShrink in Different Rendering Extensions

For a text box, the CanGrow property indicates whether the size of the text box can increase vertically according to its content. For a text box, the CanShrink property decreases the height of the text box according to its content.

You can only dynamically control the height of the report elements through these properties because the Height property cannot be bound to an expression. So, these two properties are useful to create the reports that can be scaled to both small and large contents.

In this demonstration text boxes are added to the body of a report that uses different colors and values for Cangrow and Canshrink. The FontSize of each text box is linked to a variable that can be changed through the Request Page. The following step procedure shows the Cangrow and CanShrink behavior by using different values for the FontSize parameter.

Demonstration Steps

1. Create a new report and add the following text boxes to the body of the report.
 - a. Open the CRONUS International Ltd. demonstration company.
 - b. Select Tools, Object Designer to open the Object Designer.
 - c. Select File, Import.
 - d. Browse to report R123456702.fob.
 - e. Click **OK** to start the import.
 - f. In the dialog box, select to open the **Import Worksheet** window.
 - g. In the Action column, verify that the action is set to Create.
 - h. Click **OK** to import the object in the database.
 - i. In the Object Designer, select the imported report and click Design.
 - j. Select View, Layout to open Visual Studio Report Designer.
 - k. Add a text box that uses a Value of "CanShrink" and **CanShrink** is set to **TRUE**.
 - l. Add a text box that uses a Value of "CanGrow" and **CanGrow** is set to **TRUE**.
 - m. Add a text box that uses a Value of "CanShrink and CanGrow" and **CanShrink** and **CanGrow** are both set to **TRUE**.
 - n. Add a text box that uses an empty value and **CanShrink** is set to **TRUE**.

- o. Add a text box that uses a single space as value (Value = " ") and **CanShrink** is set to TRUE.
 - p. Add a non-visible text box (Visibility.**Hidden** = **TRUE**).
 - q. Add a non-visible text box where **CanShrink** is set to TRUE.
 2. Set the **BackColor** for each text box to a different color.
 - a. Set the **BackColor** for the first text box to **Red**.
 - b. Set the **BackColor** for the second text box to **Orange**.
 - c. Set the **BackColor** for the third text box to **LimeGreen**.
 - d. Set the **BackColor** for the fourth text box to **SeaGreen**.
 - e. Set the **BackColor** for the fifth text box to **Aqua**.
 - f. Set the **BackColor** for the sixth text box to **CornflowerBlue**.
 - g. Set the **BackColor** for the seventh text box to **DarkViolet**.
 - h. Exit Visual Studio, import the RDLC report layout, and then save and compile the report.
 3. Add a variable to the Request Page.
 - a. In the **C/AL Globals** menu, there is a variable with the name **FontSize** of **Data Type Option**.
 - b. In the **OptionString** the value is 6pt,10pt,16pt,30pt.
 - c. Open the Request Page.
 - d. Add the variable **FontSize** to the request page.
 - e. Type **Request Page** in the **Caption** property of the first line.
 - f. Then add a second line of type **Group**, subtype **Group** and **Caption Options**.
 - g. Add a third line with Type Field, SourceExpr FontSize and Caption FontSize.
 - h. Close the Request Page Designer.
 - i. In the **Report Dataset Designer**, select the **Integer** data item.
 - j. In the properties window verify that the property **PrintOnlyIfDetail** is set to **No** and if not, set it to No.
 - k. In the Report Dataset Designer, verify that there's a column that contains the FontSize field. If not then, below the integer data item, add a new line of type **Column** and set the **Data Source** and **Name** to **FontSize**.

4. In Visual Studio Report Designer, add the variable `FontSize` into the `FontSize` property of each text box.
 - a. Select View Layout to open Visual Studio Report Designer.
 - b. In the first text box set the property **FontSize** to `=First(Fields!FontSize.Value, "DataSet_Result")`
 - c. Repeat this step for the other text boxes (2 to 7).
5. Add a new text box above the existing text boxes to display the selected `FontSize`.
 - a. In the report body, select the text boxes and then move them down.
 - b. Add a new text box.
 - c. In the **Value** property use the following expression:

```
= "FontSize=" & First(Fields!FontSize.Value, "DataSet_Result")
```
6. Run the report for `FontSize 6pt`.
 - a. Run the report.
 - b. Select `FontSize 6pt`.
 - c. The result will resemble this:

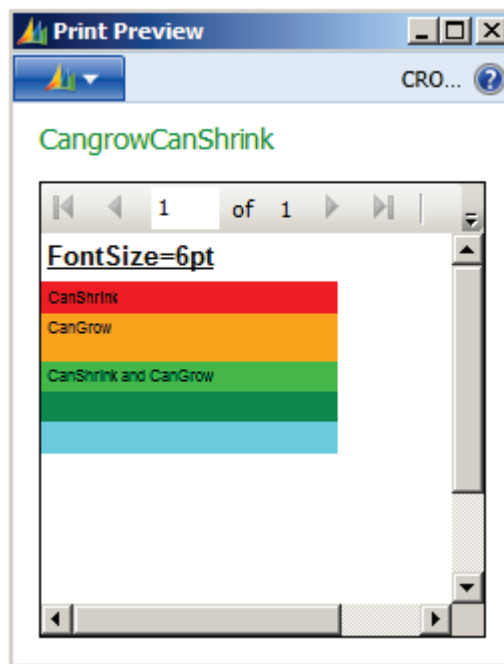


FIGURE 4.11: FONTSIZE 6PT WINDOW

- d. Close the report.

7. Run the report for FontSize 10pt.
 - a. Run the report.
 - b. Select FontSize 10pt.
 - c. The result will resemble this:

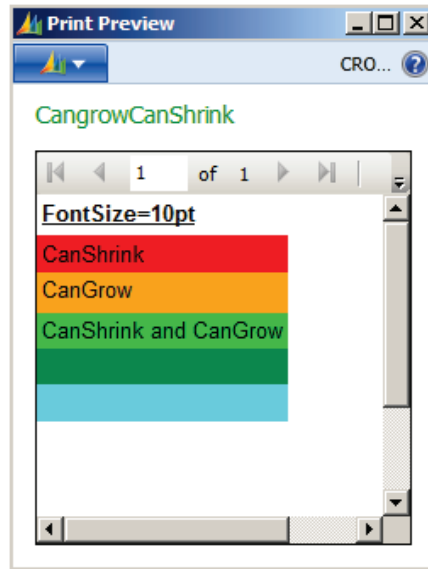


FIGURE 4.12: FONTSIZE 10PT WINDOW

- d. Close the report.
8. Run the report for FontSize 16pt.
 - a. Run the report.
 - b. Select FontSize 16pt.
 - c. The result will resemble this:

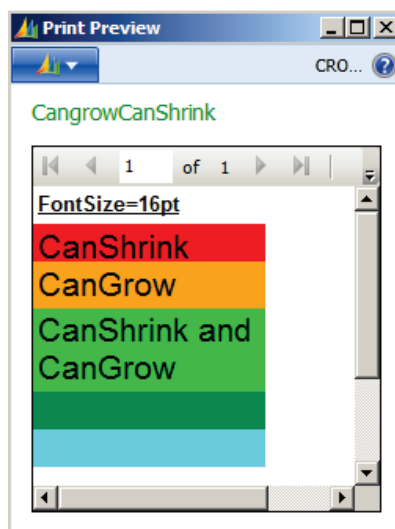


FIGURE 4.13: FONTSIZE 16PT WINDOW

- d. Close the report.

9. Run the report for FontSize 30pt.
 - a. Run the report.
 - b. Select FontSize 30pt.
 - c. The result will resemble this:

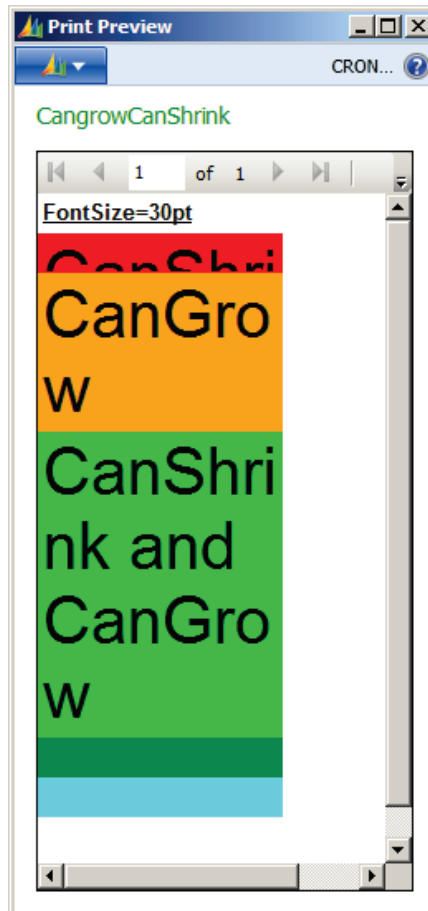


FIGURE 4.14: FONTSIZE 30PT WINDOW

- d. Close the report.

Conclusion:

- Depending on the settings of **CanGrow** and **CanShrink** the textboxes will have a different size at runtime.
- When you increase or decrease the **Width** property of the textboxes results may vary from the presented screenshots.

10. Run the report for FontSize 6pt and export it to PDF.
 - a. The result will resemble this:

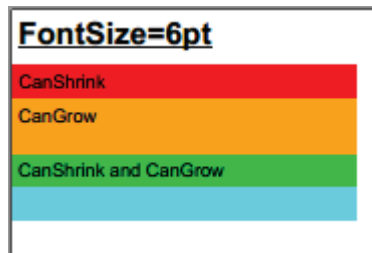


FIGURE 4.15: FONTSIZE 6PT PDF WINDOW

11. Run the report for FontSize 10pt and export it to PDF.
 - a. The result will resemble this:

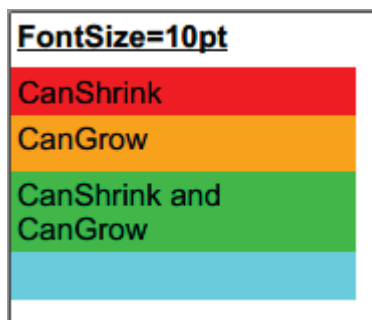


FIGURE 4.16: FONTSIZE 10PT PDF WINDOW

12. Run the report for FontSize 16pt and export it to PDF.
 - a. The result will resemble this:

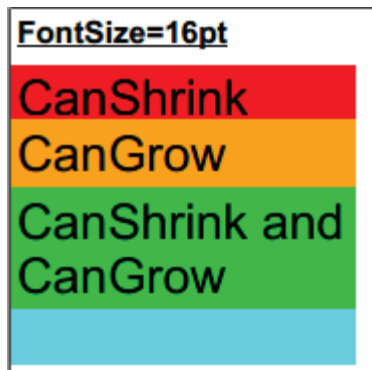


FIGURE 4.17: FONTSIZE 16PT PDF WINDOW

13. Run the report for FontSize 30pt and export it to PDF.
 - a. The result will resemble this:



FIGURE 4.18: FONTSIZE 30PT PDF WINDOW

Conclusion:

- Similarly to the **HTML** rendering, the **CanGrow** and **CanShrink** modify the heights of the text boxes.
- Depending on the **Width** of the textboxes and the selected **FontSize**, empty space might appear between the textboxes. This also depends on the amount of space there is in between the different textboxes.
- It's important to test the export of a report to PDF, because the layout might differ from what you see in Print Preview modus.
- In the next steps we will export the report to Excel.

14. Run the report for FontSize 6pt and export it to Excel.
 - a. The result will resemble this:

	A	B	C
1	FontSize=6pt		
2	CanShrink		
3	CanGrow		
4	CanShrink and CanGrow		
5			
6			
7			
8			
9			

FIGURE 4.19: FONTSIZE 6PT EXCEL WINDOW

15. Run the report for FontSize 10pt and export it to Excel.
 - a. The result will resemble this:

	A	B	C
1	FontSize=10pt		
2	CanShrink		
3	CanGrow		
4	CanShrink and CanGrow		
5			
6			
7			
8			
9			

FIGURE 4.20: FONTSIZE 10PT EXCEL WINDOW

16. Run the report for FontSize 16pt and export it to Excel.
 - a. The result will resemble this:

	A	B	C
1	FontSize=16pt		
2	CanShrink		
3	CanGrow		
4	CanShrink and CanGrow		
5			
6			
7			
8			
9			

FIGURE 4.21: FONTSIZE 16PT EXCEL WINDOW

17. Run the report for FontSize 30pt and export it to Excel.
 - a. The result will resemble this:

	A	B	C	D
1	FontSize=30pt			
2	CanShrink			
3	CanGrow			
4	CanShrink and CanGrow			
5				
6				

FIGURE 4.22: FONTSIZE 30PT EXCEL WINDOW

Conclusion:

- As you can see there might be differences in the **PDF** and **Excel** rendering.
- The **Word** rendering extension might also render **CanGrow** and **CanShrink** differently, for example here:

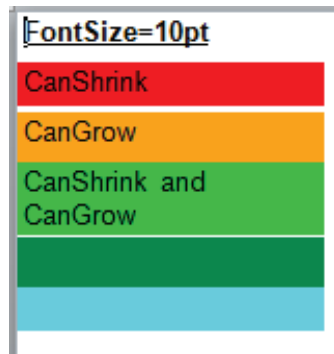


FIGURE 4.23: FONTSIZE 10PT WORD WINDOW

- When the text box shrinks, white space might appear in between the text boxes.

Check Boxes

Boolean fields can be displayed in multiple ways. You can use "**True/False**," "**Yes/No**," even work with "**1/0**" or any other combination.

You can either use the required data value of the field or work with expressions to translate the intrinsic value to a different and, or multilanguage enabled presentation. Visual Studio Report Designer does not include a separate check box control.

You can also simulate the check box control in reports. For a field to be displayed as a check box, you can change the font of the text box to Wingdings. In addition, you can set the Value property of the text box to an expression similar to the following:

Code Example

```
=IIF(Fields!Customer__Print_Statements_.Value = "1", Chr(0254), "o")
```

New Lines

To include line breaks in an expression, you can use the **vbCrLf** constant:

Code Example

```
= Fields!Customer_Name.Value & vbCrLf &  
Fields!Customer_Address.Value & vbCrLf &
```

```
Fields!Customer_Post_Code_.Value & space(2) &  
Fields!Customer_City.Value & vbCrLf &  
Fields!Customer_Country_Region_Code_.Value
```

You can use this to format text as a block in a text box. In the previous expression, the address fields will be formatted as a block inside a text box.

If the line break is the last character in a text box, it will be ignored. (If you have multiple line breaks, only the last one will be ignored.)

Other Visual Basic (VB) constants such as **vbTab** are not supported inside a text box. To insert several spaces, you can use the **Space()** function.

Use Rectangles to Keep Objects Together

Rectangles can be used either as graphical elements or as containers of objects. As object containers, they keep objects together on a page and control how objects move and push one another.

To keep multiple objects together on a page, put the objects in a rectangle. You can then put a page break before or after the rectangle by using the **PageBreakAtStart** or **PageBreakAtEnd** properties for the rectangle.

Using Rectangles to Control Item Growth and Displacement or Visibility

Items in a rectangle become peers of one another and are governed by the rules of how peer items are positioned on the page as they move or grow. For example:

- Items will push or displace one another in the rectangle.
- Items will not push or displace items outside the rectangle, because they are not their peers.
- If it is necessary, a rectangle will grow to accommodate the items it contains.

You can use this logic to your advantage when you deal with objects that expand. For example:

- If you want to leave a blank space in your report for a table to expand into, group the blank space and the table in the same rectangle. When the table grows, it will push the blank space.
- If you want to prevent a matrix from pushing items off the right side edge of the page, put the matrix in a rectangle with blank space to its right. Now, the matrix is no longer a peer to the other item on the page and cannot push it until the matrix can no longer be contained within its rectangle.

If you have multiple controls that must be hidden together, put them inside a rectangle and define the **Visibility** for the rectangle. The following code sample shows or hides the rectangle (and all controls inside the rectangle) based on the value of a **Boolean** variable (HideRectangle) on the request options page:

Code Example

```
=IIF(First(Fields!HideRectangle.Value = True), True, False)
```

To see whether a control is positioned inside a rectangle control, check the control's **Parent** property. The property cannot be edited directly in the **Properties** window. To change the **Parent** property, drag the control to another location. Default value for all controls is the section that they are included in.

Avoid Blank Pages

Sometimes, blank pages will be visible when outputting reports to a physical page format such as PDF or print. Generally, this happens when the size of the report body exceeds the size of the page.

For rendering formats that render physical pages (such as PDF), you can set the page **height** and page **width** properties for the report to control the pagination.

1. Select Report, Report Properties in Visual Studio Report Designer.
2. In the **Report Properties** window, click the **Page Setup** tab.

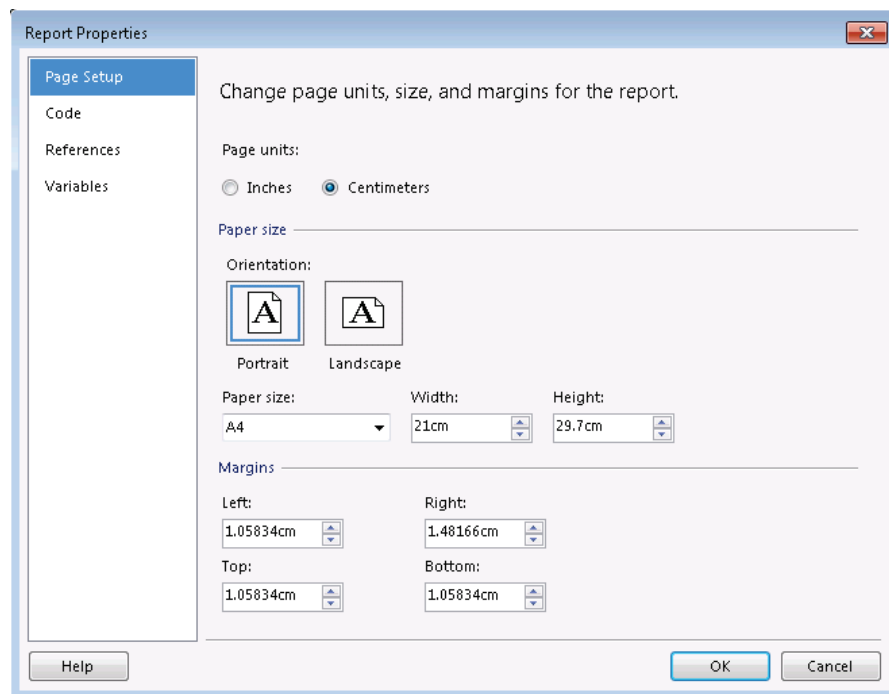


FIGURE 4.23: THE REPORT PROPERTIES WINDOW WITH THE DEFAULT PAGE SIZE AND MARGINS

Here you can define the page **height** and page **width** together with the different report margins.

To make sure that all the contents fit on a single page, the body width plus the margins must be less than the defined page width. However, some report items can grow horizontally and can cause the width of the body to exceed the page width. Typical examples are matrix data region controls and images set to automatically **Autosize** or **Fit**.

You must understand that setting the **Body** size at design time is not generally useful in controlling pagination. The body is the container for the collection of objects on the report. The design time size of the body represents the smallest amount of space that will be used by the report. However, if any objects grow as they are filled with data, the body will expand to contain them. Mostly, the body will grow taller (tables, lists, and text boxes are some items that will cause vertical growth) but can also grow wider (matrices and autosized images can cause horizontal growth).

When the runtime size of the body (plus the page header and footer size, if they are defined) exceeds the specified page size (minus the margins), a physical page break occurs. You must understand that controls on the page that do not seem to contain any information (a wide text box) can cause a blank page to be added to the report (when printed or viewed in **Print Layout** mode) if the control exceeds the page boundaries. Page breaks can also be specified explicitly before or after instances of report items (rectangles, groups, data regions).

The Excel rendering format does not explicitly support pagination. The page size and margins are mapped into the Excel document for printing within Excel. For these reports, you will have to specifically include page breaks to break the report into multiple worksheets.

Use Page Breaks to Improve Performance for Large Reports

If you do not specify a page size or page breaks for a report that returns a large amount of data, some report formats will try to render the report as a single page.

For example, Excel has no default notion of a fixed page size. Therefore, if you have a large report, Excel will try to render it as a single worksheet. Generally, by using page breaks this improves the performance for the users who are accessing the report, because they can view the first page while the rest of the report is being rendered.

Insert a Page Break Following a Specific Number of Rows in a Table

You can use the **Ceiling** function to group the rows in a table and insert a page break at the end of each group. The **Ceiling** function returns the smallest number that is greater than the argument. For example, to add a page break after every 30 rows, you can use the following expression to group the rows:

Code Example

```
=Ceiling(LineNumber(Nothing)/30)
```

To do this, add a group in the tablix and use this expression as the **GroupExpression**. Then in the group properties, enable the property to add a page break after each instance of the group.

Add Alternating Bars to a Table

You can create a report that contains a table or matrix in which every other row is shaded. This bar effect makes it easier to visually track the different rows across a page.

To more closely simulate the old "green bar" paper that were used at one time to run large reports on high-volume data center printers, you can make the alternating bars green.

To achieve this effect, use the **IIF()** function to assign the background color conditionally, based on whether the row number is odd or even. For example:

Code Example

```
=IIF(LineNumber(Nothing) Mod 2,"PaleGreen","White")
```

Pagination

One key design issue for long reports is controlling where the page breaks occur. Page breaks are controlled by two factors:

- Page size
- Page breaks specifically included before or after objects

Page Size

To control page size, set the page height and width properties for the report by using the following guidelines:

- For rendering formats that render physical pages (PDF), use the **PageHeight** and **PageWidth** properties for the report.
- Some rendering formats, such as Excel, do not support page size. For these reports, it is necessary to specifically include page breaks to break the report into multiple pages.

If the report is wider than the defined page width, then the report will break across multiple pages horizontally.

Page Breaks

You can apply page breaks at the beginning or end of a rectangle, table, matrix, list, chart, or group. Report Viewer tries to keep all the data in the item or grouped on the same page.

To include page breaks specifically before or after items, use the **PageBreakAtEnd** and **PageBreakAtStart** properties for the item.

Setup Multiple Columns

You can use the Report Designer to easily create multicolumn reports. A multicolumn report can conserve space by displaying the report data in more than one column.

No Item	Description Item		
1000	Bicycle	1976-W	INNSBRUCK Storage Unit/W.Door
1001	Touring Bicycle	1980-S	MOSCOW Swivel Chair, red
1100	Front Wheel	1984-W	SARAJEVO Whiteboard, blue
1110	Rim	1988-S	SEOUL Guest Chair, red
1120	Spokes	1988-W	CALGARY Whiteboard, yellow
1150	Front Hub	1992-W	ALBERTVILLE Whiteboard, green
1151	Axle Front Wheel	1996-S	ATLANTA Whiteboard, base
1155	Socket Front	2000-S	SYDNEY Swivel Chair, green
1160	Tire	70000	Side Panel
1170	Tube	70001	Base
1200	Back Wheel	70002	Top Panel
1250	Back Hub	70003	Rear Panel
1251	Axle Back Wheel	70010	Wooden Door
1255	Socket Back	70011	Glass Door
1300	Chain Assy	70040	Drawer
1310	Chain	70041	Shelf
1320	Chain Wheel Front	70060	Mounting
1330	Chain Wheel Back	70100	Paint, black
1400	Mudguard front	70101	Paint, yellow
1450	Mudguard back	70102	Paint, blue
1500	Lamp	70103	Paint, red
		70104	Paint, green

FIGURE 4.24: EXAMPLE OF A MULTICOLUMN REPORT

To create this kind of multicolumn report, you must make sure that the report data width does not exceed the column width.

To view the report rendered correctly in multiple columns, make sure that you preview the report by clicking the **Print Layout** button. If you preview the report, you cannot view the data that flows in columns because the preview mode does not consider the page settings.

Currently, Report Viewer supports defining multiple columns only at the report level. You cannot, for example, define a multicolumn layout by region, for example, a table region.

To define the number of columns, select **Report, Report Properties** from the menu. On the **Page Setup** tab, you can specify the number of columns, the page size and the margins.

When you set up the column widths, you must make sure that there is sufficient page space to accommodate the number of columns, by using the following formula:

$$\text{Page width} - (\text{left margin} + \text{right margin}) \geq \text{number of columns} * \text{column width} + (\text{number of columns} - 1) * \text{column spacing}$$

Report Designer eliminates the trial-and-error fitting game and shows you the outline of the columns in layout mode. Then, you can easily determine whether the report width exceeds the page width.

The Usage of Colors

In this module several techniques show how to apply colors to a report so that you can visualize information more clearly. An example is the usage of alternating colors in a tablix data region. Using alternating colors increases the readability of the report.

However a certain percentage of people could have problems distinguishing colors. A part of the report users might be unable to correctly interpret some graphs and charts. Being unable to clearly distinguish colors makes charts and tables confusing to understand.

Remember to consider the following when you design and implement report items so that they can be used by every report user in the company.

- Images that are used as key performance indicators must be different in shape rather than just color.
- In Line charts, use markers with different shapes.
- In bar graphs, include a marker on the top of each bar.
- Shapes are as important as colors.
- Greyscale and shades of the same color might be more readable for a person who has a color vision deficiency.

A way to test the readability of a report that contains colors is to print the report in black-and-white and then determine whether all the information is well presented.

Lab 4.1: Creating Green Bar Reports

Scenario

Ellen asks Mort to improve the readability of the Inventory report. Instead of having the item number and item name displayed on an orange background, she wants to have the inventory amounts printed with alternating background colors. She also prefers a lighter pastel background color (**LightSteelBlue**).

Objectives

In this lab, you will fine-tune the report that you created in Lab 1.2 (Creating a Matrix Report) by adding the green bar functionality.

Exercise 1: Implement a Green Bar Effect

Exercise Scenario

Create a green bar matrix report that shows the item inventory by location. The rows must have alternating background colors. To select the correct colors, use a function that returns either "**LightSteelBlue**" or "**White**" every time that a new item number is printed.

Task 1: Import Report 123456701

High Level Steps

1. Import the object in the demonstration database.

Detailed Steps

1. Import the object in the demonstration database.
 - a. Open the Microsoft Dynamics NAV 2013 demonstration database.
 - b. Open the CRONUS International Ltd. demonstration company.
 - c. Select **Tools, Object Designer** to open the Object Designer.
 - d. Select **File, Import**.
 - e. Browse to the report R123456701.fob.
 - f. Click **OK** to start the import.
 - g. In the dialog box, select to open the **Import Worksheet** window.
 - h. In the **Action** column, verify that the action is set to **Create**.

Click **OK** to import the object in the database.

Task 2: Add a Custom Function to the Report

High Level Steps

1. Add a custom function to the report.

Detailed Steps

1. Add a custom function to the report.
 - a. Open report 123456701 in Visual Studio.
 - b. Select Report, Report Properties.
 - c. On the **Code** tab, add the following custom variable and function, if not already present in the report and review the code:

Code Example

```
Private bOddRow As Boolean

Public Function AlternateColor(ByVal OddColor As String, ByVal EvenColor as
String, ByVal Toggle As Boolean) As String
    If Toggle Then
        bOddRow = Not bOddRow
    End If
    If bOddRow Then
        Return OddColor
    Else
        Return EvenColor
    End If
End Function
```

- d. Click **OK** to close the **Report Properties** window.

Task 3: Change the Background Color for the Row Header Cells

High Level Steps

1. Change the background color for the cells in the first column.

Detailed Steps

1. Change the background color for the cells in the first column.
 - a. Select the first cell on the first table row (the cell that contains the **Item No.** and **Description** fields).
 - b. In the **Properties** window, select the **BackgroundColor** property.
 - c. Change the Value of the **BackgroundColor** property to the following expression:

```
=Code.AlternateColor("LightSteelBlue", "White",
True)
```

- d. Repeat this step for the textbox on the second row, first column.

- e. Select **Rectangle1** in the Properties window.
- f. Set the **BackgroundColor** property to of Rectangle1 to No Color

Task 4: Change the Background Color for the Other Row Cells

High Level Steps

1. Change the background color for the other row cells.

Detailed Steps

1. Change the background color for the other row cells.
 - a. Select the second cell on the second table row (the cell that contains the **Inventory** field (textbox Item_Ledger_Entry_Quantity), at the right bottom of the Tablix. This textbox has the following expression for its value:

```
=Fields!Item_Ledger_Entry_Quantity.Value
```

- b. In the **Properties** window, select the **BackgroundColor** property.
- c. Change the Value of the **BackgroundColor** property to the following expression:

```
=Code.AlternateColor("White", "LightSteelBlue", False)
```

- d. In the properties window, select textbox Color.
- e. Set the **Width** property to **0cm**.
- f. Exit Visual Studio, import the new RDLC layout in the object, save the object and run it.

The result will resemble this:

	BLUE	GREEN	OUT. LOG.	OWN LOG.	RED	WHITE	YELLOW
1896-S ATHENS Desk		49	25			52	160
1900-S PARIS Guest Chair, black	52	47				46	160
1906-S ATHENS Mobile Pedestal	70	108		40		63	
1908-S LONDON Swivel Chair, blue	234	47		10		5	
1920-S ANTWERP Conference Table	38	67				8	
1924-W CHAMONIX Base Storage Unit	1	3				2	15
1928-S AMSTERDAM Lamp	149	-5				57	97
1928-W ST.MORITZ Storage Unit/Drawers	4	4				-1	41
1936-S BERLIN Guest Chair, yellow	36	50	4	4		50	
1952-W OSLO Storage Unit/Shelf	2	-1				2	
1960-S ROME Guest Chair, green	136					32	
1964-S TOKYO Guest Chair, blue	59	14		14		29	
1964-W INNSBRUCK Storage Unit/G.Door	6	5				-1	8
1968-S MEXICO Swivel Chair, black	236	14				28	
1968-W GRENOBLE Whiteboard, red		4				6	10
1972-S MUNICH Swivel Chair, yellow	37	-1				5	90
1972-W SAPPORO Whiteboard, black	4	2				5	
1976-W INNSBRUCK Storage Unit/W.Door	3	-2				3	3
1980-S MOSCOW Swivel Chair, red	65	14				24	
1984-W SARAJEVO Whiteboard, blue	3	3				4	
1988-S SEOUL Guest Chair, red	41	83					43

FIGURE 4.25: GREEN BAR REPORT PREVIEW

Lab 4.2: Creating a Top X Report

Scenario

Prakash is searching for available resources for one of his projects and he wants to check the resource usage. The current report 1106, **Resource -Utilization**, does not immediately provide this information, because it is sorted by resource number and sorting cannot be changed.

Prakash wants to have the existing report content replaced by two charts. The first chart must show the three most used resources. The second chart must show the five least used resources.

Finally, he asks Mort not to delete the current report content. Instead, he proposes to hide the content, and include a button or a label that can be clicked to show or hide the details dynamically.

After he creates a copy of the original report, Mort starts to develop the report.

Objectives

In this lab, you will add Top X functionality to report 123456706, **Resource Utilization - Chart**.

Exercise 1: Implement a Top X Dashboard

Exercise Scenario

Make a copy of report 1106, **Resource - Utilization** and save it as report 123456706, **Resource - Utilization Chart**. Extend report 123456706 so a Top 3 chart for the three most used and a Top 5 chart for the least used resources show at the same time. Add a "Show Details" text box that you can use to toggle the visibility of the current report content (the table control). The table must be displayed under the charts.

Task 1: Import Report 123456706

High Level Steps

1. Import the object in the demonstration database.

Detailed Steps

1. Import the object in the demonstration database.
 - a. Open the Microsoft Dynamics NAV 2013 demonstration database.
 - b. Open the CRONUS International Ltd. demonstration company.
 - c. Select **Tools, Object Designer** to open the Object Designer.
 - d. Select **File, Import**.

- e. Browse to the report R123456706.fob.
- f. Click **OK** to start the import.
- g. In the dialog box, select to open the **Import Worksheet** window.
- h. In the **Action** column, verify that the action is set to **Create**.
- i. Click **OK** to import the object in the database.
- j. Click **OK** in the Imported Objects window.

Task 2: Reorganize the Body Section

High Level Steps

1. Reorganize the body section.

Detailed Steps

1. Reorganize the body section.
 - a. Open report 123456706 in Visual Studio Report Designer.
 - b. Select the body.
 - c. Change the height of the body section to **9.10001cm**.
 - d. Select the **Table1** Tablix control.
 - e. Move it to the bottom of the body section.

Task 3: Include a "Show Details" Text Box to Show/Hide the Table

High Level Steps

1. Add a "Show Details" text box to Show/Hide the Table.

Detailed Steps

1. Add a "Show Details" text box to Show/Hide the Table.
 - a. In the **Toolbox** window, select the **Textbox** control.
 - b. Click the body to insert a text box control.
 - c. Move the newly added text box control to the left of the body section. Position it immediately under the **Resource_TABLECAPTION_____ResFilter** text box.

Task 4: Format the Show Details Text Box

High Level Steps

1. Format the Show Details text box.

Detailed Steps

1. Format the Show Details text box.
 - a. Select the newly added text box.
 - b. In the **Properties** window, set the **Name** property to **ShowDetails**.

- c. In the **Properties** window, set the **Value** property to **Show Details**.
- d. In the **Properties** window, set the **BackgroundColor** property to **LightGrey**.
- e. In the **Properties** window, set the **Color** property to **Black**.
- f. In the **Properties** window, set the **TextAlign** property to **Center**.
- g. In the **Properties** window, set the **FontWeight** property to **Bold**.

Task 5: Hide the Current Report Content

High Level Steps

1. Hide the report content.

Detailed Steps

1. Hide the report content.
 - a. In the body, select the **Table1 Tablix** control.
 - b. In the **Properties** window, set the **Hidden** property to **True**.
 - c. In the **Properties** window, set the **ToggleItem** property to **ShowDetails**.

Task 6: Add a Top 3 Chart for the Most Used Resources

High Level Steps

1. Add a chart to the report.

Detailed Steps

1. Add a chart to the report.
 - a. In the **Toolbox** window, select the **Chart** control.
 - b. Click the body to insert the **Chart** control.
 - c. In the **Chart Type** field, select **Stacked Column** and then click the **OK** button.
 - d. In the **Properties** window, change the **Left** property to **0 cm**.
 - e. In the **Properties** window, change the **Top** property to **1.446cm**.

Task 7: Add Data Elements to the Chart Control

High Level Steps

1. Add data elements to the Chart.

Detailed Steps

1. Add data elements to the Chart.
 - a. Select View, Report Data to open the Report Data window.
 - b. Double-click the chart control.
 - c. In the **Report Data** window, select the **Resource_No__** element.

- d. Drag it to the **Drop Category Fields Here** area.
- e. In the **Report Data** window, select the **Resource_Usage_Qty__** element.
- f. Drag it to the **Drop Data Fields Here** area.

Task 8: Format the Top 3 Chart

High Level Steps

1. Format the Top 3 chart.

Detailed Steps

1. Format the Top 3 chart.
 - a. Select the chart control and select **Chart Properties**.
 - b. Open the Titles property collection in the **Properties** window.
 - c. In the **Caption** field enter the following text: **Top 3 - Resource Utilization**.
 - d. Click the **OK** button to close the **ChartTitle Collection Editor** window.
 - e. In the **Palette** property of the chart, select **Excel**.
 - f. Double-click the **Chart** and in the **Drop Category field here**, right-click the **Resource_No__** field.
 - g. Select the option **Category Group Properties**.
 - h. On the **Filters** tab , select Add to add a filter.
 - i. Select **[Resource_Usage_Qty__]** in the **Expression** field.
 - j. In the **Operator** column, select Top N.
 - k. In the Value column, enter =3.
 - l. On the **Sorting** tab , select Add to add a filter.
 - m. On the **Sorting** tab, in the **Sort by** column, select **[Resource_Usage_Qty__]**.
 - n. In the **Direction** column, select **Z to A**.
 - o. Click the **OK** button to close the **Category Group Properties** window.
 - p. On the **Vertical Axis**, in the **Title** field, enter **Usage (Qty)**.
 - q. On the **Horizontal Axis**, in the **Title** field, enter **Resource**.
 - r. On the **Chart**, right-click and select **Legend Properties**.
 - s. In the **Visibility** tab, select Hide.
 - t. Click **OK** to close the **Legend Properties** window.

Task 9: Add a Top 5 Chart for the Least Used Resources

High Level Steps

1. Add a top 5 chart for resources that are used the least.

Detailed Steps

1. Add a top 5 chart for resources that are used the least.
 - a. In the **Toolbox** window, select the **Chart** control.
 - b. Click the body to insert the **Chart** control.
 - c. In the **Chart Type** field, select **Stacked Column** and then click the **OK** button.
 - d. In the **Properties** window, change the Left property to **7.5cm**.
 - e. In the **Properties** window, change the Top property to **1.446cm**.

Task 10: Add Data Elements to the Chart Control

High Level Steps

1. Add the fields **Resource_No__** and **Resource_Usage_Qty__** to the chart.

Detailed Steps

1. Add the fields **Resource_No__** and **Resource_Usage_Qty__** to the chart.
 - a. Double-click the chart control.
 - b. In the **Website Data Sources** window, select the **Resource_No__** element.
 - c. Drag it to the **Drop Category Fields Here** area.
 - d. In the **Website Data Sources** window, select the **Resource_Usage_Qty__** element.
 - e. Drag it to the **Drop Data Fields Here** area.

Task 11: Format the Top 5 Chart

High Level Steps

1. Apply formatting to the chart.

Detailed Steps

1. Apply formatting to the chart.
 - a. Select the chart control and select **Chart Properties**.
 - b. Open the **Titles** property collection in the **Properties** window.
 - c. In the **Caption** field enter the following text: **Bottom 5 - Resource Utilization**.
 - d. Click the **OK** button to close the **ChartTitle Collection Editor** window.

- e. In the **Palette** property of the chart, select **Excel**.
- f. Double-click the **Chart** and in the **Drop Category field here**, right-click the **Resource_No** field.
- g. Select the option **Category Group Properties**.
- h. On the **Filters** tab, select Add to add a filter.
- i. On the **Filters** tab, select **[Resource_Usage_Qty]** in the **Expression** field.
- j. In the **Operator** column, select **Bottom N**.
- k. In the **Value** column, enter **=5**.
- l. On the **Sorting** tab, select Add.
- m. On the **Sorting** tab, in the **Sort by** column, select **[Resource_Usage_Qty]**.
- n. Click the **OK** button to close the **Category Group Properties** window.
- o. On the **Vertical Axis**, in the **Title** field, enter **Usage (Qty)**.
- p. On the **Horizontal Axis**, in the **Title** field, enter **Resource**.
- q. On the Chart, right-click and select **Legend Properties**.
- r. In the **Visibility** tab, select **Hide**.
- s. Click **OK** to close the **Legend Properties** window.

Task 12: Run the Report

High Level Steps

1. Save and run the report.

Detailed Steps

1. Save and run the report.
 - a. Exit Visual Studio.
 - b. Save and import the RDLC changes.
 - c. Save and compile the report in the Report Designer.
 - d. Click **Run**.

After you click the **Show Details** button, the report resembles this:

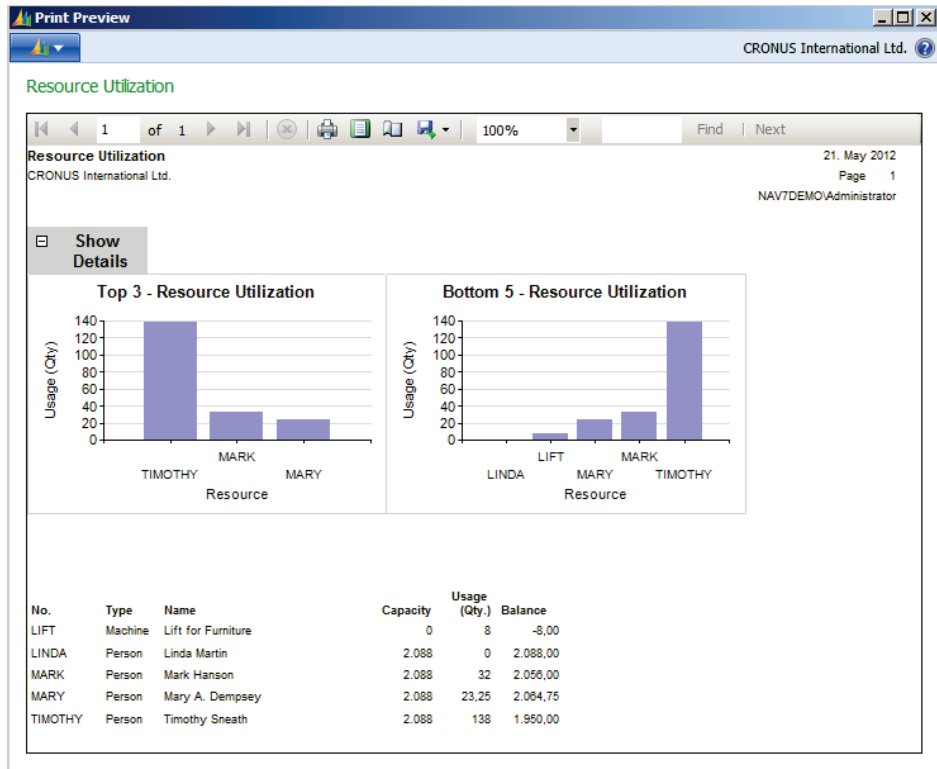


FIGURE 4.26: TOP X REPORT PREVIEW

Module Review

Module Review and Takeaways

This module demonstrated how to implement some of the most used report patterns and recipes:

- How to apply a green bar effect.
- How to create a dashboard report.
- Using a Chart in a report.
- Simulate transheader and transfooter sections.

Furthermore the module contains useful considerations about report rendering and pagination and interesting tips and tricks in advanced report design.

Test Your Knowledge

Test your knowledge with the following questions.

1. What happens when a report is exported with a multilevel document map to PDF?
 - () The document map is not exported.
 - () The document map is exported without changes to the Bookmarks pane.
 - () The document map is exported but the hierarchy is lost and all items appear on the same level in the Bookmarks pane.
 - () An error message is displayed because you cannot export reports that use multilevel document maps.
2. What happens when a report that uses a chart is exported to Excel?
 - () The chart is converted to an Excel chart.
 - () The chart is converted to an image.
 - () The chart is converted to a table.
 - () An error message is displayed because you cannot export reports that use charts.

3. Which C/AL function for exporting reports is now added in Microsoft Dynamics NAV 2013?
- SAVEASHTML
 - SAVEASPDF
 - SAVEASWORD
 - SAVEASEXCEL
4. Which operator is used to filter a recordset to the three highest values?
- Max(3)
 - Max N
 - Top 3
 - Top N
5. Which operator is used to filter a recordset to the three lowest values?
- !Top N
 - Top N
 - Bottom N
 - <> Top N

Test Your Knowledge Solutions

Module Review and Takeaways

1. What happens when a report is exported with a multilevel document map to PDF
 - () The document map is not exported.
 - () The document map is exported without changes to the Bookmarks pane.
 - (√) The document map is exported but the hierarchy is lost and all items appear on the same level in the Bookmarks pane.
 - () An error message is displayed because you cannot export reports that use multilevel document maps.
2. What happens when a report that uses a chart is exported to Excel?
 - () The chart is converted to an Excel chart.
 - (√) The chart is converted to an image.
 - () The chart is converted to a table.
 - () An error message is displayed because you cannot export reports that use charts.
3. Which C/AL function for exporting reports is now added in Microsoft Dynamics NAV 2013?
 - () SAVEASHTML
 - () SAVEASPDF
 - (√) SAVEASWORD
 - () SAVEASEXCEL
4. Which operator is used to filter a recordset to the three highest values?
 - () Max(3)
 - () Max N
 - () Top 3
 - (√) Top N

5. Which operator is used to filter a recordset to the three lowest values?

!Top N

-Top N

Bottom N

<> Top N