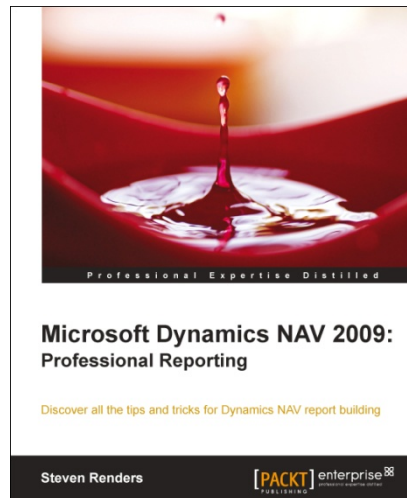


# Microsoft Dynamics NAV 2009: Professional Reporting

**Steven Renders**



## Chapter No. 2 "Creating a Report in the Classic Client"

## In this package, you will find:

A Biography of the author of the book

A preview chapter from the book, Chapter NO.2 "Creating a Report in the Classic Client"

A synopsis of the book's content

Information on where to buy this book

## About the Author

**Steven Renders** is a Microsoft Certified Trainer (MCT) with a wide range of skills spanning business and technical domains. He later specialized in Microsoft Dynamics NAV and Microsoft SQL Server.

He has more than 15 years of business and technical experience and provides training and consultancy focused on Microsoft Dynamics NAV, Microsoft SQL Server, Business Intelligence Solutions, Microsoft SQL Server Reporting Services, and Database Performance Tuning.

Furthermore, he is also an expert on Microsoft Dynamics 2009, on which he has already delivered many training sessions. Steven was an author of some of the official Microsoft training materials on Dynamics NAV Reporting, Dynamics NAV SQL Server performance tuning, and development. Steven was also a reviewer of the book: Programming in Microsoft Dynamics NAV 2009.

Steven was also a presenter at various Microsoft MSDN and TechNet evenings, conferences, communities, events, and the MCT Summit in Prague.

**For More Information:**

[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)

Steven has been awarded the Microsoft Certified Trainer (MCT) status since 2007, and is a Microsoft Certified Technology Specialist (MCTS) in Microsoft SQL Server, Microsoft Certified IT Professional Developer for Microsoft Dynamics NAV (MCITP), and Microsoft Certified IT Professional Installation and configuration for Microsoft Dynamics NAV (MCITP).

Recently, Steven started his own company, Think About IT, which is specialized in training and consultancy, helping companies learn, implement, understand, and solve complex business requirements related to IT, in Belgium and abroad.

Specialties:

- Microsoft Dynamics NAV
- Microsoft SQL Server
- Business Intelligence & Management Reporting

E-mail: [steven.renders@thinkaboutit.be](mailto:steven.renders@thinkaboutit.be)

LinkedIn: <http://be.linkedin.com/in/stevenrenders>

**For More Information:**

**[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)**

# Microsoft Dynamics NAV 2009: Professional Reporting

Microsoft Dynamics NAV gives you direct access to real-time, business-critical information, and a wide range of analytical tools to help you manage budgets, create and consolidate reports, and look for trends and relationships.

What's more, Microsoft Dynamics NAV is built on industry-standard Microsoft technology and integrates with other Microsoft Business Intelligence (BI) products and technologies. So, you can start with the basic modules and Microsoft Office Excel and then add functionality and tools as you need them.

This is how Microsoft describes the Business Intelligence capabilities of the Dynamics NAV product. As we will see in this book, this description is not far from the truth. The only problem is that most people don't have a clear idea of how much is available in the box and what's available out of the box.

Implementers are usually very good in setting up the application so that users can input their data into the system according to the processes and flows in their organization / company. A lot of time and effort is spent on adapting the application to the flows and processes of the organization. But, after the data is then finally inside the database, how do we then get it out again? That's a question I get a lot from customers. In this book, I will give you an answer to this question.

After reading this book it should be clear how to manipulate Dynamics NAV for it to produce the reports and analytical data that you want, when you want it, and in the format you want it!

## What This Book Covers

*Chapter 1, What's Available in the Box?*, gives an overview of the types of reports available in the application, where to find them, and how to use them. You can create your custom reports without the need for a developer or report designer. This can be done by making use of dimensions, analysis views, and account schedules and so on. It's important to know what's already available for free in the application, before you start spending time, money, and resources on developing custom reports.

*Chapter 2, Creating a Classic Report*, explains the classic layout, how to create it, and all the capabilities of the good old classic report designer. The creation aspect of the classic report will be kept to a minimum, with more focus on new reporting possibilities of the Role Tailored Client. You will need this information to design reports for the Role Tailored Client, as development of reports starts with a classic report and knowledge about the classic designer is still required.

**For More Information:**

[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)

*Chapter 3, Creating a Role Tailored Client Report*, dives into the Role Tailored Report designer, or the RDLC report layout as it is called. It starts with an introduction to Visual Studio, its different flavors (versions), the toolbars, the environment, and useful shortcuts. It will explain the different ways to create a report from scratch using the Create Layout suggestion feature. You will see different kinds of problems you may encounter when developing reports for the Role Tailored Client and how to troubleshoot them.

*Chapter 4, Visualization Methods*, explains that creating reports is not just extracting and formatting data from the database and dropping it onto a layout. The way you visualize the information is equally important. The report will stand or fall depending on the way the information is rendered and presented to the user. That's why a big portion on the chapter will be about data visualization techniques and how to apply them in RDLC.

*Chapter 5, Developing Specific Reports*, explains how the RDLC report layout for documents, such as a sales invoice, is full of workarounds. We will explore it in detail with the most important workarounds, how and why they are required, and explore some alternative solutions. Creating dashboards and top x reports are also covered in this chapter.

*Chapter 6, Other Reporting Tools and Business Intelligence*, explains the database behind the Dynamics NAV application. How can you create an ER model? How are the tables related to each other? This chapter will address all these questions. The chapter then dives into SQL Server Reporting Services and explains how you can create an SSRS report. Other BI tools from the Microsoft stack, like for example PowerPivot, Excel Data Mining and Business Analytics, are also covered in this chapter. The purpose is to get a good overview on the other tools that are out there and the added value they have to offer on top of a Dynamics NAV database.

*Chapter 7, A View to the Future*, shows the future panorama of reporting in Dynamics NAV. Besides the Dynamics NAV application, the other BI applications are also evolving and becoming more integrated. This chapter will try to give you an overview on what will or might happen and the kind of impact or added value it might offer for Dynamics NAV.

**For More Information:**

[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)

# 2

## Creating a Report in the Classic Client

When Dynamics NAV 2009 was introduced, it also came with the capability to add a Role Tailored Layout for a report. The previous version had only the Classic Layout. This chapter is all about the Classic Layout, how to create it, and all of the capabilities of the good old Classic Report Designer.

The knowledge obtained in this chapter is required for when you migrate towards the Role Tailored Client or when you are already using the Role Tailored Client with the new report layout. This is because the Classic Report Designer is, currently, the starting point for the development of all types of reports in Dynamics NAV. First, we learn how to walk, then how to run and win the race.

In this chapter, we will learn about:

- Using the report wizard
- Creating a simple list report
- Creating an Excel-like layout for a report
- Printing a report to Excel
- Report functions

### The Report Designer

Reports are used to present information from the database, structure and summarize information, and print documents such as invoices. Reports can also be used to process data without printing anything.

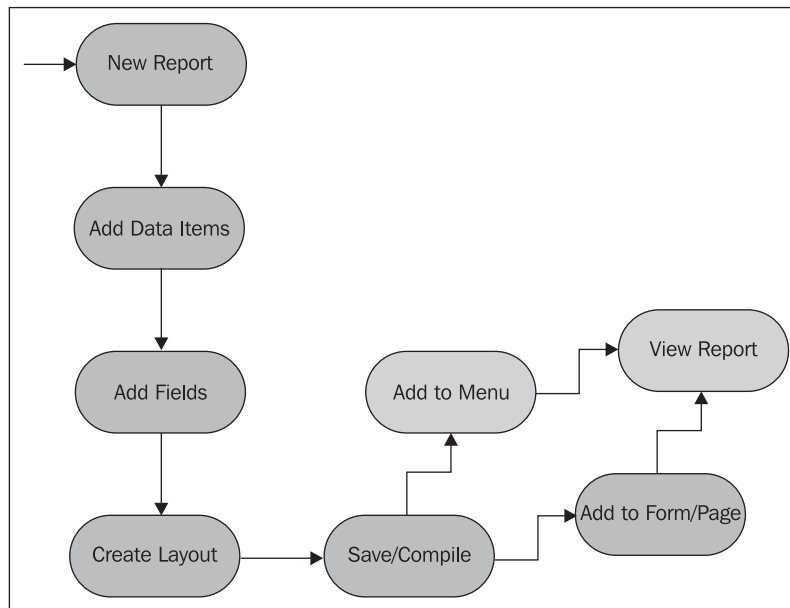
Reports can be created or customized via the **Report Designer**, which can be found in the **Object Designer**.

**For More Information:**

[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)

The Report Designer is the development environment that is available in the Dynamics NAV Classic Client to create or customize report objects. It contains a data item designer to define the data model for the report, a section designer to design the layout of the report for the Classic Client, and a request form designer to create an optional request form in which the user can select options when running the report.

The workflow for designing a Classic report in Dynamics NAV 2009 can be visualized with the following diagram:



There are two ways to create a new report: from scratch or by using the wizard. I will start with an explanation of the report wizard and then dive into the process of manually creating a report from scratch.

## Using the report wizard

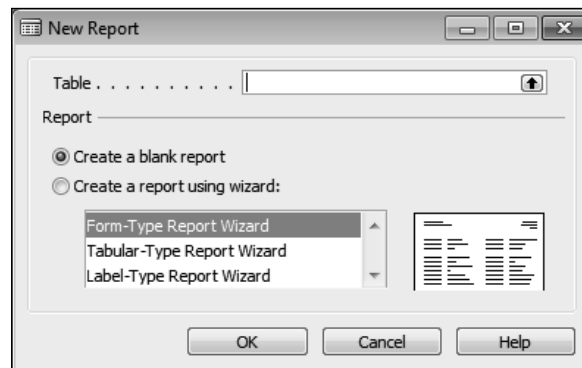
Dynamics NAV has a built in report wizard that you can use to automatically create a report with just your mouse. The report wizard can create three types of reports:

- **Form Type Reports:** Use this template to create a form type report. The report will have multiple columns of information and the look and feel of the report resembles the look and feel of a Card type form.
- **Tabular Type Reports:** Use this template to create a list type report. The report will resemble the look and feel of a list type form.

- **Label Type Reports:** Use this template to create a report used to print labels. You can use these labels from the report for example on letters or on envelopes.

You launch this wizard by clicking on the **New** button in the **Object Designer** window, after selecting the Report objects button on the left side of the Object Designer.

This will open the following window:



When you select a report type in this window, then the image to the right of the report type will change to reflect the layout of the chosen report type.

## Form type report

At this stage, you have to enter the name or number of the table you want to display information from in the textbox at the top of the report wizard window.

Then, click on the **next** button to open the next window in the report wizard. Here, you can select the fields from the table. You can select a field from the list on the left and use the buttons in the middle to add a field (or multiple fields at once) to the report. At the right-hand side, there are two buttons: **Separator** and **Column Break**. You can use these buttons to add columns to the report and to insert a separator.

In the next window, you can optionally select the sorting order or the **Key** the report will use.

When you click on the **Finish** button, the report is ready and opens in the report designer. In here, you can save the report or modify it further.



## Tabular type report

When you select Tabular Type Report, the next two windows are the same as the one for the Form Type Report. You select the fields you want to be available on the report and then the Key to use to sort them.

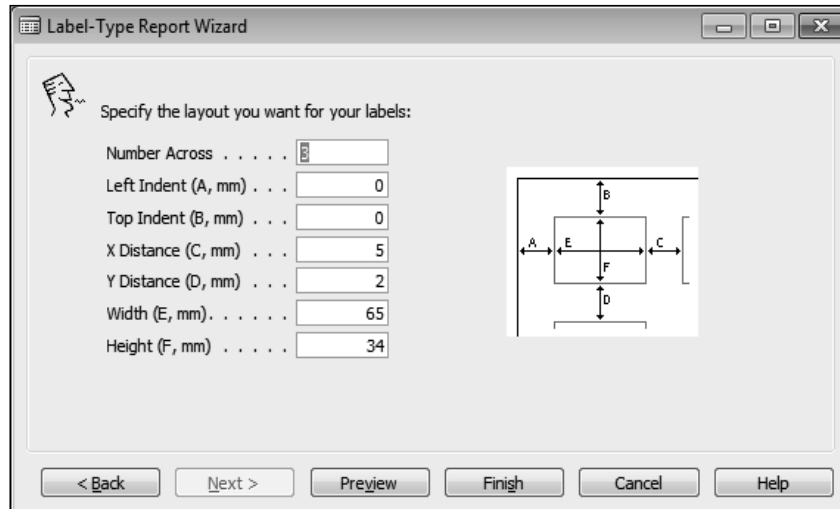
When you then click on the **Next** button, a window opens in which you can select the fields to use to group the data of the report. This is an option, it's not mandatory. In the next window, you can select a report style: List or Document. This will change the look and feel. Clicking on either of these options will change the preview image at the right of the window.

When you click on the **Finish** button, the report is ready and opens in the report designer. Here, you can save the report or modify it further.

## Label type report

When you select Label Type Report, the next two windows are the same as the one for the Form Type Report. You select the fields you want to be available on the report and then the Key to use to sort them.

In the next window, you can specify the size of the labels, as you can see in the following screenshot:



**For More Information:**  
[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)

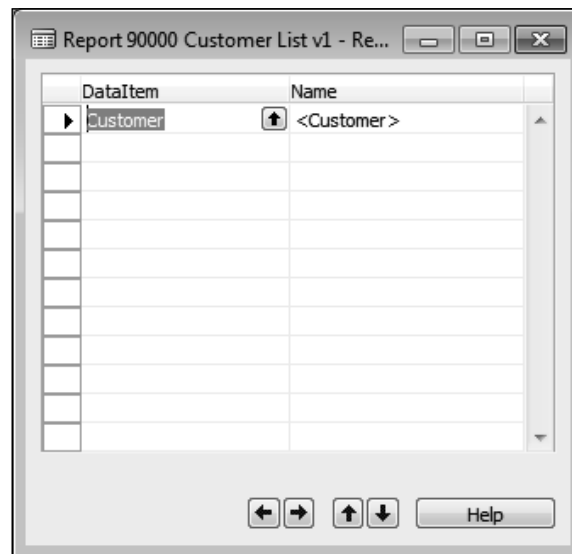
## Creating a simple List report

The first thing you have to do when you want to design a new custom report is create the data model. Where is the data coming from that I want to show in the report? Is it all in one table or do I need multiple tables? If there are multiple tables, how will I connect the records from the different tables? A good suggestion that I can give is to first make a draft drawing on a piece of paper of the layout of the report you want to create. Write down the fields that need to be visible on the report and then find out which table they are coming from. After that, if there are multiple tables find out how the tables are related and write that down.

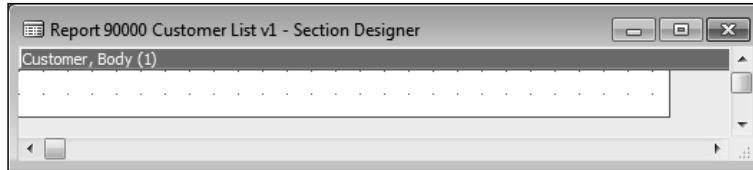
This way, when you open the designer you already know what you need to do. Many novice and experienced developers make the mistake of not thinking before they begin. And then it can get confusing very quickly in the report designer.

Now that we have a good idea of the tables that we will require, it's time to open the report designer. The first thing that you will see is the **Data Item Designer**. This is where you define the Data Model of the report.

Let's start with a simple example of a **List report**. Suppose you would like to create a list of customers. Then, the table from which the data is coming out of is the **Customer table (18)**. This will be our **DataItem**, as you can see in the following screenshot:

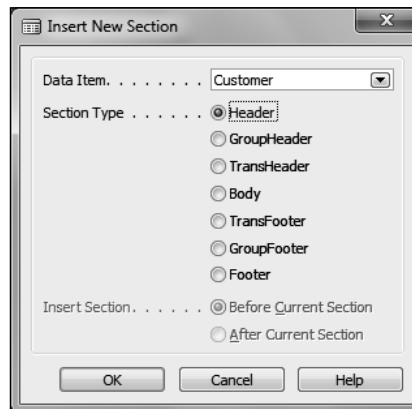


Now, it's time to design the actual layout of the report. For this, we need to open the Section Designer. There's no shortcut for this, so you will need to click on the menu and select **View | Sections**. When the **Section designer** opens, you will notice a **Body** section with the same name as your data item, like in the following screenshot:



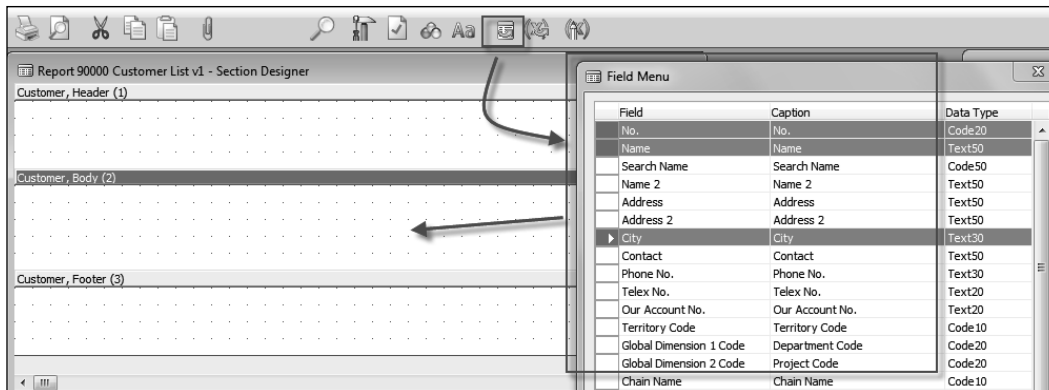
Sections are linked to data items. A body section will repeat itself for every record of the data item. When you click the *F3* function key, the **Insert New Section** window will open.

Here, you can select to add a section to the report:

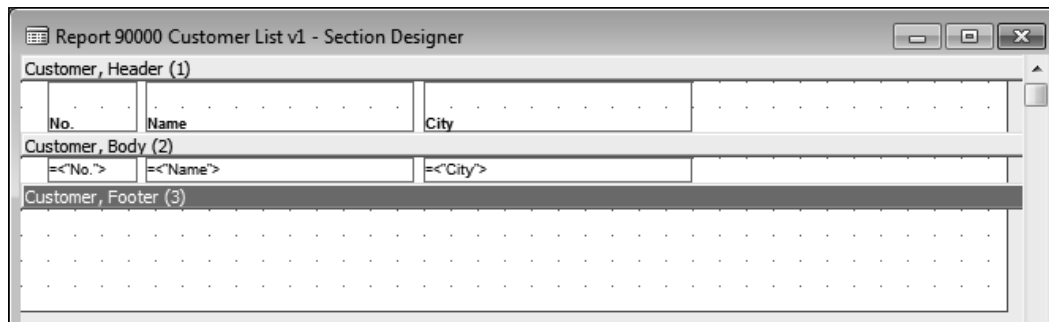
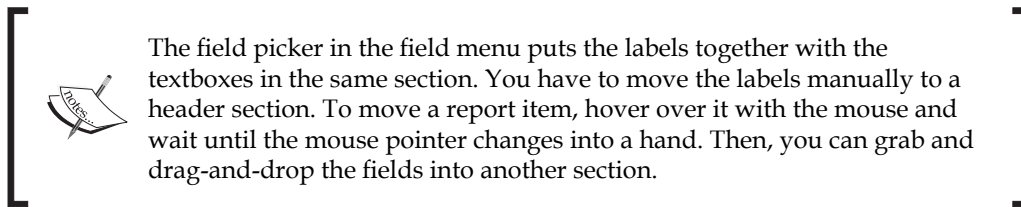


A **Header** section will be shown only once, before the **Body**. A **Footer** section will be shown only once, below the **Body** section. A header section is usually used to display column labels and a footer section is usually used to display totals. You can add as many sections as you like. For example, you can create multiple Header (and/or Body, Footer...) sections for a Data Item.

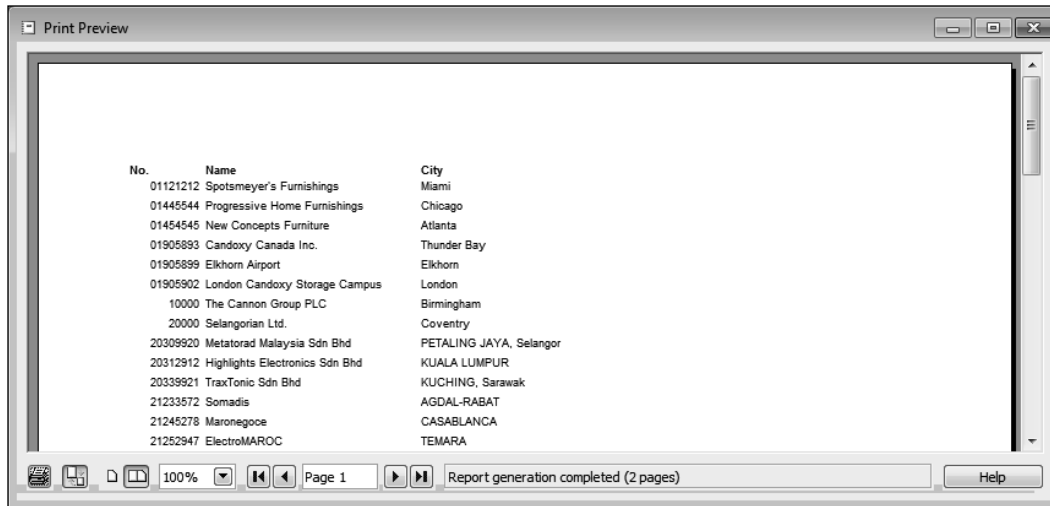
We will insert a header and footer section for our Customer Data Item as follows. Then, click on the **Field Menu** button on top of the screen. The **Field Menu** window will open. This will list all of the fields of the **Customer table**, our Data Item. In here, you select the fields that you want to add to the report using the mouse in combination with the *Ctrl* or *Shift* key. Then, click two times inside the **body** section, like in the following screenshot:



The selected fields will now be added to the body section. Every field will have a label and a textbox. Drag and drop the labels onto the header section, as in the following screenshot:



To run the current report, type *Ctrl-R*, or click on **File** and then on **Run**. The report viewer will open. After you select **Preview** in the **Request Form** the following Customer List will display:



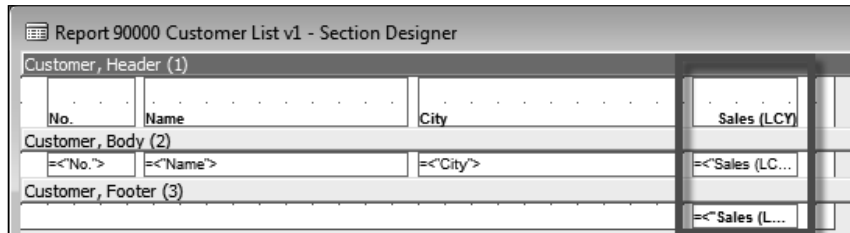
Let's now add something in the footer section, for example a numerical field like Sales (LCY), to see a total of what has been sold to the customers in the list. Make sure you add the Sales (LCY) field to the body and the footer sections and put the label on the header section as in the following screenshot:

To do this, you can follow these steps:

1. Open the field menu.
2. Find the Sales (LCY) field and click on it in the list.
3. Click in the body section; this will add the label and textbox.
4. Move the Label to the header section.
5. Select the Sales (LCY) textbox in the body.
6. Type: *Ctrl-C* and *Ctrl-V* to do a copy/paste of the textbox. You now have two textboxes containing the Sales (LCY) in the body.
7. Copy and paste a Sales (LCY) textbox from the header to the footer section.
8. Select the label in the header and make it bold.

You can do this with the **Aa** button in the menu bar. When you click on the **Aa** button, a popup window opens that you can use to format a textbox on the report.

After following these steps, the report will look like the next screenshot:



Now, run the report again to see the result. Scroll towards the last page of the report and have a look at the total. It's not correct. The total is the same as the value of the last Sales (LCY) in the body Section. Why is that?

Well, this is because you have to explain to Dynamics NAV for which fields you want to calculate totals. You can do this in the data item properties by using the property: `TotalFields`.

You can open the properties window via the button in the menu at the top of the screen, or via **Menu | View | Properties**, or via the shortcut *F4*.

When you enter the field Sales (LCY) in this property, Dynamics NAV will calculate totals for this field. When this field is then added to a footer section, the calculated total will be shown. Instead of using the `TotalFields` property, you could also obtain the same effect via C/AL code using the function `CreateTotals()`. This C/AL code should go into the `OnPreReport` trigger of the report.

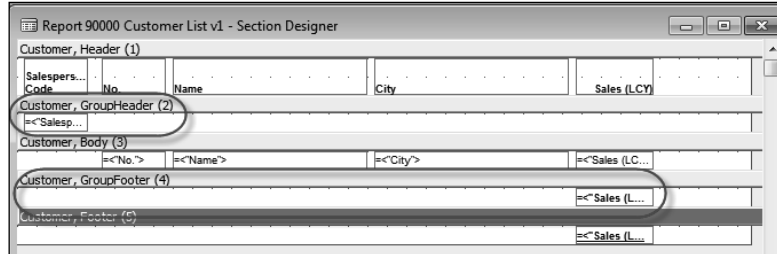
No.	Name	City	Sales (LCY)
01445544	Progressive Home Furnishings	Chicago	1.499,02
10000	The Cannon Group PLC	Birmingham	17.100,98
20000	Selangorian Ltd.	Coventry	6.510,84
30000	John Haddock Insurance Co.	Manchester	6.142,90
32858585	Antarcticopy	Antwerpen	2.582,81
35451238	Gagn & Gaman	Hafnafjordur	877,32
35983852	Heimilispyrdi	Reykjavik	2.024,21
40000	Deerfield Graphics Company	Gloucester	1.083,10
42147258	BYT-KOMPLET s.r.o.	Bojkovice	1.602,90
43687129	Designstudio Gmunden	Gmunden	2.498,10
46897889	Englunds Kontorsmöbler AB	Norrköping	673,71
47583218	Klubben	Haslum	11.772,20
49633863	Autohaus Mielberg KG	Hamburg 38	281,40
50000	Guildford Water Department	Guildford	533,40
			<b>55.162,67</b>

## Sorting and grouping data in a report

Now, we are ready to group this report. Actually, we want to group the data according to salesperson. So for every salesperson we want to see his/her customers, with a subtotal of the total sales by salesperson and a grand total per customer. To do this you need to include a group section in the report. A group header section can be used to show the salesperson and a group footer section can then be used to display the subtotal(s):

1. Type *F3* to open the Insert New Section window.
2. Select the **GroupHeader** option.
3. Click on **Ok** to add the **GroupHeader** section to the report.
4. Repeat these steps to add a **GroupFooter** section to the report.
5. Now, use the Field Menu to add the salesperson field to the **GroupHeader** section.
6. Now, use the Field Menu to add the Sales (LCY) field to the **GroupFooter** section.
7. You can use the **Aa** button to underline the Sales (LCY) field in the **GroupFooter** and to make the salesperson bold in the **GroupHeader** section.

You should now have this result in the **Section Designer**:



And when you run the report, this is the layout:

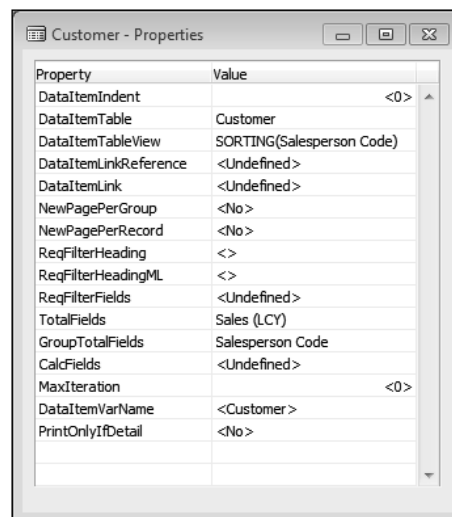
Salesperson Code	No.	Name	City	Sales (LCY)
	01445544	Progressive Home Furnishings	Chicago	1.499,02
	10000	The Cannon Group PLC	Birmingham	17.100,96
	20000	Selangorian Ltd.	Coventry	6.510,84
	30000	John Haddock Insurance Co.	Manchester	6.142,90
	32650565	Antarcticoopy	Antwerpen	2.582,81
	35451236	Gagn & Gaman	Hafnafjordur	877,32
	35903852	Heimilispyrdi	Reykjavik	2.024,21
	40000	Deerfield Graphics Company	Gloucester	1.063,10
	42147258	BYT-KOMPLET s.r.o.	Bojkovice	1.602,90
	43687129	Designstudio Gmunden	Gmunden	2.498,10
	46897889	Englunds Kontorsmöbler AB	Norrköping	673,71
	47563218	Klubben	Haslum	11.772,20
	49633863	Autohaus Mielberg KG	Hamburg 36	281,40
	50000	Guildford Water Department	Guildford	533,40
				<b>0,00</b>

But wait, this is not what we expected. There's no salesperson group or subtotal. Why is that?

Well, that's because you also have to explain to Dynamics NAV that a grouping needs to be calculated when the data item is executed.

You do this via the property: `GroupTotalFields`. In this property, you need to include the field on which we are grouping: `Salesperson Code`. It is a property of the data item. To change it, click on the data item and then on `F4`.

When you do this and then run the report again, you will still see no difference. Well, that's because Dynamics NAV is only capable of calculating group totals when the key that is used to sort the data item contains the field on which you are grouping. So you also need to include the salesperson in the `Key` property of the `DataItemView` property. And if there's no key in the table, then you will need to create one. The following is what the property list of the data item should look like:





And then, you get this result when you run the report:

Salesperson Code	No.	Name	City	Sales (LCY)
JR				
	01445544	Progressive Home Furnishings	Chicago	1.499,02
	32856565	Antarcticopy	Antwerpen	2.582,81
	35451236	Gagn & Gaman	Hafnafjordur	877,32
	35963852	Heimilispyrdi	Reykjavik	2.024,21
	42147258	BYT-KOMPLET s.r.o.	Bojkovice	1.802,90
	43887129	Designstudio Gmunden	Gmunden	2.498,10
	48897889	Englunds Kontorsmöbler AB	Norrköping	673,71
	47563218	Klubben	Haslum	11.772,20
	49633663	Autohaus Mielberg KG	Hamburg 36	281,40
				<b>23.811,67</b>
PS				
	10000	The Cannon Group PLC	Birmingham	17.100,96
	20000	Selangorian Ltd.	Coventry	6.510,64
	30000	John Haddock Insurance Co.	Manchester	6.142,90
	40000	Deerfield Graphics Company	Gloucester	1.063,10
	50000	Guildford Water Department	Guildford	533,40
				<b>31.351,00</b>
				<b>55.162,67</b>

What is that, a **Key**?

Well, a **Key** is defined in a table. Basically, in Dynamics NAV there are two types of keys:

- A primary key
- Secondary key(s)

When you open a table in design, by clicking on the **Design** button in the object designer after selecting the table objects, you can open the key window via the Menu: **View | Keys**. In this window, you can see and define the keys for the table.

The very first key in this list of keys will always become the primary key of the table (most of the time abbreviated as **Pk**). The other keys in the list become secondary keys.

A primary key defines what makes a record unique. For example in the customer table the primary key is "No.". This means that every record in that table, in other words every customer, must have a unique "No.".



### Unique keys

In fact in Dynamics NAV all keys are unique, including secondary keys. This is because when you create a secondary key, the fields from the primary key are automatically added at the end of all secondary keys. Because the Pk is unique, the secondary keys also become unique. You cannot see this in the key window in the Classic Client, but if you open the database design in the SQL Server Management Studio, then you can see this.

Keys are used for mostly three purposes in Dynamics NAV:

- Sorting
- Finding records faster
- Sum Index Fields

When you open a form or a page in Dynamics NAV and you want to change the sorting order of the data displayed on screen, then you click on a sort button and a popup window opens that shows a list of keys from the underlying table. The key you select will then determine the sort order.

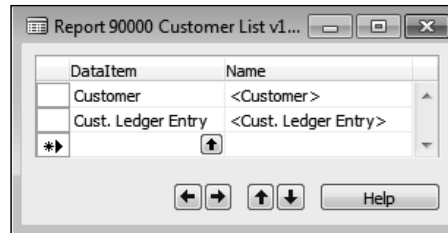
For example, if you want to be able to sort the customer list page by city, then you need to select a key that contains (or begins with) the city field. If there is no such key available, then it will have to be created in the table.

When you fetch data from a table, via C/AL code or via the user interface, selecting a good key will result in faster search results. For example, imagine that you have a big encyclopedia containing thousands of pages of information about painters and you are looking for the information of all painters living in your city. If you do not have an index in city in your encyclopedia, then the only way to find the information you request is to read every single page.

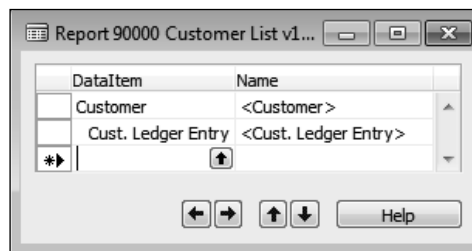
Having a good index results in faster results and less scanning of tables. This is of course also the case in reports. I should say that especially in reports it is very important to select good keys because it can make the difference between a very slow reports causing lots of frustration to the end users, and a very fast report everyone is pleased with.

Keys are also used in reports for grouping purposes. When you want to apply sorting or grouping in a report, you will need to select the appropriate key in the data item, as described in the section here above.

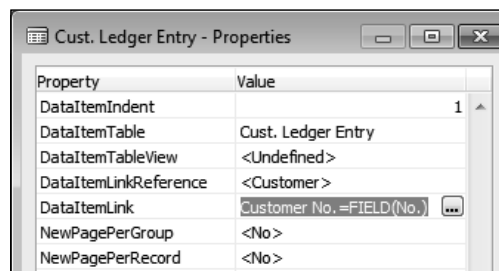
As you can imagine, when using multiple **data items**, reports get more interesting. For example, you might want to include information about the Customer Ledger Entries on the report. To do this, you can create a second data item below the **Customer** data item, like in the following screenshot:



When two data items are below each other in the **data item designer**, at runtime Dynamics NAV will first process all the records from the first data item (Customer) and then the second data item (Customer Ledger Entry) and so on. In some reports this is exactly what is required. But in this report we want the Customer Ledger Entries by customer. To accomplish this we will need to reflect this in the DataItem designer by indenting the second data item one level to the right. You can do this by clicking on the arrow → at the bottom of the data item designer:

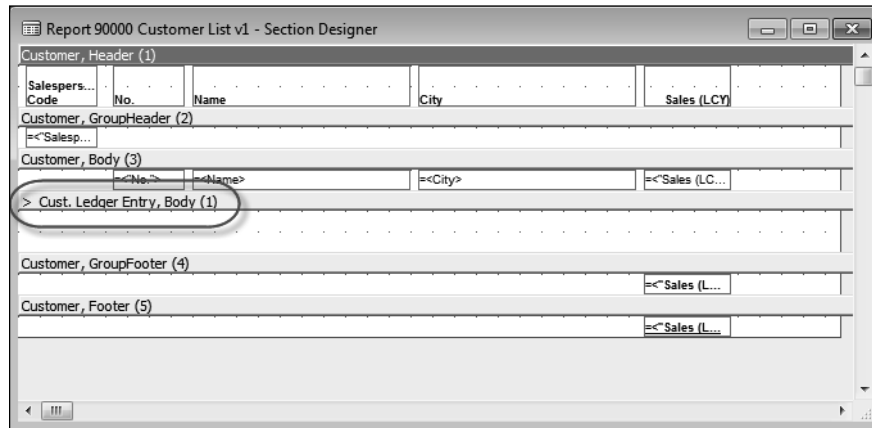


The next thing we need to do is explain to Dynamics NAV how to connect the two tables. This can be achieved via the `DataItemLink` property on the indented data item:

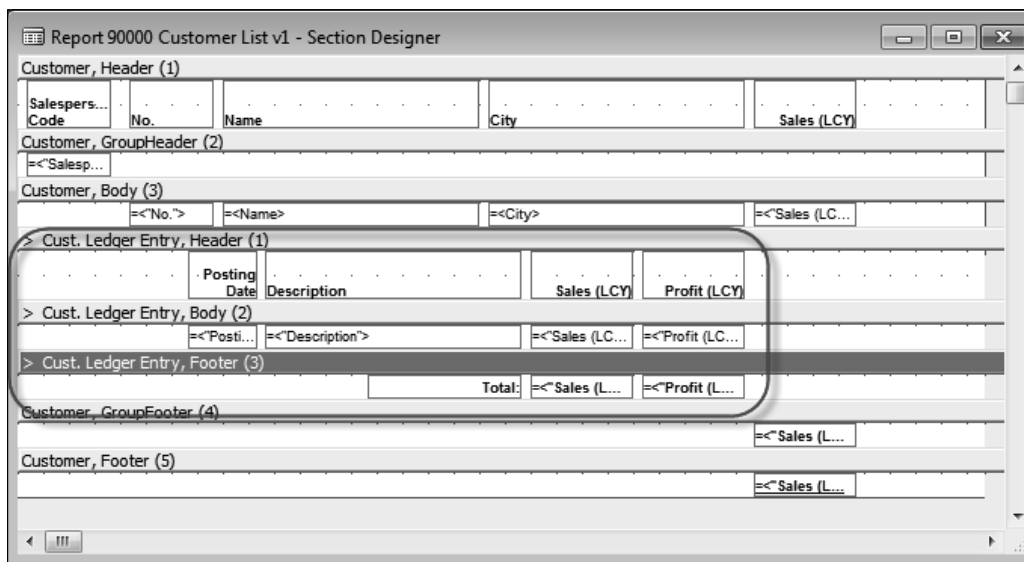


If there are multiple fields that connect the two data items, then you can enter multiple connections on the `DataItemLink` property.

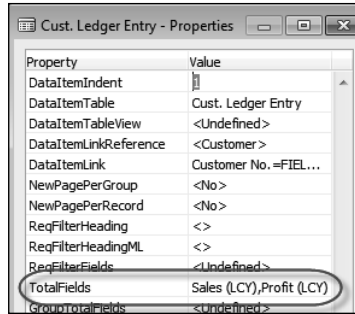
If you have a look now at the **Sections** of this report you will see this:



As you can see, there's now also a section for the **Customer Ledger Entry** data item, and the > sign indicates the level of indentation. If required, you can use the *F3* shortcut to add extra sections for the Customer Ledger Entry data item, like for example a Header and Footer. You can now put the fields from the Customer Ledger Entry Section(s), like in the following screenshot:



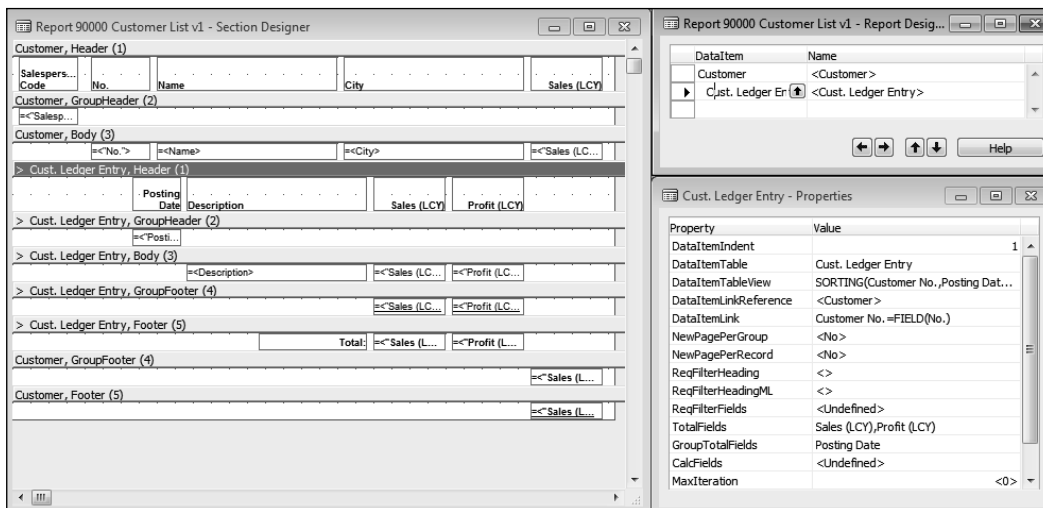
The same rules apply to the indented data item(s) as for the first data item regarding Totalling and/or Grouping. In this example, this means that for the Totals for Sales (LCY) and Profit (LCY) to be calculated, the `TotalFields` property needs to be filled in as in the following screenshot:



And if you would like to group the Customer Ledger Entries by Posting Date you will need to:

- Put the Posting Date in the `GroupTotalFields` property of the Customer Ledger entry data item
- Add a `GroupHeader` and/or `GroupFooter` section to the layout of the report to display the Posting Date and/or Group Totals
- Select a Key in the `DataItemTableView` property of the Customer Ledger entry data item

This is shown in the following screenshot:



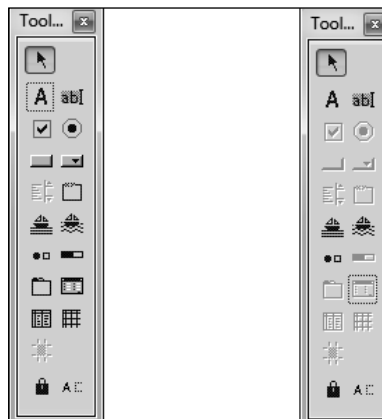
**For More Information:**  
[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)

After which you will get this result when you run the report:

Posting Date	Description	Sales (LCY)	Profit (LCY)
31/12/11	Opening Entries, Customers	0,00	0,00
	Opening Entries, Customers	0,00	0,00
	Opening Entries, Customers	0,00	0,00
	Opening Entries, Customers	0,00	0,00
	Opening Entries, Customers	0,00	0,00
	Opening Entries, Customers	0,00	0,00
		<u>0,00</u>	<u>0,00</u>
08/01/12	Order 101001	6.615,23	1.184,33
		<u>6.615,23</u>	<u>1.184,33</u>
15/01/12	Credit Memo 104001	-234,27	-42,07
	Payment 2012	0,00	0,00
	Payment 2012	0,00	0,00
	Payment 2012	0,00	0,00
		<u>-234,27</u>	<u>-42,07</u>
18/01/12	Order 6005	3.281,50	1.385,00
		<u>3.281,50</u>	<u>1.385,00</u>
23/01/12	Invoice 103001	7.438,50	2.813,00
		<u>7.438,50</u>	<u>2.813,00</u>
	<b>Total:</b>	<b>17.100,96</b>	<b>5.340,26</b>

What we have now is a simple report with two indented data items, groupings, totals, and subtotals.

To improve the layout of the report, you could now add some formatting, company logo, company information, labels, rectangles, and so on. To do this, you have a toolbox at your disposal. It's the same toolbox that is used when designing a Form, but not all of the controls are enabled for Reports, as you can see in the next screenshot:



An example of a report where you can see the usage of the Line and Rectangle controls is report 752 Work Order. When you run the report, this is what it looks like:

<b>Work Order</b> CRONUS International Ltd.		14. February 2011 Page 8 Steven				
The Cannon Group PLC Mr. Andy Teal 192 Market Square Birmingham, B27 4KT Great Britain		<b>Sales Order No.</b>	<b>101016</b>			
		Shipment Date	26/01/12			
<b>Quantity used during work (Posted with the Sales Order)</b>						
Type	No.	Description	Quantity	Unit of Measure	Quantity Used	Unit of Measure
Item	1920-S	ANTWERP Conference Table	1	Piece		
<b>Comments</b>						
Code	Comment			Date		
	dfgbb			26/01/12		
<b>Extra Item/Resource used during work (Posted with Item or Resource Journals)</b>						
Type	No.	Description	Quantity	Unit of Measure	Date	


As you can see, there's a header section containing general information like the name of the report, the company, date, page number, username, and so on. Then, you can see three tables containing information about Sales Orders. The actual content of the report is not our interest right now; let's focus on how it is designed.

More information about how to create the Excel look and feel of this report can be found in this chapter in the section: *How can you create an Excel-like layout for a report?*

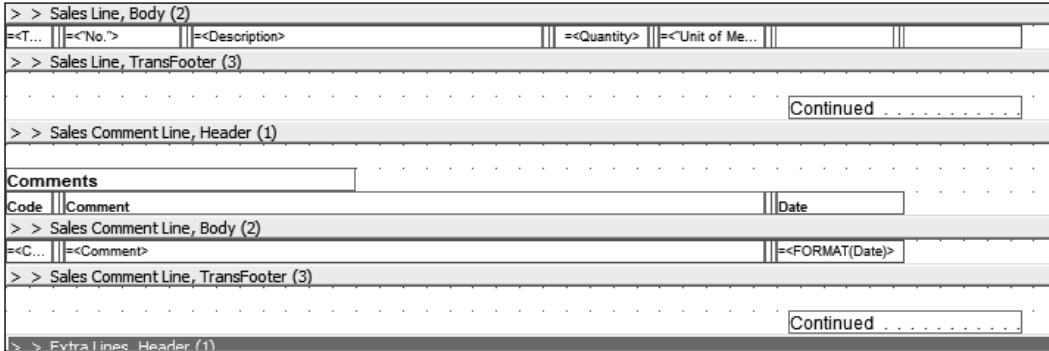
## Sections in a classic report

You might have noticed in the Work Order report, in the sections, the occurrence of a TransFooter section for several data items. The purpose of adding a TransHeader or TransFooter section for a data item is that when, during the execution of the report, a page break occurs during the data item loop, the TransHeader/TransFooter sections are printed. If there's no page break during the processing of the data item, then the TransFooter and/or TransHeader sections will not be printed.

**For More Information:**  
[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)

 **Running totals**  
 Another common usage for TransHeaders and TransFooters is to show running totals.

This is a screenshot of the **TransFooter** section(s) for the Work Order report:



And this is the result at runtime:

Item	1000	Bicycle		Piece		
Item	1000	Bicycle		Piece		
Item	1000	Bicycle		Piece		
Item	1000	Bicycle		Piece		
Item	1000	Bicycle		Piece		
Item	1000	Bicycle		Piece		
Item	1000	Bicycle		Piece		
Item	1000	Bicycle		Piece		

Continued. ....

**For More Information:**  
[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)



To summarize, the following table lists the different report sections and their purpose:

Section Type	Description
Body	Prints for each iteration of the data item loop. When there is an indented data item, the complete loop for this data item begins after the body section of the higher level data item has been printed.
Header	Prints before a data item loop begins. If the <code>PrintOnEveryPage</code> property of the section is set to Yes, the header is also printed on each new page.
Footer	Prints after the loop has ended. If the <code>PrintOnEveryPage</code> property of the section is set to Yes, the footer is also printed on each new page. If the <code>PlaceInBottom</code> property of the section is set to Yes, the footer section is printed at the bottom of the page, even if the data item loop ends in the middle of a page.
GroupHeader	Beginning of a new group.
GroupFooter	Ending of a group.
TransHeader	Prints if a page break occurs during a data item loop; the header is printed at the top of the new page. This section is printed after any header section of the data item.
TransFooter	Prints if a page break occurs during a data item loop, the header is printed before the page break. This section is printed before any footer section of the data item.

## Controls

The information that is printed in the sections is made of controls. The available controls are as follows:

- **Text boxes** – These are used for printing the results of the evaluation of any valid C/AL expression, such as the contents of a table field. They are also used for printing the results of complex calculations.
- **Labels** – These are used for printing static text, such as a caption for a column of data.
- **Shapes, images, and picture boxes** – These are used for printing bitmap pictures and graphical elements, such as lines and circles.

Controls are also available in the **request form** and the **request page**.

## Triggers

In reports, triggers are typically used to perform calculations and to control whether or not to output sections. This depends, for example, on the value in a field, or a choice the user made in the request form. But the most important point about triggers is that they allow you to control how data is selected and retrieved in a more complex and effective way than you can achieve by using properties.

Reports can contain the following types of triggers:

- Report triggers
- DataItem triggers
- Section triggers
- Control triggers (Request Form/Page)

Report triggers apply to the report itself:

Trigger	When is it executed?
OnInitReport	When the report is loaded.
OnPreReport	Before the report is run but after the request form has been run.
OnPostReport	After the report has run, but not if the report was stopped manually or by the Break function.
OnCreateHyperlink	After the user creates a URL to a form or a report.
OnHyperlink	After the OnInitReport trigger is executed for a report. The trigger executes a URL string.

DataItem triggers apply to each data item of the report:

Trigger	When is it executed?
OnPreDataItem	Before the data item is processed, but after the associated variable has been initialized.
OnAfterGetRecord	When a record has been retrieved from the table.
OnPostDataItem	When the data item has been iterated for the last time.

Section triggers apply to each of the sections of a data item:

Trigger	When is it executed?
OnPreSection	Before processing a section.
OnPostSection	After processing a section but before printing it.

## What happens when a report runs?

When you run any report, the `OnInitReport` trigger is called first. This trigger performs any processing that is necessary before the report is run.

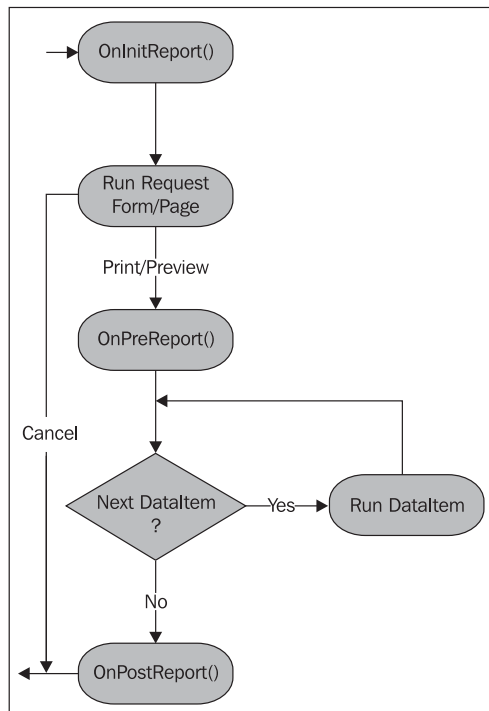
Next, the request form for the report is run if it is defined. Here, you select the options that you want for this report.

If you decide to continue, the `OnPreReport` trigger is called. At this point, no data has yet been processed. You can use this trigger for example to initialize variables or fetch information from the database via C/AL code. In this trigger, usually the Company Information table is queried to retrieve company information like the name, vat number, company logo, and so on.

When the `OnPreReport` trigger has been executed, the first data item is processed. When the first data item has been processed, the next data item, if there is any, is processed in the same way.

When there are no more data items, the `OnPostReport` trigger is called to do any necessary post processing.

The following is a visual representation of the Report Execution Flow:



## How is a data item processed?

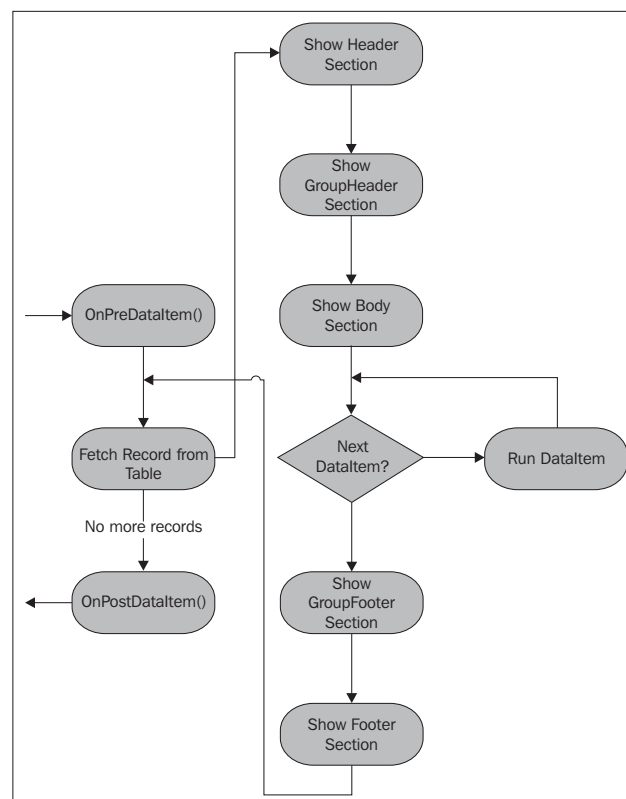
Before the first record is retrieved, the `OnPreDataItem` trigger is called, and after the last record has been processed, the `OnPostDataItem` trigger is called.

Between these two triggers, the data item records are processed. Processing a record means executing the record triggers and outputting sections. C/SIDE also determines whether the current record should cause the outputting of a special section, such as a header, footer, group header, or group footer.

If there is an indented data item, a data item run is initiated for this data item. These are processed for each record in the parent data item. (Data items can be nested 10 levels deep.)

When there are no more records to be processed in a data item, control returns to the point from which processing was initiated. For an indented data item, this means the next record of the data item on the next highest level. If the data item is already on the highest level indentation, control returns to the report.

The following is a visual representation of the DataItem Execution Flow:



Understanding the flow of the report triggers and data item triggers is crucial to decide where to put C/AL code. The idea should always be not to execute C/AL code when it is not necessary. For example, if you can choose between the `OnAfterGetRecord` trigger or the `OnPreDataItem` trigger you should choose the `OnPreDataItem` trigger. This is because the `OnAfterGetRecord` trigger will execute for every single record that is retrieved from the database.

## Properties in a report

Every object in a report has properties, including:

- The report itself
- Data items
- Sections
- Controls in the section
- Request forms
- Controls on a request form



### How to see the properties

To see the properties of a textbox or data item, or section, you have to first select it. Make sure it is selected by clicking on it with your mouse. Then, you can click on the properties button at the top of the screen, or press *Shift* and the *F4* function key. Now, the property window opens and displays the appropriate properties.

The following table briefly describes the report properties. The *C/SIDE Reference Guide Online Help* contains more detailed information about these properties and you can get context-sensitive Help for a property by opening the Properties window for the report, selecting the property in question, and pressing *F1*.

Property	Description
ID	ID of the report. Must be unique among reports.
Name	Name of the report.
Caption	Caption (shown on the Request Form window, for example). The default is the same as Name.
CaptionML	The translation of the caption.
ShowPrintStatus	Determines whether the printing status window should be displayed during printing (with the opportunity to cancel printing).

<b>Property</b>	<b>Description</b>
UseReqForm	Determines whether the request form should be run before the report.
UseSystemPrinter	If Yes, then the system default printer is used to print the report. If No, then the printer defined for the combination User/Report in the setup is used.
ProcessingOnly	If No, the report is a processing only report. If Yes, the report cannot have sections.
TransactionType	The behavior of a transaction takes effect from the beginning of a transaction. There are four basic transaction type options: Browse, Snapshot, UpdateNoLocks, and Update. There is a Report option that maps to one of the basic options and enables a report to use the most concurrent read-only form of data access.  When you use Classic Database Server, it maps to Snapshot. When you use SQL Server, it maps to Browse.
Description	For internal purposes.
TopMargin	Top margin in 1/100 mm.
BottomMargin	Bottom margin in 1/100 mm.
LeftMargin	Left margin in 1/100 mm.
RightMargin	Right margin in 1/100 mm.
HorzGrid	Distance between horizontal gridlines (1/100 mm).
VertGrid	Distance between vertical gridlines (1/100 mm).
Permissions	The permissions of the report to access database objects. (The report can have wider permissions than the individual user, thereby enabling the user to print reports that retrieve information from tables which normally cannot be accessed.)
Orientation	Sets the page orientation for the report, Portrait or Landscape.
PaperSize	Sets the paper size for the report.
PaperSourceFirstPage	Specifies the paper source to use for printing the first page of the report. This is useful if the report has a cover.
PaperSourceOtherPage	Specifies the paper source to use for the remaining the pages in the report.
DeviceFontName	Use this property for reports that are designed specifically for dot matrix printers, to prevent the printer from switching into graphics mode when printing text. Specify the name of a device font (a font that is built into a printer).

The next table briefly describes the data item properties.

Property	Description
DataItemIndent	The indentation level, which can be set in the designer when creating data items.
DataItemTable	The table that the data item is based on, which can be set in the designer when creating data items.
DataItemTableView	The key, sort order, and filters to apply.
DataItemLinkReference	The DataItemVarName of a less-indented data item that this data item will be linked to.
DataItemLink	Link between the current data item and the data item specified by DataItemLinkReference.
NewPagePerGroup	Determines whether each group should be printed on a separate page.
NewPagePerRecord	Determines whether each record should be printed on a separate page.
ReqFilterHeading	The caption for the request form tab that relates to this data item. The default is the name of the table that the data item is based on.
ReqFilterHeadingML	Translation of the caption for the request form tab.
ReqFilterFields	Name of the fields that will be included in the request filter form.
TotalFields	Name of the fields for which totals will be calculated.
GroupTotalFields	Name of the fields that will be used for grouping data.
CalcFields	Name of the fields that will be calculated after a record has been retrieved.
MaxIteration	Maximum number of data item loop iterations.
DataItemVarName	Name of record as a variable. The default is the name of table that the data item is based on.
PrintOnlyIfDetail	Print item only if sublevels generate output.

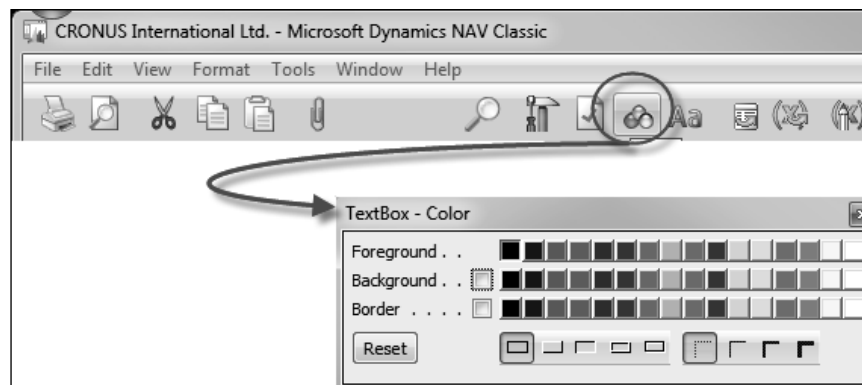
The next table briefly describes the section properties:

Property	Description
PrintsOnEveryPage	Determines whether the header and footers should be printed on every page.
PlaceInBottom	Determines whether the footer should be placed below the last line or at the bottom of the page.
SectionWidth	Width in 1/100 mm.
SectionHeight	Height in 1/100 mm.

Controls in reports have the same properties as controls on forms. The Properties window of a control shows the properties, and they are described in the C/SIDE Reference Guide Online Help.

## Adding color to a classic report

When you are designing a report, in the menu bar there's a button to assign colours to a control, as you can see in the following screenshot:



In the designer the object for which you set a color will get the selected color, but when you run the report the color is not shown. Although colors are visible for controls that are set in the **Section Designer**, these colors do not print when the report is run. The only way to print colors is to use **Image** or **PictureBox** controls.

When you create an RDLC layout for a report and run it in the Role Tailored Client, then colors will be visible. It is only one advantage of an RTC report over a Classic report, but it is sometimes considered as a good enough reason to switch from a Classic report to an RTC report.

## What is a ProcessingOnly report?

A processing-only report is a report that does not print but instead only processes data or C/AL code. Processing table data is not limited to processing-only reports. Reports that print can also change records. This section applies to those reports as well.

It is possible to specify a report to be "Processing Only" by changing the `ProcessingOnly` property of the Report object. The report functions as it is supposed to (processing data items), but it does not generate any printed output.



When the `ProcessingOnly` property is set, the request form/page for the report changes slightly, as the **Print** and **Preview** buttons are replaced with an **OK** button. The **Cancel** and **Help** buttons remain unchanged.

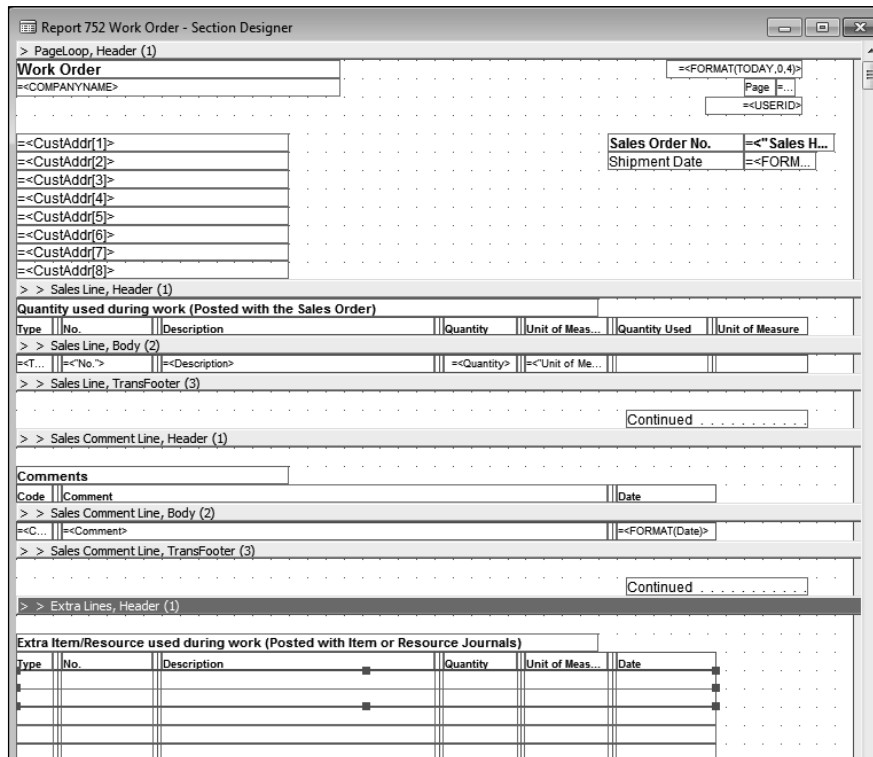
When the `ProcessingOnly` property is set you also cannot create any sections.

There are advantages to using a report to process data rather than a code unit:

- The request page functionality that allows the user to select options and filters for data items is readily available in a report, but difficult to program in a code unit
- Using the features of Report Designer ensures consistency
- Instead of writing code to open tables and to retrieve records, report data items can be used

## Creating an Excel-like layout for a report

This is how the sections look like when you take the Work Order report into design mode:

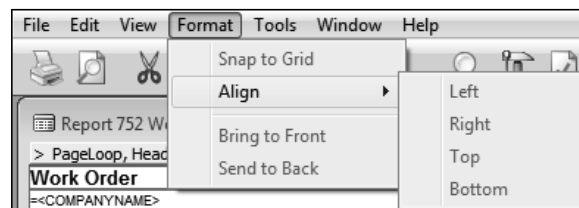


When you look a little closer and click on one of the cells, you will notice that when you have a look at its properties, they are the properties of a **Shape** control. When you select a **Shape** control in the toolbox and put it on a section you can decide via the property `ShapeStyle` which type of **Shape** you are adding.

These are the possible values for the `ShapeStyle` property of a Shape control:

- Rectangle
- Rounded Rectangle
- Oval
- Triangle
- NWLine
- NELine
- HorzLine
- VertLine

You have to put the **Shape** control on top of the other controls (labels, textboxes) that contain the actual data. To do this you can select a Shape and bring it to front or send it to back using the **Format** menu on top of the report designer like this:



### Overlapping controls



As you can see, putting two controls on top of each other can provide an added value. Another example of when this technique is applied is when you want to give a **textbox** a **backgroundcolour**. We already know that colours are never printed, but when you use a **Picturebox** or **Image** control you can overcome this issue. That is why in some reports a **Picturebox** control or an **Image** control is put behind a **Textbox** or **Label**.

This menu can also be used to align multiple controls to the Left, Right, Top, or Bottom.

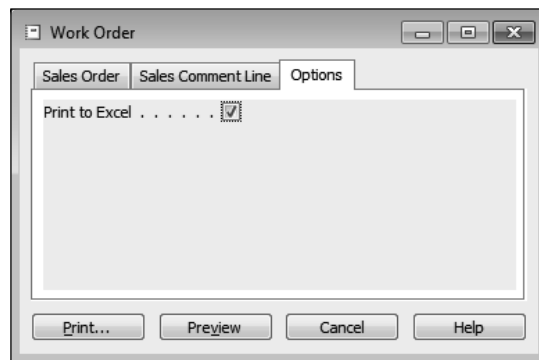
So by using the **Shape** Control, you can draw the rectangles or lines that will make up the Excel look and feel of the report. But there are a few disadvantages:

- When you put a Shape on top of another control, you can no longer select the control unless you move or resize the Shape control
- To be able to produce an Excel look and feel for a report, it will take a long time to draw the Shapes and put them in the correct position
- Anytime you need to make a change to this kind of report, you will need to take time into account to check whether the Shapes have not moved so the layout at runtime still resembles the Excel look and feel

Actually, when someone asks me to create this kind of report with an Excel look and feel I would estimate it taking about twice the amount of time than without the Excel look and feel.

Also, when this kind of report that uses a lot of **Shape** controls is exported to HTML, it will usually cause unwanted effects and sometimes even an unusable layout.

When you run this report and the **Request Form** opens, there's also an **Options** tab in there. In here you can see an option box – **Print to Excel**:



Let's have a look now at how this can be achieved.

## Printing a report to Excel

Behind the **PrintToExcel** option box on the **Request Form** there's a Boolean variable **PrintToExcel**, which will be true if the user selects the option and false otherwise.

If you have a look at the Report triggers, you will see the following code in the **OnPreReport** and **OnPostReport** trigger:

```

Report - OnPreReport()
IF PrintToExcel THEN
  ExcelBuf.CreateBook;

Report - OnPostReport()
IF PrintToExcel THEN BEGIN
  ExcelBuf.CreateSheet(Text000,Text000,COMPANYNAME,USERID);

  IF NOT "Sales Header".ISEMPTY THEN BEGIN
    IF NOT "Sales Line".ISEMPTY THEN
      ExcelBuf.AutoFit('WorkOrderLineRange');
    IF NOT "Sales Line".ISEMPTY THEN
      ExcelBuf.BorderAround('WorkOrderLineRange');
    IF NOT "Sales Comment Line".ISEMPTY THEN
      ExcelBuf.BorderAround('WorkOrderCommentLineRange');
    IF NOT "Sales Line".ISEMPTY THEN
      ExcelBuf.BorderAround('WorkOrderExtraLineRange');
  END;

  ExcelBuf.GiveUserControl;

  ERROR('');
END;

```

To be able to send information into Microsoft Office Excel, a variable `ExcelBuf` is used. This is a record variable that points towards the **Excel Buffer table**. When you run the Excel Buffer table in the object designer, you will see that the table is empty. That is because it is always used as a temporary table. In the Work Order report, you will see in the properties of the `ExcelBuf` variable that **Temporary** has been put set to Yes.

## What is so special about the Excel Buffer table (370)?

These are the fields of the Excel Buffer table:

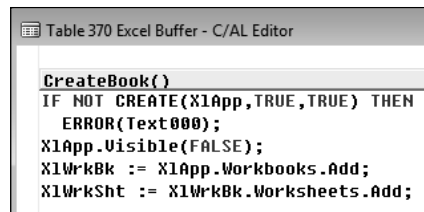
E..	Field No.	Field Name	Data Type	Length	Description
▶	1	Row No.	Integer		
✓	2	xlRowID	Text	10	
✓	3	Column No.	Integer		
✓	4	xlColID	Text	10	
✓	5	Cell Value as Text	Text	250	
✓	6	Comment	Text	250	
✓	7	Formula	Text	250	
✓	8	Bold	Boolean		
✓	9	Italic	Boolean		
✓	10	Underline	Boolean		
✓	11	NumberFormat	Text	30	
✓	12	Formula2	Text	250	
✓	13	Formula3	Text	250	
✓	14	Formula4	Text	250	

But the fields are not the most interesting part of the Excel Buffer table. Let's have a look at the C/AL functions of the table. This is the list of some of the functions of this table:

- Name
- CreateBook
- OpenBook
- CreateSheet
- ReadSheet
- FilterToFormula
- SumIf
- AddToFormula
- NewRow
- AddColumn
- AutoFit
- BorderAround

The Excel Buffer table contains no data, some fields, and a lot of functions. These functions can be used to work with Microsoft Excel. For example, the function `CreateBook` will create a new empty Excel workbook; the function `CreateSheet` will create an Excel sheet in the current workbook, and so on...

How is this possible? Well, let's have a closer look at one of the functions of the Excel Buffer table, for example the `CreateBook` function:



```
Table 370 Excel Buffer - C/AL Editor
CreateBook()
IF NOT CREATE(XlApp,TRUE,TRUE) THEN
  ERROR(Text000);
XlApp.Visible(FALSE);
XlWrkBk := XlApp.Workbooks.Add;
XlWrkSht := XlWrkBk.Worksheets.Add;
```

This function makes use of variables like `xlApp`, `xlWrkBk`, `xlWrkSht`, and so on...

Name	DataType	Subtype	Length
InfoExcelBuf	Record	Excel Buffer	
XLApp	Automation	Unknown Automation Server.Application	
XLWrkBk	Automation	Unknown Automation Server.Workbook	
XLWrkSht	Automation	Unknown Automation Server.Worksheet	
XLWrkshs	Automation	Unknown Automation Server._Worksheet	
XLRange	Automation	Unknown Automation Server.Range	
FormulaUnitErr	Text		250
RangeStartXlRow	Text		30
RangeStartXlCol	Text		30
RangeEndXlRow	Text		30
RangeEndXlCol	Text		30
CurrentRow	Integer		
CurrentCol	Integer		
UseInfoSheed	Boolean		

As you can see in the screenshot, these variables are of type `Automation`.

#### What is Automation?



Automation is a client/server infrastructure that allows one application to access and communicate with another application. With Automation, an application, such as Microsoft Office Word, exposes its internal functions and routines as Automation objects that Microsoft Dynamics NAV 2009 can access through an Automation controller in C/SIDE. The application that exposes the Automation object, such as Word, acts as the Automation server, and the C/SIDE Automation controller acts as the client.

So basically, by using an automation variable we can control Excel and use the Microsoft Excel Object Library to do it, and do this from within Dynamics NAV.

Because there might be multiple reports, or other objects, that need to be able to export/print information towards into Excel, the functions used to control the Excel automation server have been created in the Excel Buffer table. Now, all one needs to do is put some data in the temporary table and then use the 'built in' functions of the Excel Buffer table to send the data towards into Excel.

Of course, the real work is then in the report, to call these functions in the report and data item triggers, so that the data is sent.

The Excel Buffer table is an example on how you can use a table to centralize functions, much like a codeunit does, but with the advantage of also having fields to store information.

Now, you do not have to use the Excel Buffer table if you don't want to. Of course you could create automation variables in the report and write all of the automation code in the report.



**Warning**

Be careful if you make customizations or modifications on the design and/or C/AL code of the Excel Buffer table. This table is being used by a lot of reports and other objects in the Dynamics NAV application to manage Excel via automation. Changing the Excel Buffer table might result in changing the behaviour in all of these objects.

The idea here is not to go into detail but to give you an overview on what's involved when you want to export towards Microsoft Office applications from within a Dynamics NAV Classic report.

## Report functions

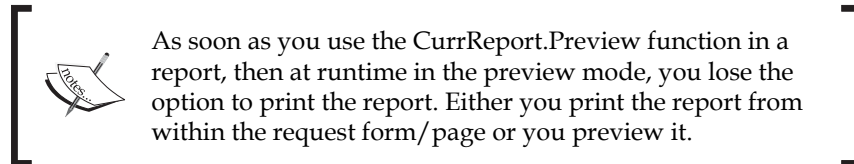
Certain functions can only be used in reports. These functions can be useful for complex reports:

- **CurrReport.SKIP**  
Use this function to skip the current record of the current data item. If a record is skipped, it is not included in totals and it is not printed. Skipping a record in a report is much slower than never reading it at all, so use filters as much as possible.  
A good trigger to use this function in is the `OnAfterGetRecord` trigger of a data item.
- **CurrReport.BREAK**  
Use this function to skip the rest of the processing of the data item currently being processed. The report resumes processing the next data item. All data items indented under the one that caused the break are also skipped.  
A good trigger to use this function in is the `OnAfterGetRecord` trigger of a data item.
- **CurrReport.QUIT**  
This function skips the rest of the report. It is not an error, however. It is a normal ending for a report. Also, the `OnAfterReport` trigger will not be executed if you use this function in the `OnAfterGetRecord` or `OnPreDataItem` trigger.

- `CurrReport.PREVIEW`

Use this function to determine whether a report is being printed in preview mode.

This function can be useful when you want to execute C/AL code when a report runs, but not when it is executed in preview mode. For example if you want to keep track of how many times a report has actually been printed on paper.



- `CurrReport.PAGENO`

Use this function to return the current page number of a report and/or to set a new page number.

- `CurrReport.CREATETOTALS`

Use this function to maintain totals for a variable in the same way as totals are maintained for fields by using the `TotalFields` property. This function must be used in the `OnPreDataItem` trigger of the data item in the sections where the totals are displayed.

- `CurrReport.TOTALSCAUSED`

Use this function to determine which field caused a break to occur. The return value is the field number of the field that the data item is grouped on that changed and caused a Group Header or GroupFooter section to print. This function is usually used in the `OnPreSection` trigger of Group Header and Group Footer sections. This function must always be used when grouping more than one field for a single data item.

- `CurrReport.NEWPAGE`

Use this function to force a page break when printing a report. This is usually found in the data item triggers.

- `CurrReport.SHOWOUTPUT`

Use this function to return the current setting of whether a section should be outputted or not, and to change this setting. This function should only be used in the `OnPreSection` trigger of a section. If `TRUE` is passed to the function, nothing changes and the section prints as normal. If `FALSE` is passed to the function, the section is not printed for this iteration of the data item.



## Summary

In this chapter, we have covered the basics on how to create a simple report. We've seen how to implement sorting, grouping, totalling, and how to indent data items. You also have an idea on what's involved to be able to print a report to Microsoft Excel. And when you want to make changes or customizations to these kinds of reports it can quickly become complicated.

Furthermore, you should also have an idea by now of the limitations of the Classic report designer. A good example is the lack of colors at runtime. Also, the steps that need to be performed to have an Excel look and feel can become very tedious. Regarding the interactivity features of Classic reports, those are limited to what you can do with the request form.

A lot, if not all, of the shortcomings or difficulties that are inherent to the Classic report designer are solved, more intuitive and more user and developer friendly in RDLC reports for the Role Tailored Client.

The idea of this chapter was to give you a good introduction to classic report design and a view on some of the difficulties.

In the next chapter, we will have a look at the steps involved in creating a report for the Role Tailored client and the advantages the new report designer and Reporting Services technology has to offer.

## Where to buy this book

You can buy Microsoft Dynamics NAV 2009: Professional Reporting from the Packt Publishing website: <http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book>

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our [shipping policy](#).

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.



[www.PacktPub.com](http://www.PacktPub.com)

**For More Information:**

[www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book](http://www.packtpub.com/microsoft-dynamics-nav-2009-for-professional-reporting/book)