

Portrait / Landscape

This is a quick way of setting the orientation of the report on the page. Another method is to go into the File > Page Setup menu item and change it on the Paper Size tab.

Pass Setting

There are two options here – One Pass or Two Pass. This is an advanced setting that beginner Clarity users shouldn't be too concerned with. Basic reports should run fine on either setting, although for large reports you may notice reports run more slowly when set to Two Pass. Without getting into too much detail here, Two Pass and this is as the name suggests – the computer goes through the data twice – doing all the calculations out on the first pass and then formatting and printing the report on the second pass. The one pass does it all in one go. You may find that in complex reports this setting needs to be taken into account to ensure, for example, that variables are not calculated twice (when set to Two Pass), or that report totals are not calculated correctly before generating (if set to One Pass).

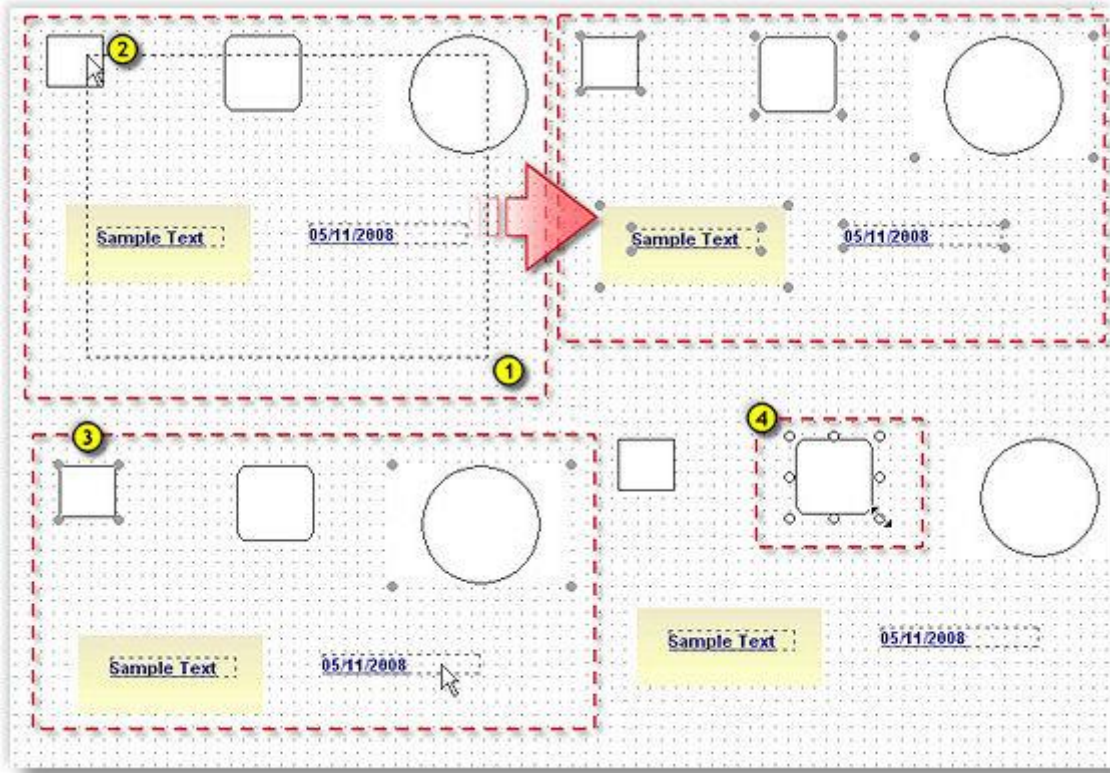
Units

A number of different units of measurement are available to select from. These apply to all reporting features (custom page size, position of labels etc.).

Selecting Components

There are numerous benefits to the correct selection of components in the design canvas. The overall benefit is time saving, reducing repetitive keystrokes and mouse manipulations. Other benefits are improved precision on the layout of the report, vital to a functional report.

There is a combination of methods for selecting components in the design canvas. This may sound straight forward, however to effectively use the design tools available, it's important to understand these different methods of selecting components.



Selecting Groups

The first method is to select a group of components, to do this:

- ① Place the mouse cursor at the lower right of the group.
- ② Click and hold the mouse cursor so that the dotted rectangle encompasses and crosses all items required for this selection.

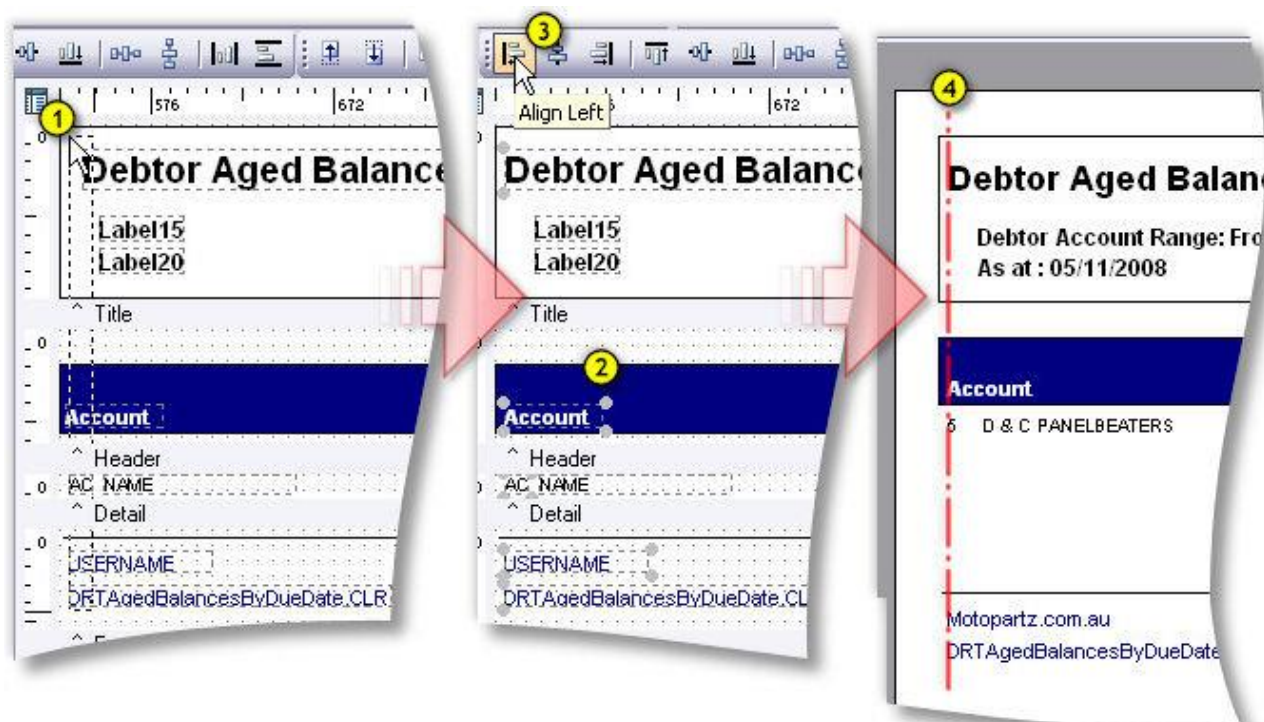
Release the mouse button and the selected items will be highlighted with grey grips.

Selecting groups is useful when you need to:

- Move a group of components around the canvas, after the selection, click on an item - hold the mouse button down and drag the selected group to a new location on the canvas. For precise movements of this selection, use the Nudge Toolbar (see page 18).
- Align (see page 20) the selected object to a specific location on the canvas. Using this selection method is not recommended for this as there is no control on which is the first item selected. A combination selection method (see page 31) is recommended for this option.
- Changing multiple attributes, such as font styles, line types and fills etc. Changing font size will work even though there are shape components combined in this selection. The shapes will be ignored and only the applicable attribute changes will be made to the relevant components in this selection.

- Make these components a uniform size using the size toolbar (see page 19). Using this selection method is not recommended for this as there is no control on which is the first item selected. A combination selection method (see page 31) is recommended for this option.

Note: Items can be selected across the **bands** (see page 2) using this method. See the illustration below:



1. Select the components across the bands.
2. De-select (see page 31) the rectangles.
3. Align the selected items left.
4. On preview, it is apparent that the selected items are precisely in line.

Individual Selections to Groups

To select components individually, hold down the shift key when selecting components with a single left click. This is illustrated in **3** top. You can do the same things as a group selection, however selecting individually gives you the added control over which components are to be modified.

Note: Selected items can be removed using the same method, i.e. you could select all components on the page **Ctrl + A** then remove the ones from the selection that aren't required by holding down shift and clicking them.

Combination of Both Selection Methods

You can either select individual items as above and while holding down the shift key, use the group selection method and vice versa. This method is recommended when using the Align (see page 20) and Size (see page 19) tools. These tools use the first object selected as a reference object and shift + selecting the first object, followed by shift group selection, gives you control as to which is the first object selected.

Selecting and Individual Component

When a single component is selected, the grips change appearance, **4** top. This indicates that you can manipulate the component using these grips, scale, width and height. The cursor will change to a double arrowed icon when on a grip, click, hold and drag to transform the selected component.

Creating your First Report

This simple overview provides an introduction to the MYOB EXO Clarity Report Designer basics, the Query Designer, and the process of building reports. The purpose of this overview is to show you what it takes to build a report and to get an overall idea of the concepts. If you feel a little lost, don't worry. The meaning behind the actions will become clearer as you progress through the topics.

Follow the steps given in this section to create a sample report contain the following items:

- A list of stock items
- A selling price for each item
- A Latest cost for each item
- A GP% for each item

Plan the Data Source

Before starting a report, take the time to plan the data source.

Which fields are required to create this report?

Item	Fields Required
A list of stock items	Stock_Item.Stockcode, Stock_Item.Description
A selling price for each item	Stock_Item.Sellprice1
A latest cost for each item	Stock_Item.Latestcost
A GP% for each item	No further fields required (GP% can be calculated by $(\text{Sellprice1} - \text{Latestcost}) / \text{Sellprice1} * 100$)

Which tables are needed to supply these fields?

All of the above fields can be found in one table: Stock_Items.

Start Clarity

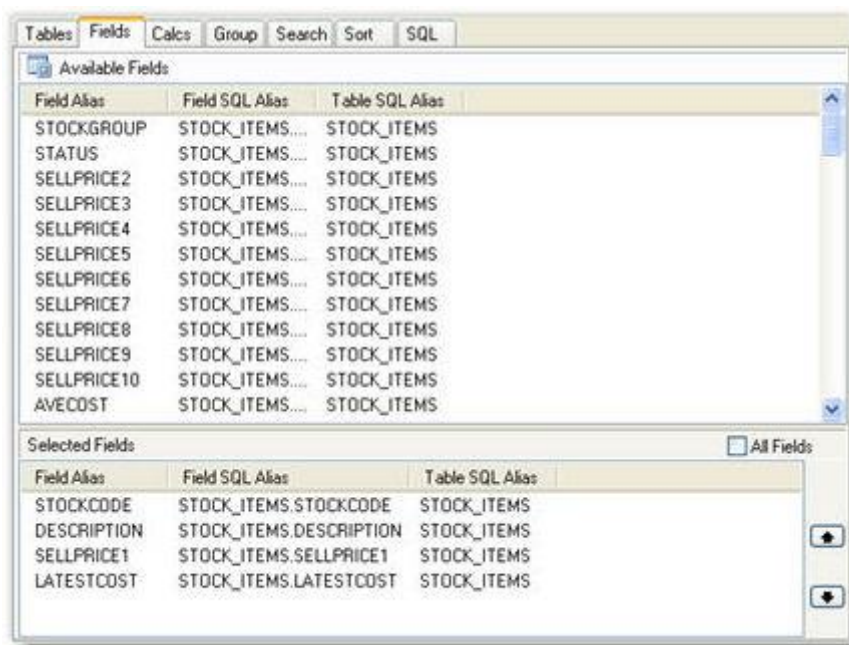
It will be helpful for you to become familiar with the following steps, because you will repeat this process to begin any report that you are writing from scratch (except FMT conversions).

Launch the Clarity Report Designer from the MYOB EXO Business Utilities menu.

1. Click on **File > New Report**, or the blank document icon on the main toolbar.
2. Click on the Data tab (see page 3).
3. Select **File > New** in order to access the New Items dialog.
4. Double-click on the Query Designer icon. The Query Designer will come up with a list of available tables.

In the Query Designer

1. Choose the Stock_Items table by double-clicking on it. This table should now appear in the list of Selected Tables.
2. We have finished selecting tables for now, so select the Fields tab (see page 8).
3. Note that all fields are preceded by the table name and a dot.
4. Select the fields as listed below for this example by double clicking on them.



5. Move to the Calculations tab (see page 10).
6. Double-click on the Latestcost field.
7. From the drop-down list box in the Function column, Change from Sum to Expression.
8. Enter the following calculation into the Expression edit box:

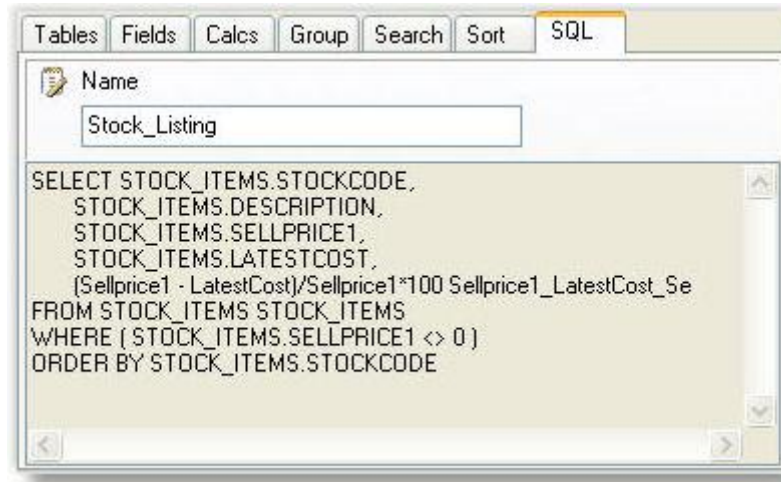
$$(\text{Sellprice1} - \text{LatestCost}) / \text{Sellprice1} * 100$$
9. Change the name of the Field Alias to GP%.

Field Alias	Field SQL Alias	Table SQL Alias	Function	Expression
GP%	(Sellprice1 - Late...	STOCK_ITEMS	Expression	(Sellprice1 - LatestCost)/Sellprice1*100

10. Move to the search tab (see page 11) and double click on Sellprice 1. In the box below specify that Sellprice1 <> 0. This is because there may be some zero selling price1's in the database and the computer is unable to divide by 0.
11. Move to the sort tab and select the Stock Code field.

MYOB EXO Clarity

12. Move to the SQL tab (see page 13) and change the name of the data source to Stock_Listing.



13. Click **OK**

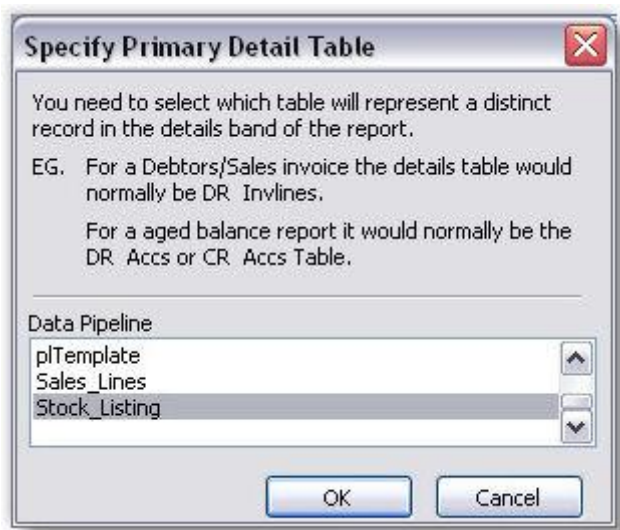
You've officially completed your first query via the Query Designer. You'll notice a new window in the upper left-hand corner of the data tab workspace. This is a data view of the query, and represents the data that will be selected from the database each time the report is generated.

14. Preview the data produced by this data source by using the page and magnifying glass icon:



We will now add another query to the same report and you will see that this additional query need not in fact be joined to our first query at all.

1. Click on **File > New**, and choose query designer.
2. Choose the General_Info table by double clicking on it. Note that you can type 'G' and the cursor will jump to General_Info in the list. General_Info contains the registered MYOB EXO Business users company information and tax registration details required on many reports.
3. Move to the Fields tab and select the Username only. Click **OK**.
4. You are back in the Data tab and General_Info appears as a dataview on your Query tab. Now let's begin laying out the report:
5. Click the Design tab.
6. Clarity will need to know which query on the Data view tab represents the Primary detail section of your report. This means it wants to know the main table on which you will be building the report. This is essential. In our case it is the first query we created via the wizard, called Stock_Listing.



Note: If you select the wrong table don't panic – click on the Report menu and choose **Data...** The above box will appear once again and you can change the table.

7. Select **File > Save As...** and save your report as MyReport.CLR.

Design Workspace

The Design workspace (see page 2) (also known as the report canvas) is the environment in which you will build your report layout. The workspace is divided up into “sections” or “bands”, with each band's title and height-adjustment bar immediately below.

- The **header band** (the white space above the word 'Header') will appear at the top of each page of the report.
- The **detail band** will be the body of the report.
- The **footer band** will appear at the bottom of each page of the report.

There is a further band that can be chosen from the Reports menu called Summary and this prints either below the last line of data on the final page or just above the footer on the final page depending whether the 'Summary to Bottom' option has been chosen from the Tools menu. It is important to note that even though the summary band appears below the footer band on the Design tab, it will always appear above the footer band on the report. When a report has a footer, it is always the last thing on the page.

To begin the report:



1. Locate the label icon on the Standard Component Palette (see page 15) toolbar and click on it.
2. Click in the white space of the header band (see page 2) to create a label.
3. Create two more labels in the header band using the same principles. Don't concern yourself with alignment.
4. Select 'Label1'.
5. Locate the Edit toolbar (see page 25) (below the label component icon on the toolbar). It should contain the text 'Label1' in a white box, which is the caption of the currently selected label.
6. Highlight the text in the Edit toolbar and type 'Product'.
7. Select Label2 and type 'Description' into the edit toolbar.
8. Select Label3 and type 'Selling Price' into the edit toolbar.

MYOB EXO Clarity

9. Continue placing labels for Latest Cost and GP% across the band.
10. Press CTRL + S to save your work.

Now add some data fields to the report:

1. Select the Product label.
2. Hold down the SHIFT key and then click on the other labels. All the labels should now be selected (see page 29). You can tell that they are selected by the small grey boxes at the corners of each label. These boxes are called selection handles.
3. Click the bold icon. All the label captions should turn bold.
4. Click the DBText icon on the toolbar.



5. Place a DBText component in the detail band (do this by clicking in the Detail band).
6. Change the text from bold to regular by clicking on the **bold** icon.
7. Place four more DBText components in the detail band.
8. Select DBText1. Notice that there are two drop-down list boxes in the Edit toolbar (see page 25). The drop-down list box on the left contains all the report sources (Stock_Listing, General_Info and the three special data sources). The drop-down on the right contains the fields.
9. Select Stock_Listing from the left drop-down list and select 'Stockcode' from the right drop-down list box.
10. Continue selecting fields 'Description', 'Sellprice1', 'LatestCost' & 'GP%' in turn for DBText components 2-5.



Things are probably looking a bit scattered - let's tidy it up a bit.

1. Launch the Align or Space (see page 20) toolbar by selecting **View > Toolbars** and clicking on Align or Space. (It may already be selected.)
2. The toolbar should appear under the drop-down list boxes.



3. Right-click on the 'Product' label (in the header band) and select **Position**.
4. Set the Left to 5 and the Top to 5
5. Click the 'GP%' label and move it almost to the right hand margin of the report.

Now to align the labels:

1. Select the Product label again, then hold down the SHIFT key and click on all of the labels remaining labels in the Header section.
2. Click the Space Horizontally icon.  The labels should now be spaced evenly across the report. (You can move them about manually later so that the fields that require more space on the report (such as description) will display nicely).
3. Click the Align Top icon.  The labels should all align with the first label you selected.

- De-select all the labels by clicking on an empty space on the canvas. Select the 'Product' label again and then hold down SHIFT and select DBText1 (containing the StockCode field) and click the Align Left icon. Repeat this step for the other labels and corresponding DBTexts but choose the Align Right icon for the numeric fields. (Ensure that the numeric components and their labels are right justified also).
- Align the tops of all of the DBText components, as we did in step 3 for the labels.
- Press CTRL + S to save your work.

Preview Window

The Preview window in the Report Designer environment works the same way as the preview capability in most other Windows applications. It shows you what your report will look when it is printed.

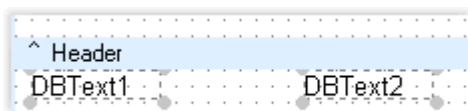
- Click on the Preview tab (see page 3) and look at your report. Make sure that the columns are nicely spaced with plenty of room for long names.
- Everything should look good except for the rows, which are double- or triple-spaced. Double spacing takes up too much room and will waste paper when the report is printed. Also you may wish to alter the number formats.

Note: You cannot edit in the preview window.

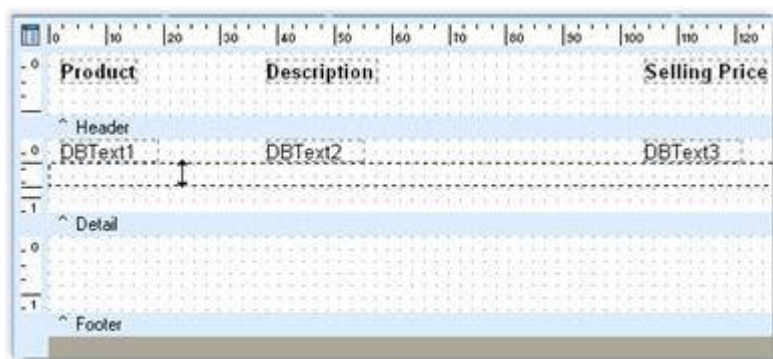
Tidying Up

Even an advanced report writer will usually find several things that can be improved when previewing a new report. Let's fix the spacing to begin with.

- Return to the design workspace by clicking on the Design tab (see page 2).
- Press the left mouse button and draw a square around the fields in the details band of your report. The fields should all have grey dots ("handles") at each corner. This indicates that it is part of a selection of multiple items (see page 29).
- Move one of the fields and notice how all of the fields move in unison, remaining in their relative positions to the field you are dragging. Move the fields to the top of the Details band.



- Place your cursor over the bar labelled Detail. Your cursor will change to an up/down arrow, indicating that you can drag the section divider up and down. (See below).



- Drag the divider (grey band) up until it meets the bottom of the components in the detail band.
- To change the formatting of the number fields, right-click on the DBText component and select display format option. Choose the format you require and click **OK**.

MYOB EXO Clarity

7. Preview the report (see page 3).
8. If any of the fields are truncating their data then right click and select the Autosize option.
9. You might wish to put the MYOB EXO Business user's company name on the top of this report. See if you can do this based upon what you've just learned.
10. Close the Report Designer window by clicking the button at the upper right corner of the window.
11. A dialog box asking you to save changes will appear. Click **Yes**.

Congratulations! You've built your first MYOB EXO Clarity report!

Grouping Data in a Report

Groups

Groups are added to a report so that summary information can be extracted from specified subsets of data. Grouping must be performed on discrete data fields; in other words, the data fields you group by must be able to be separated into distinct, identical groups. An example of a field that cannot easily be grouped is a "datetime" field like Transdate. This is because a datetime value can be any date or any time (down to fractions of a second). An example of a field that can be grouped is Accno – in other words, all records relating the same account will have the same Accno.

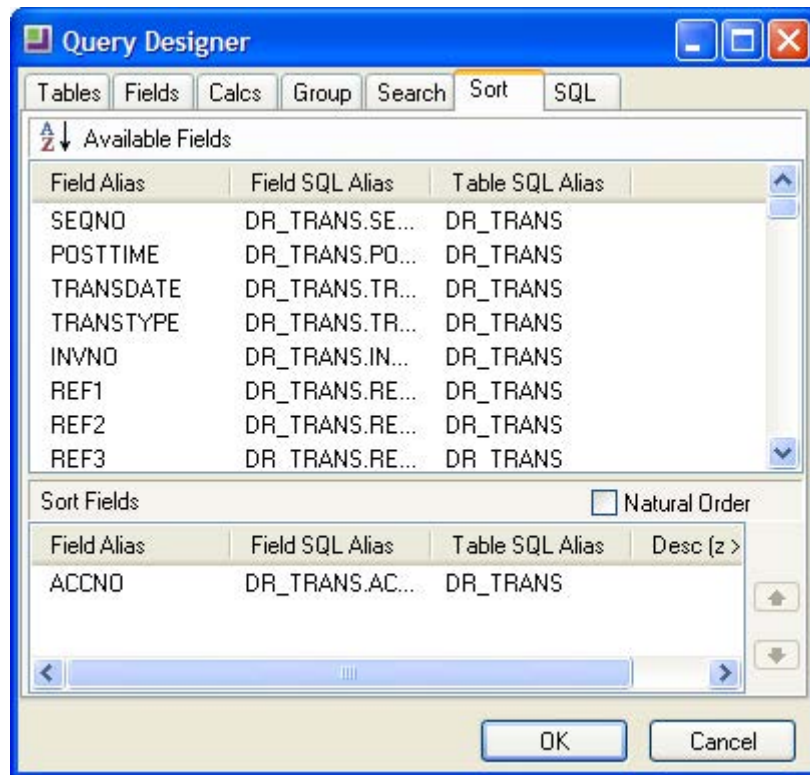
Groups are represented in the report layout by a matched pair of bands: the group header and group footer. The group header prints when the group begins and the group footer prints immediately after the last detail band of the group. This process repeats itself until all groups have been processed.

You are able to group data in a report by more than one field but it is imperative that the fields that you group your data by are also fields that you sort by and in the same order or your report will not make sense. Once the sort fields are properly set for the groups, extra sort fields can be added. This may be required, for example, to sort by the DUE DATE field for debtor invoices which have been grouped by branch then by debtor. In this case, your sort fields might be (in this order): DR_TRANS.BRANCHNO, DR_ACCS.ACCNO, DR_TRANS.DUE DATE.

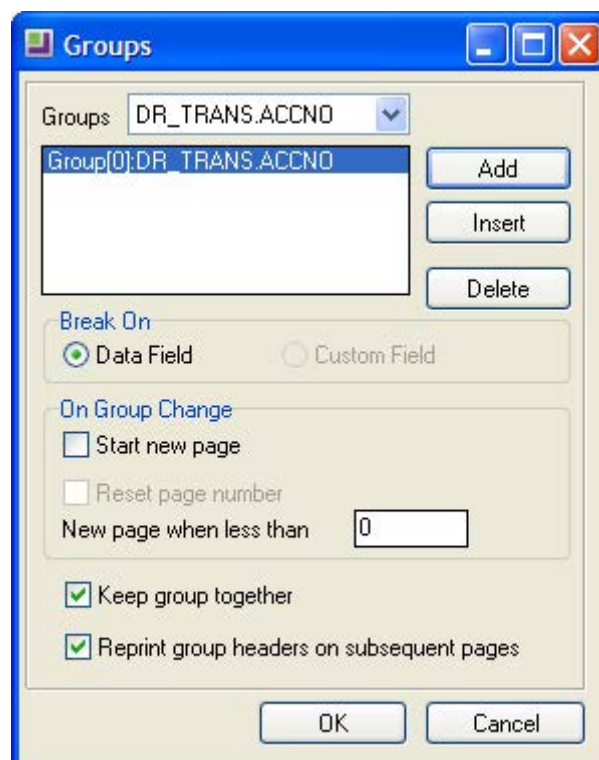
A Simple Grouping Example

To view the invoices generated by a business, organized (grouped) by each Debtor:

1. Use the Query Designer to create a data view consisting of required fields from the DR_TRANS table and sort the data by ACCNO:

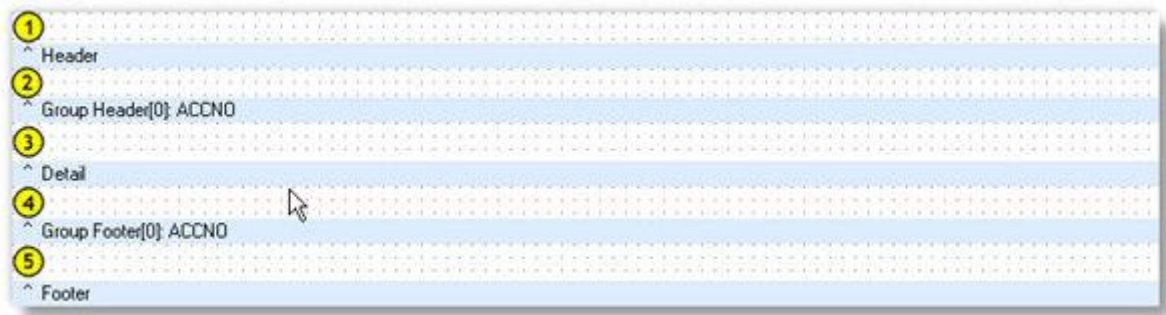


2. Close the Query Designer, then select **Groups** from the Report menu. Create a group based on the ACCNO field as shown below, then click **OK**.



MYOB EXO Clarity

The following design layout is produced. Compare the numbers on the screenshot below to the report preview to get an idea of how the report groups are generated.



3. In the Group Header section, place the repeating data for each group, i.e. the account data (because the ACCNO is the same for each item in the group).
4. In the Group Footer, place the Summary data for each group, i.e. a dbCalc component displaying `sum(Dr_Trans.amount)`.
5. In the Detail section, place the invoice data (invno, amount etc...).
6. Your report should look something like this (perhaps with different fields). Compare the design view and preview below.

The screenshot shows a report titled "Invoice Listing by Debtor Account". The report is designed with a table structure. The table has five columns: Account, Trans. Date, Invoice No., Due Date, and Amount. The report is divided into five sections: Header, Group Header[0]: ACCNO, Detail, Group Footer[0]: ACCNO, and Footer. The Group Header section contains the fields ACCNO and NAME. The Detail section contains the fields TRANSDATE, INVNO, DUEDATE, and AMOUNT. The Group Footer section contains the field Sum(AMOUNT).

Account	Trans. Date	Invoice No.	Due Date	Amount
ACCNO	NAME			
ACCNO	TRANSDATE	INVNO	DUEDATE	AMOUNT
				Sum(AMOUNT)

EXO Business Report Designer:

File

Data Calc Design Preview Detail

62% 1 Cancel

Invoice Listing by Debtor Account

Account	Trans. Date	Invoice No.	Due Date	Amount
0 6. CASH SALES	15.01.2008	10171	15.01.2008	227.53
	17.01.2008	10173	20.01.2008	207.55
	17.01.2008	10174	22.01.2008	266.50
	17.01.2008	10176	27.01.2008	282.87
	17.01.2008	10177	29.01.2008	22.33
	17.01.2008	10193	28.01.2008	227.53
	17.01.2008	10194	28.01.2008	346.95
	16.01.2008	10187	16.01.2008	83.15
	16.01.2008	10188	16.01.2008	83.15
	14.01.2008	10185	14.01.2008	32.86
				1789.42
1 1. KNIGHT NICOL AUTOS	17.01.2008	1001	26.02.2008	124.99
	17.01.2008	1002	28.03.2008	54.59
	17.01.2008	1003	28.03.2008	54.59
				234.17
2 2. ALL CAR PARTS	17.01.2008	1004	23.05.2008	127.89
	17.01.2008	1005	23.05.2008	6,608.00
	17.01.2008	1006	23.05.2008	182.70
	17.01.2008	10189	21.01.2008	210.98
	17.01.2008	10190	21.02.2008	107.55
	17.01.2008	10191	21.02.2008	346.95
				7634.87
3 3. AUSSIE SPARES	13.01.2008	10170	21.02.2008	1,538.00
				1538
7 7. JAMES BARRY	17.01.2008	10172	17.01.2008	533.70
				533.7
8 8. MARK LAWRENCE	17.01.2008	10178	31.01.2008	1,530.00
				1530
13 13. SUVA PARTS	17.01.2008	10175	21.02.2008	752.00
				752
15 15. TIMARU CAR SERVICES LTD	14.01.2008	10186	21.01.2008	189.77
	17.01.2008	10192	30.01.2008	204.78
				394.55

Page 1 of 2

Runtime Parameters

Parameters Editor

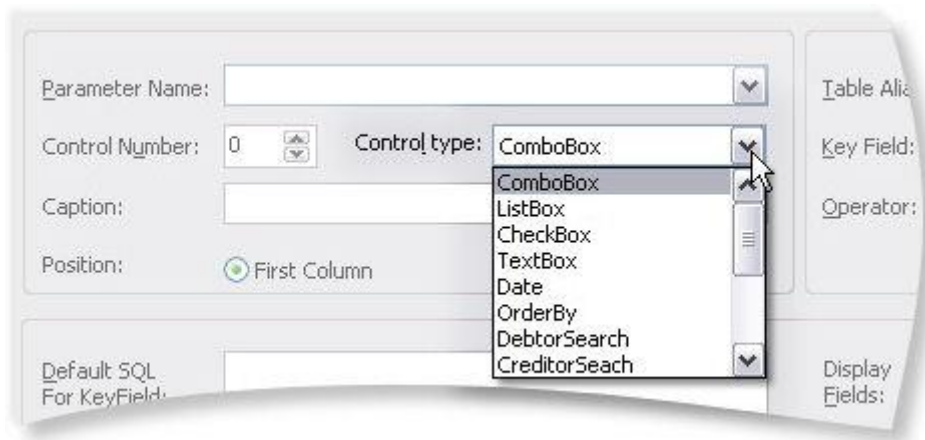
To view the Parameters Editor, select **Tools > Runtime Parameters**. The Parameters Editor window is displayed. A description of each field on the window is shown below.

Field	Description
Parameter Name	This is the name of the parameter that will be used when the parameter value is accessed in code on the Calc tab.
Control Number	This determines the order of the components when they show on the parameter dialogue box. 0 is first, 1 is second, and so on.
Control Type	The type of control (see page 43) that will appear on the Clarity Report Launcher (text box, combo box, etc).
Caption	The caption that will appear alongside the control on the Clarity Report Launcher. The caption is important because it will describe to the user which field the selection criteria will control. See item 6 in the example below.
Table Alias	The name of the data pipeline that the selection criteria affect.
Key Field	The field within the data pipeline that is filtered by the selection criteria.
Operator	The search operator that is applied to the key field, e.g. equal / greater than / less than.
Default SQL for	Used with Listbox, Combobox, Checkbox, Textbox, OrderBy, DebtorPeriod, CreditorPeriod, StockPeriod and GLPeriod components. The value that is

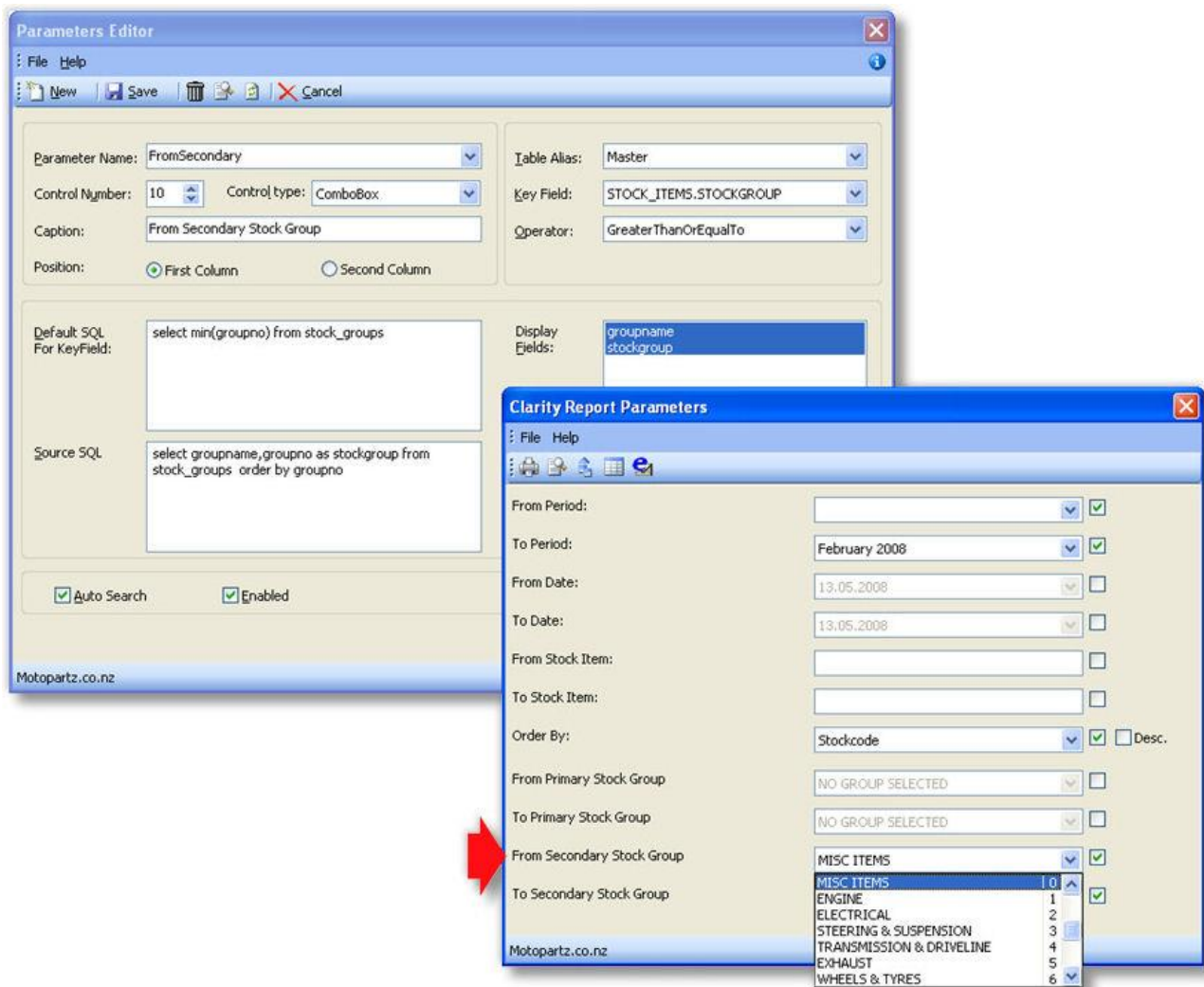
Key Field	<p>entered into this box determines what is selected by default in the Listbox / Combobox when the Clarity Report parameter window pops up.</p> <p>For OrderBy fields, this is a comma-separated list of fields that you can select to order your data by (the labels on this window change to reflect this when you select an OrderBy type parameter).</p> <p>For the period selection control types, a short SQL query must be used to specify the default, such as:</p> <pre>SELECT SEQNO FROM Period_Status WHERE Ledger = 'D' and AGE = 1</pre>
Source SQL	<p>Only used with Listbox, Combobox and OrderBy components. The SQL that will retrieve the values to be displayed in the Listbox / Combobox when the Clarity Report Launcher pops up. The name of the field for your "Key Field" must match one of the names of the fields in your source SQL statement. If there's a difference (e.g. your Key Field is HDR_SEQNO and your combo has the field SEQNO), change "SEQNO" to "SEQNO AS HDR_SEQNO". This is called aliasing a field and allows Clarity to match that field to the Key Field.</p> <p>For OrderBy parameters, this determines the default OrderBy field.</p>
Display Fields	<p>The fields that are highlighted in this box are the fields that will be displayed in the Listbox / Combobox when the Clarity Report Launcher pops up. To enter values into this box, double click on the SQL statement in the Source SQL box. If it is a valid SQL statement, the list will automatically populate. The key field must be included in this list to be able to save the parameter. Of the selected fields, you can determine which ones will actually show by CTRL-clicking to select and de-select fields.</p>
Enabled	<p>Whether or not the component is enabled by default. Enabling / disabling can also be achieved at runtime by checking or un-checking the checkbox displayed alongside the component.</p>

Control Types

The control types listed here are derived from the Control Type drop-down in the Parameters Editor window (see page 42).

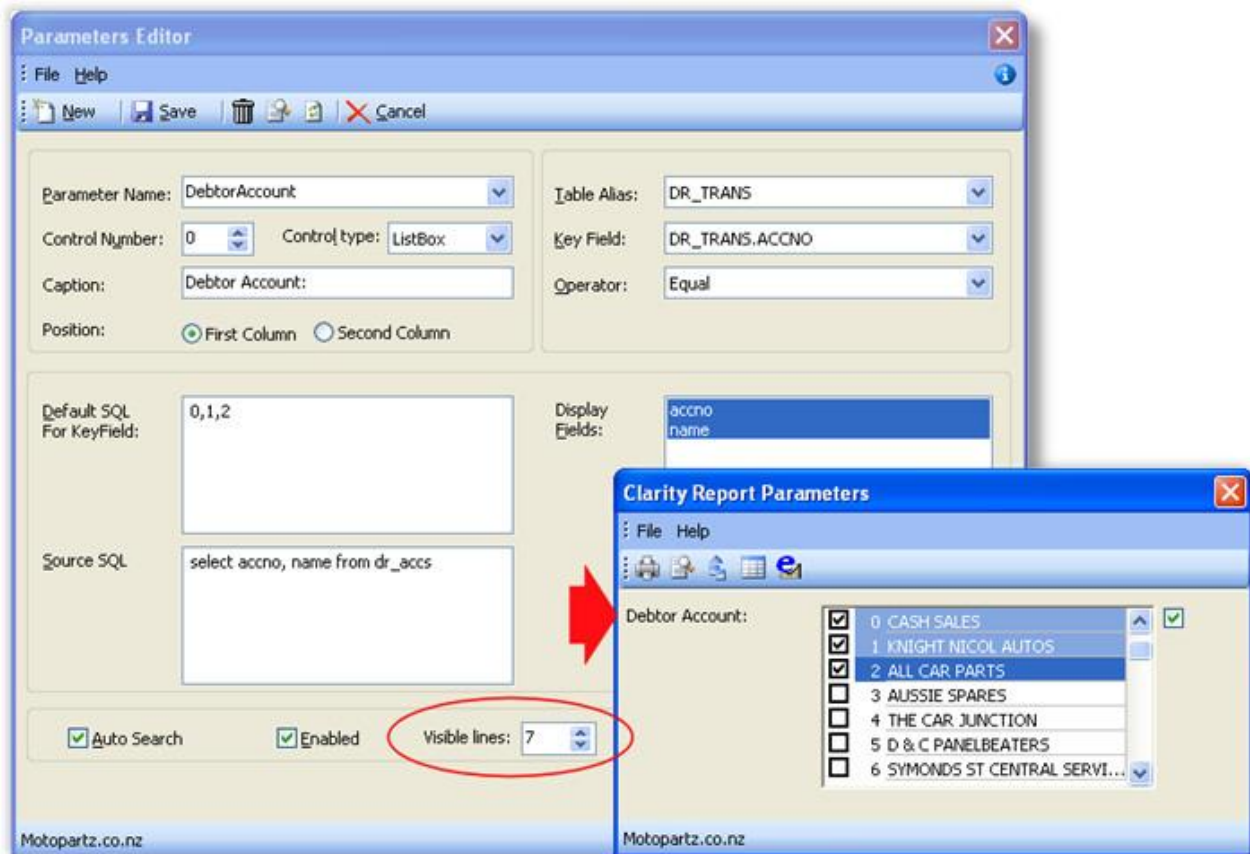


Combo Box



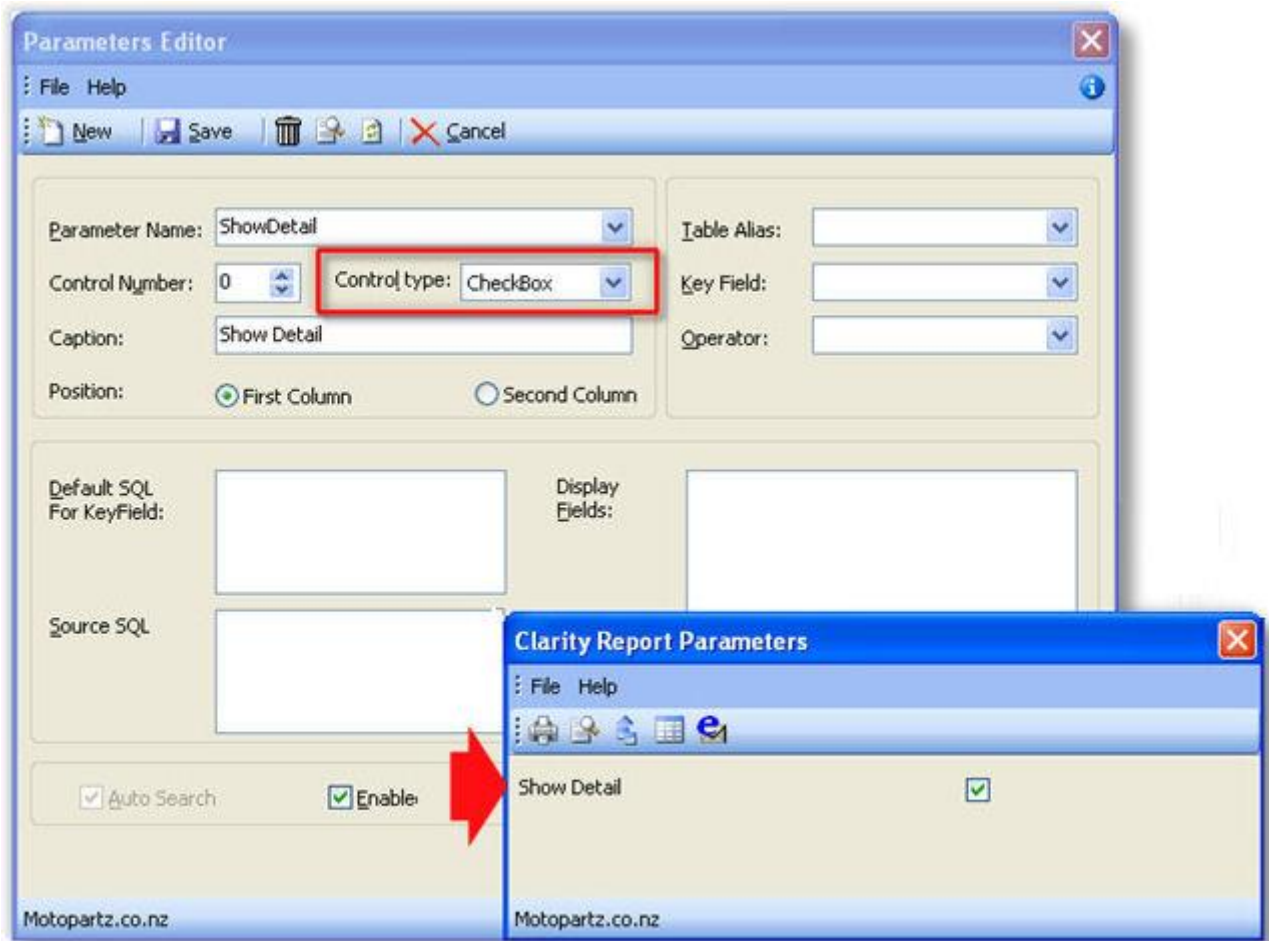
This parameter would select all periods where the STOCK_ITEMS.STOCKGROUP field is greater than or equal to the stock group selected by the user in the “From Secondary Stock Group” combo box.

Notice that the default SQL uses “select min(groupno) from stock_groups”, which gives us the first stock group as a default. Groupname and Groupno are the fields that are displayed in the combo box.

List Box

This parameter would select all records from DR_TRANS where ACCNO equals any of the accounts that the user has selected in the list box. Notice the extra field “Visible lines”, which tells Clarity how many lines high you would like the list box to be.

Check Box



This parameter would pass a true value through to the report if the user checks the check box. Notice how this parameter is not tied to any data fields, so it does not affect the filtering. Instead, this field is intended for use within Calc code (in this case it would be used to determine whether or not the detail is visible).

Text Box

The screenshot displays the Clarity Report Designer interface for configuring a parameter. The main window has a top section with fields for 'Parameter Name' (HeaderText), 'Control Number' (0), 'Control type' (TextBox, highlighted with a red box), 'Caption' (Header Text), and 'Position' (First Column selected). Below these are sections for 'Default SQL For KeyField', 'Source SQL', and 'Display Fields'. At the bottom, there are checkboxes for 'Auto Search' and 'Enabled'. A red arrow points from the 'Enabled' checkbox to a smaller dialog box titled 'Clarity Report Parameters'. This dialog box has a menu bar with 'File' and 'Help', and a toolbar with icons for print, save, undo, redo, and email. It contains a single text field labeled 'Header Text:' with a green checkmark icon to its right.

This is another 'unbound' parameter. It could be used to pass some text to the report which could then be printed in the header using a DBText field pointed to the plParams data source and the HeaderText field. You could also add some default text which would print every time unless the user decides to edit it.

Date

The screenshot shows the 'Clarity Report Parameters' window in the foreground, overlaid on a larger parameter configuration window in the background. The background window has fields for 'Parameter Name' (ToDate), 'Control Number' (4), 'Control type' (Date, highlighted with a red box), 'Caption' (To Date), 'Position' (First Column), 'Table Alias' (Master), 'Key Field' (DR_INVLINES.TRANSDATE), and 'Operator' (LessThanOrEqualTo). The foreground window shows various date and stock item filters, with a calendar for May 2008. A red arrow points from the 'Auto Search' checkbox in the background window to the 'Clarity Report Parameters' window.

Clarity Report Parameters

From Period: [] [✓]
 To Period: February 2008 [✓]
 From Date: 28.04.2008 [✓]
 To Date: 02.05.2008 [✓]
 From Stock Item: [] []
 To Stock Item: [] []
 Order By: [] [✓] Desc.
 From Primary Stock Group [] []
 To Primary Stock Group [] []
 From Secondary Stock Group MISC ITEMS [✓]
 To Secondary Stock Group SPORTY SWEAT PANT [✓]

Calendar: May 2008
 Today: 13/05/2008

This parameter would select all records from Master where DR_INVLINES.TRANSDATE is less than or equal to the date selected by the user. One alternative here is to put the word "TODAY" in the Default SQL box. This would always default the selection box to the current system date.

Others

Other parameter types are also available and some are new to Clarity in recent versions. These include:

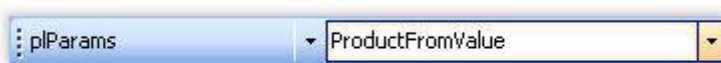
- **OrderBy** - OrderBy parameters allow you to specify which field to sort the data that comes to your report by at runtime. It also offers a "Desc" or descending sort order flag.
- **DebtorSearch, CreditorSearch** - These parameter types allow the user to type a question mark "?" to bring up the MYOB EXO Business Debtor or Creditor Search screen to select an account. These parameters always filter on Accno.
- **StockSearch** - This parameter type, like the previous type, brings up an MYOB EXO Business search box but for stock items.
- **GLAccSearch, GLAccGroupSearch** - Same as above but for GL Accounts and Account Groups.
- **DateRange** - This parameter allows users to select a date range, rather than just a single date value.

- **Spacer / Line** - These two are special parameters; they actually aren't parameters at all, but act as a vertical gap between parameters. They have no functional use, and are added for aesthetic reasons, allowing you to group similar parameters logically in the dialog box.
- **AnalysisCodeSearch** - This parameter allows users to select one or more Analysis Codes. Analysis Codes are arbitrary codes that can be assigned to transactions for grouping and reporting.
- **DebtorPeriod, CreditorPeriod, StockPeriod, GLPeriod** - These parameters allow users to select a period from the Debtors, Creditors, Stock or General Ledger.

Displaying Parameter Values

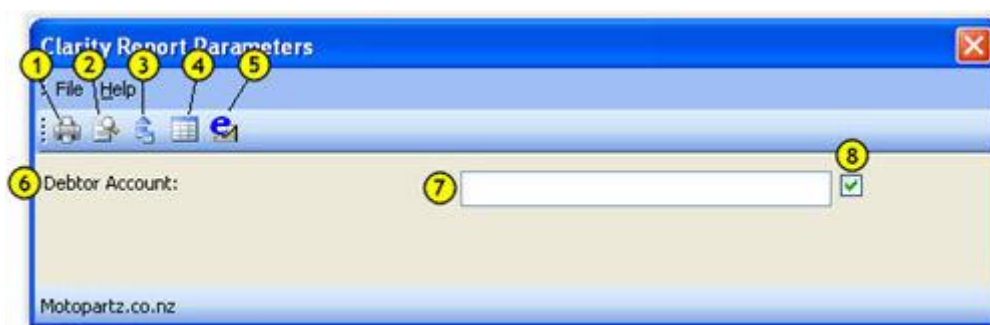
Parameter values may be displayed in a report using the DBCalc Component.

Instead of selecting a data source, select 'plParams' from the data pipeline name drop down box, and select ParameterNameValue from the data field drop down box. Clarity automatically suffixes the parameter name with 'Value' and then assigns it the value entered by the user. In the example below, the parameter name is "ProductFrom".



Parameter Examples

An example parameter window is shown below. Items 1 – 5 are selected when the desired parameters are set, to determine where the report is output to.



The options are:

1. Print to printer/file
2. Preview to screen
3. Export data to file
4. View grid, and
5. Send as attachment.

The other fields are:

6. Caption
7. Parameter field
8. Enabled checkbox

Using Parameters with Stored Procedures and Functions

If SQL editing has been enabled on the SQL tab of the Query Designer (see page 13), it is possible to take data from a function or stored procedure. In this case, you can use a runtime parameter as an input parameter to the stored procedure/function. To do this:

1. Create the parameter as normal.
2. On the Data tab, create a data source using the Query Designer.
3. On the SQL tab of the Query Designer, right-click and select **Edit SQL** to enable manual SQL editing.
4. Enter the SQL query that uses the stored procedure or function. To substitute a runtime parameter, enter the parameter name preceded by a colon parameters, e.g. SELECT *FROM FN_CR_AGEDBALANCES(:Age).
5. Click **OK**.
6. Go to the Calc tab (see page 51) and find the OnInitializeParameters event in the Events list. Right click on this event and select **New**.
7. Enter a script similar to the following to copy the values set in plParameters to the report parameters:

```
procedure ReportOnInitializeParameters(var aCancel: Boolean);
begin
    aCancel := False;
    { set Age parameter to value entered by user }
    Report.Parameters['Age'] := plParams['AgeValue'];
end;
```

Calculations

Calculations Overview

The calculations tab is where we perform the more complex calculations. It requires knowledge of Delphi. We will be looking at a few simple calculations here that will hopefully give you a better understanding and set you on the way to bigger & better calculations.

Programming Basics

Clarity uses a small subset of the Delphi programming language. While limited in one sense, this subset still allows us to perform complex calculations, string manipulation, database interaction as well as many built-in MYOB EXO Business-specific functions.

In order to use the Calcs tab effectively, it is important for you to become familiar with a few concepts.

Data Types

There are several fundamental data types used in databases and programming, but we will only describe five main ones here very briefly.

Integers – are whole numbers with no decimal portion, e.g. an account number.

Floating Point numbers – (or “Floats”) are like integers, but they do allow numbers after the decimal point, e.g. an amount field. You may see floats being called “Extended” in Clarity.

Boolean values – which only have two possible values, True or False (on/off, 1/0, they are all just representations of the same thing).

Date / DateTime – these data types store years, months, days, and for DateTime hours, minutes, seconds and fractions of a second. Sometimes you only need to work with a date (you can use “Date”); other times you only need to work with a time (You can use “DateTime”, and the date part will be irrelevant).

Strings – are just text. Even if you have a string with some numbers in it, you can't do any maths on the numbers without forcing the computer to interpret it as such. Look at some of the conversion functions to convert string values if you need to.

Objects

In the simplest sense, Objects are just abstract entities that represent “things”. Your report is an object. A label on your report is an object. A picture on your report is an object. Lines, regions and anything else that make up your report are also objects.

Properties

Objects have properties that either store information about the object, or determine how they look or behave. For example, your report has a property called “PageNo” that can be used by your code at any time during the generation of the report to tell you which page it is currently on. Be aware that properties also have a data type associated with them.

Events

Objects also have events tied to them. Events are granular steps in the generation of your report, and each time an event happens, an “event handler” is run if it exists. Event handlers are pieces of code that tell Clarity what to do when the event happens. An example of an event is a label's “OnGetText” event, which allows you to manipulate the label text at runtime. You might add an event handler to change the label text based on some other parameter, or to update a counter.

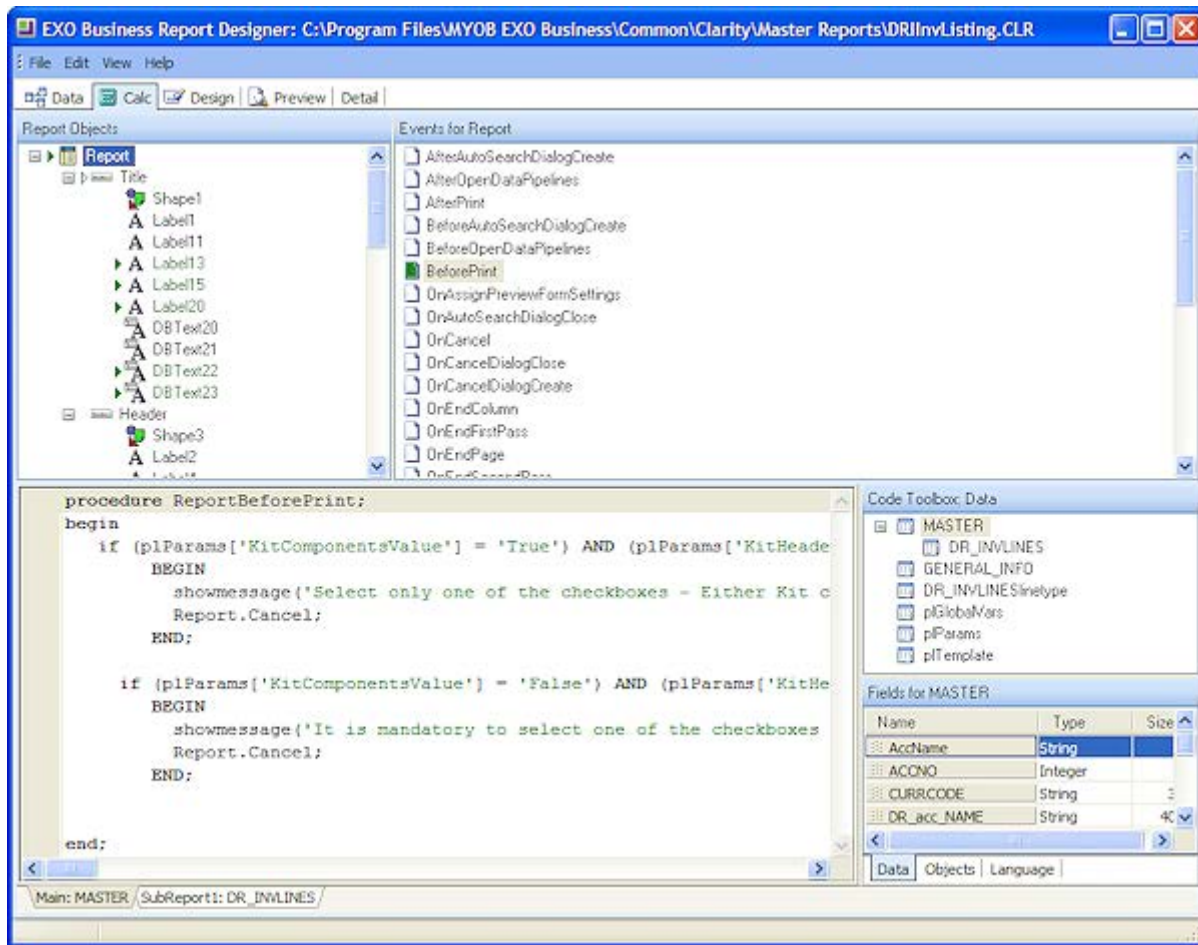
Variables

Clarity has two types of variables. While initially this can be confusing, you should come to know the difference fairly quickly once you start using them. The first type is a *component* that you can add to your canvas from the toolbar. The second type is not visible on the report itself, but it is a *value* that is manipulated “under the hood”. This value can be a counter, a text string, or a list of strings. If you want to display the value of the variable in the report, you need to manually assign it to an existing component on the report, such as a label. Like Object properties, variables also have a data type associated with them.

In the remainder of this document a variable will generally mean a component when we are using the canvas, and a value when using the Calcs tab.

The Calc Tab

If you cannot see the Code Toolbox in the lower right, go to the View menu and select **Toolbox**.



Tree View

The upper left window pane has three modes to determine what kind of data you can see, both in this pane and the Event Pane (2). To select the view mode, right click in Pane 1 or chose one of the following options from the View menu:

- **Variables** – When this option is selected, the visible report bands are listed in the Tree View and any variable components that are currently in those bands are shown in the Events Pane. This gives you the ability to quickly change multiple variables' calculation code.
- **Events** – This is the default Calcs view, and it shows a tree view of all the components of the report. You can then select any component to show the events that are triggered in relation to that component in the Events Pane (2). This is the mode that we will work in mostly for the remainder of this document.
- **Module** – The module view is very powerful, in that it shows all the current code that has been entered against events or calculations in the report. You can edit code from here, but you cannot add code against a new event, you need to go to the Events view for this. The tree view here shows four items of interest:
 - *Declarations* is where we declare any “global variables” or constants. Declaring a variable or constant here will ensure that we will be able to access it from any of our functions or procedures.

- *Events* lists only two events in the Events pane, OnCreate and OnDestroy. OnCreate is triggered as soon as the report is run, and OnDestroy is triggered as soon as the report is closed (or finishes printing if there's no user input).
- *Programs* is an area where you can write your own code, either as a function (which returns a value) or as a procedure (which does not return anything). Using custom procedures means that you can write a procedure once and call it as many times as you like throughout your report. It has the benefit that if you ever need to update your procedure, you only update the one piece of code, not all the different variations of the same thing scattered throughout the report.
- *Event Handlers* lists all the currently populated event handlers in the Events pane. While you can modify existing ones from here, you cannot create a new handler from here. You need to go into the Events mode to do this (see above).

Events Pane

This is a list of all the events belonging to the selected item in the Tree View pane (1). Events without an event handler show as a white icon in the Events pane. A green icon means that there is a valid event handler, and a red icon means that there is a problem interpreting the code in the event handler.

The two event handlers that we will use the most are "OnCalc" and "OnGetText". You will note that different events appear when you click on a DBCalc field, Variable field and label field. Our calculations will mostly use Variables. We will usually use the "OnCalc" statement when we are performing a calculation and the "OnGetText" field when we are manipulating text fields.

Code Pane

This is where we write the code (event handler) that will be executed on the selected event. To begin writing code, click on an event in the Events pane and then click in the code writing pane. A skeleton procedure will be displayed for you to complete.

Code Toolbox

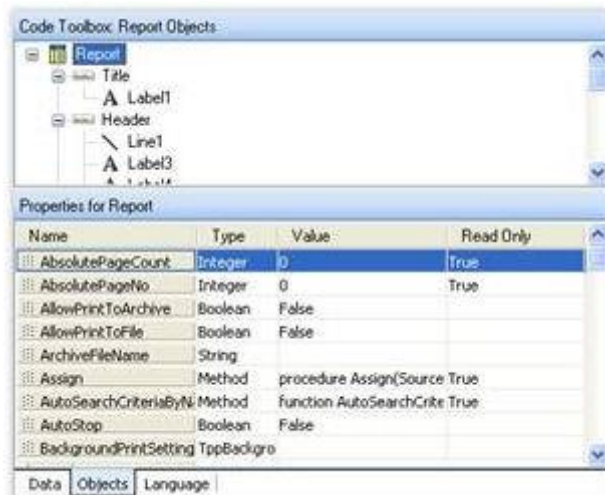
The Code Toolbox has three tabs: Data, Objects and Language. Any items listed in the bottom pane of any of the three tabs is able to be clicked on and dragged onto the code writing area to form part of your code (this just saves typing time).

Data Tab



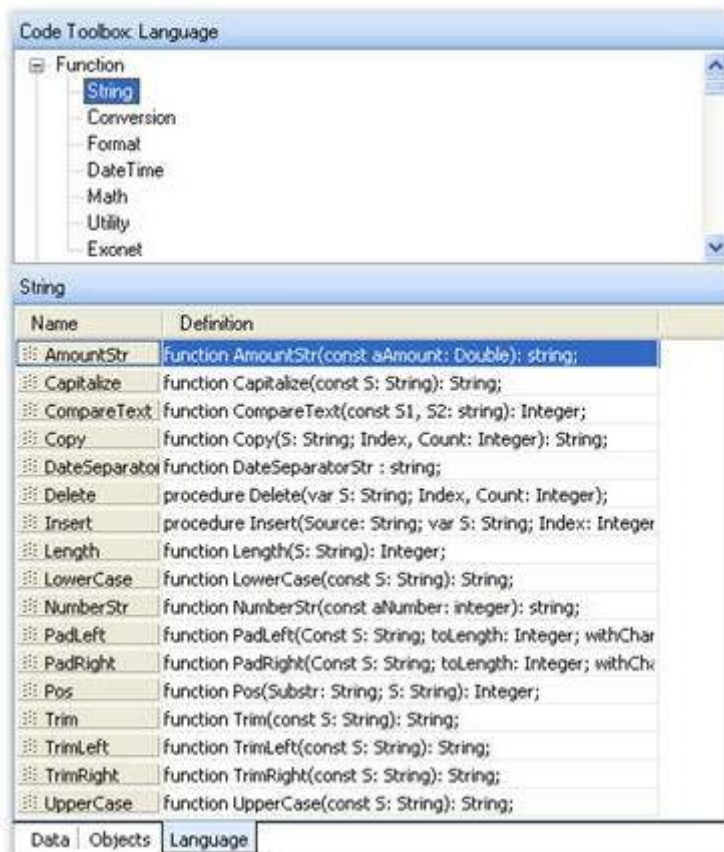
The top pane lists our available data sources. As you click on a data source the fields available in that data source appear below.

Objects Tab



The top pane shows a tree view of the report, similar to what you see in other areas of Clarity. When you click on a report object the properties belonging to that object appear in the bottom pane. You are able to use these values in your code, either reading from them or changing them, if it is not a “Read Only” property.

Language Tab



The language tab contains templates for the many functions and values that are available for use within Clarity. These functions and values are separated into different areas and you click on the area you require in the top pane to reveal the list of function templates / values that are available.

For exampl, to set all letters of a string to lowercase, click on the String item in the upper pane and then drag the LowerCase item from the lower pane onto the code area.



You can usually tell from the definition column how to use each function. In the example above you need to enter the string to convert in between the brackets and that the function will return a string.

Example:

Text := LowerCase ('THIS is my String');

Would result in the Text property becoming:

this is my string

Basic Delphi Syntax

This topic provides an overview of Delphi syntax.

Comments

As a starting point you should know how to make comments. A comment is text that is ignored by Clarity and is simply for making a note about the code. This can be to explain how a complex piece of code works or perhaps give a reason for doing something unusual in the code. This helps both you and others after you who may come back to the code long after it was written to fix or modify it.

Don't make comments too long-winded - you don't have to explain every single step, just potentially confusing ones. It's also a good idea to add the date of the change and name or initials to the comment if you are making modifications to an existing report.

You can enclose a comment in curly braces. It is best to put comments at the end of a line of code, or on a line on their own. For example,

```
Value := 1; {This is a comment here, after a line of code}
           {This is a comment on its own line}
```

Variables

Variables need to be declared before they can be used. Use the `var` keyword to declare variables. The format is:

```
var variableName : type;
```

where `type` is one of the defined data types.

Semicolon and Layout

The semicolon is a statement separator. Every clause must end in a semicolon, with a few exceptions which are noted below. While you could put all your code on one line (the semicolons help Clarity know where one statement stops and the next one starts), it is not very practical to do so. Don't be afraid to give your code plenty of space, lay it out neatly, and indent properly. This helps with readability.

Delphi uses `begin` and `end` to create *code blocks*. A code block is a logical sequence of commands that occur one after the other. You need to use the `begin` and `end` with all major structures in Delphi when you need to do more than one command. A code block always ends in a semicolon, and each clause within the block (including embedded blocks) must end in a semicolon.

A procedure block (same for functions):

```
procedure MyProcedure(var inputvar: string);
var
    a,b,c : integer;
```

MYOB EXO Clarity

```
begin
    {Do some stuff with a,b,c,s and inputvar here}
end;
```

Notice that the variable declaration occurs *before* the procedure code block. In the code examples in this document, the procedure header, begin and end keywords are mostly omitted.

Operators

Operators are the symbols in your code that enable you to manipulate all types of data. The various types of operators that are most commonly used are explained below. Note the use of comments in this document is to explain the concepts, and commenting as verbosely is not usually necessary.

Assignment operators

To assign a value to a variable use the `:=` operator.

Example:

```
Variable1.Value := 5; {assigning value of 5 to a variable component}
Dbtext2.text := Master['Name']; {assigning a value of a database field to a dbtext component}
```

Comparison operators

Comparison operators consist of `=` (equals), `<` (less than), `>` (greater than), `<=` (less than or equal to), `>=` (greater than or equal to), `<>` (not equal to). These are most often used in “if” statements.

Example:

```
If dr_trans['transtype'] = 1 then      {Comparison used}
    Variable2.value := 'Invoice';      {Assignment used}
```

Note: The `:=` operator is used to assign a value to a variable. The `=` operator compares the values of two operands.

Logical operators

“AND” and “OR” logical operators are most commonly used as a part of an “if” statement or loop as demonstrated in the following two examples:

```
If (a = 2) AND (b < 3) then
    { ... Do something ... }
If (a = 2) OR (b = 2) then
    { ... Do something ... }
```

Note: If you have an “if” statement that makes multiple comparisons, make sure that you enclose each set of comparisons in parentheses, as shown above, or it may not work correctly.

Arithmetic operators

These are standard mathematical operators used in the same way you would on a calculator.

	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	Mod

Note: Because of Delphi's *operator precedence*, the order in which arithmetic operations are calculated is not necessarily the same as the order you type them, just like on a scientific calculator. You can use parentheses to group operations, and the parenthesised operations will happen first. While operator precedence is outside the scope of this document, two examples are used below to illustrate:

```
A := 1 + 2 * 3; { A is 7 after this statement, not 9! }
```

```
A := (1 + 2) * 3; { A is 9 after this statement }
```

String Concatenation Operator

Strings also have an operator, the concatenation operator, which is the same as arithmetic addition:

```
Text := 'Dear ' + Dr_Contacts['Firstname'] + ' ' +  
        Dr_Contacts['Lastname'];
```

If you need extra text or more than one data field in a line, this is the best way to do it. It makes the report look a lot tidier than it would if you just dumped the fields next to each other.

Testing Conditions

Testing conditions allow you to give your code some “flow control”:

If Statements

An “if” statement enables you to determine whether certain conditions are met before executing some code. If statements don't need semicolons, but the statements after them (or blocks of statements) do.

Example:

```
If x = 4 then {no semicolon here}  
    y := x; {there's a semicolon here because it ends the if statement};
```

Use the **begin** and **end** keywords if you want to execute multiple lines of text when a given condition is true. This is called a “code block”.

Example:

```
If x = 6 then  
begin  
    DoSomething; {semicolons after each of these lines in the block}
```

MYOB EXO Clarity

```
DoSomethingElse;  
DoAnotherThing;  
end; {semicolon after the end keyword}
```

You can combine multiple conditions using the “if ... else” construct:

Example:

```
If x = 100 then  
    Somefunction {note: NO SEMICOLON here for a single line of code!}  
else if x = 200 then begin  
    Someotherfunction1;  
    Someotherfunction2;  
end else begin  
    Somethingelse1;  
    Somethingelse2;  
End; {End of the if statements, so there's a semicolon here}
```

Case Statements

A case statement provides a means for choosing one condition among many possibilities without needing lots of embedded “if...else” constructs. Case statements only work with numeric values though, not strings of text.

Example:

```
case Cr_Trans['Transtype'] of {Depending on this value,}  
    1: Text := 'Invoice'; {One of these 4 lines are run}  
    4: Text := 'Payment';  
    5: Text := 'Adjustment';  
    else Text := '';  
end;
```

Event Programming and Useful Functions

The Message Box

As a report runs different events are fired (another way of saying triggered). To see in which order the events fire you can enter this code in the code pane of each event. A message box will then pop up when the event is fired.

```
ShowMessage('Write the event name here');
```

ShowMessage() is also useful for debugging your code. You can put messages in to find out what value variables have, when code is being run and when it's not, and many other things. Just remember that ShowMessage needs a string, so you can't pass it a number without converting it first:

```
ShowMessage('The value of x is: ' + IntToStr(x));
```

IntToStr() is a function that converts an integer value into a string. Other such type conversions are possible, have a look in the Language tab of the Code Toolbox, under “Conversion”.

Using Variable Components

A variable-type component on your report canvas has to be assigned a value in a calculation.

1. Drop a variable onto the form.
2. Select the data type from the Edit toolbar.
3. Go to the Calc tab and select the event (usually OnGetText for strings and OnCalc for numeric data types).
4. Write the code to assign value to the variable.

Another way of quickly accessing the code window from the canvas is by right-clicking on the variable on the canvas and selecting Calculations.

The OnCalc Event

Let's look at the logic required for a running total. Basically what we want is a value field to store the total in, and we need to add the value for the next row to the total of the previous rows. Using the OnCalc event of a Variable component, the code to cumulatively add the Amount field of the Master data pipeline would be:

```
Value := Value + Master['Amount'];
```

The line of code to suppress zero values might look like this for a DBCalc component:

```
DBCalc1.Visible := DBCalc1.Value <> 0;
```

While there is an easier way to do this (right click on the component and select "Blank When Zero"), this illustrates an interesting point. When the DBCalc1.Value <> 0 part evaluates to "True", that is what is assigned to DBCalc1's Visible property making the component visible; otherwise it's hidden.

The following shows how to convert credit values to positive numbers in the trial balance. It is a slightly extended from the code line above:

```
procedure TeeChart1OnClick(ppCustomTeeChart: TppCustomTeeChart); begin
  If DBCalc2.Value <= 0 then
    Value := DBCalc2.Value * -1;
  else
    Value := 0;
  AccnoCreditValue.Visible := DBCalc2.Value < 0;
```

This makes any negative values positive, and hides any positive (now negative) values.

Date Manipulation

Getting the Current Date

There is a special function to return the current date (with no time) and the current date and time. These are `CurrentDate` and `CurrentDateTime`. These will be demonstrated later on.

Formatting a Date

Go to **Language Tab > Functions > DateTime**:

`FormatDateTime(format, DateTime);`

Format can be any string, using any combination of these keywords (for a complete list, see). The example is based on the date and time "6 January 2008 1:23:45pm":

Format	Description	Example
d	single-digit date	6
dd	double-digit date	06
ddd	short day name	Sun
dddd	long day name	Sunday
m	single-digit month	1
mm	double-digit month	01
mmmm	full month name	January
yy	two-digit year	08
yyyy	four digit year	2008
hh	hours	01
MM	minutes	23
ss	seconds	45
am/pm	am or pm	pm

Example 1:

This formats the date as in: 'Monday, 19 April 2008':

```
Text := FormatDateTime('dddd, dd mmmm yyyy', Dr_Accs['Startdate']);
```

Example 2:

This function can also be used to get part of a date:

```
Text := FormatDateTime('mm', CurrentDate);
```

This would return the month part of the current date only (e.g. it would show '04' for April).

Manipulating Dates

There are functions that allow you to manipulate dates, doing things such as adding or subtracting a number of days, months or years. These follow all the rules regular calendars, including taking into account leap-years.

Days

To add or subtract days to or from a date, simply use the + and – arithmetic operators. For example, this adds 30 days to the current date:

```
Variable1.Value := CurrentDate + 30;
```

Note that similarly simple algebra can also be used to find out the number of days between two dates by subtracting one from the other. Without going into too much detail, all dates are stored internally as a number of days since 30 December 1899, so you can use this knowledge to your advantage in complex date calculations (The reasons for this date in particular are well beyond the scope of this document).

Months

To add or subtract months to or from a date, you need to use the IncMonth() function. This takes into account the number of days per month including February 29th in leap years when applicable. For example, this subtracts 3 months from the current date:

```
Variable1.Value := IncMonth(CurrentDate, -3);
```

Years

To add or subtract years to or from a date, it's easiest to just use IncMonth() with a multiple of 12 months. This takes into account leap years when applicable. For example, this adds 3 years to the current date:

```
Variable1.Value := IncMonth(CurrentDate, 3 * 12);
```

Other Date Functions

Clarity also has other more advanced Date functions for creating or breaking down Date and DateTime values (EncodeDate / EncodeTime and DecodeDate / DecodeTime) as well as finding out how many weeks, months or years between two dates (WeeksBetween, MonthsBetween and YearsBetween), and even finding out what day of the week a given date falls on (DayOfWeek). This kind of advanced date manipulation is outside the scope of this document but they are standard functions in Delphi and any good Delphi tutorial will cover the topics in detail.

Conditional Formatting

Hiding a Section

The following code sets the visibility of the detail section. Place this in the DetailBeforePrint event handler (click on the Detail section in the Events-mode tree view, then select the BeforePrint event.

```
if DR_TRANS['amount'] < 0 then
    detail.visible := false;
```

Control DR / CR Style Formatting of Amount

Often you need to show debit/credit amounts on a report, but when a line is a debit, you don't want the credit amount showing as "0.00", you just want to hide it, and vice-versa. Here is an easy piece of code that will do that. Place it in the DetailBeforePrint event handler, with two variable components, one called debitVar and the other called creditVar in your detail band.

```
if DR_TRANS['amount'] < 0 then
    debitVar.visible := true;
    creditVar.visible := false;

    debitVar.Value := DR_TRANS['amount'];
end else begin
    debitVar.visible := false;
    creditVar.visible := true;

    creditVar.Value := DR_TRANS['amount'];
end;
```

Changing a Field Format Based on its Value

You may wish to change the format (i.e. colour, bold, etc.) of a field based on its value. This can be useful for highlighting negative values, or values that are out of the ordinary. You can use the Font property of the object to set the style of font for the field. This will work with any text field except RichText-type fields. Putting this in the DetailBeforeGenerate event handler will change Label1 to red and bold for negative values of the Amount field:

```
if DR_TRANS['amount'] < 0 then
    Label1.Font.Color := clRed;
    Label1.Font.Bold := True;
end;
```

Notice the use of the built-in constant value for the colour red. You can choose anything from the list of standard colors:

Constant	Colour
clAqua	Aqua
clBlack	Black
clBlue	Blue
clDkGray	Dark Gray
clFuchsia	Fuchsia
clGray	Gray
clGreen	Green
clLime	Lime

clLtGray	Light Gray
clNavy	Navy
clOlive	Olive
clPurple	Purple
clMaroon	Maroon
clSilver	Silver
clTeal	Teal
clWhite	White
clRed	Red
clYellow	Yellow

Alternatively, if you can't find the colour you want, you can use the RGB function to return the exact colour you need, based on the components of red, green and blue in the colour. The format is:

`RGB(red, green, blue);`

Where *red*, *green* and *blue* can be any value from 0 (none) to 255 (full). Now we could replace the line above with this to achieve the same result:

`Label1.Font.Color := RGB(255, 0, 0);`

Checking for Null Values

The syntax for checking for null values is:

```
Datapipeline.FieldObjects['Fieldname'].IsNull
```

Be aware that strictly speaking, null values are different than both zero (0) and the empty string (""). Clarity makes things slightly easier with strings, however, in that null strings are just converted to empty strings. You may have to check for both nulls and zero in some cases for numeric fields, though. This may look daunting but consider the following examples where 'Master' is our DataPipeline (our primary detail table), 'Addr1' is a string field and X_MyAccountType is a custom extended integer field with both nulls and zeros (which in this case mean the same thing):

```
if Master['Addr1'] = "" then
    AddrLabel.Text := 'No address';

if (Master.FieldObjects['X_MyAccountType'].IsNull) then
    AccTypeLabel.Text := 'AccType Not Set'
else
    if (Master['X_MyAccountType'] = 0) then
        AccTypeLabel.Text := 'AccType Not Set'
    else
        AccTypeLabel.Text := 'AccType is ' +
            IntToStr(Master['X_MyAccountType']);
```

Passing Values between Reports

Values can be passed between main reports and sub reports so that they can be used in subsequent calculations and can be accessed in code anywhere in a report.

Global Variables (i.e. variables that can be used anywhere in the report) must be declared in the following format:

```
Var [Variable Name] : [Variable Type];
```

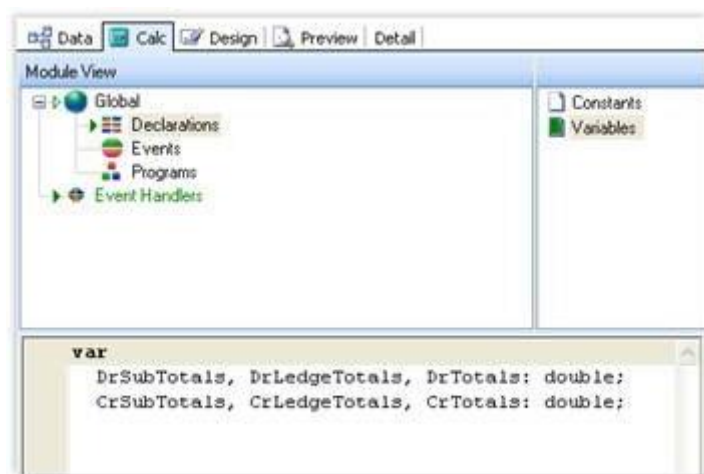
Examples:

```
MyGlobalString: String;
MyGlobalInteger: Integer;
```

A standard Clarity report that has made good use of this is the TaxByRateType.CLR. If you open it up on your screen and go to the Calcs tab of one of the sub reports, then change View to Modules you will see a list of the Calculations under the heading "Event Handlers"

Back in the main report, under "Declarations" you will see a list of all the totals that will be passed through from the Sub Reports. They have all been declared as doubles. Then under the "Event Handlers" you will see how the main report takes the totals from the Sub Reports & uses them in calculations.

Select **View > Module** on Calculations tab, then select Declarations in the Tree View pane and Variables in the Events Pane to declare global variables:



Other Useful Functions

Showing an Amount in Words

This function is useful for printing cheques. You can find it in **Language Tab > Functions > String**:

```
AmountStr(aAmount);
```

Where *aAmount* is the numeric amount you want to convert.

Example in a Variable's OnGetText event handler:

```
Text := AmountStr(DR_Trans['Amount']);
```

An amount of 300.00 would return 'Three hundred dollars only'.

Loading a Picture From a File Path

Sometimes file paths to pictures are stored in database fields. This function enables images to be loaded into image components at run time via a file path. This function is not listed in the code toolbox.

```
[image component name].picture.loadfromfile([filename]);
```

Example to load the image of a stock item into a picture component, using an extended field on STOCK_ITEMS:

```
image1.Picture.LoadFromFile(STOCK_ITEMS['X_PicturePath']);
```

Creating an AutoSearch Criteria in Code

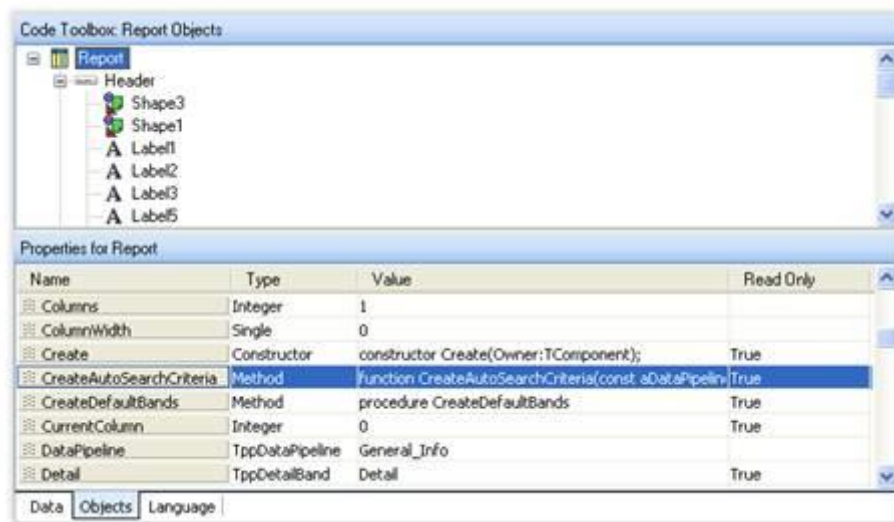
This function would be required where two different search criteria needed to be applied from one selected parameter value e.g. a trial balance report for a selected period requires \leq period value for balance sheet accounts and \leq Period value and $=$ period year age for profit and loss accounts. Having two runtime parameters and getting the user to select the same value twice could accomplish this but it is a lot tidier and more professional to have only one.

Another example would be where join restrictions between data sources make the usual method of assigning search criteria (via runtime parameters and joins) impossible.

This function must be placed in the OnGetAutosearchValues event of the Report Object.

Objects Tab > Report > Create Autosearch Criteria

MYOB EXO Clarity



Syntax:

Report.CreateAutoSearchCriteria(*aDPName*, *aFieldName*, *aOperator*, *aExpression*, *aMandatory*)

Where:

aDPName = name of the data source

aFieldName = name of field within data source

aOperator = pick from the list under Language tab > EnumeratedType > TppSearchOperatorType

aExpression = value for the search criteria

Example:

```
Report.CreateAutoSearchCriteria('Master', 'Age', soEqual,  
    plParams['PeriodValue'],true);
```

This would create an Autosearch criteria on the data source called 'Master' where field 'Master.Age' = [value selected for parameter 'Period'].

Note: This function is a *method* of the report object (see the code toolbox, Objects tab and select 'Report'). This is why it starts with "Report.". The Objects tab lists both *methods* (functions) and *properties* (attributes) associated with each object.

Execute SQL Command from Within Clarity

You can also execute almost any SQL Server command from within Clarity. An example is shown below:

```
ExecuteSQL('update stock_items set status = "L" where stockcode = ' +  
    "'LABOUR'");
```

While this is potentially very powerful (and a handful of reports use it for non-critical data processing), it is also potentially very dangerous. For this reason, it is only allowed if the associated MYOB EXO Business profile setting has been enabled. The profile setting 'Allow Clarity ExecSQL Function' controls the visibility and access to ExecSQL function in Clarity.

Profile options:

- **None** - Does not show in the Clarity designer and will not execute at runtime
- **Runtime** - Does not show in the Clarity designer but will execute at runtime
- **Design and Runtime** - Shows in the Clarity designer and will execute at runtime

Sub Reports

- Data has a master / detail type of relationship e.g. Serial numbers on a document line.
- The report requires two different data sources that do not link in any way
- Sets of data require different search criteria e.g. DR_Control.CLR
- Report requires line level & summary data

There are many cases with master/detail reports where subreports and grouping are interchangeable, but with experience you will come to know which method is the best to use for your particular purpose.

A good example of non-linking data sources is our GST report, which obviously needs to draw on information from both the Debtors and the Creditors tables. There is no way that we can or would even want to join the tables together, so we construct two separate reports and pass the results of our calculations to a third report which summarises the information.

Sub Reports Using Non-linked Data Sources

Sub reports can be used to report on two sets of data that are not linked in any way. The report that is going to be built shall give a list of transactions (debtors as well as creditors) grouped by tax rate.

Summary Sub Reports

Sub reports are also used in scenarios where there is different line and summary data.

Example:

Debtors Transaction Listing					
Invno	Transdate	Account	Subtotal	Tax	Amount
^ Header					
INVNO	TRANSDATE	NAME	SUBTOTAL	TAXTOTAL	AMOUNT
^ Detail					
^ Footer					
SubReport1: DebtorSales					

Drill Down Reports

Clarity can build reports which summarise information, but which can be “drilled into” to view more detailed information if required; in other words, the sub report is hidden until the user activates it. This is obviously very powerful and extremely useful.

Drilldown can be added to any sub report, but it is obviously more useful in some cases than others. When implementing drilldown on an existing report, you simply need to set the component on which the user is to click to drill down.

Note: Drill down reports should always be set to one pass or else the entire report will be re-generated each time a line is drilled into.

As with many other sub reports, the main and sub reports are usually each built using separate data sources based on the same table. The data pipelines are separately named and one will be grouped to give summaries, the other will not, giving details.

Crosstab Reports

The Crosstab Format

There are many different formats a spreadsheet may take. The Crosstab format is one of the most popular. Crosstab stands for Cross tabulation, a process by which totals and other calculations are performed based on common values found in a set of data.

In Microsoft Excel™ the term “Pivot Table” is used for a Crosstab.

For example, let’s assume you have a set of data that describes the sales for a company. Each sale is represented by a row of data. Each row of data contains a customer name, company type, Geographical Area, sale date or period, and sale amount. Now, assume you want to know the total sales for each month by area.

Here is one way you could present the data:

Area: Sydney

Year: 2000

Total Sales: \$2577

State: Auckland

Year: 2001

Total Sales: \$3548

This format is OK, but it makes a state-to- state comparison difficult. Another format is as follows:

	Region		
Areas	Auckland	Sydney	Total
2000	\$7816	\$5327	\$13,143
2001	\$10,500	\$9750	\$20,250

This format is easier to read and more compact there is more information in less space. It is easy to make state-to-state and year-to-year comparisons. This format is a Crosstab. The Year and Region values are called “dimensions” because they orient the data in rows and columns. The values in the cells are the calculations created when the sales data is summarized and are sometimes referred to as measures.

The simple Crosstab we've outlined here can be taken a couple of steps further to create a very informative report. What if we wanted to know the sales by Customer type within the region, as well as the total number of distinct sales per Customer type. We can present this information by adding another dimension to the columns (i.e. Customer Type) and another calculation to the values (i.e. count of sales). The resulting Crosstab would look like the one below:

		Region										
		Auckland					Sydney					Total
Year	Data	Dairies	Super-markets	Mini-marts	Take-aways	Region Total	Dairies	Super-markets	Mini-marts	Take-aways	Region Total	
2000	Count Sales	43	27	19	103	192	23	24	12	83	142	334
	Sum of Sales	\$2076	\$1764	\$1524	\$1672	\$7816	\$1001	\$1502	\$1423	\$1401	\$5327	\$13,143
2001	Count Sales	41	56	14	164	275	30	36	10	102	178	453
	Sum of Sales	\$3084	\$4500	\$1029	\$2097	\$10,500	\$2124	\$4500	\$1029	\$2097	\$9750	\$20,250

Notice the new subtotal columns after each Region. This Crosstab shows all of the information of the initial Crosstab, plus more detailed information by Customer Type. You can see that Crosstabs can express a lot of information in a very small amount of space. Clarity has a built-in facility for creating Crosstabs. The Crosstab component is designed to handle the most common Crosstab requirements with minimal effort on your part. For example, let's assume you wanted to create the Crosstab we just described. You would take the following steps:

1. Select the sales data from the database
2. Create a Crosstab component in the report layout.
3. Select the Region, Customer type, and Period as dimensions.
4. Select count and sum as values.
5. Preview the report. It would look identical to the one above.

Note: Be aware that while it is tempting to create Crosstabs to browse and analyse your transaction data, you should be careful to filter out unnecessary transactions. On a large database, calculating a Crosstab could take a long time, and it could affect performance if you are running it on the database server. Do some small test runs first to analyse the performance of the query.

Creating Crosstabs

This example provides a step-by-step process for creating a Crosstab report. By studying this example it will give practical overview in relating to the process for creating these Crosstabs.

Task 1: Select Data

1. Create a new report.
2. Hide the Data Tree and the Report Tree if they are visible.
3. Access the Query Designer.
4. Select the following tables and join by the key fields as indicated in the query here:

```
DR_ACCS INNER JOIN DR_TRANS ON (DR_TRANS.ACCNO = DR_ACCS.ACCNO)
INNER JOIN PERIOD_STATUS ON (PERIOD_STATUS.AGE = DR_TRANS.AGE)
INNER JOIN DR_ACCGROUPS ON (DR_ACCGROUPS.ACCGROUP =
    DR_ACCS.ACCGROUP)
INNER JOIN DR_ACCGROUP2S ON (DR_ACCGROUP2S.ACCGROUP =
    DR_ACCS.ACCGROUP2)
```

5. Click the Fields tab and select the following fields:

```
DR_ACCS.ACCNO
DR_ACCS.NAME,
PERIOD_STATUS.PERIOD_SHORTNAME
DR_ACCGROUPS.GROUPNAME
```

6. Scroll down to the Period_Shortname field in the selected fields pane and select it by clicking once, wait a second until the field turns black then click again. Clicking too soon will exclude the field from the selected fields pane.
7. Change the name to "Period Label" (This is just to illustrate that you can set an alias for any field if it suits your column headings).
8. Select the Calcs tab and double-click on the Dr_trans.Amount field in the Available fields pane.
9. Maximize the Query Designer, then choose Expression from the Function drop-down list and enter the following expression:

```
DR_Trans.Subtotal / DR_Trans.ExchRate
```

Note: The Calculation could be in either of the ways- `DR_Trans.Subtotal * DR_Trans.ExchRate` or `DR_Trans.Subtotal / DR_Trans.ExchRate`, depending upon the way the exchange rate is calculated and stored in the database. In MYOB EXO Business currently the amount (in local currency) is calculated by dividing the foreign amount by the exchange rate.

This will convert any invoices in foreign currency to the equivalent value in local currency based upon the exchange rate at the time that the invoice was generated.

10. Change the Field Alias to Sales Value.

MYOB EXO Clarity

11. Go to the Search tab. Here we will enter data refining the criteria. Select the fields:

Period_Status.Ledger, Operator '=', and Value D

DR_Trans.Transtype, operator '=', and Value 1

DR_Accs.Accno, Operator '=', and Value 0

This will limit the data returned to the report to period information relevant to the debtors ledger and invoice & credit (type 1) transactions for Accno 0 only. Clarity assumes that these conditions are 'AND' type conditions. You could change these to OR and specify parentheses as required by pressing right-mouse click in the Criteria pane.

12. Go to the SQL tab. The SQL query that is generated should look as follows:

```
SELECT DR_ACCS_1.ACCNO, DR_ACCS_1.NAME,  
       PERIOD_STATUS_1.PERIOD_SHORTNAME,  
       DR_ACCGROUPS_1.GROUPNAME,  
       DR_TRans.Subtotal*DR_TRans.ExchRate DR_TRans_Subtotal_DR_TRan  
FROM DR_ACCS DR_ACCS_1  
     INNER JOIN DR_TRANS DR_TRANS_1 ON  
       (DR_TRANS_1.SALES_ACCNO = DR_ACCS_1.ACCNO)
```

13. Change the name of this query to Account_Sales.

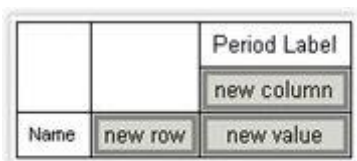
14. Click **OK** at the bottom right.

Task 2: Create a Crosstab

1. Access the design tab. Do not specify a primary details table for a cross-tab query, as there are no Detail lines to be printed. If you do select this you may end up with your crosstab report printing many times. Choose None and click **OK**.
2. Click once on the Crosstab icon on the Advanced component palette. And then click in the Detail band to create the crosstab



3. Use the Edit toolbar to assign the Crosstab to the Customer pipeline Account_Sales.
4. Select **Report > Landscape**.
5. Right-click on the Crosstab and select Configure. The Crosstab Designer will be displayed. Read the instructions at the top of the Crosstab Designer.
6. Select the Period_Label (at the bottom of the list) and drag it over the new column cell. Look for the little black, triangular indicators that visually confirm where the dimension will be created:
7. Drag the name field over and drop it on the new row element.
8. When the indicators appear to the left of the new column element, drop the field into the diagram. The diagram should look like this:



9. Click on period label element and press Sort Ascending from the layout palette.