crystal decisions™

Crystal Reports

Everything You Need to Know about Running Totals

Overview

This paper explains how to calculate running totals using the three possible methods available in Crystal Reports. Included are an outline of the uses for running totals, a comparison of the advantages and disadvantages of the three methods, and plenty of sample code for all three methods. This paper applies to all versions of Crystal Reports.

Also included in SCR_Running_Total.zip (the ZIP file containing this document):

- Sample reports illustrating the different running total methods on which the screenshots in this document were based.
- The Craze.mdb sample database

Contents

Introduction

Summary and subtotal fields total up all the records in a group or the entire report. Running totals are similar to summary or subtotal fields, but they allow you to control how the total is calculated and when it is reset.

Why use running totals?

Running totals are a very powerful and versatile feature which are used quite often in Crystal Reports to:

- See the total accumulate as it is calculated record by record
- Total a value independently of the report's grouping
- Total a value conditionally
- Total a value returned by WhilePrintingRecords formulas
- Total a value after a group selection formula has been applied, and
- Create grand totals for Top or Bottom N reports
- Total only distinct values when a report contains duplicate records

The different methods for creating running totals

All running totals need to be initialized, evaluated, then displayed on the report. Crystal Reports has three methods to do this:

Method	Available in CR and SI versions
Manual running totals (AKA Formulas with variables)	All
Running total functions (AKA RT Functions)	5 or higher (Although they are not installed automatically in version 7 or higher.)
Running Total Expert	7 or higher (However, there are several known limitations in CR 7, which were fixed in CR 8.)

Which method should I use?

The chart below is designed to help you decide which is the best running total method is for your reporting purposes.

Method	Advantages	Disadvantages
Manual running totals (AKA Formulas with variables)	Very flexible – can be used to with WhilePrintingRecords formulas, and can do page-by-page running totals Has the least known limitations out of the three running total methods	Requires basic knowledge of variable syntax Requires creating three separate formulas To display multiple types of running totals (such as both a running count & a running total), you need to create a set of three formulas for *each* type
Running total functions (AKA RT Functions)	Very flexible – can summarize WhilePrintingRecords formulas, and can do page-by-page running totals Easy to display multiple types of running totals (for Ux5rt.dll functions only)	Requires creating three separate formulas Several known limitations for RT functions
Running Total Expert	Only requires one field placed on the report (as opposed to three with the other methods). Since you're using an 'Expert' rather than creating a formula, running total fields are easier to create and modify	Cannot be used for page-by-page running totals Cannot summarize WhilePrintingRecords formulas Several known limitations in CR 7; better to use the Running Total Expert with CR 8.

Samples that accompany this document

The ZIP file containing this document also includes four sample reports that illustrate the three running total methods (there are two for RT functions).

- They are based on the Customers and Orders tables of the Craze.mdb sample database, also included in the ZIP file.
- They have a record selection to only include data from UK and France.

SCR5_Manual_RunningTotal.rpt	Illustrates running totals created using formulas and variables. Can be opened in version 5 or higher.
SCR5_RTfunctions_Ux5rt.rpt	Illustrates running totals created using formulas and RT functions. Can be opened in version 5 or higher.
SCR6_RTfunctions_Ux5total.rpt	Illustrates running totals created using formulas and RT functions. Can be opened in version 6 or higher.
SCR7_RunningTotalExpert.rpt	Illustrates running totals created using Running Total fields. Can be opened in version 7 or higher.

Manual Running Totals

This method is available in all versions of Crystal Report Designer (CR). It is the most basic method for creating a running total. It involves creating three formulas that reference the same variable, which stores the running total value.

- **The first formula initializes a variable that is to contain the running total**
- **the second formula increments the running total**
- **The third formula prints the running total.**

Before you begin, you need to understand how to declare a variable and assign it values within CR formulas, as variables are the heart of this method for calculating running totals.

Variable overview

Variables are used like a container. A variable holds information which has been assigned to it, and then can be called again further along in the formula (or in another formula) to return that value. They can also be manipulated in formulas so that they will return other values. Variables need to be declared according to which type of information you want it to return.

The data type of a variable determines the type of data that can be stored as a value in that variable. You can declare a variable with one of seven data types:

- number (100000)
- currency ($30,000.00)
- Boolean (True)
- date (January 1, 1996)
- string ('Hello')
- time (11:59:01)
- dateTime (96/12/31 11:59:59 P.M.)

In addition to declaring a variable, you also need to assign it a value. The assignment operator for CR variables is a colon followed by an equals sign (:=)

For more information on using variables, please read the chapter on Formulas in the *Crystal Reports User's Guide*.

Components of a manual running total formula

Here is the initialization formula for calculating manual running totals:

```
WhilePrintingRecords;
NumberVar RunningTotal;
RunningTotal := 0;
```

`WhilePrintingRecords;`	This function specifies when to evaluate the formula during CR's two-pass reporting process. Without this function, the formula would process during CR's first pass over the data, which is too early to calculate running totals correctly.
`NumberVar RunningTotal;`	This line declares a Number variable, and the name of this variable is RunningTotal.
`RunningTotal := 0;`	This line assigns the variable RunningTotal a value of zero. To assign a value to variables, you need to use a **:= which is** the assignment operator for variables in CR.

NOTE	For more information on the two-pass report processing model, please read our technical brief *Evaluation Times and the Two-Pass Report Processing Model* (SCR7_EvaluationTime.pdf). You can download this document from http://support.crystaldecisions.com/docs

Hands-on: Manual running totals

Let's say you work in a retail business, and have created a report for the inventory you have sold for the past year, grouped by month.

You would like to display a running total of the orders for each month. For this example, and all other examples in this document, we will use the Orders table of the Craze.mdb sample database that installs with Crystal Reports version 5 and 6.

The variable will be initialized in the report header section, incremented in the detail section, and displayed in the group footer section. You must create corresponding formulas to initialize the variable, to increment the variable, and to display the variable.

The corresponding sample report for this method is:

SCR5_Manual_RunningTotal.rpt.

Creating a manual running total

1. On the Insert menu, click Formula Field.
2. Click Formula Fields, and then click New.
3. Create these three formulas:

//@Initialize

```
//Assigns the RunningTotal variable an initial value of
//zero.
WhilePrintingRecords;
```

```
NumberVar RunningTotal;
RunningTotal := 0;
```

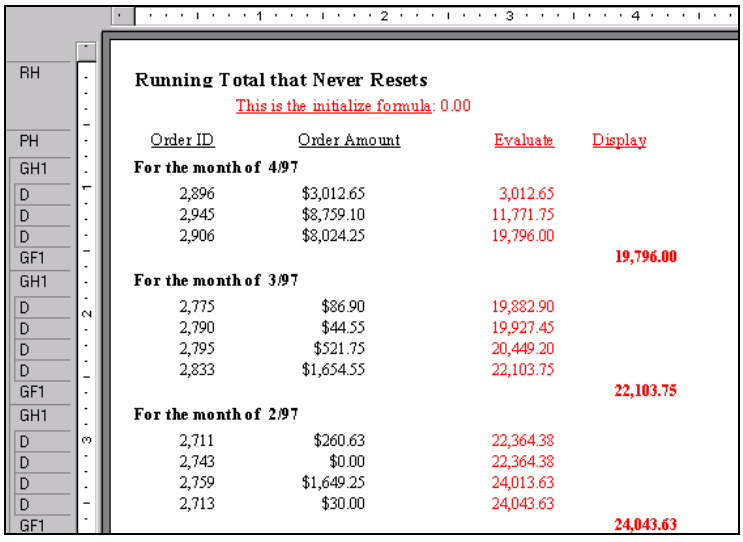
//@Evaluate

```
//Increments the RunningTotal variable by the
//current value for Orders.Order Amount}.
WhilePrintingRecords;
NumberVar RunningTotal;
RunningTotal := RunningTotal +ToNumber({Orders.Order
Amount});
```

```
//@Display
// Prints the cumulative value of the RunningTotal
// variable.
WhilePrintingRecords;
NumberVar RunningTotal;
RunningTotal;
```

4. Place the formulas on the report:

- **Place @Initialize in the report header section. Since the report header section only occurs once in the report, the variable is initialized only once.**
- **Place @Evaluate in the details section. This increments the variable for each record in the report.**
- **Place @Display in the group footer section. Since the report is grouped by month, placing the display formula in the group footer displays the cumulative value for the variable for each month.**



Manual running total that accumulates record-by-record. You can suppress the @Evaluate formula so the report only displays the month-by-month running totals.

Suppressing running total formulas

After you have verified that the formulas are returning the correct values, you can suppress the @Initialize and @Evaluate formulas so your report looks less cluttered. CR still processes the formulas, even if they aren't visible on the report.

To suppress the initialization and incrementing formulas:

1. On the report, right-click *@Initialize*, and then click **Format Field**.
2. On the **Common** tab, select **Suppress**.
3. Repeat these steps for *@Evaluate*.

Suppressing these formula fields will not affect the running total calculations.

Creating running counts

To convert the running total to a running count, all you need to do is modify the @Evaluate formula from the above example.

1. On the report, right-click *@Evaluate* and then click **Edit Formula**.
2. Modify the formula so it now says:

```
//Increments the value of the RunningTotal variable by one

WhilePrintingRecords;
NumberVar RunningTotal;
RunningTotal := RunningTotal + 1;
```

This is the only change you need to make. Keep the formula placement the same as described above.

Displaying multiple summaries for the same data

To display multiple summary types (for example, both a running total and a running count):

- **Create another set of three formulas to initialize, increment and display another variable, for each additional summary type.**
- **Insert these formulas in the same sections as the corresponding formulas for the original running total.**

The variable name in each set of three formulas must be distinct from other sets. For example, the three running total formulas would reference a variable called RunningTotal, while the three running count formulas would reference a different variable, called RunningCount.

Alternatively, you can modify the @Initialize and @Evaluate formulas to process the variables for both the running total and the running count. Note that while the formulas process both variables, they will only display the value of the variable referenced in the last line of the formula. For this reason, you'll need to create separate display formulas for each variable.

```
//@Initialize

// Initializes both the RunningTotal and

// RunningCount variables.
```

```
WhilePrintingRecords;
NumberVar RunningTotal;
NumberVar RunningCount;
RunningTotal:=0;
RunningCount:=0;

//@Evaluate
//Increments both the RunningTotal and RunningCount
variables
WhilePrintingRecords;
NumberVar RunningTotal;
NumberVar RunningCount;
RunningTotal:=RunningTotal + {Orders.Order Amount};
RunningCount:=RunningCount + 1;

//@DisplayTotal
//Displays RunningTotal final value
WhilePrintingRecords;
NumberVar RunningTotal;
RunningTotal;

//@DisplayCount
//Displays RunningCount final value
WhilePrintingRecords;
NumberVar RunningCount;
RunningCount;
```

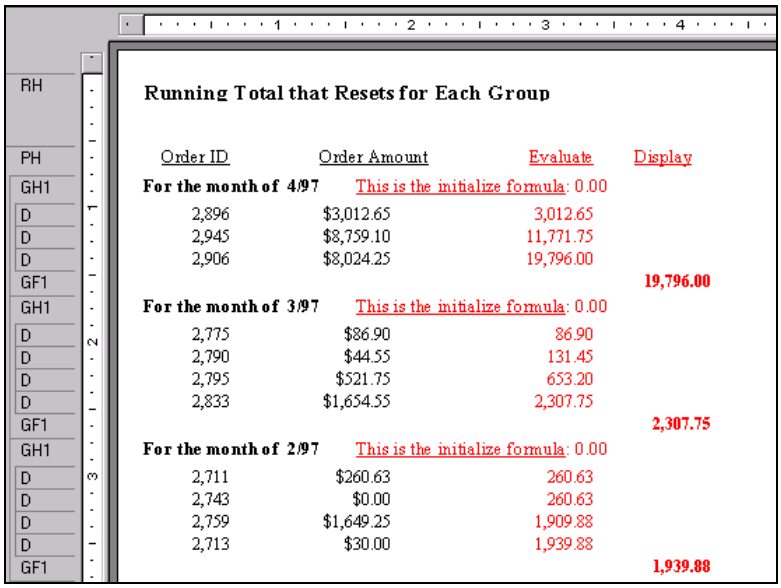
Placing these formulas on the report displays both the running total and running count for the same data.

How formula placement determines running total values

Let's look at how the running total results change when you move the formulas to different sections.

- **Move @Initialize to the group header section. This now resets the RunningTotal variable to zero when CR reaches a new group.**
- **Keep @Evaluate in the Detail section.**
- **Keep @Display in the group footer section.**

The running total now mimics a group summary, in that it only summarizes the records for a particular group, then resets.



Note how moving the @Initialize formula affects the results of the running total field. It now mimics an inserted group summary.

For more details on how the placement of your initialize, evaluate, and display formulas affects the result of a running total, see Appendix A.

Creating conditional running totals

You're probably wondering 'Why bother creating a running total for the previous example, when inserting a summary or a subtotal produces the same results?'

Say each line in the details section displays an order amount. If you only want to sum up those Order Amounts if the order was shipped, you need to somehow exclude those orders that were not shipped from the sum. The Orders table of Craze.mdb contains the Boolean {Orders.Shipped} field, which returns a True value if an order has been shipped.

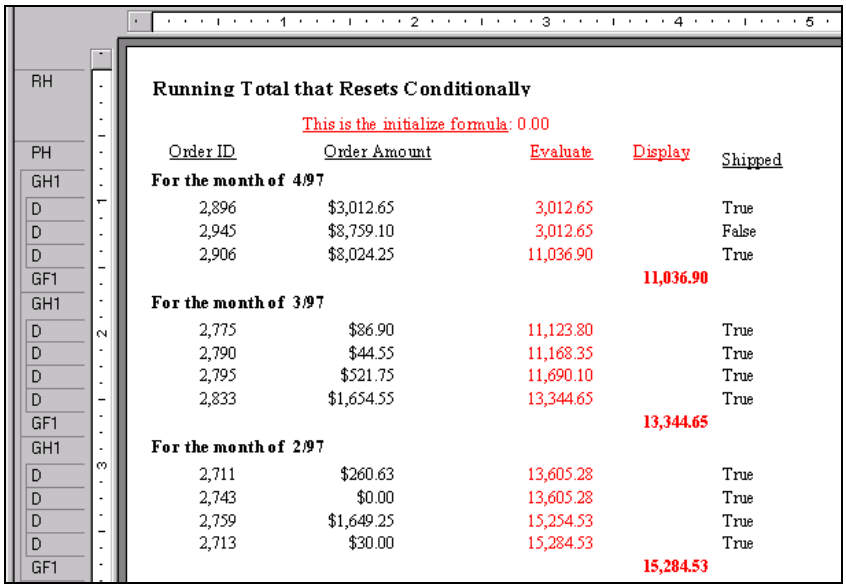
The existence of this field enables us to create a conditional running total, where the *@Evaluate* formula increments the running total *only if* the current record's shipped field equals True. Modify the *@Evaluate* formula to say the following:

```
WhilePrintingRecords;

NumberVar RunningTotal;

If {Orders.Shipped} = True

then

RunningTotal := RunningTotal + ToNumber({Orders.Order
Amount} )

Else

RunningTotal := RunningTotal
```

As in the previous example,

- ***@Initialize* is in the group header**
- ***@Evaluate* is in the details**
- ***@Display* is in the group footer.**

However, the results of the *@Display* formula have changed, because *@Evaluate* only increments RunningTotal for records where the order has been shipped.



A conditional running total. Since order ID 2945 has not yet shipped, the running total doesn't increment for that record.

For information on creating Manual Running Totals using the Store and Fetch functions, go to http://support.crystaldecisions.com/docs and search for UFLSTORE.PDF

Running Total Functions

Crystal Reports versions 5 and 6 include U25total.dll (for 32-bit) or Uf5total.dll (for 16-bit), which are responsible for three built-in running total functions. Throughout this document, and throughout Technical Support, these functions are referred to under the general term 'RT functions'.

The next generation of more streamlined RT functions was created for use in version 6 or 7 of Crystal Reports. The U25rt.dll (for 32-bit) and Uf5rt.dll (for 16-bit) files are responsible for these newer functions. The DLL files do not install automatically when you install either version 6 or version 7; you must add the DLL file yourself before you can use these newer RT functions.

The existence of the two different DLL files and the two generations of RT functions causes a lot of confusion. We hope to clarify things in this section by giving an overview of the both generations of RT Functions, then providing sample code for creating running totals and running counts.

Here is a table that summarizes the RT functions available for version 5, 6, 7 and 8. For version 8, however, we recommend using the Running Total Expert to create Running Total fields, rather than using RT functions.

Version	Available RT Functions (Installs with the program)	Optional RT Functions (Requires copying an additional file to your hard drive)
5	U*x*5total.dll RT functions	none
6	U*x*5total.dll RT functions	U*x*5rt.dll RT functions
7 & 8	Running Total Expert	U*x*5rt.dll RT functions (not tested with version 8)

As with the 'manual' method, you still need to create 3 formulas – one to initialize and reset the running total variable, one to increment it, and one to display the value. However, RT functions reduce the amount of typing required for each formula.

RT functions that install with version 5 & 6

Installing Crystal Reports version 5 and 6 automatically installs the User Function Library (UFL) file U25total.dll (for 32-bit CR) or Uf5total.dll (for 16-bit CR), which is responsible for three RT functions in the Formula Editor:

- **ResetTotal**
- **RT**
- **RTWho**

ResetTotal

This Function resets the variable, or creates it if it does not already exist

```
ResetTotal("RunningTotal")
```

`RunningTotal`	This is a text string that passes a name identifying the variable to be counted in the running total.

This resets the running total variable "RunningTotal" to a value of zero. Placing the ResetTotal function in the report header resets the running total once during the report, allowing the running total to tabulate for the entire report. Placing it in the Page Header resets the running total on every page, providing a running total for every page. Similarly, if it is placed in the Group Header it will provide a running total for every group.

For more information on the relation between formula placement and running total values, see Appendix A

RT – used to increment running total variable

This function calculates the running total by incrementing the variable by the value of a number, record-by-record.


```
RT("RunningTotal", "Function" Number Value, "WhoString")

//Adds the Value to the "RunningTotal" and keeps track
//of the associated WhoString.
```

"RunningTotal"	This variable stores the running total value.
"Function"	This function specifies what kind of calculation the formula is to perform. There are five choices: Min, Max, Sum, Count, and Average.
Number Value	This is a placeholder for a Numeric database field or Numeric value on which you wish to perform the running total. In the case of running counts, set Number Value to 1 so CR adds 1 to the running total variable for each record in the section.

"WhoString" (optional)	This is the database field used to get the value for the function. This function is useful only if you are calculating maximum or minimum values for a section. If you aren't interested in maximum or minimum values, pass an empty string ("") for the WhoString.
	
	Example without a WhoString:
	
	RT("RunningTotal", "Sum", ToNumber({Orders.Order Amount}), "")
	//Calculates the RunningTotal variable and increments
	//it by value of {Order.Order Amount} for each record.
	
	Example with a WhoString:
	
	RT("RunningTotal", "Sum", ToNumber({Orders.Order Amount}), {Customer.Customer Name})
	//Calculates the RunningTotal variable and increments
	//it by the value of {Order.Order Amount} while
	//keeping track of {Customer.Customer Name} for each
	//record.

RT – used to display the final running total value

The RT function is also used in a separate formula to retrieve and display the final calculated value of the running total. The syntax and arguments in this case are slightly different than when incrementing the running total variable:

```
RT("RunningTotal", "FetchFunction", Number Value,
"WhoString")
```

"RunningTotal"	This variable stores the running total value.
"FetchFunction"	This function specifies what kind of summary to retrieve. There are five choices: FetchMin, FetchMax, FetchSum, FetchCount, and FetchAverage.
Number Value	Set this to zero. When using the RT function to increment the running total variable, you need to specify which value to increment the running total by, such as the value of {Orders. Order Amount}. In this case, the function is being used for retrieval and display purposes only; no more incrementing is required, so the Number Value is zero.
"WhoString" (optional)	Set this to an empty string (""). Since the RT function is being used to retrieve and display the final value for the running total only.

Example:

```
RT("RunningTotal", "FetchSum", 0, "")
//Displays the sum for the variable called RunningTotal.
```

If you want to display the associated WhoString for a maximum or minimum, see RTWho, below.

RTWho

To display the maximum or minimum for a section and the associated WhoString, use the RTWho function. It looks at the WhoString in the first RT formula to determine the string value that corresponds to the maximum or minimum value (depending on what you specified for Function).

```
RTWho("VarName", "Function")
```

"VarName"	This is the variable that stores the running total value.
"Function"	Specifies whether to retrieve the maximum or minimum value for a range of records.

Example:

@Evaluate

```
RT("RunningTotal", {Orders.Order Amount},
{Customer.Customer Name})
//Adds the {Orders.Order Amount} field to a variable called
"RunningTotal" //and keeps track of the
{Customer.CustomerName}
```

@WhoString

```
RTWho("RunningTotal", "Max")
//Returns the name of the customer who had the
//maximum (highest) order amount.
```

New RT Functions for version 6 or 7 (need manual install)

Another DLL was created to use with version 6 for a new generation of RT functions. The DLL is called U25rt.dll (32-bit) or Uf5rt.dll (16-bit). These DLLs add these functions to the Crystal Reports Formula Editor:

- RTReset
- RTSet
- RTGet
- RTGetWho

These updated functions are compatible with version 7 as well. They have not been tested with version 8. These DLL files are not automatically installed, so you must obtain the DLL file and copy it to the appropriate folder before you can use these functions in the Formula Editor**.**

Obtaining the RT functions DLL for version 6 or 7

You can obtain the self-extracting EXE containing this newer DLL file from our support website. You can also find it on the Crystal Reports version 7 CD.

1. **From the web**: Go to http://support.crystaldecisions.com/downloads
   From the CD: Insert the Crystal Reports CD into your CD ROM drive.
2. **From the web**: Search for *Uf5rt.exe*, and save it to your hard drive.
   From the CD: In Windows Explorer, go to the CD drive, open the UFL folder, then open the Uf5rt folder.
3. Run the self-extracting executable *Uf5rt.exe*, accepting the defaults. This copies 3 files to the C:\Seagate\UFLS folder:
4. Uf5rt.dll – use with 16-bit Crystal Reports

- U25rt.dll – use with 32-bit Crystal Reports
- Readme.txt - explains how to install the files and use the functions they include.

5. **Windows 9x**: Copy the appropriate DLL (either 16- or 32-bit) to the C:\Windows\Crystal directory.

   Windows NT: Copy the appropriate DLL (either 16- or 32-bit) to the C:\Winnt\Crystal directory.

NOTE	If you are distributing this file for use with a compiled report or an application, copy the appropriate DLL file to your client's \WINDOWS\SYSTEM directory.

RTReset

This Function resets the variable, or creates it if it does not already exist. The syntax for this function is as follows:

```
RTReset("VarName")
```

"VarName"	This is the variable that stores the running total value.

This resets the running total variable “RunningTotal” to a value of zero. Placing the RTReset function in the report header, resets the running total once during the report, allowing the running total to tabulate for the entire report. Placing it in the Page Header resets the running total on every page, providing a running total for every page. Similarly, if it is placed in the Group Header it will provide a running total for every group.

For more information on the relation between formula placement and running total values, see Appendix A

RTSet

This function calculates the running total by incrementing the variable by the value of a number, record-by-record. The syntax for this function is as follows:

```
RTSet(“VarName”, Number Value, “WhoString”)
```

This function adds the Value to the "RunningTotal" and keeps track of the associated WhoString. It also summarizes the values for the running total in five ways:

- Count
- Sum
- Average
- Min
- Max

“VarName”	This is the variable that stores the running total value.
Number Value	A Numeric database field or Numeric value that you wish to perform the running total on. In the case of running counts, set Number Value to 1 so CR adds 1 to the running total variable for each record in the section.
“WhoString” (optional)	This is the database field used to get the value for the function. This function is useful only if you are calculating maximum or minimum values for a section. If you aren’t interested in maximum or minimum values, pass an empty string ("") for the WhoString.

Example without a WhoString:

```
RTSet(“RunningTotal”, ToNumber({Orders.Order Amount}), “”)
//Calculates the variable named “RunningTotal” and
increments it by the value //of {Order. Order Amount} for
each record in the report.
```

Example with a WhoString:

```
RTSet(“RunningTotal”, ToNumber({Orders.Order Amount}),
{Customer.Customer Name})
//Calculates the variable named “RunningTotal” and
increments it by the value //of {Order.Order Amount} while
keeping track of the {Customer.Customer //Name} for each
record in the report.
```

RTGet

This function is used in formulas to retrieve and display the value of the running total. The syntax and arguments for this function are shown below:

```
RTGet("VarName", "Function")
```

"VarName"	The variable that stores the running total value.
"Function"	FunctionName: Since the RTSet function calculates five different summary types, you must specify which type of summary to display in this formula. *Sum* prints the final total. *Min* prints the minimum value in the running total. *Max* prints the maximum value in the running total. *Average* prints the average value in the running total. *Count* prints the final count in the running total. To do a count, you need to replace the Value in the RTSet formula with the number 1.

Example:

```
RTGet("RunningTotal", "Sum")

//Displays the sum for the variable called RunningTotal.
```

RTGetWho

You can also use RTGetWho, which looks at the WhoString part of the RTSet() formula. This function determines who corresponds to the function used in the "FunctionName". This function is useful only when retrieving maximum or minimum values.

```
RTGetWho("VarName", "Function")
```

"VarName"	This is the variable that stores the running total value.
"Function"	Specifies whether to retrieve the maximum or minimum value for a range of records.

Example:

@RTSet

```
RTSet("RunningTotal", {Orders.Order Amount},
{Customer.Customer Name})

//Adds the {Orders.Order Amount} field to a variable called
"RunningTotal" //and keeps track of the
{Customer.CustomerName}
```

@DisplayWho

```
RTGetWho("RunningTotal", "Max")

//Returns the Customer Name which had the Maximum (highest)
order amount.
//The {Customer.Customer Name} field was specified as the
WhoString in //{@Evaluate}
```

Hands on: Running totals using RT functions

Let's say you have an inventory report grouped by month, and you want to display a running total for the {Orders.Order Amount} field for each month.

To keep a running total of this field, create a variable. The variable will be initialized in the report header section, incremented in the detail section, and displayed as a result in the group footer section. You must create corresponding formulas to initialize the variable, to keep track of the variable, and to display the variable.

The corresponding sample reports for this method are:

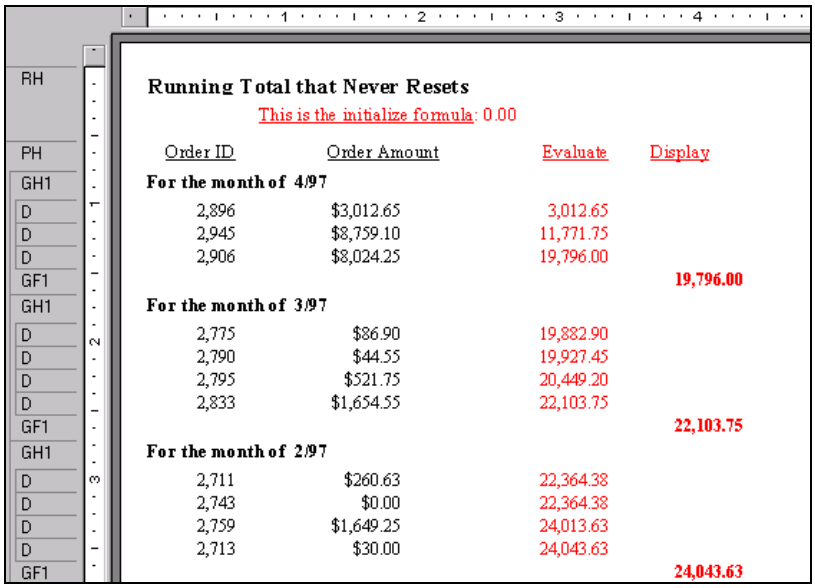
SCR5_RTfunctions_Ux5rt.rpt and *SCR6_RTfunctions_Ux5total.rpt*.

Creating running totals using RT functions

- On the report, open the **Insert** menu and click **Formula Field**.
- Click the **New** button and type a name for your formula.

Create the three formulas to initialize, evaluate and display the running total, as outlined in the table below.

Formulas for version 5 or 6 (Ux5total.dll)	Formulas for version 6 or 7 (Ux5rt.dll)	What it does, and where you place it
@Initialize ResetTotal("RunningTotal")	@Initialize RTReset("RunningTotal")	Initializes and resets (if necessary) the variable called RunningTotal Place @Initialize in the report header section. This assigns the RunningTotal variable an initial value of zero.
@Evaluate RT("RunningTotal", "sum", ToNumber({Orders.Order Amount}), {Customer.Customer Name})	@Evaluate RTSet("RunningTotal", ToNumber({Orders.Order Amount}), {Customer.Customer Name})	Sums the {Orders.Order Amount} field and stores it in the variable called RunningTotal. Also tracks the customer name associated with each value in {Orders.Order Amount}. Place @Evaluate in the Details section. This formula tabulates the value of the RunningTotal variable by noting the value of the {Orders. Order Amount} field.
@Display RT("RunningTotal", "fetchsum", 0, "")	@Display RTGet("RunningTotal", "Sum")	Displays the variable called RunningTotal, which contains the sum of the {Orders.Order Amount} field. Place @Display in the Group Footer section.
@DisplayWho RTWho("RunningTotal", "Max")	@DisplayWho RTGetWho("RunningTotal", "Max")	Displays the customer name for the record with the maximum value for {Orders.Order Amount. Place @DisplayWho in the Group Footer section.



RT function running total that accumulates record by record. You can suppress the @Evaluate formula so the report only displays the month-by-month running totals.

Suppressing running total formulas

After you have verified that the formulas are returning the correct values, you can suppress the *@Initialize* and *@Evaluate* formulas so your report looks less cluttered. CR still processes the formulas, even if they aren't visible on the report.

To suppress the formula fields:

1. On the report, right-click *@Initialize*, and then click **Format Field**.
2. On the **Common** tab, select **Suppress**.
3. Repeat these steps for *@Evaluate*.

Suppressing these formula fields will not affect the running total calculations.

Creating a running count

A running count works pretty much the same as other running totals. You must modify the *@Evaluate* and *@Display* formulas, but formula placement is the same as with running totals.

Formulas for version 5 or 6 (Ux5total.dll)	Formulas for version 6, 7, and 8 (Ux5rt.dll)	What it does, and where you place it
@Initialize ResetTotal(“RunningTotal”)	@Initialize RTReset("RunningTotal")	Initializes and resets (if necessary) the variable called RunningTotal Place @Initialize in the report header section. This assigns the RunningTotal variable an initial value of zero.
@Evaluate RT(“RunningTotal”, “count”, 1, “”)	@Evaluate RTSet("RunningTotal", 1, “”)	Counts the {Orders.Order Amount} field and stores it in the variable called RunningTotal. Since this is a count, we don’t need to track the customer name as a WhoString, so we leave that argument as an empty string (“”) Place @Evaluate in the Detail section. This formula increments the value of the RunningTotal variable by 1 for each value in the {Orders. Order Amount} field.
@Display RT(“RunningTotal”, “FetchCount”, 0, “”)	@Display RTGet("RunningTotal", "Count")	Displays the variable called RunningTotal, which contains the count of the {Orders.Order Amount} field. Place @Display in the Group Footer section.

Displaying multiple summaries

When using Ux5total.dll RT functions, to display multiple summary types (for example, both a running total and a running count), you need to create another set of three formulas to initialize, increment and display another variable, for each additional summary type.

The variable name in each set of three formulas must be distinct from other sets. For example, the three running total formulas would reference a variable called RunningTotal, while the three running count formulas would reference a different variable, called RunningCount.

When using Ux5rt.dll RT functions, it is easier to display multiple summary types for your data than with the Ux5total.dll RT functions and the manual running total method.

Instead of creating an additional three formulas, all you need to do is create another display formula to display another summary. The *@Evaluate* and *@Initialize* do not need to be modified.

For example:

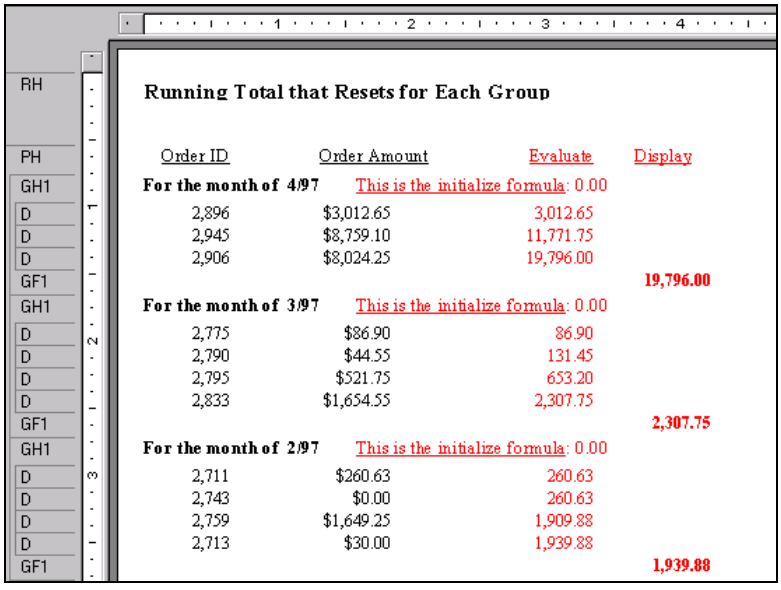
```
//@DisplaySum
RTGet("RunningTotal", "Sum")
//Displays the sum contained in the variable RunningTotal
//@DisplayMax
RTGet("RunningTotal", "Max")
//Displays the max contained in the variable RunningTotal
```

Place *@DisplaySum* and *@DisplayMax* in the group footer to display the sum total and maximum for the group.

How section placement determines running total values

Let's look at how the running total results change when you move the formulas to different sections.

- Move *@Initialize* to the group header section. This will set the RunningTotal variable to zero when CR reaches a new group.
- Keep *@Evaluate* in the Detail section. This assigns the RunningTotal variable the value equal to the current value of RunningTotal *plus* the current value in the Order Amount field.
- Keep *@Display* in the group footer section. This prints the value of RunningTotal for that group, as calculated in the *@Evaluate* formula on the last detail line in the current group.



Note how changing when the running total resets affects the results of the running total field. It now mimics an inserted group summary.

For more details on how the placement of your reset, evaluate, and print formulas affects the result of a running total, see Appendix A.

Creating conditional running totals with RT functions

You're probably wondering 'Why bother creating a running total for the previous example, when inserting a summary or a subtotal produces the same results?'

Say each line in the details section displays an order amount. If you only want to sum up those Order Amounts if the order was shipped, you need to somehow exclude those orders that were not shipped from the sum. The Orders table of Craze.mdb contains the Boolean {Orders.Shipped} field, which returns a True value if an order has been shipped.

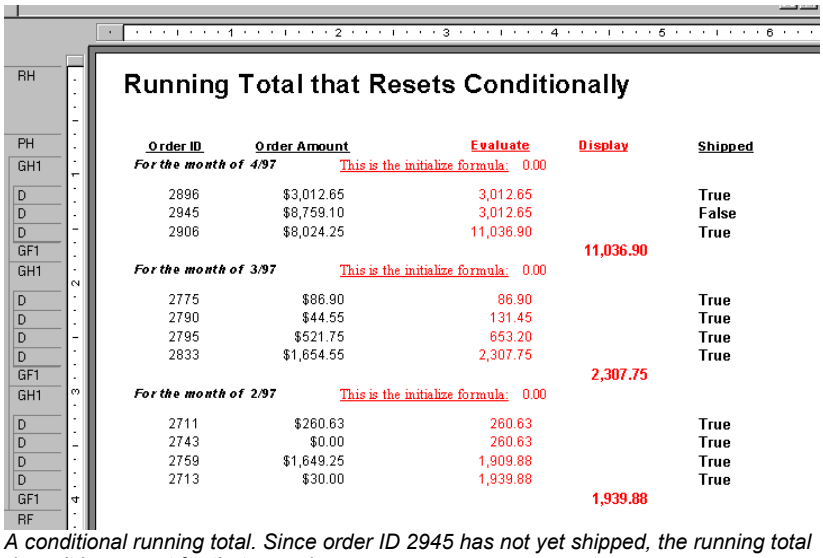
The existence of this field enables us to create a conditional running total, where the *@Evaluate* formula increments the running total *only if* the current record's shipped field equals True. Modify the *@Evaluate* formula to say the following:

```
If {Orders.Shipped} = True then

RTSet("RunningTotal", {Orders.Order Amount}, "WhoString")

//Adds the {Orders.Order Amount} field to a variable called
"RunningTotal"

//only if the order was shipped, and keeps track of the
associated customer name
```

Placement of the formulas remains the same as in the previous example:

- *@Initialize* is in the group header
- *@Evaluate* is in the details
- *@Display* is in the group footer.

However, the results of the *@Display* formula have changed, because *@Evaluate* only increments RunningTotal for detail lines where the Orders.Shipped field is equal to True (that is, the order has been shipped).



A conditional running total. Since order ID 2945 has not yet shipped, the running total doesn't increment for that record.

Limitations of RT functions

RT formulas in report created in CR 6 produce error message when opened in CR 7 or higher.

Problem

In CR 7, when opening a report with saved data, or when previewing a report that was created in CR 6 using any of the RT( )functions:

```
RTGet()
RTGetWho()
RTReset()
RTSet()
RT()
RTWho()
```

CR 7 generates the error message: "The remaining text does not appear to be part of the formula"

At this point, CR 7 displays the formula containing the unrecognized function in the Formula Editor.

Solution

The Formula Editor in Crystal Reports version 7 does not recognize the functions mentioned above because CR 7 does not automatically install the RT function DLL files. This is because CR 7 includes a built-in Running Total Expert.

This feature is easier to use and therefore eliminates the need for the RT() functions. However, there are issues with the Running Total Expert in Version 7 that were not fixed until Version 8.

One solution is to remove any references to these functions from all formulas in the report and use the Running Total Expert instead (if using Version 8 or higher).

However, if you want to continue using these RT functions:

- RTReset
- RTSet
- RTGet
- RTGet
- RTGetWho

please see the section, *Obtaining* the RT functions DLL for version 6 or 7, in this document.

If you want to continue using these other RT functions:

- ResetTotal
- RT
- RTWho

you will need to copy the U25total.dll (for 32-bit) or Uf5total.dll (for 16-bit) from your existing CR version 5 or 6 installation as these DLL files are not included in the CR 7 installation CD.

Exporting a report with suppressed sections prevents RT from calculating

Problem

When sections containing fields/formulas vital to a running total's calculation are suppressed or hidden, the running total does not process, appearing as "0" in the exported file. This happens for these export formats only:

- CHR (character-separated values)
- CSV (comma-separated values)
- TSV (tab-separated values)
- REC (record-style)
- DIF (data-interchange) formats.

All other formats export correctly, whether the sections involved are suppressed, hidden or visible.

Solution

Rather than suppressing or hiding those sections containing the fields/formulas used in the running total, do the following:

1. Suppress all the fields in the section(s)
2. Select Format | Section (Section Expert) and enable "Suppress Blank Section" for each of those sections.
3. Preview the report to make sure it looks the same as it did before (when the sections were hidden or suppressed)
4. Export the new report and the running total(s) should display correctly.

Suppressed Details section affecting results of page-by-page running total

Problem

A running count runs for each page. The count resets in the page header, the evaluate formula is in the detail section doing a record-by-record count and the display formula is placed in the page footer.

When the detail section is not suppressed, the page count is accurate. When the detail section is suppressed, the page total equals the records on the page *plus* the group total of the first group on the following page (i.e., page 2).

The following page total equals the sum of all the groups on the page excluding the first group (the first group sum is included on the previous page's running total) and the first group's total for the next page.

Solution

Because the hidden detail sections occupy no space, they are included at the bottom of each page while its group footer is being evaluated and printed on the following page. Workaround is to do a running total of the group footers.

Original formula in the detail section was:

```
WhilePrintingRecords;

NumberVar Counter;

Counter := Counter + {table.FIELD}
```

Instead, place the following formula in the group footer and hide while printing:

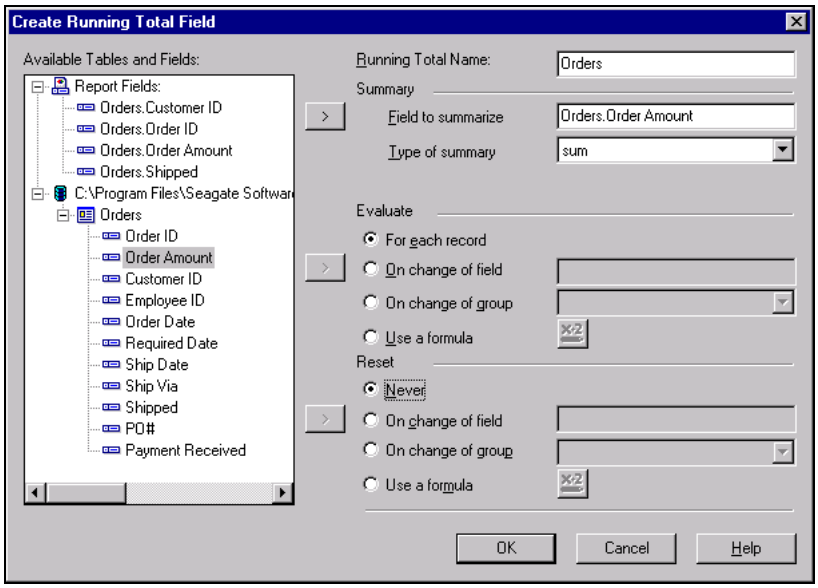
```
WhilePrintingRecords;

NumberVar Counter;

Counter := Counter + Sum ({table.FIELD} ,
{table.GROUP_FIELD})

//{table.GROUP_FIELD} is the database field you

//are grouping on.
```

This forces the running total to be evaluated when the group footer is evaluated and not on the hidden detail sections.

Running Total Expert

Crystal Reports version 7 introduced another way to calculate running totals. This new feature is called the Running Total Expert.

The Running Total Expert is a fast, efficient new way of creating running totals. A person simply needs to enter all their information into the input, and drop-down boxes to create a running total calculation on a numeric field.



*Clicking **Running Total Field** on the **Insert** menu opens the Running Total Expert dialog box.*

Hands on: Running total fields

The corresponding sample report for this method is:

SCR7_RunningTotalExpert.rpt.

1. With your report open, go to the Insert menu and select Insert Running Total Field. This opens the Insert Fields dialog box, on the Running Total tab.
2. Click New. This opens the Running Total Expert.
3. Specify a name in the Running Total Name box (for example, RunningTotal).
4. Select the type of summary you want performed in the Type of Summary box.
5. Select the field you wish to perform the running total on by moving it from the Available Tables and Fields list to the Field to Summarize box.
6. Select how frequently CR is to increment the running total field by choosing an option in the Evaluate section. There are four choices, described on the list below.

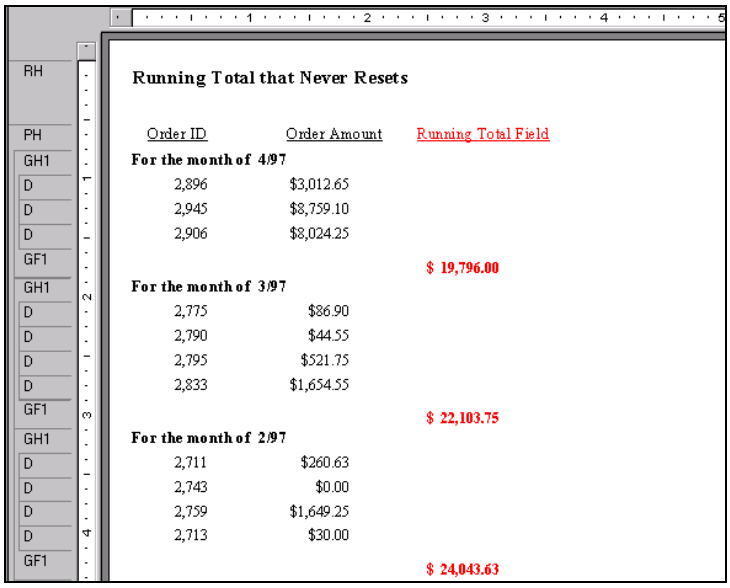
For more information on the relation between when a running total evaluates and the values that are returned, see Appendix A.

Selecting	Results in
For each record	Running total field evaluates for every record
On change of field.	Running total field evaluates only if the value in the specified field has changed. This is useful for reports that have duplicate records, or if you want to create running totals that are independent of a report's grouping.
On change of group	Running total evaluates only when the specified group has changed.
Use a formula	Running total field evaluates only when the current record has met a certain criteria. Choosing this option activates the X+2 button. Selecting the X+2 button opens the conditional formula editor window, where you can enter a formula that determines when to increment the running total. The conditional formula must return a Boolean - either True or False).

7. Select how frequently CR resets the running total field by choosing an option in the Reset section. Similar to the Evaluate section, there are four choices.

Selecting	Results in
Never	CR performs a running total for the entire report without ever resetting to zero.
On Change of field	Running total field resets each time CR encounters a change in the value for a field you specify.
On Change of group	Running total field resets each time CR encounters a new group.
Based on a formula	Use this option when there is certain criterion a record must meet before the running total field can be reset. Choosing this option activates the X+2 button. Selecting the X+2 button opens the conditional formula editor window, where you can enter a formula that determines when to reset the running total.

8. Click OK to close the Running Total Expert.
9. Click Close to close the Insert Fields dialog box.
10. Insert the running total field in the group footer section.



Running total field that cumulatively sums records and displays them for each group

Unlike the other running total methods, there is only one field you need to insert. You can insert this field multiple times on to the report.

For example, you can place another copy of this running total field in the report footer to display the grand total. To see the running total increment for each record, place another copy of the running total field in the details section.

Creating running counts

To convert the running total to a running count, all you need to do is modify the summary type:

1. On the report, right-click the {#RunningTotal} field, and then click **Edit Running Total**. This opens the Running Total Expert dialog box.
2. In the **Type of Summary** box, change the value to *Count*.
3. Click **OK** to return to the report.

Displaying multiple summaries

To display multiple summary types (for example, both a running total and a running count), create another running total field as described in the section, Hands on: Running total. Make sure to:

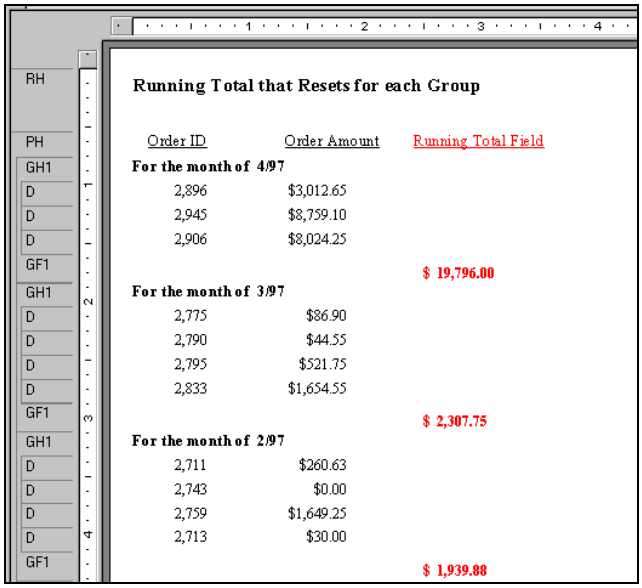
- Give it a different name than the first running total you created (for example, RunningCount).
- Select the desired summary type in the Type of Summary box (for example, Count).

Insert this new running total field on the report.

How changing the Evaluate or Reset options affects running total values

Let's look at how the running total results change when you change when the running total field resets:

1. On the report, right-click the {#RunningTotal} field, and then click **Edit Running Total**. This opens the Running Total Expert dialog box.
2. In the **Reset** section, select the option **On Change of Group**, accepting the default value.
3. Click **OK** to return to the report.



Note how changing when the running total resets affects the results of the running total field. It now mimics an inserted group summary.

The running total now mimics a group summary, in that it only summarizes the records for a particular group, then resets. Changing the option in the Evaluate section will also affect running total field results. This is illustrated in the following section.

For more details on how changing the evaluate and reset options for running total fields affects their results, see *Appendix A – When to initialize and display running totals*.

Creating conditional running totals

You're probably wondering, 'Why bother creating a running total for the previous example, when inserting a summary or a subtotal produces the same results?'

Say each line in the details section displays an order amount. If you only want to sum up those order amounts if the order was shipped, you need to somehow exclude those orders that were not shipped from the sum. The Orders table of

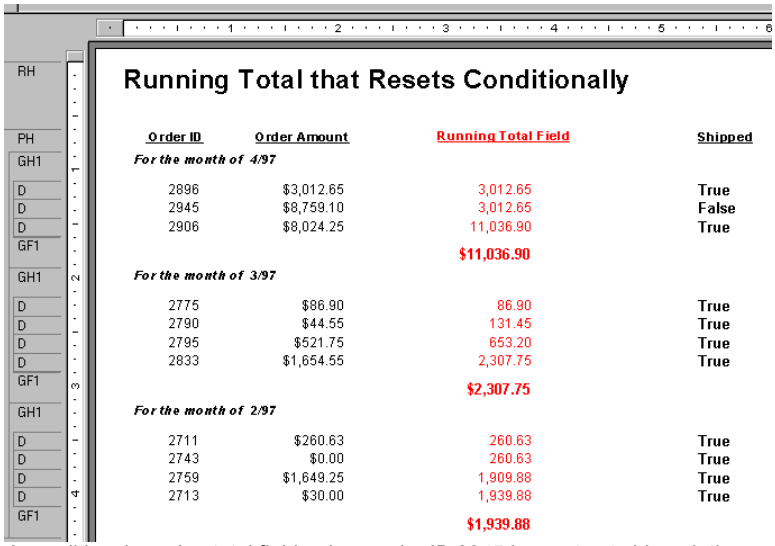
Craze.mdb contains the Boolean {Orders.Shipped} field, which returns a True value if an order has been shipped.

The existence of this field enables us to create a conditional running total field, which evaluates only if the current record's {Orders.Shipped} field equals True. Modify {#RunningTotal} to evaluate conditionally:

1. On the report, right-click the {#RunningTotal} field, and then click **Edit Running Total**. This opens the Running Total Expert dialog box.
2. In the **Evaluate** section, select the option **Use a Formula**, then click **X+2**. This opens the Conditional Formula Editor.
3. Type the following criteria:

```
//Tells the Running Total Expert to increment the running
total field

//only if the order has been shipped

{Orders.Shipped} = True
```

4. Save the formula, then close the Formula Editor.
5. Click **OK** to return to the report.



A conditional running total field - since order ID 2945 has not yet shipped, the running total field doesn't increment for that record. For illustration purposes, another copy of the running total field was placed in the details section.

Limitations of Running Total fields

- You cannot create Running Total fields based on summary fields, or WhilePrintingRecords formulas.
- You cannot create page-by-page Running Total Fields. For example, you cannot choose to reset the running total on change of a page break. Use the formulas with variables (manual running totals) or the RT function methods instead.
- Running total value displayed is incorrect when Keep Group Together is turned on in the original release of Version 7 (7.0.x.192). In the original release of Version 7 (7.0.x.192), when Keep Group Together is turned on under Report | Change Group | Options, Crystal would place imaginary files at the bottom of the group to fill up the empty space. When the Running total is displayed on the next page, the value is off by how many records would have fit in the previous page. This was fixed in the Maintenance release for Version 7.
- Running total value displayed is incorrect when a group spans pages and the details section is suppressed. If the details section is suppressed and a group spans multiple pages, the running total field will display incorrectly. Use the formulas with variables (manual running totals) or the RT functions method, or un-suppress the details section.

General Issues With Running Totals

The limitations described in this section apply to all three methods of calculating running totals.

Exporting reports containing running totals

Crystal Reports supports the exporting of reports containing running totals to *certain export formats only*. This is most likely due to the evaluation time of running totals.

Crystal Reports evaluates running totals 'WhilePrintingRecords'; that is, at the end of the second pass of the data. With certain export formats, it seems CR exports the report before it has had a chance to evaluate the running totals. As a result, the running total calculations are incorrect in the exported file. To export reports containing running totals successfully, use one of these formats:

Tab Separated Text	HTML Extended
RTF	HTML Standard
Word Document	HTML Draft
Text	Data Interchange Format (.DIF)
Paginated Text	Lotus 1-2-3 WK3
Excel Standard	Lotus 1-2-3 WKS
Excel Extended Non-Tabular	Record Style
Excel Extended Tabular	Tab Separated Text

Page-by-page running totals

On every new page, the running total resets to zero and group total becomes incorrect.

The group in the report has the option Repeat Group Header on Every Page selected. Since the *@Reset* formula is often placed in the group header, repeated group headers will trigger a resetting of the running total variable.

To get around this, insert an additional group footer, and move the *@Reset* formula to that section. By doing so, the *@Reset* formula is executed before entering the next group.

When using running totals to count items per page, the *@Display* formula adds 1 to the correct count.

This is due to the way Crystal Reports determines a page break; it reads an additional detail line before identifying there is a page break, and increases the variable incorrectly.

This is the display formula in the Page Footer:

```
//@Display

WhilePrintingRecords;

NumberVar counter;

counter;
```

The solution is to change the Display Formula to the following, leaving it in the Page Footer:

```
//@Display

WhilePrintingRecords;

NumberVar counter;

counter:=counter
```

Using this formula displays the correct count.

Appendix A – When to initialize and display running totals

This table outlines when to reset, evaluate, and display running totals to achieve different results, for all three running total methods covered in this paper. It also applies to running counts and conditional running totals.

For a running total to:	Using manual method or RT functions	Using Running Total Expert
Create a grand total for all the records in the report	@Initialize – report header @Evaluate – details @Display – report footer	Evaluate - for each record Reset – never *Place running total field in the report footer.*
Subtotal records by page	@Initialize – page header @Evaluate – details @Display – page footer	N/A
Subtotal records by group	@Initialize – group header @Evaluate – details @Display – group footer	Evaluate – for each record Reset – On Change of Group *Place running total field in group footer*
Subtotal values in group footer or group header for the entire report	@Initialize – report header @Evaluate – group footer / header @Display – report footer	Evaluate – for each group Reset – Never *Place running total field in report footer*
Subtotal values in group footer 2 or group header 2 for display in group footer 1	@Initialize – group header 1 @Evaluate – group footer / header 2 @Display – group footer 1	Evaluate – for each group (select group 2) Reset – for each group (select group 1) *Place running total field in group footer 1*

Appendix B – Excluding duplicate or suppressed records

To prevent duplicate records appearing in a report, you need to conditionally suppress the details section if it is equal to the previous record. By suppressing a section, the values are still contained in the report. They are only hidden from sight. Therefore, when inserting a Crystal Summary on a field, all of the values are being added up even if you cannot see them on the report.

In order to create a summary on a field, excluding those records that are conditionally suppressed, you need to create conditional running totals.

Example

Let's say that the details section is being conditionally suppressed if the Customer name field is equal to the previous Customer name field. In other words, duplicates regarding Customer names are being suppressed. You would like to display a grand total of the Last Years Sales field. However, you would only like to include the Last Years Sales in the Sum if it is shown on the report (the details section is not suppressed).

Note that the formula in the conditional suppression of the details section to suppress duplicate records (Format | Section | Details | x+2 button beside Suppress No Drill Down) is as follows:

```
//This formula conditionally suppresses the details section
//if the record is not //the first record and the customer
//name is equal to the previous customer name.

Not onfirstrecord and {Customer.Customer Name} =
previous({Customer.Customer Name})
```

To set up a conditional running total field, you would need to conditionally add the Last Years Sales value to the running total if the record is not being suppressed. The conditional formula for the running total would be opposite of the formula that you are using to suppress the details section. By placing the NOT() function around the entire formula, you will create the opposite of what the formula originally says.

The following three sections describe what you must do to create running totals that operate conditionally, for each running total method.

Manual running totals method

Create 3 formulas; *@Initialize*, *@Evaluate*, and *@Display*.

@Initialize is placed in the Report Header Section

```
//This formula sets a variable called RunningTotal to 0

WhilePrintingRecords;

CurrencyVar RunningTotal := 0
```

@Evaluate is placed in the Details Section

```
//If the conditional suppression is NOT true, then add the
Last Years Sales //value to the current value of the
variable.
```

```
WhilePrintingRecords;

CurrencyVar RunningTotal

If NOT(Not onfirstrecord and {Customer.Customer Name} =
previous({Customer.Customer Name})) then

RunningTotal := RunningTotal + {Customer.Last Years Sales}
```

@Display is placed in the Report Footer section to display the grand total

```
//Displays the variable called RunningTotal

WhilePrintingRecords;

CurrencyVar RunningTotal
```

RT Functions method

Create 3 formulas; *@RTReset*, *@RTSet*, and *@RTGet*

@RTReset is placed in the Report Header section

```
//Resets the variable RunningTotal to 0

RTReset("RunningTotal")
```

@Evaluate is placed in the Details Section

```
//If the conditional suppression is NOT true, set the
variable called Running //Total to itself plus the Last
Years Sales value.
//NOTE: ToNumber must be placed around this field to
convert it to a number //because the {Customer.Last Year's
Sales} is a currency field.
//RTSet only accepts number value as its second argument.

If NOT(not onfirstrecord and {Customer.Customer Name} =
previous({Customer.Customer Name})) then

RTSet("RunningTotal", ToNumber({Customer.Last Year's
Sales}), "")
```

@Display is placed in the Report Footer Section

```
//Displays the variable RunningTotal

RTGet("RunningTotal")
```

Running Total Expert method

1. On the Insert menu, click Running Total Field.
2. Click the New button/icon, and give the running total a name.
3. Move {Customer.Last Years Sales} into the Field To Summarize box.
4. Select *Sum* in the Type Of Summary box.
5. In the Evaluate section, select the On Change of Field option, and add the {Customer.Customer Name} field into the corresponding box.
6. In the Reset section, select the Never option, because you want to display a Grand Total (that is, the total for the entire report).
7. Click OK to return to the report.

Contacting Crystal Decisions for Technical Support

We recommend that you refer to the product documentation and that you visit our Technical Support web site for more resources.

Self-serve Support:

http://support.crystaldecisions.com/

Email Support:

http://support.crystaldecisions.com/support/answers.asp

Telephone Support:

http://www.crystaldecisions.com/contact/support.asp