lucity™

TRAINING GUIDE

Advanced Crystal 3

clarity through connections www.lucity.com 10561 Barkley, Suite 500 Overland Park, KS 66212 913.341.3105

Using Crystal Reports with Lucity

Advanced Examples - 3

The last in the series, this workbook is designed for experienced Crystal Reports® users. At the end of this series, you'll have the skills needed to create custom reports for your organization. In this workbook, we'll discuss variables and their use.

Table of Contents

Why Use Variables?

One of the more important uses of variables is in reports that may be run with a filter of a "child" or "grid" field that could have multiple values. Another use is in running subtraction calculations. There are many uses that only become apparent when the use of the standard formulas don't work.

Variables

A variable is a type of component that may be used in a formula. A variable represents a specific item of data or a value. It then acts as a placeholder for that value. When a formula encounters a variable, it searches for the variable's value and uses it in the formula.

Unlike a constant value, which is fixed and unchanging, a variable can repeatedly be assigned different values. When you assign a value to a variable, it maintains that value until you later assign it a new value. Because of this flexibility, it is necessary for you to declare variables before you use them.

Variables can be challenging to understand and use. Often when you create a report, the logic of what you are doing seems correct but the report just isn't responding the way you thought it would. This is when you need to start thinking about variables. If this doesn't solve the problem on first try, start moving the variables around or adding sections to put the variables in.

Declaring a Variable

Each variable must be assigned a data type (string, number, currency, time, or date) and a name. You also need to know the scope or degree to which the variable will be used:

- **Local –** Variable will be used in a single formula.
- **Global –** Variable will be used throughout the main report. This is Default if the scope isn't stated.
- **Shared –** Variable will be used throughout the main report and any subreports.

1. To declare a variable, create a formula using the *Formula Workshop*.
   - Within the *Formula Workshop,* there is an option under *Operators* called *Variable Declarations*. This lists the correct format used in stating the Data Type.
   - The correct format requires that the scope be placed before the Data Type.

     <Scope> <Data Type><VariableName>;

     Example: **Shared NumberVar Security;**

- In our example, the Scope is **Shared** and the Data Type is **NumberVar**. The Variable name is **Security**.

2. After the formula is created it needs to be placed in the report.

Assigning a Value to a Variable

After a variable has been declared, a value can be assigned to it. The assigned value can be a direct value, formula, parameter, or the value of a field.

<VariableName> := <Value>

Example: **Security := {?ViewSecuredFields};**

Declare and Assign

You can also declare a variable and assign a value to it in one step.

Example: **Shared NumberVar Security := {?ViewSecuredFields};**

Evaluation Time

These are statements that tell when the formula is to be evaluated. If nothing is stated, Crystal Reports guesses what is appropriate for the data being used in the formula.

BeforeReadingRecords: formulas are evaluated before the database records are read.

WhileReadingRecords: formulas are evaluated while it is reading the database records.

WhilePrintingRecords: formulas are evaluated while it is printing the database record data.

Evaluate After (x): formula forces this formula to calculate after the "x" formula.

Notes:_____

_____

_____

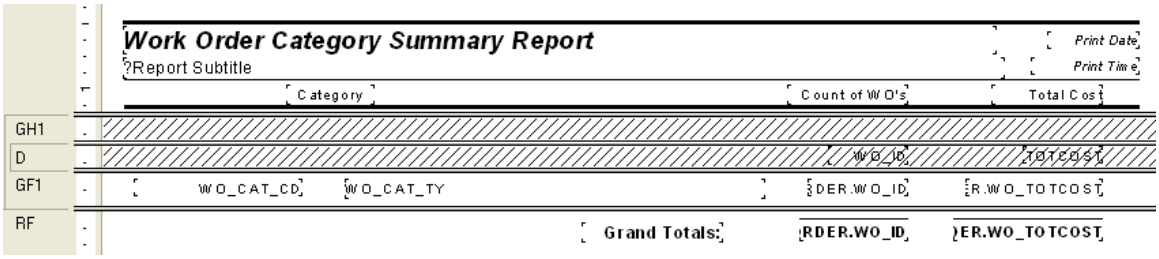
_____

_____

_____

Using Variables in a Report

The original Work Order Category Summary report looked like this:



It was a very simple report which grouped on Category and used the Crystal Summary tool to calculate the Group Total and Grand Total. This worked fine if run without filters. There were certain filters that caused duplicate Work Order costs. The problem filters are the fields that come from grid data. In the Work Order module these would include Location, Assets, Tasks and Resources. This occurs because of the filter statement being passed from Lucity to Crystal. If there are two Tasks on a Work Order that are true for a Task filter being run then the report will run the record twice. To correct this issue we used grouping and variables. Open WOCatSum.rpt and follow the steps used to correct the report:

1. Added a second grouping on the Work Order Number, WO_NUMBER.
2. Created a formula for the Work Order Cost (**WOCost**) and placed it in the WO Number *Group Footer 2*.

   ```
   WhilePrintingRecords;
   Shared numberVar WOCost ;
   WOCost:={WKORDER.WO_TOTCOST}
   ```

3. Created a formula to summarize the cost for the Category (WOCostTot). Added a new section below the WO Number *Group Footer* 2. This is a second *Group Footer* (*GF2b*) for this section. Place this new formula in this section.

   ```
   WhilePrintingRecords;
   Shared numberVar WOCostTot ;
   Shared numberVar WOCost ;
   WOCostTot:= WOCostTot + WOCost
   ```

4. Created a formula to summarize the total cost for the Work Orders (TotSum). This was placed in *GF2b*.

   ```
   WhilePrintingRecords;
   Shared numberVar GrWOCost ;
   Shared numberVar WOCost ;
   GrWOCost:= GrWOCost + WOCost
   ```

5. Created a formula to Zero the Work Order cost variable (**ZeroWO**). Placed this in the WO Number *Group Header #2*.

   Shared numberVar WOCost :=0;

6. Created a formula to Zero the Category Cost (**Zero**). Placed this in the Category *Group Header #1*.

   Shared numberVar WOCostTot :=0 ;

   Shared numberVar WOCost :=0;

7. Created a formula to show the Category total cost (**TaskTot**). Placed this in the Category *Group Footer #1*.

   WhilePrintingRecords;

   Shared numberVar WOCostTot ;

   WOCostTot

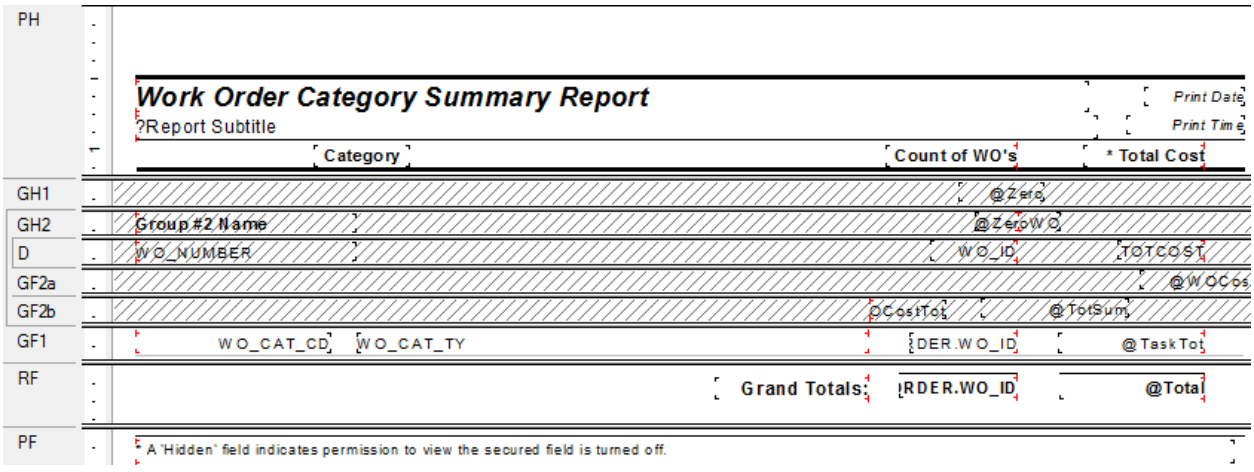
8. Created a formula to show the Grand total cost (**Total**). Placed this in the *Report Footer*.

   WhilePrintingRecords;

   Shared numberVar GrWOCost ;

GrWOCost

Suppressed the new *Group Header* and *Footer* sections.



Notes:_____

_____

_____

_____

Secured Fields in Subreports

When you add the simple **ViewSecuredFields** parameter formula to a field format in a subreport, it does not carry over to the main report without first asking for the security of the subreport. This defeats the purpose of security. The subreport should be responding to the Lucity Security setup. The main body of the report needs to communicate with the subreport. This requires the use of variables.

We will look at the Work Order Detail report and see how security was added to the cost fields in the Task/Resource subreport.

1. First, we used a shared variable called **Security**. It was declared in both the report and subreport.
    - Open **MS_WODetail.rpt**. The **ViewSecuredFields** parameter was created. For additional information, please refer to the related workbook, Beginning Crystal 2.
    - A new formula was created called **Security** to declare the variable.

    ```
    Shared NumberVar Security := {?ViewSecuredFields};
    ```

2. The new Security formula was placed into the ***Report Header*** section. The field size was reduced and the field text formatted to have white font. This ensured that the formula was not visible in the report.
3. In the **TaskRes.rpt** subreport a new formula called **SecuritySub** was created.

    Once again, the Security variable was declared .

    ```
    Shared NumberVar Security;
    ```

    - The formula was dragged into the suppressed ***Report Header***.

4. There are four fields set up to show "Hidden" if the user does not have proper security. See the Beginning Crystal 2 handbook to see how the fields were set up.

Notes:_____

_____

_____

_____

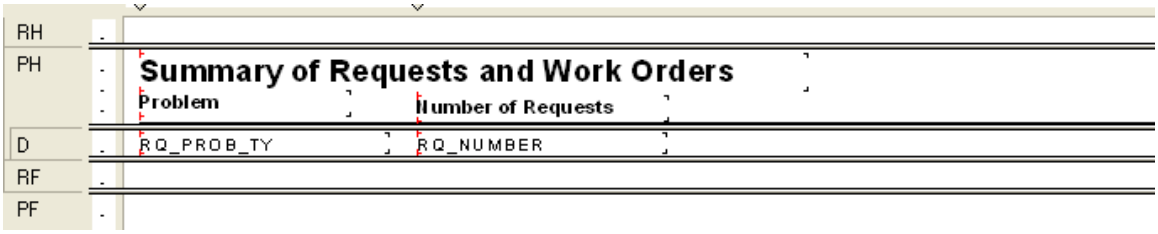
_____

_____

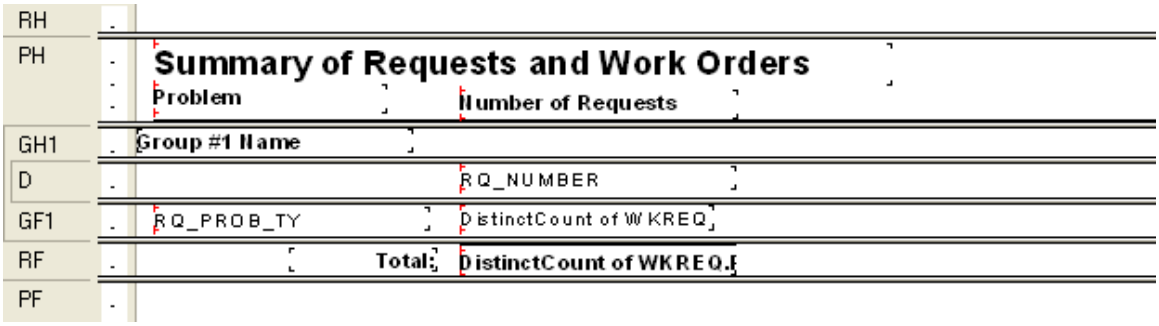
More on Variables – Passing information from a subreport to the main report

Let's create a report that shows the number of Requests and Work Orders for each Request Problem type. The number of Requests is straightforward enough with the use of a Running Total field. The number of Work Orders gets a little trickier because Work Orders are attached to Requests through a grid and thus need to be brought into the report as a subreport. Information from a subreport to a parent report can be done by using variables.

1. Create a new Work Report and name it MS_ReqSumRQWO.
2. Bring in the WKREQ table. Bring in the Problem field and the Request Number:



3. Group by Problem Type. Move the Problem field to the *Group Footer*.
4. Add a **Number of Requests** column title. Then, create a Distinct Count of the Request Numbers for each Problem Type and place it under the column header in the *Group Footer* section.
5. Add a "Total" text in the Report Footer. Then put in a Distinct Count of the Request Numbers into the *Report Footer*.



6. Add the **WKWOMWO** table in the *Database Expert* and link as shown in the related workbook, Advanced Examples 2.
7. Create a **Work Order** subreport with the WKORDER and the WKWOMWO tables and place it in the *Detail* section. Link as shown in the Advanced Examples 2 workbook.
8. In the subreport, drag the **WO_Number** field into the *Detail* section. Create one formula (**WOCount**):

```
WhilePrintingRecords;

Shared numberVar WOCount;


WOCount ;=DistinctCount({WKORDER.WO_NUMBER})
```

9. Place the formula and the **WO_NUMBER** field as follows and suppress the subreport sections. This subreport is counting the Work Orders for each Request. WOCount is a shared variable that is available to the main report.

10. In the main report, create five formulas:
    - Zero

      WhilePrintingRecords;

      Shared numberVar WOCount :=0;

      Shared numberVar ProbWOCount :=0 ;

    - WOCount (The shared variable WOCount is being passed in from the subreport)

      WhilePrintingRecords;

      Shared numberVar WOCount;

      WOCount

    - WOSum (WOCount is being summed to calculate the Problem Total for Work Orders and the Grand Total for Work Orders.)

      WhilePrintingRecords;

      Shared numberVar WOCount ;

      Shared numberVar ProbWOCount ;

      Shared NumberVar GrTotWOCount;

      ProbWOCount:=ProbWOCount + WOCount;

      GrTotWOCount:=GrTotWOCount + WOCount;

    - ProbTot

      WhilePrintingRecords;

      Shared numberVar ProbWOCount ;

      ProbWOCount

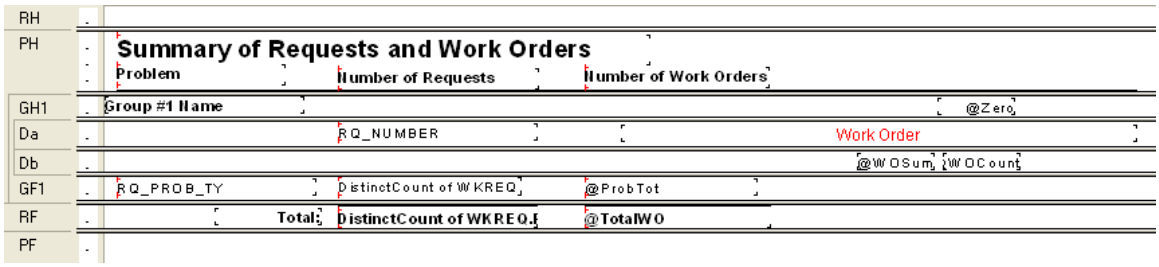
- TotalWO

WhilePrintingRecords;

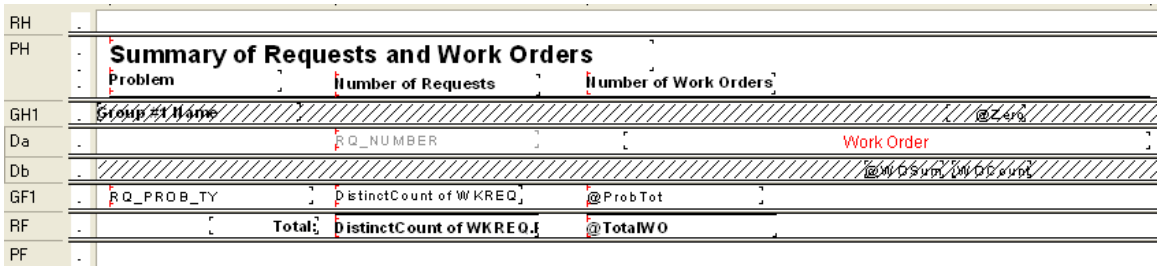
Shared numberVar GrTotWOCount ;

GrTotWOCount

11. Add another ***Detail*** section.
12. Create a column header, **Number of Work Orders**, and place the formulas as follows:



13. In ***Section Expert***, make sure you select *Suppress Blank Section* for the ***Detail*** section with the **Work Order** subreport.
    - Suppress sections as follows:



Preview

Summary of Requests and Work Orders

Problem	Number of Requests	Number of Work Orders
Concrete Sidewalk Repair	2	0
TS - Flashing	3	2
TS - Misalignment	3	0
TS - Red Out	3	2
TS - Timing	4	3
Total:	**15**	**7**

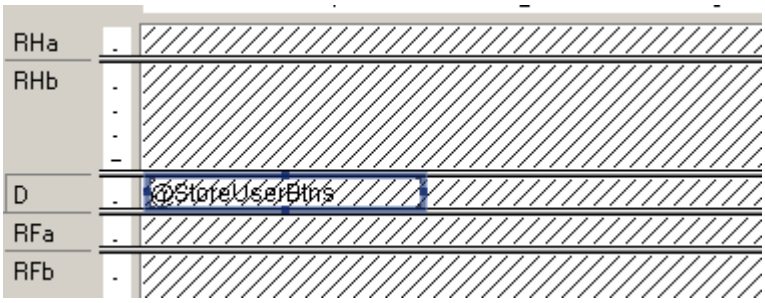
Unlinked Subreports

There are times when the subreports do not need to be linked. For example, you won't need to link subreports if there are no connections to the main report. In this situation, the subreport can share data using variables but does not necessarily share a database.

In *Lucity* software there are two ways to alter button captions to suit individual needs:

- Edit the appropriate text object in the report as discussed in the related Beginning Crystal workbooks.
- Pull the value that is stored in the database and display it on the report using an unlinked subreport.

Lucity uses this type of unlinked subreport in many of the detailed reports. It is hidden in a Report Header subreport. It contains a formula declaring variables that are associated with the User button captions in the Custom tab.



The **@StoreUserBtns** formula appears on the following page. Note that this is only part of the formula. We've included this to make you aware of what happens behind the scenes.

Any line starting with // is just a comment; it is not part of the formula.

The formula for each User button caption looks something like this:

```
WhilePrintingRecords;
Shared StringVar strUser1;
strUser1
```

There is more information on unlinked subreports and variables in the Advanced Crystal, Supplemental Resources handout.

```
//The subreport that this formula belongs to must contain the following
//Select Expert statement:
//{WTFIELDS.FieldName} like ["BD_USE*"]
//The Fields table reference should match your database.  The two character
//data field prefix and suffix must be changed for each report.
//You must also enter your specific Fields Table name in the "IF" statements
//found below (e.g. - WTFIELDS):
WhilePrintingRecords;
//Enter two-character field prefix here (e.g. - 'BD'):
StringVar ModPrefix:='BD';
StringVar strUser1Field:=ModPrefix+'_USER1CD';
StringVar strUser2Field:=ModPrefix+'_USER2CD';
StringVar strUser3Field:=ModPrefix+'_USER3CD';
StringVar strUser4Field:=ModPrefix+'_USER4';
StringVar strUser5Field:=ModPrefix+'_USER5';
StringVar strUser6Field:=ModPrefix+'_USER6';
StringVar strUser7Field:=ModPrefix+'_USER7';
StringVar strUser8Field:=ModPrefix+'_USER8';
StringVar strUser9Field:=ModPrefix+'_USER9';
StringVar strUser10Field:=ModPrefix+'_USER10';
StringVar strUser11Field:=ModPrefix+'_USER11';
StringVar strUser12Field:=ModPrefix+'_USER12';
StringVar strUser13Field:=ModPrefix+'_USER13';
StringVar strUser14Field:=ModPrefix+'_USER14';
StringVar strUser15Field:=ModPrefix+'_USER15';
Shared StringVar strUser1;
Shared StringVar strUser2;
Shared StringVar strUser3;
Shared StringVar strUser4;
Shared StringVar strUser5;
Shared StringVar strUser6;
Shared StringVar strUser7;
Shared StringVar strUser8;
Shared StringVar strUser9;
Shared StringVar strUser10;
Shared StringVar strUser11;
Shared StringVar strUser12;
Shared StringVar strUser13;
Shared StringVar strUser14;
Shared StringVar strUser15;
If {WTFIELDS.FieldName} = strUser1Field then
    If not IsNull ({WTFIELDS.UserName}) and {WTFIELDS.UserName} <> '' then
    strUser1:={WTFIELDS.UserName}+':'
    Else strUser1:={WTFIELDS.DefaultName}+':';

If {WTFIELDS.FieldName} = strUser2Field then
    If not IsNull ({WTFIELDS.UserName}) and {WTFIELDS.UserName} <> '' then
    strUser2:= {WTFIELDS.UserName}+':'
    Else strUser2:= {WTFIELDS.DefaultName}+':';

If {WTFIELDS.FieldName} = strUser3Field then
    If not IsNull ({WTFIELDS.UserName}) and {WTFIELDS.UserName} <> '' then
    strUser3:= {WTFIELDS.UserName}+':'
    Else strUser3:= {WTFIELDS.DefaultName}+':';
```