**Repair Production Orders**

**Mascidon, LLC**

**March 2022**

Table of Contents

[Repairing Equipment 4](#_Toc96591525)

[Figure 1.1 BOM for a Fictitious Pump 5](#_Toc96591526)

[Figure 1.2 Warehouses Used in repair Process 6](#_Toc96591527)

[Receive Repair Item(s) From Customer 7](#_Toc96591528)

[Figure 1.3 Goods Return 7](#_Toc96591529)

[Figure 1.4 Validate Warehouse 8](#_Toc96591530)

[Figure 1.5 Force Warehouse to be REWORK 8](#_Toc96591531)

[Figure 1.6 Quote Created from A Return 9](#_Toc96591532)

[Figure 1.7 Create Button on the Return 10](#_Toc96591533)

[Figure 1.8 Validation to Click on Create Quote Button 10](#_Toc96591534)

[Figure 1.9 Boyum UFs Using Line Loop to Create the Quote 11](#_Toc96591535)

[Figure 1.10 Item Property Setting 11](#_Toc96591536)

[Figure 1.11 Item Master Identifying a Repair Part 12](#_Toc96591537)

[Figure 1.12 Create the Quote Header 13](#_Toc96591538)

[Figure 1.13 Create the Quote Line Items 13](#_Toc96591539)

[Figure 1.14 Save the Quote and Create Reference to the Return 14](#_Toc96591540)

[Figure 1.15 Altered Quote 15](#_Toc96591541)

[Figure 1.16 Buttons Added to Sales Quote Form 16](#_Toc96591542)

[Figure 1.17 Create PDO from Quote 17](#_Toc96591543)

[Figure 1.18 Message if No Items are Repair Items 18](#_Toc96591544)

[Figure 1.19 Message if Selected Item is not a Repair Item 18](#_Toc96591545)

[Figure 1.20 PDO for Quote Item is Already on File 18](#_Toc96591546)

[Figure 1.21 First Step in PDO Create 19](#_Toc96591547)

[Figure 1.22 Create A PDO from a Quote Item 19](#_Toc96591548)

[Figure 1.23 Get PDO Costs to Associate with Quote 21](#_Toc96591549)

[Figure 1.24 Search for Component Lookups 22](#_Toc96591550)

[Figure 1.25 BOM Display from the Double Click 22](#_Toc96591551)

[Figure 1.26 UF Query to Display the BOM for the PDO 23](#_Toc96591552)

[Figure 1.27 Resource Labor Setup 24](#_Toc96591553)

[Figure 1.28 Update the PDO Reference in the Sales Quote 25](#_Toc96591554)

[Figure 1.29 PDO Created from Quote 25](#_Toc96591555)

[Figure 1.30 Review the PDO Referenced Document 26](#_Toc96591556)

[Figure 1.31 PDO(s) Referenced from the Sales Quote 27](#_Toc96591557)

[Figure 1.32 PDO Costs 27](#_Toc96591558)

[Figure 1.33 Sales Quote with Pricing Included 28](#_Toc96591559)

[Customer Approves the Repair 28](#_Toc96591560)

[Figure 1.34 Validation on the Update Button to Populate PDO Sales Order 29](#_Toc96591561)

[Figure 1.35 UF Macro to Update the Sales Order on the PDO 30](#_Toc96591562)

[Figure 1.36 PDO Cost Review 31](#_Toc96591563)

[Figure 1.37 PDO Item Placement Button 31](#_Toc96591564)

[Figure 1.38 B1 Validation Trigger for Cost Estimate Button on PDO 32](#_Toc96591565)

[Figure 1.39 PDO Cost Estimate Query Report 33](#_Toc96591566)

[Figure 1.40 Sales Order with PDO Costs Button 34](#_Toc96591567)

[Figure 1.41 PDO Costs Related to a Sales Order 34](#_Toc96591568)

[Figure 1.42 Sales Order PDO Cost Button Placement 35](#_Toc96591569)

[Figure 1.43 B1 Validation for Sales Order PDO Cost Button 35](#_Toc96591570)

[Figure 1.44 UF Used to Show Sales Order PDO Costs 36](#_Toc96591571)

[Appendix A 38](#_Toc96591572)

# Repairing Equipment

SAP Business One utilizes production orders to track manufacturing costs. In a remanufacturing or repair facility the production orders require that the component item and the manufactured item be the same item. The ‘core’ is being re-worked, so it is its own component material. In the test SAP database I have added 8 items to represent a complete ‘PUMP’. An actual pump would consist of many more parts but this shortened BOM will provide a good example of how to repair a pump. This documentation references the ‘Pump’ as a repair item. It could be brake shoes, generators, or any other piece of machinery that needs repair. In some instances you will not have a BOM for the item being repaired, in other instances you will.

Repairs will use ‘Special’ PDOs to process the repair. ‘Standard’ production orders do not allow users to include a component that is the same item as the ‘manufactured’ item. Also, earlier versions of SAP B1 do not allow even special PDOs to have a component the same item as the manufactured item.

The basic processing of a repair is shown in the diagram below.

Customer Return

Sales Quote

Production Order(s)

Sales Order

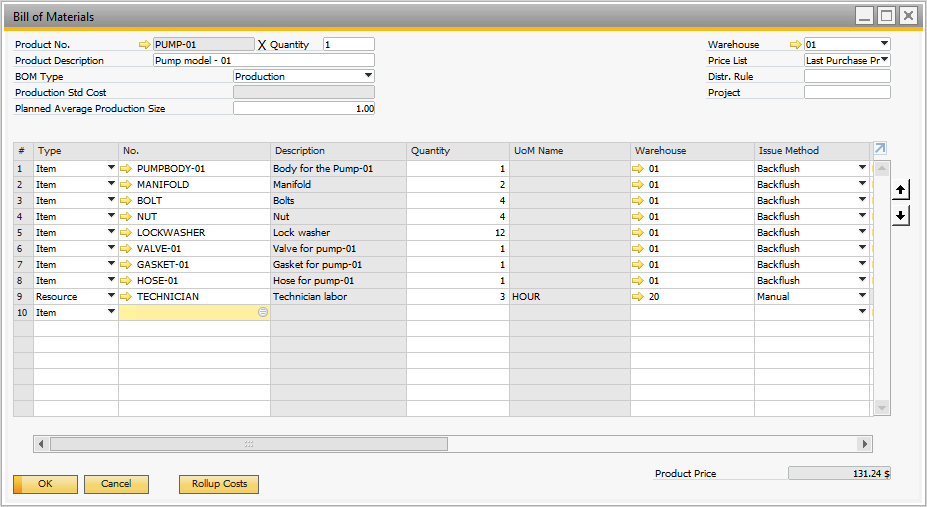
Delivery Order

Invoice

Normal SAP relationships do not account for all of the linkages for repairs. For instance, a customer return is not related directly to a sales quote. The sales quote is not related to the PDO. I have added these ‘References’ so that the trail from the invoice back to the receipt of the repair items in the customer return can be tracked easily.

This repair functionality is based on the Boyum B1 Useability package and it is required.

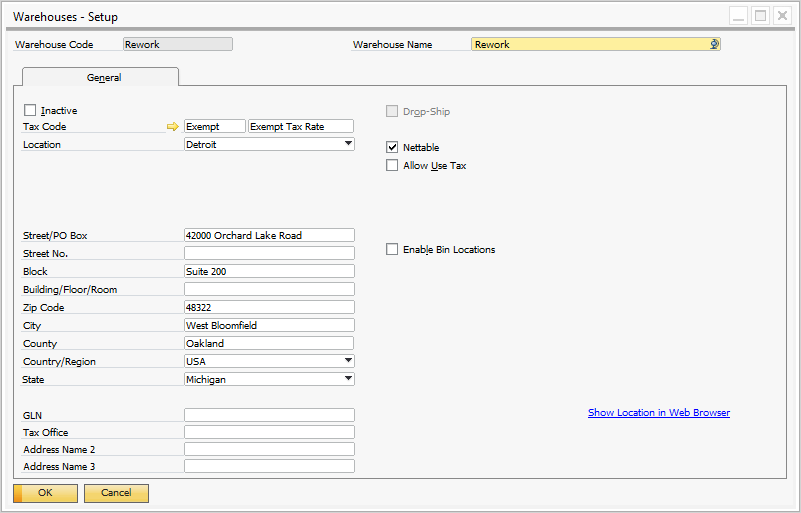
The BOM for the PUMP-01 is shown in Figure 1.1.

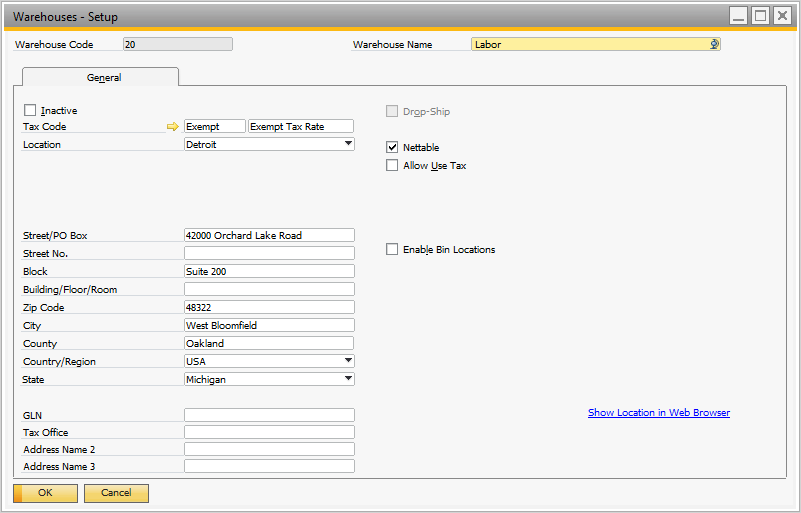


## Figure 1.1 BOM for a Fictitious Pump

Let’s say that your company repairs pumps for customers. The customer sends you 20 pumps and asks that these be repaired and put into refurbished condition. You will throw away washers and gaskets and any other component parts that are damaged on the pump. Then you will expend labor and perhaps purchase more components to repair the pump.

In order to accommodate both labor and rework inventory items let’s create 2 warehouses in SAP– Labor and Rework. These are shown in Figure 1.2. I used a numeric warehouse code for labor (20) and an alpha warehouse code for rework (REWORK). You can create these 2 warehouses as numbers or alpha. Note: the queries and Boyum validations and universal functions use these warehouses within the code. Alter the code if you change the ids for the rework and labor

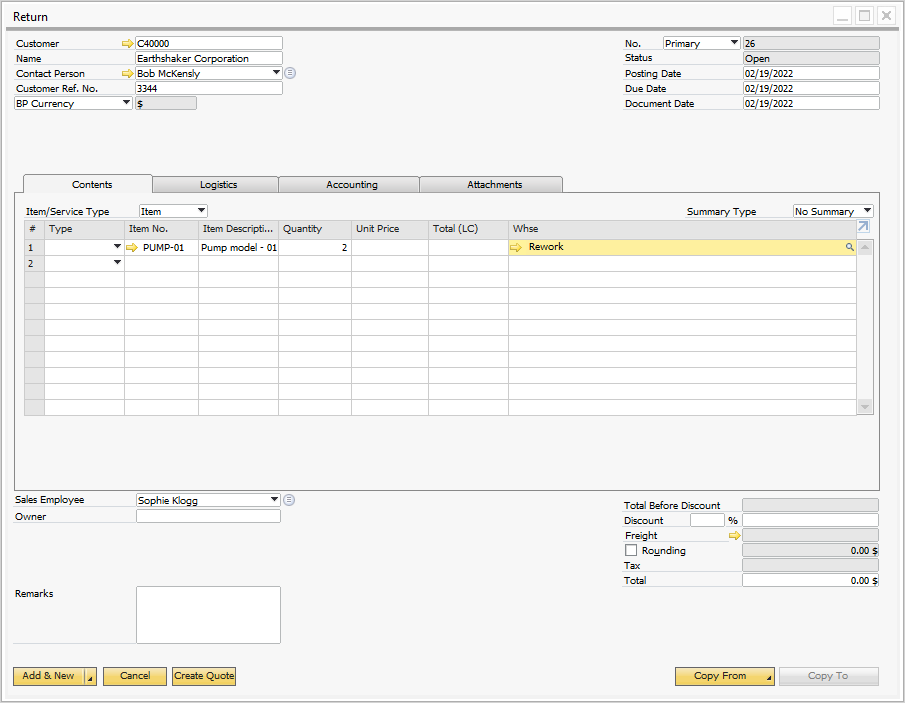




## Figure 1.2 Warehouses Used in repair Process

# Receive Repair Item(s) From Customer

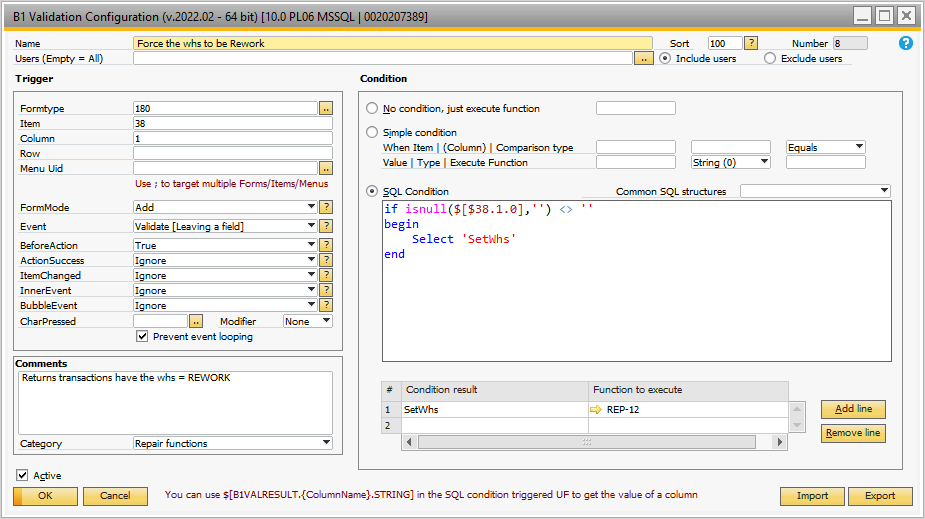
The first step in the repair process is that you receive the pumps to be repaired from a customer. From our viewpoint we need to identify the pumps as belonging to a specific customer and we should place the pump in the ‘Rework’ warehouse to isolate it from new pumps that are being sold to customers. A goods return transaction is used to do this. Refer to Figure 1.3. Note that the unit price is zero – because you do not own the pump being placed into Rework inventory. The warehouse defaults to ‘Rework’ using a Boyum UF and Validation. It is shown in Figure 1.4 and 1.5 respectively.



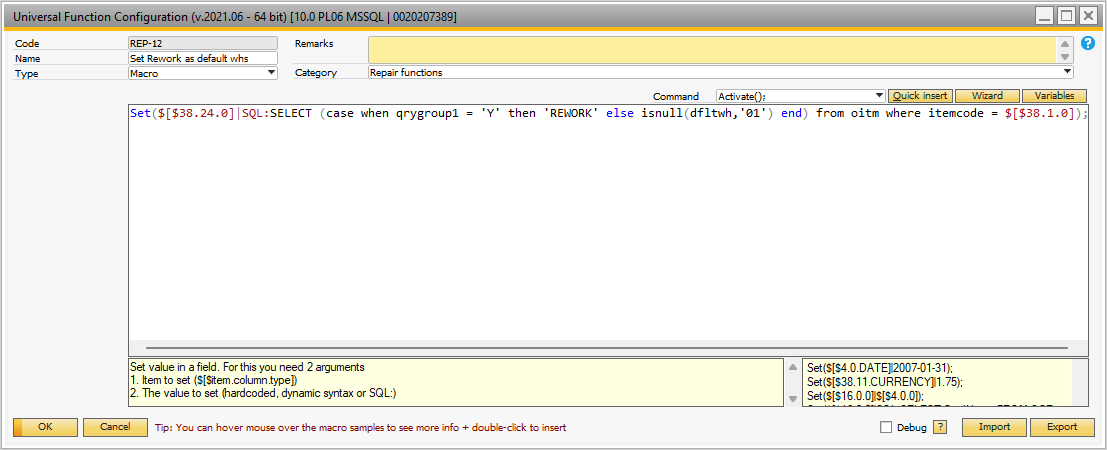
## Figure 1.3 Goods Return

The following fields must be ‘exposed’ for the Return form, the sales quote form and the sales order form:

* Item code
* Quantity
* Warehouse
* Free Text



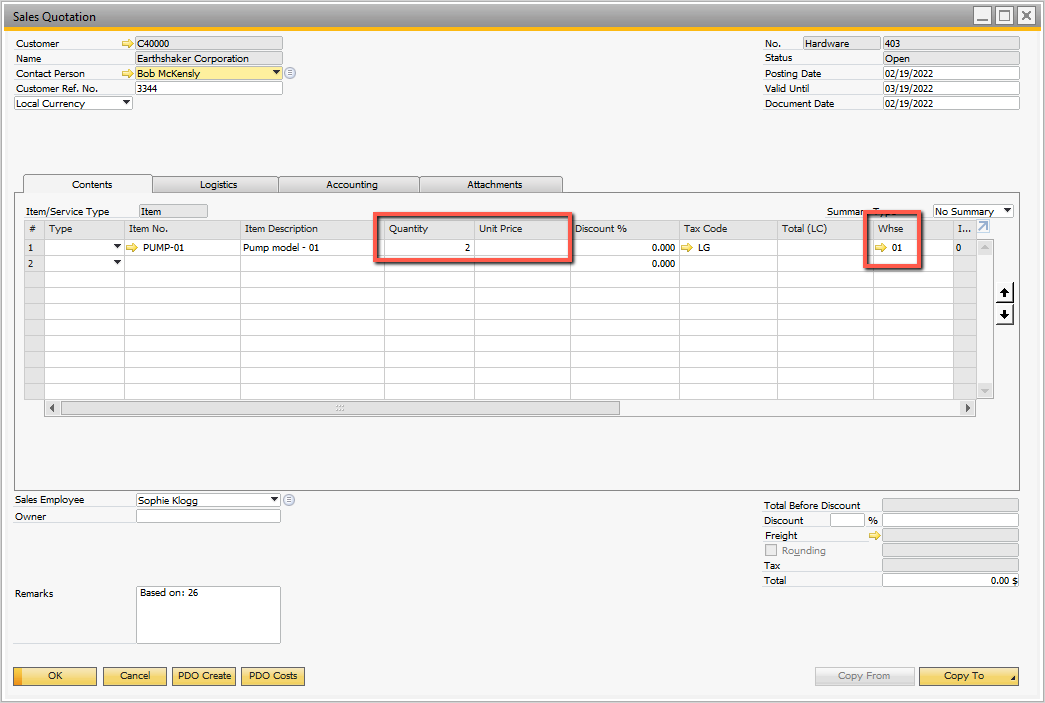
## Figure 1.4 Validate Warehouse



## Figure 1.5 Force Warehouse to be REWORK

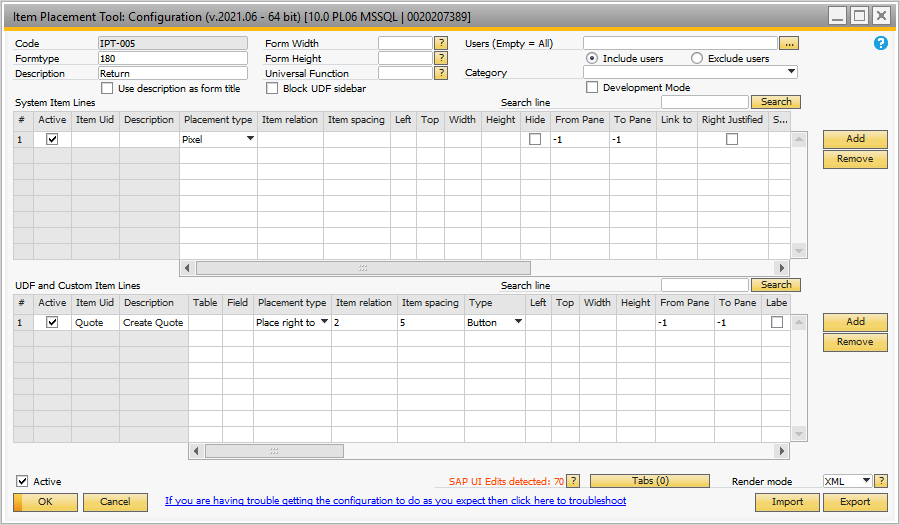
You may need to change this query to match available warehouses within your SAP.

Now that the pump(s) has been received into the ‘Rework’ warehouse someone can be assigned to look over the pump(s) and determine the extent of the repairs required. As they review the pump repair requirements they will create a sales quote for the customer. Note that there is a button on the Sales Return screen ‘Create Quote’. When this button is clicked a sales quote is created. This is shown in Figure 1.6. The quote created still has zero in the price field. It has changed the warehouse from ‘Rework’ to ‘01’ – main warehouse because the repaired pump will eventually be shipped from that warehouse. (Modify this to match the actual warehouses within your SAP).

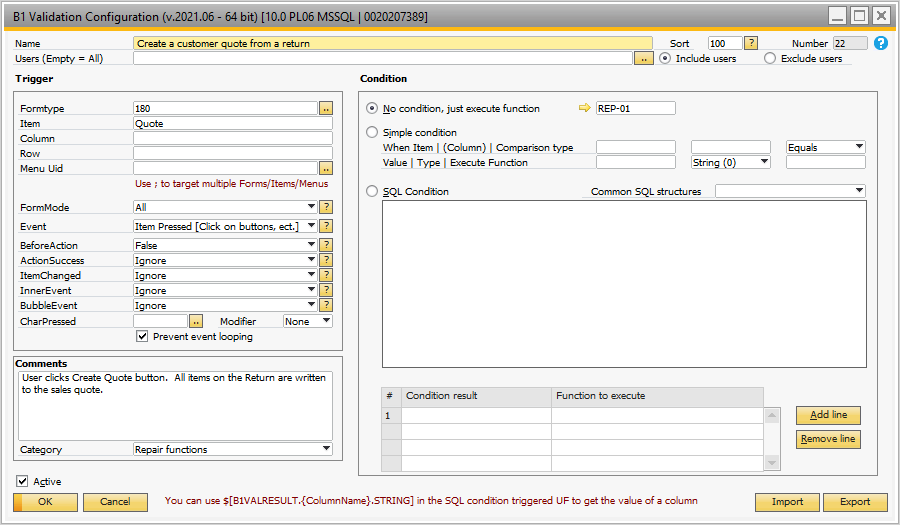


## Figure 1.6 Quote Created from A Return

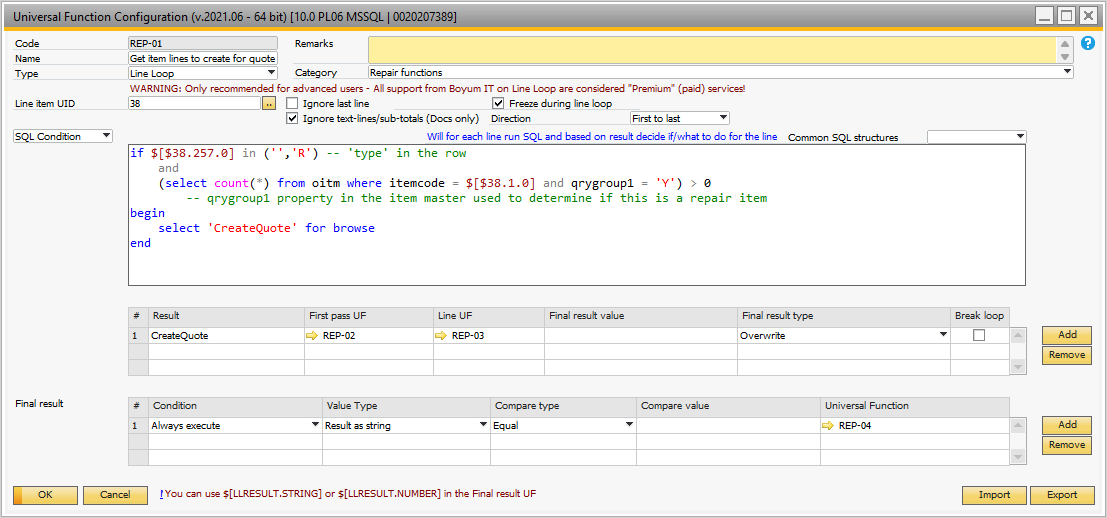
The Boyum functions used to create the sales quote are shown in Figure 1.7 through 1.14.



## Figure 1.7 Create Button on the Return

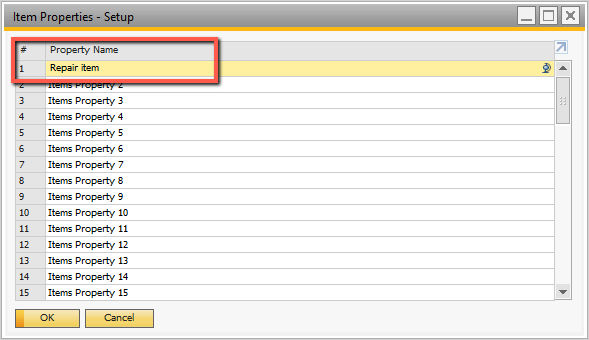


## Figure 1.8 Validation to Click on Create Quote Button



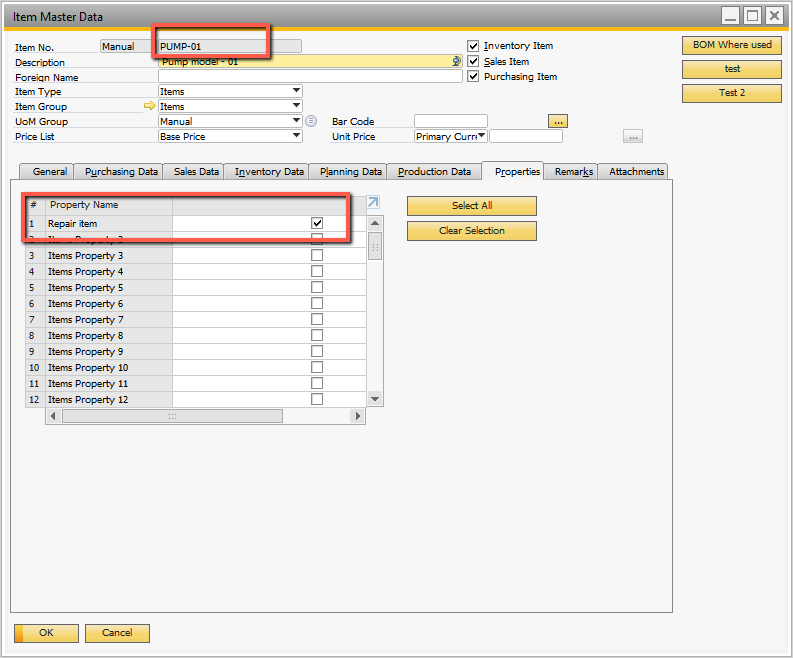
## Figure 1.9 Boyum UFs Using Line Loop to Create the Quote

Note: I have arbitrarily assigned ‘Property 1’ in the item master to identify the item as a repair part. The item property can be renamed within SAP. Refer to Figure 1.10. You see a reference to the qrygroup1 field in Figure 1.9. If you need to use a different property because of current usage of inventory properties, make changes to all queries and Boyum code where ‘qrygroup1’ is referenced to reflect the usage of a different item property.

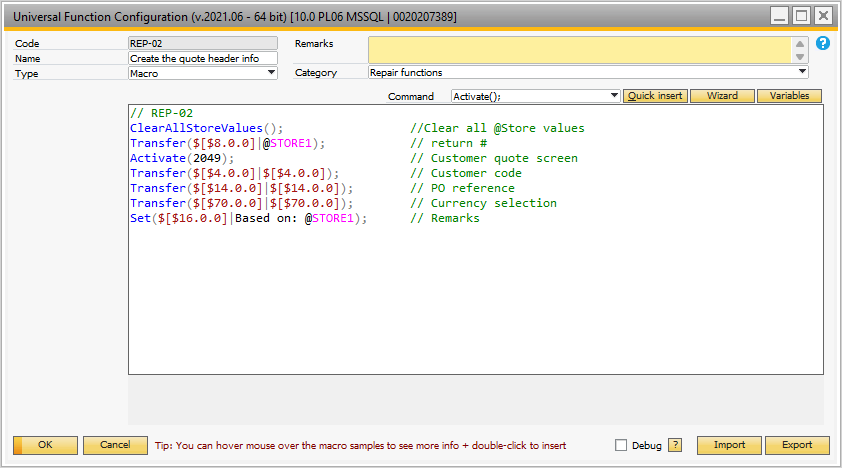


## Figure 1.10 Item Property Setting

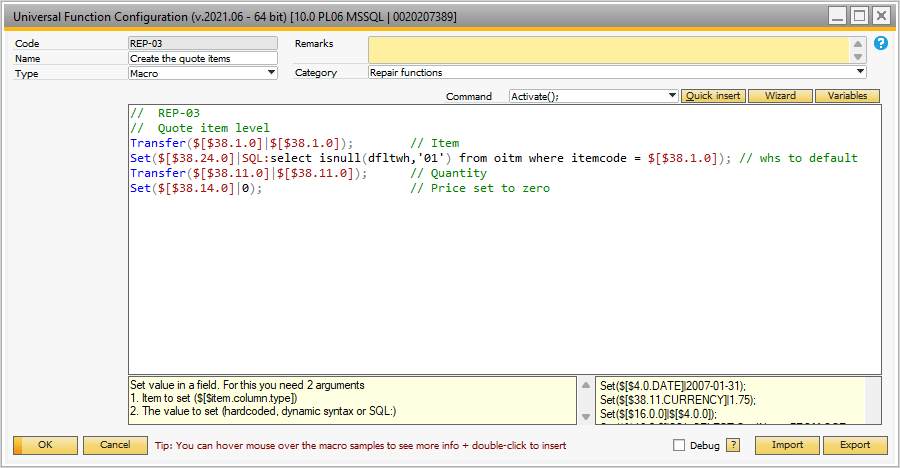
Any items that will be processed as ‘Repair’ items should have this property set. Refer to Figure 1.11 for an example.



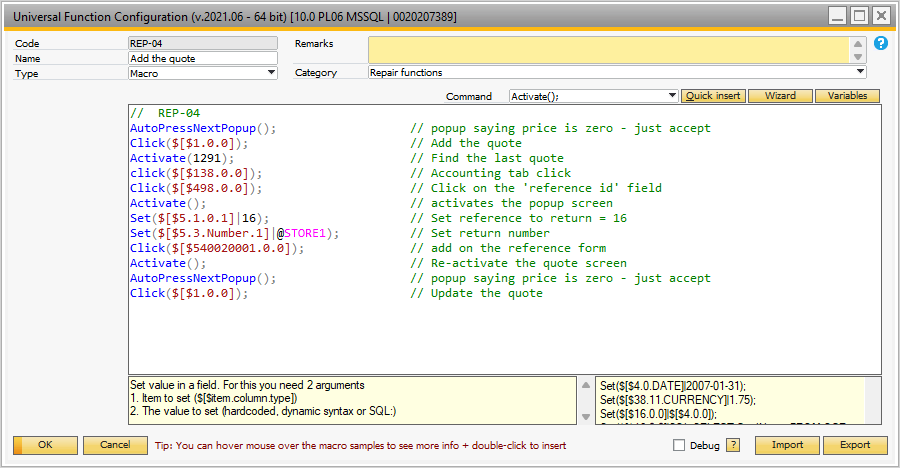
## Figure 1.11 Item Master Identifying a Repair Part



## Figure 1.12 Create the Quote Header

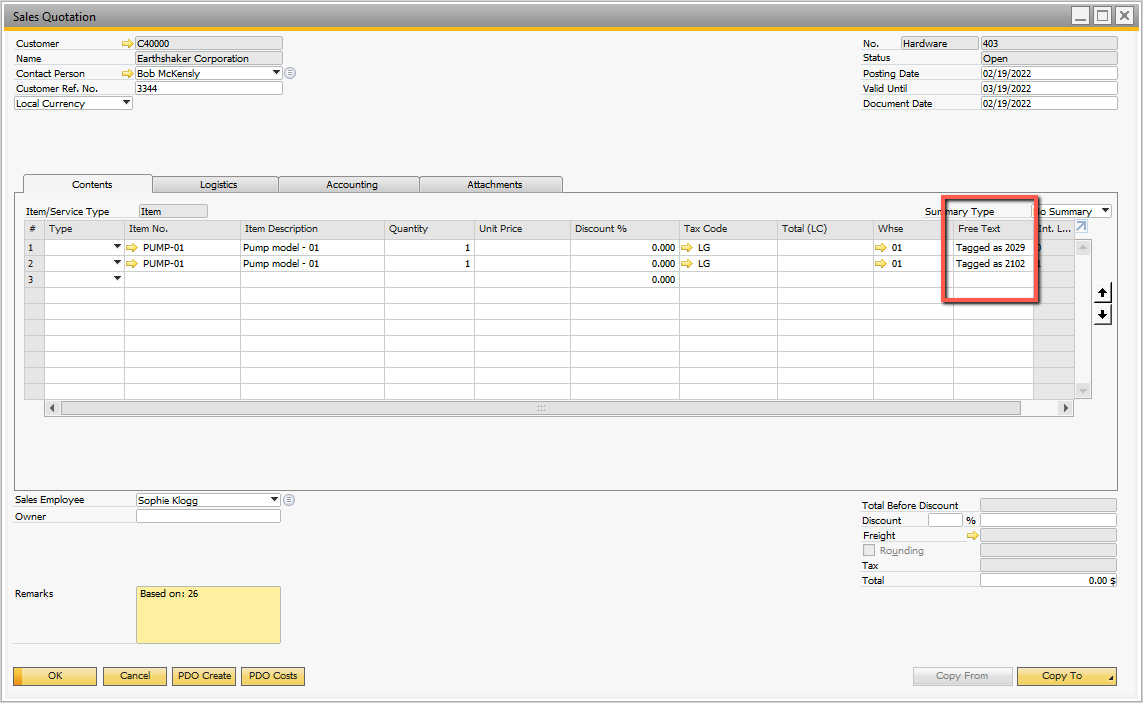


## Figure 1.13 Create the Quote Line Items



## Figure 1.14 Save the Quote and Create Reference to the Return

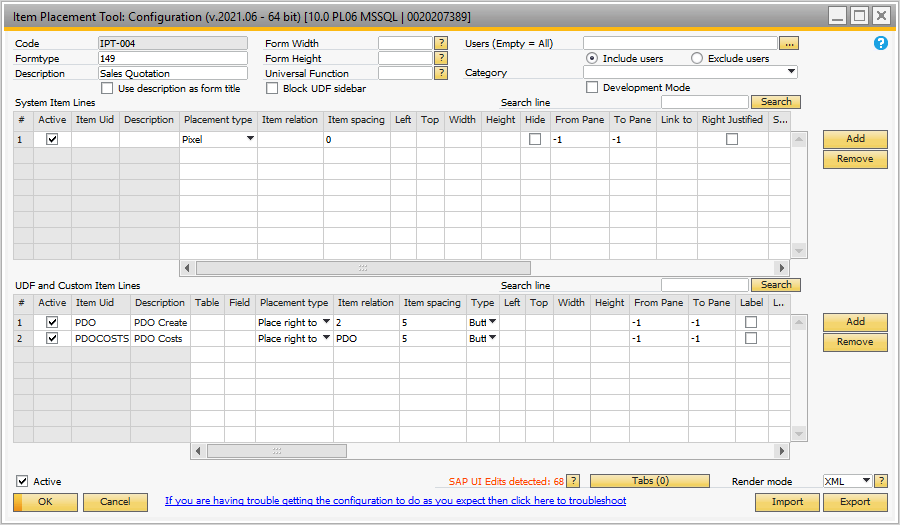
Referring back to Figure 1.7, a quote has been created from the sales return. The pricing is still set at zero. The quantity on this quote is 2. If the pumps need to undergo basically the same refurbishing process, the quote is ready to be updated with pricing. If the pumps require radically different repairs, then the quote should be altered to have 2 lines of ‘PUMP-01’, each with a quantity of 1 – so that we can quote 2 different repair prices to the customer. I have altered the quote to include 2 lines as this is the more detailed repair type and the easier single line quote is a subset of this process. Refer to Figure 1.15. Note: I used ‘free text’ to identify the specific pump.



## Figure 1.15 Altered Quote

There are 2 buttons at the bottom of the Quote screen – PDO Create and PDO Costs. The item placement tool was used to create these 2 buttons. See Figure 1.16.

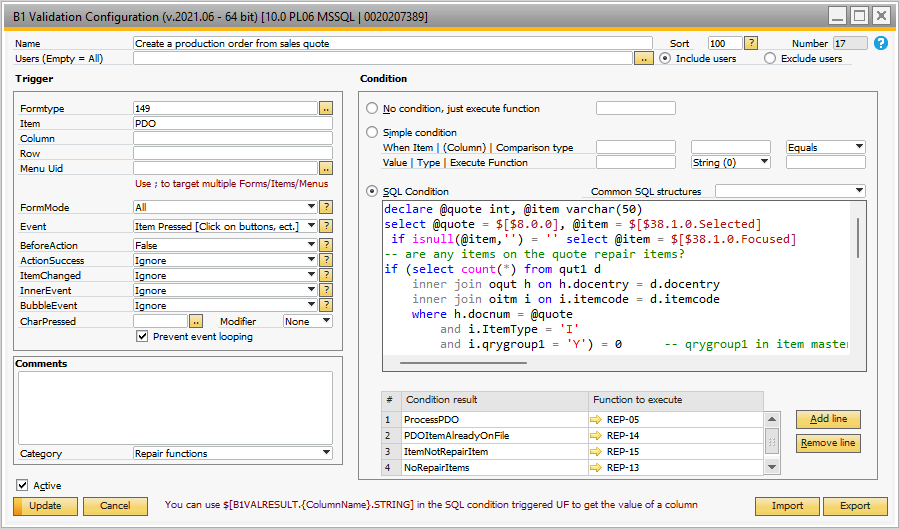
The B1 Validations for these 2 buttons are shown in Figure 1.17 and 1.23.



## Figure 1.16 Buttons Added to Sales Quote Form

The SQL condition in Figure 1.17 is doing the following:

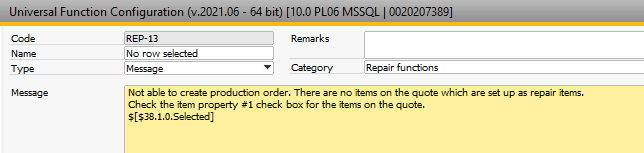
1. If none of the quote items have the flag ‘qrygroup1’ (item property 1) set to ‘Y’, then this quote does not have any items that are repair items. The UF REP-13 message alerts the user and stops the creation of the PDO – see Figure 1.18.
2. If the selected (highlighted) item on the quote has the flag ‘qrygroup1’ (item property 1) set to ‘Y’, then this quote item is not a repair item. The UF REP-15 is called with a message – see Figure 1.19.
3. If a PDO has already been created for the selected item the UF REP-14 sends a message to the user as shown in Figure 1.20.
4. If the PDO is to be created, the UF REP-05 is called. Refer to Figure 1.21 for the steps required.



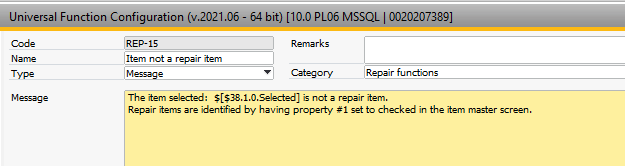
## Figure 1.17 Create PDO from Quote

The SQL script in the validation in Figure 1.17 is:

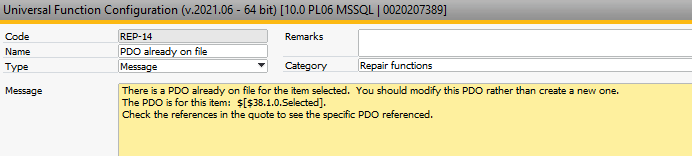
declare @quote int, @item varchar(50)select @quote = $[$8.0.0], @item = $[$38.1.0.Selected] if isnull(@item,'') = '' select @item = $[$38.1.0.Focused]-- are any items on the quote repair items?if (select count(\*) from qut1 d inner join oqut h on h.docentry = d.docentry inner join oitm i on i.itemcode = d.itemcode where h.docnum = @quote and i.ItemType = 'I' and i.qrygroup1 = 'Y') = 0 -- qrygroup1 in item master is used to indicate a repair partbegin select 'NoRepairItems' for browseendelse -- is the selected item a repair item? if (select qrygroup1 from oitm where itemcode = @item) = 'N' begin select 'ItemNotRepairItem' for browse end else -- check if a PDO for this item and quote is already referenced (qut21) if (select count(\*) from qut21 r inner join oqut h on h.docentry = r.docentry inner join owor w on w.docnum = r.refdocnum and r.RefObjType = '202' and w.status <> 'C' where w.ItemCode = @item and h.docnum = @quote and w.comments = $[$38.163.0.Selected]) > 0 begin select 'PDOItemAlreadyOnFile' for browse end else begin select 'ProcessPDO' for browse end



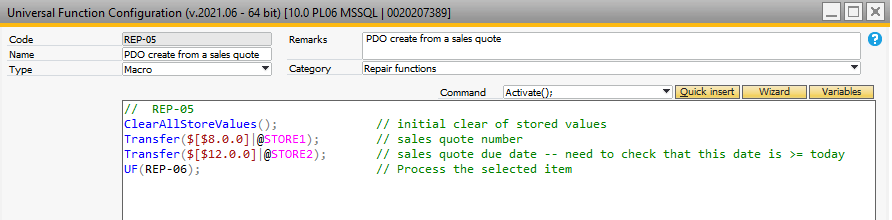
## Figure 1.18 Message if No Items are Repair Items



## Figure 1.19 Message if Selected Item is not a Repair Item

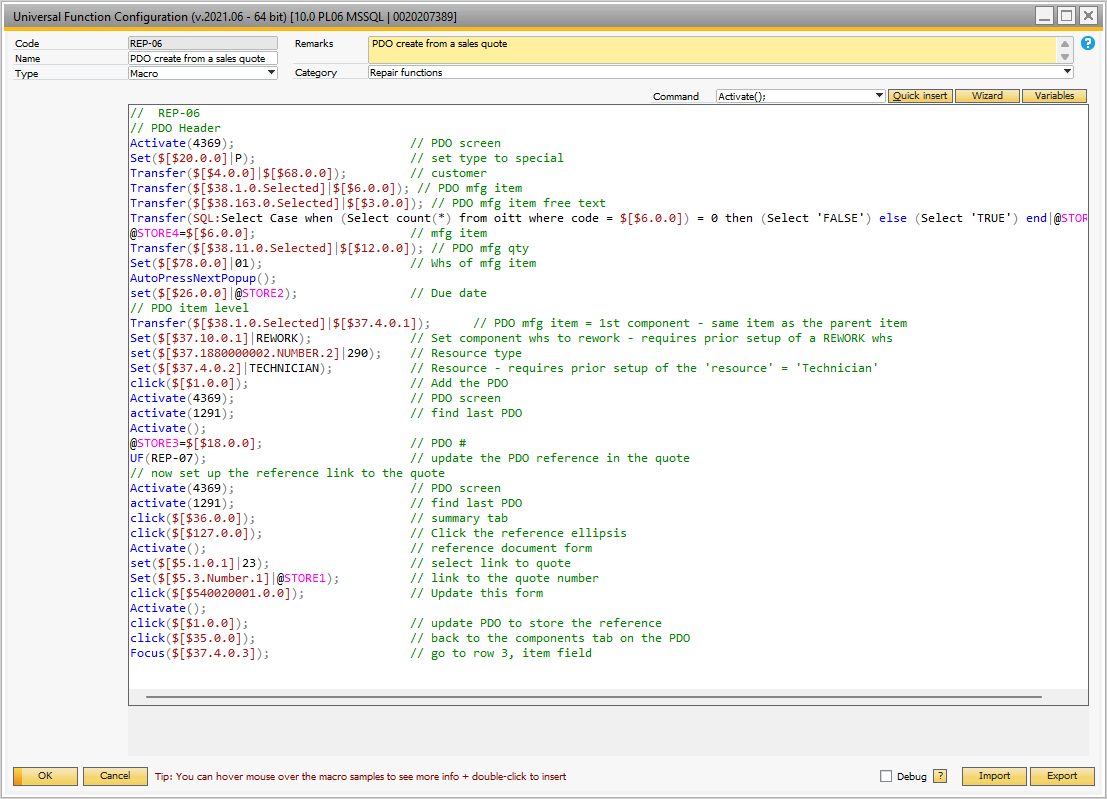


## Figure 1.20 PDO for Quote Item is Already on File



## Figure 1.21 First Step in PDO Create

The last macro command shown in Figure 1.21 calls the UF REP-06. This is shown in Figure 1.22.



## Figure 1.22 Create A PDO from a Quote Item

The macro shown in REP-06 (Figure 1.22) is complex. Let’s review the code.

The PDO header comes from this code:

// PDO Header

Activate(4369); // PDO screen

Set($[$20.0.0]|P); // set type to special

Transfer($[$4.0.0]|$[$68.0.0]); // customer

Transfer($[$38.1.0.Selected]|$[$6.0.0]); // PDO mfg item

Transfer($[$38.163.0.Selected]|$[$3.0.0]); // PDO mfg item free textTransfer(SQL:Select Case when (Select count(\*) from oitt where code = $[$6.0.0]) = 0 then (Select 'FALSE') else (Select 'TRUE') end|@STORE6);@STORE4=$[$6.0.0]; // mfg itemTransfer($[$38.11.0.Selected]|$[$12.0.0]); // PDO mfg qtySet($[$78.0.0]|01); // Whs of mfg itemAutoPressNextPopup();set($[$26.0.0]|@STORE2); // Due date Activate(4369); // PDO screenSet($[$20.0.0]|P); // set type to specialTransfer($[$4.0.0]|$[$68.0.0]); // customerTransfer($[$38.1.0.Selected]|$[$6.0.0]); // PDO mfg itemTransfer(SQL:Select Case when (Select count(\*) from oitt where code = $[$6.0.0]) = 0 then (Select 'FALSE') else (Select 'TRUE') end|@STORE6);@STORE4=$[$6.0.0]; // mfg itemTransfer($[$38.11.0.Selected]|$[$12.0.0]); // PDO mfg qtySet($[$78.0.0]|04); // Whs of mfg itemAutoPressNextPopup();set($[$26.0.0]|@STORE2); // Due date

The header populates the item, quantity, and places the ‘Free text’ pump specific information in the PDO remarks. It sets the manufactured item warehouse equal to the ‘01’ warehouse (the default whs in my database). The PDO is set up as a special PDO because we want to control the components used and not just use the default BOM as a ‘Standard’ PDO would use.

The components for the PDO are set up in the next part of the macro code:

// PDO item levelTransfer($[$38.1.0.Selected]|$[$37.4.0.1]); // PDO mfg item = 1st component - same item as the parent itemSet($[$37.10.0.1]|REWORK); // Set component whs to rework - requires prior setup of a REWORK whsset($[$37.1880000002.NUMBER.2]|290); // Resource typeSet($[$37.4.0.2]|TECHNICIAN); // Resource - requires prior setup of the 'resource' = 'Technician'click($[$1.0.0]); // Add the PDOActivate(4369); // PDO screenactivate(1291); // find last PDOActivate();@STORE3=$[$18.0.0]; // PDO #

I set up a resource to account for the labor portion of the PDO. I have entered 2 components: the first component is the same pump as the manufactured item. The warehouse used is the ‘REWORK’ warehouse because when the customer return is processed this is where the pump resides in inventory. The second is the ‘Technician’ resource as set up in Figure 1.27.

You could add a third labor charge for ‘Evaluation’ if you had a charge for evaluating the repair requirements of the pump. You would need to set up ‘Evaluation’ as another resource.

The next portion of the macro calls the UF REP-07 which adds the ‘reference link’ to the PDO into the sales quote. This will be shown later in this document.

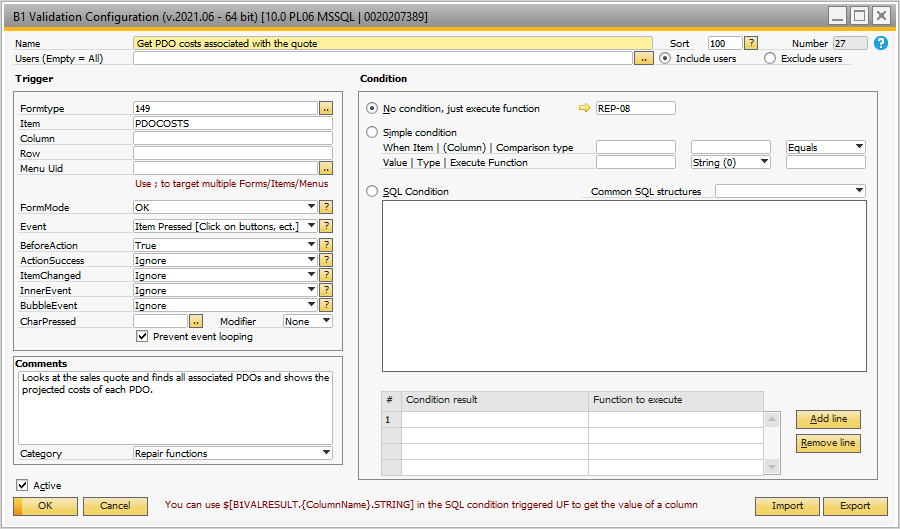
The next portion of the code sets up the reference link to the quote within the PDO. It accesses the link reference popup and stores the sales quote information.

// now set up the reference link to the quoteActivate(4369); // PDO screenactivate(1291); // find last PDOclick($[$36.0.0]); // summary tabclick($[$127.0.0]); // Click the reference ellipsisActivate(); // reference document formset($[$5.1.0.1]|23); // select link to quoteSet($[$5.3.Number.1]|@STORE1); // link to the quote numberclick($[$540020001.0.0]); // Update this formActivate();click($[$1.0.0]); // update PDO to store the referenceclick($[$35.0.0]); // back to the components tab on the PDO

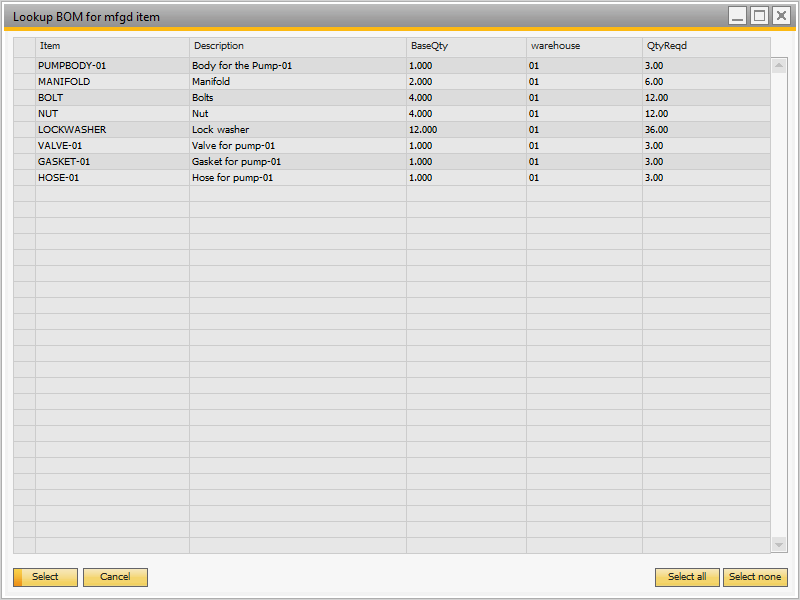
Focus($[$37.4.0.3]); // go to row 3, item field

The last step in the macro is to leave the PDO focus as the component item. From there, the user can double click on the item to bring up a list of all of the components of the manufactured item.

When the user double clicks on the ‘No.’ field ion the PDO, the ‘report’ shown in Figure 1.24 is displayed.

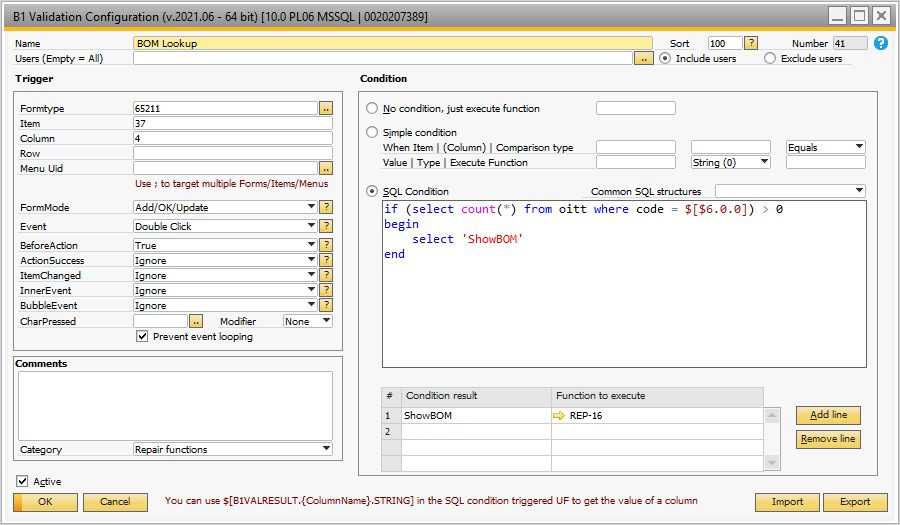


## Figure 1.23 Get PDO Costs to Associate with Quote

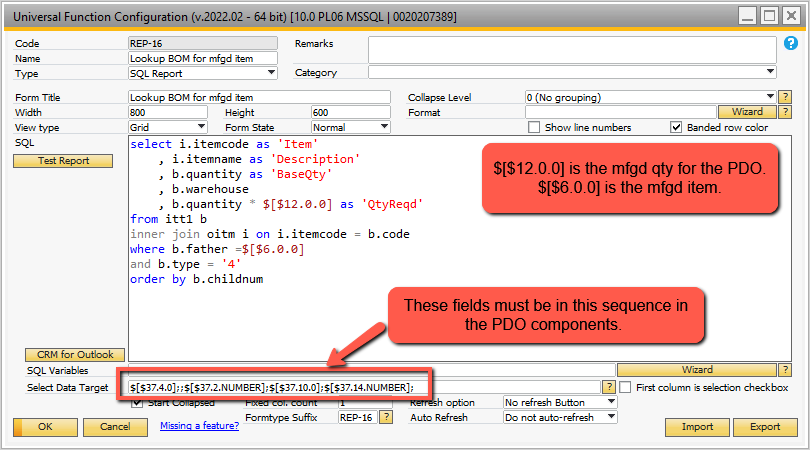


## Figure 1.24 Search for Component Lookups

The ‘double click’ initiates the B1 Validation shown in Figure 1.25. The universal function REP-16 is initiated if the manufactured part has a related BOM structure. The format of the universal function is shown in Figure 1.26.



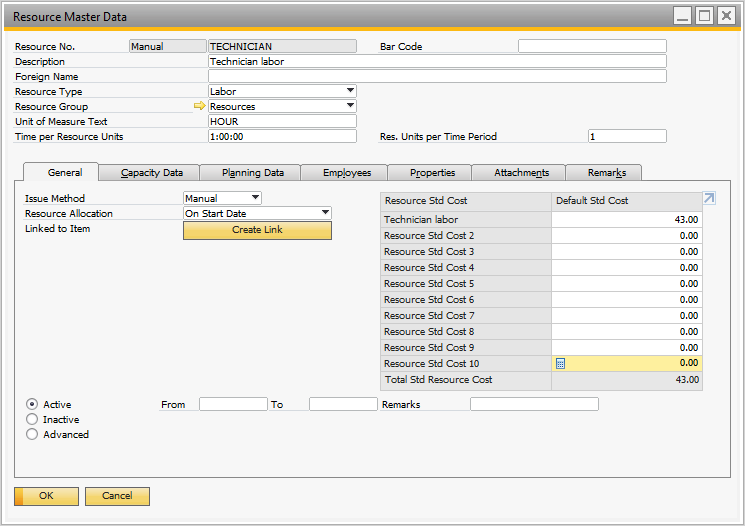
## Figure 1.25 BOM Display from the Double Click

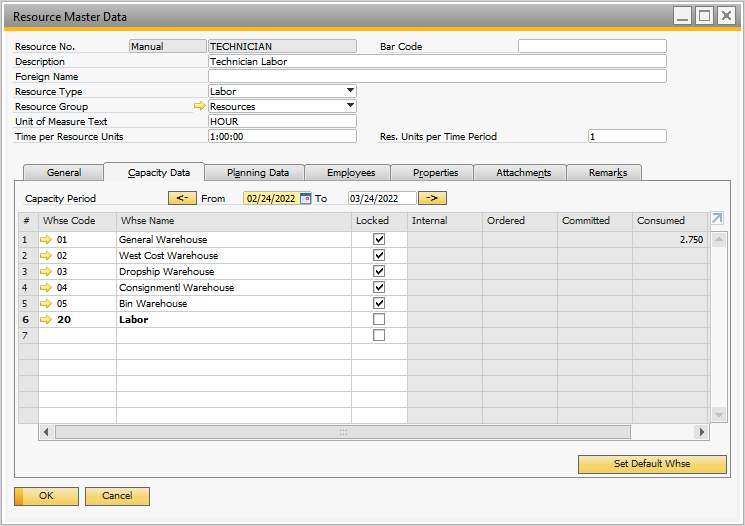


## Figure 1.26 UF Query to Display the BOM for the PDO

Note: the ‘select target’ portion of the universal function moves the ‘selected’ information from the query display to the PDO component lines. i.e. when the report from Figure 1.24 is displayed, the user can highlight one or more lines and then click the ‘Select’ button. The data selected will be written to the component portion of the PDO.

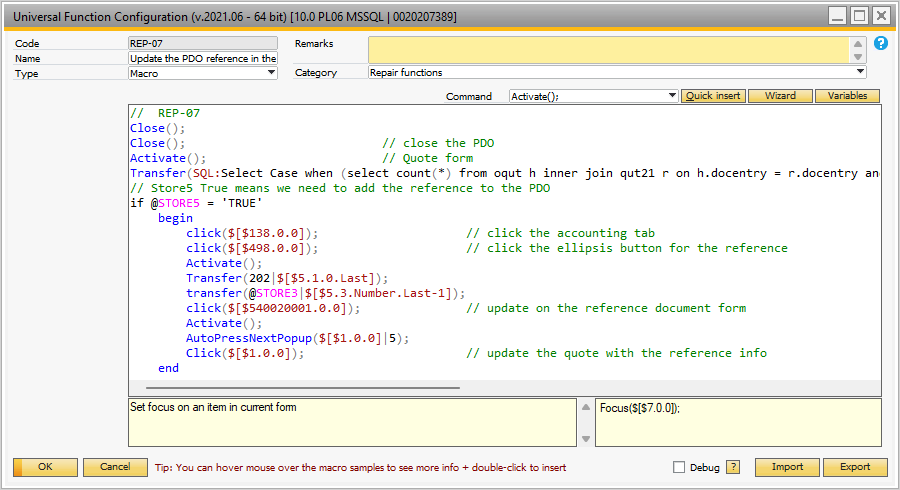
The ‘Select Target’ fields shown must be in the same sequence as the fields. In this case the fields are: Component Item No. - $[$37.4.0]; The item description - $[$37.3.0]; the base quantity - $[$37.2.Number]; the warehouse - $[$37.10.0]; and the planned qty - $[$37.14.Number]. If your PDO form is organized with these fields in a different order on the PDO, then these same fields need to be adjusted in the ‘Select Target’ area of the REP-16 universal function. Note: if the description field is not an active field, then replace $[$37.3.0]; with just ‘;’.





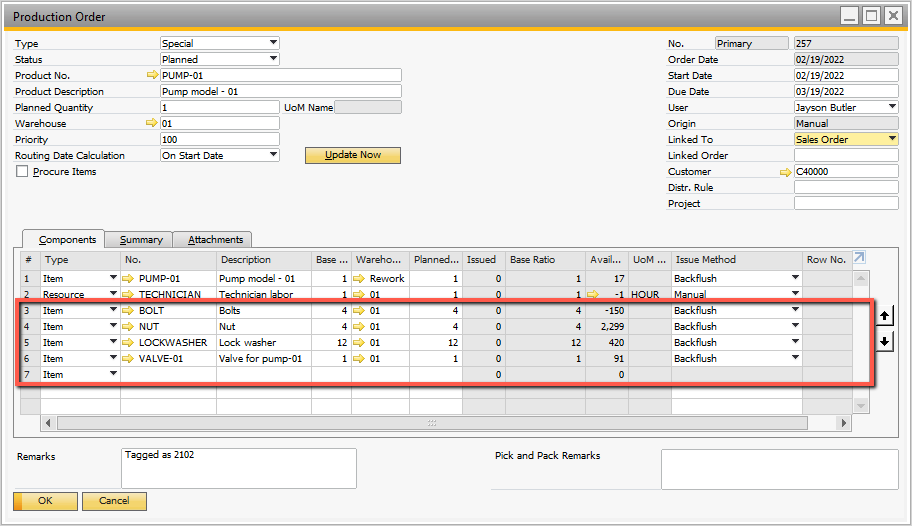
## Figure 1.27 Resource Labor Setup

The UF REP-07 is used to add the reference information regarding the PDO to the sales quote.



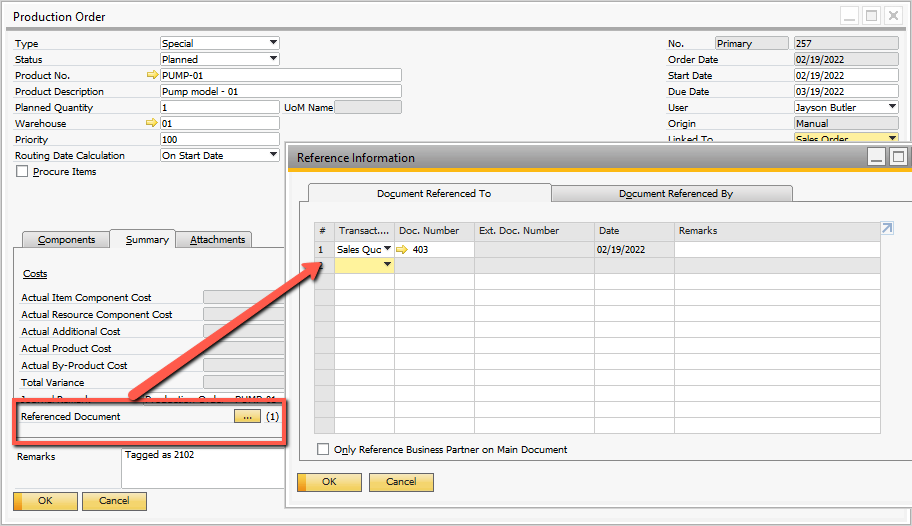
## Figure 1.28 Update the PDO Reference in the Sales Quote

Let’s look at the PDO created for the second line of the quote – see Figure 1.29.



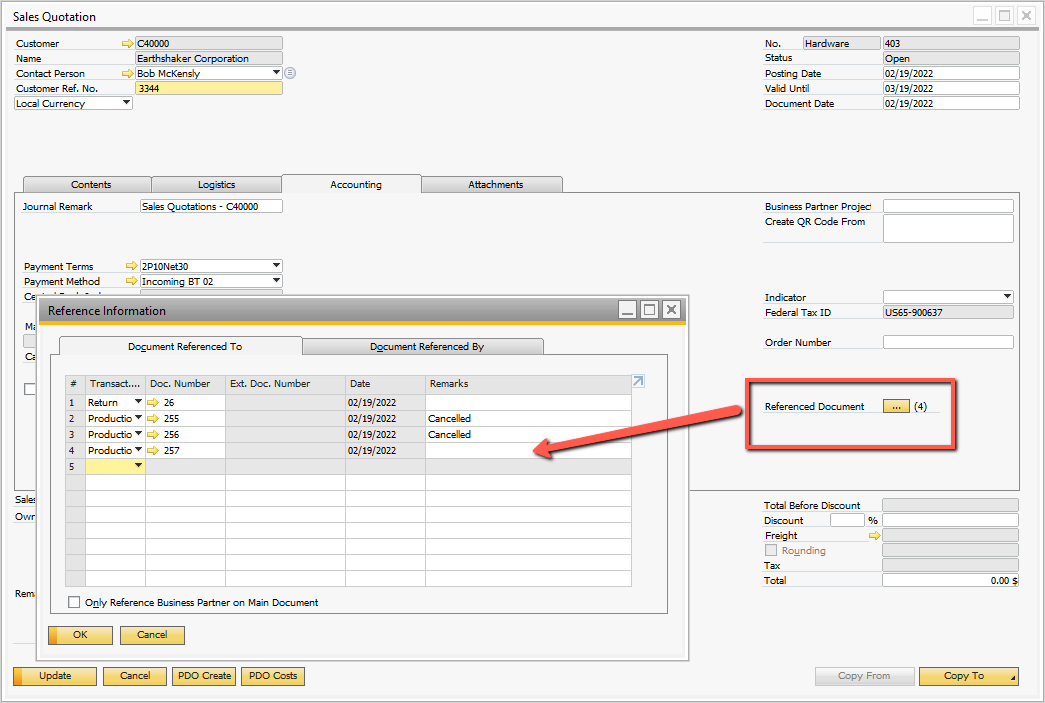
## Figure 1.29 PDO Created from Quote

Note: the line items boxed in Red were selected as components that need to be replaced during the repair. (the double click functionality from the component no. field on the PDO.



## Figure 1.30 Review the PDO Referenced Document

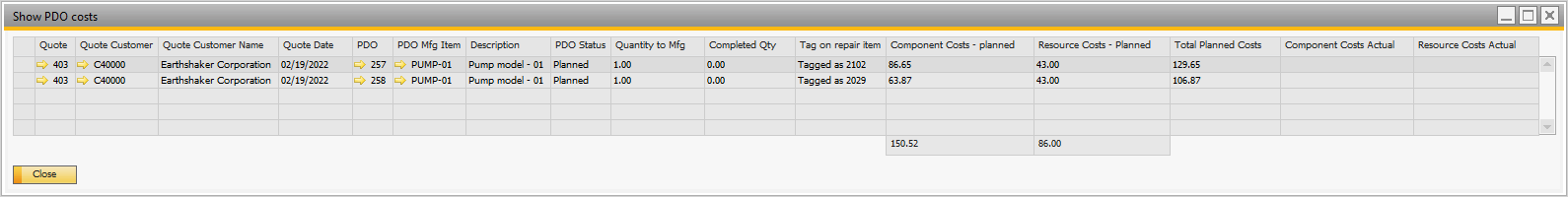
Note: the PDO references the sales quote used to create the PDO.



## Figure 1.31 PDO(s) Referenced from the Sales Quote

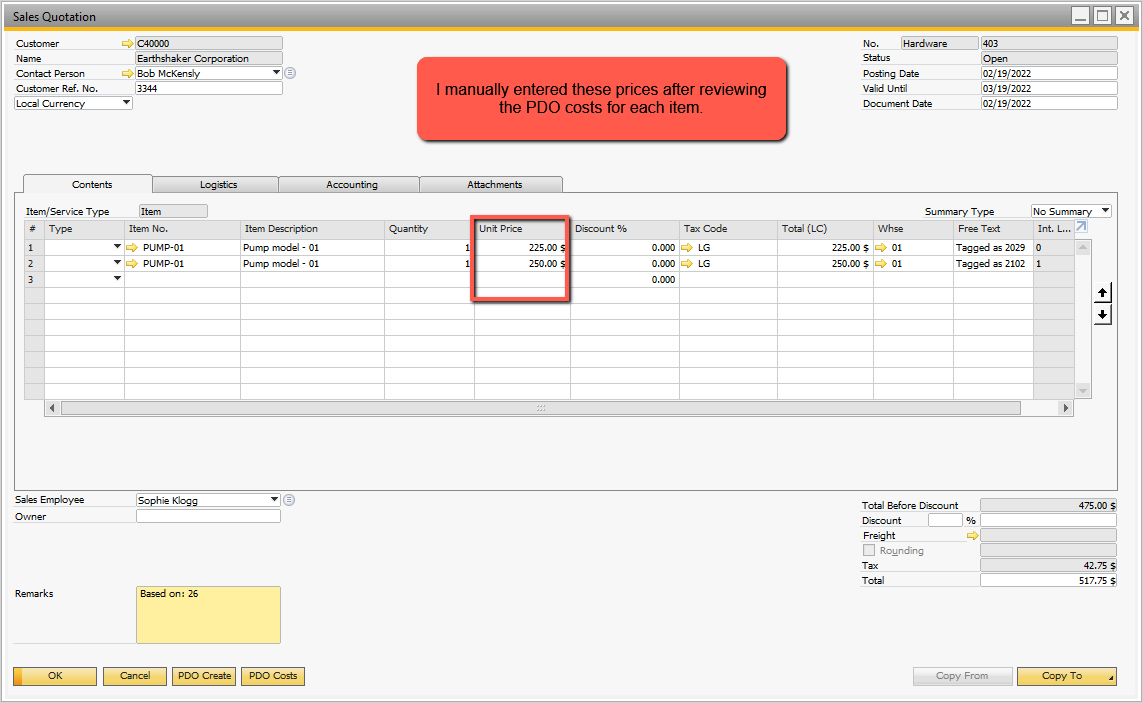
Note: I manually cancelled PDOs created during testing to verify that cancelled PDOs were ignored.

At this point in the process the sales quote costs are still $0. The second button on the sales quote screen ‘PDO Costs’ allows us to review the costs associated with the repair. Clicking on the PDO Costs button shows the estimated PDO repair costs – see Figure 1.32.



## Figure 1.32 PDO Costs

At this point there could be a ‘formula’ added to calculate the line item sales quote prices from the component and resource costs, or the user could enter the pricing. If automating this process, add another Boyum button next to the ‘PDO Costs’ and execute a validation that updates the sales quote lines with the costs based on your formulas. For our example quote I added the prices $250 and $225 to the sales quote. This is shown in Figure 1.33.



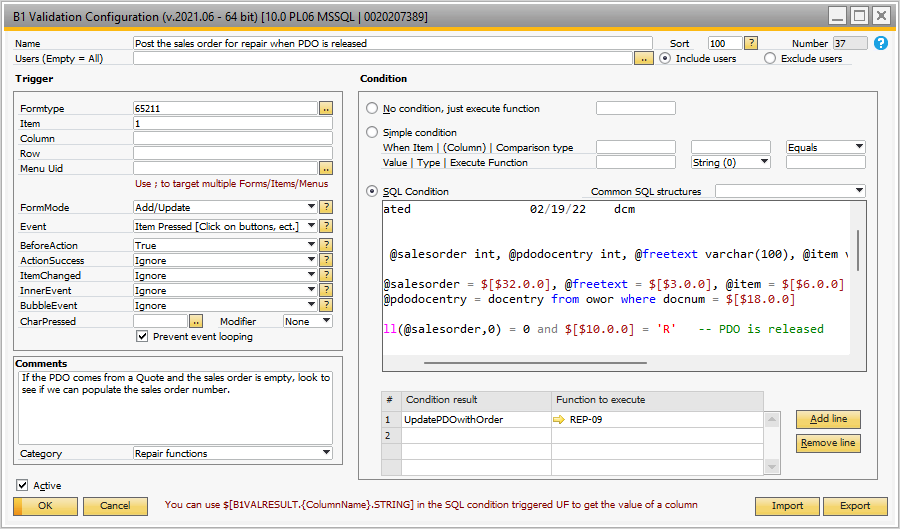
## Figure 1.33 Sales Quote with Pricing Included

# Customer Approves the Repair

The sales quote was sent to the customer and they approved the sales quote.

The sales quote is still open. Use normal SAP functionality to create the sales order from the sales quote.

At this point we can assign the PDOs to the repair technicians. Some time may have passed and you’ve forgotten the PDO numbers. Access the sales quote and then access the ‘reference ids’ and drill to each of the PDOs. The PDOs have a status of ‘Planned’. They do not reference the sales order at this point. Change the status to ‘Released’ and click update. During the ‘Update’ a Boyum validation function is called to update the link on the PDO to the sales order associated with the quote. The Boyum validation is shown in Figure 1.34. The associate SQL condition is shown below Figure 1.34.



## Figure 1.34 Validation on the Update Button to Populate PDO Sales Order

SQL Condition Statement:

/\* Look to see if the sales order number can be added to the PDO Created 02/19/22 dcm\*/declare @salesorder int, @pdodocentry int, @freetext varchar(100), @item varchar(50), @quotedocentry intselect @salesorder = $[$32.0.0], @freetext = $[$3.0.0], @item = $[$6.0.0]select @pdodocentry = docentry from owor where docnum = $[$18.0.0]if isnull(@salesorder,0) = 0 and $[$10.0.0] = 'R' -- PDO is releasedbegin if (select count(\*) from wor5 where docentry = @pdodocentry and RefObjType = '23' and objtype = '202') > 0 -- PDO is related to a quote begin select @quotedocentry = RefDocEntr from wor5 where docentry = @pdodocentry and RefObjType = '23' and objtype = '202' select @salesorder = h.docnum -- get the sales order relationship to the quote for item - free text link from qut1 d inner join ordr h on h.docentry = d.TrgetEntry where d.docentry = @quotedocentry and d.itemcode = @item and (convert(varchar(100),d.FreeTxt) = @freetext or isnull(@freetext,'') = '') if isnull(@salesorder,0) > 0

begin

select 'UpdatePDOwithOrder'

end

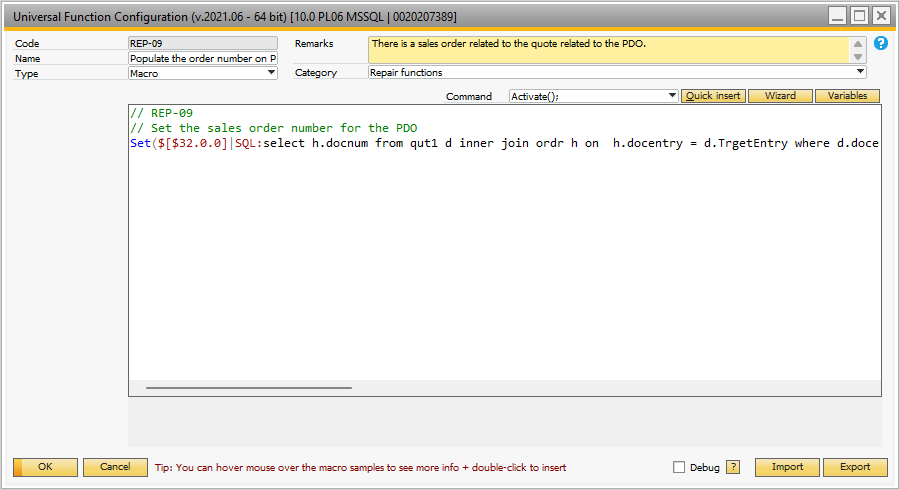
end

end

Assuming the validation conditions are met:

* No current sales order
* PDO is released
* PDO is related to a quote
* Quote is related to a sales order

Then the macro UF REP-09 shown in Figure 1.35 is executed.



## Figure 1.35 UF Macro to Update the Sales Order on the PDO

The SQL associate with the macro is:

select h.docnum

from qut1 d

inner join ordr h on h.docentry = d.TrgetEntry

where d.docentry = (select r.RefDocEntr

from owor w

inner join wor5 r on r.docentry = w.docentry

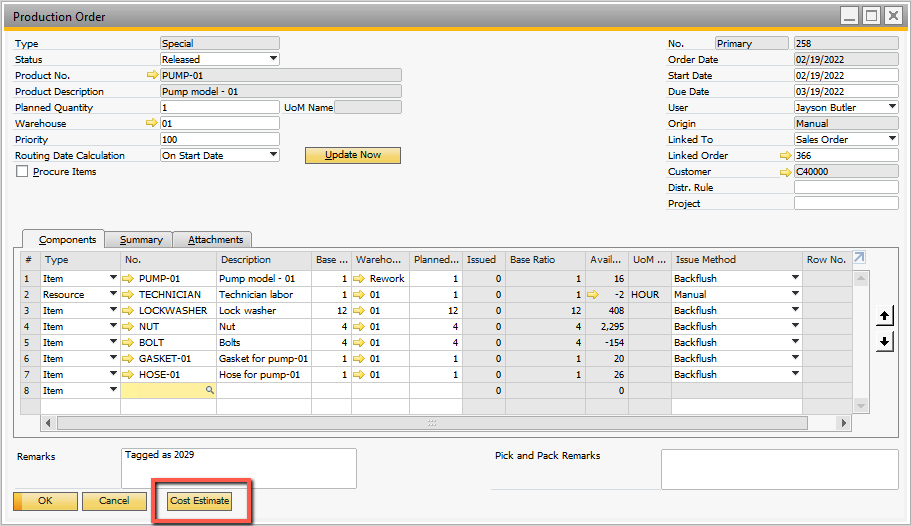
and r.RefObjType = '23'

and r.objtype = '202'

where w.docnum = $[$18.0.0])

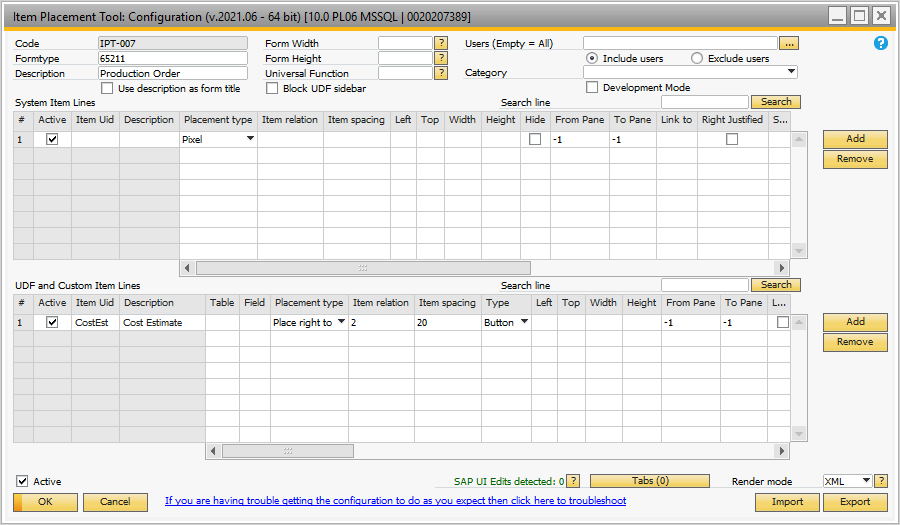
and d.itemcode = $[$6.0.0]

The PDO has a button to review the costs. The ‘Cost’ button is shown in Figure 1.36.



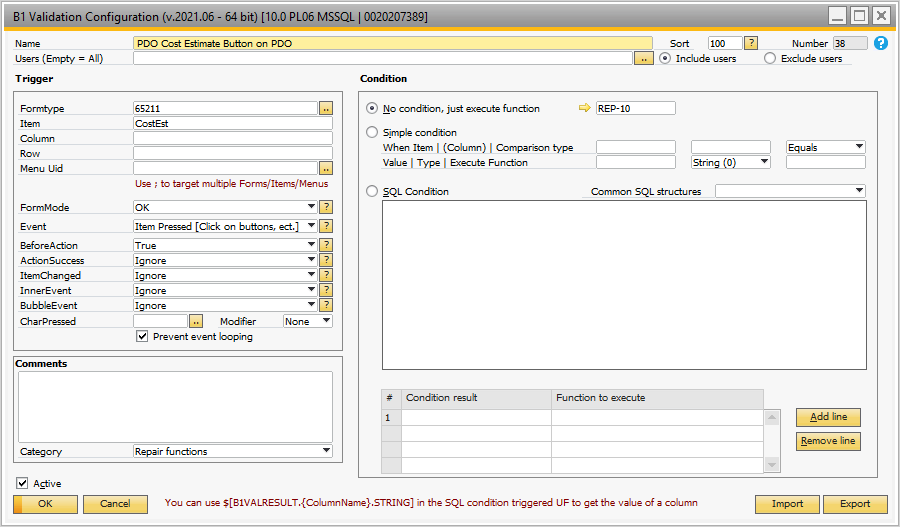
## Figure 1.36 PDO Cost Review

The item placement tool used to display this button is shown in Figure 1.37.



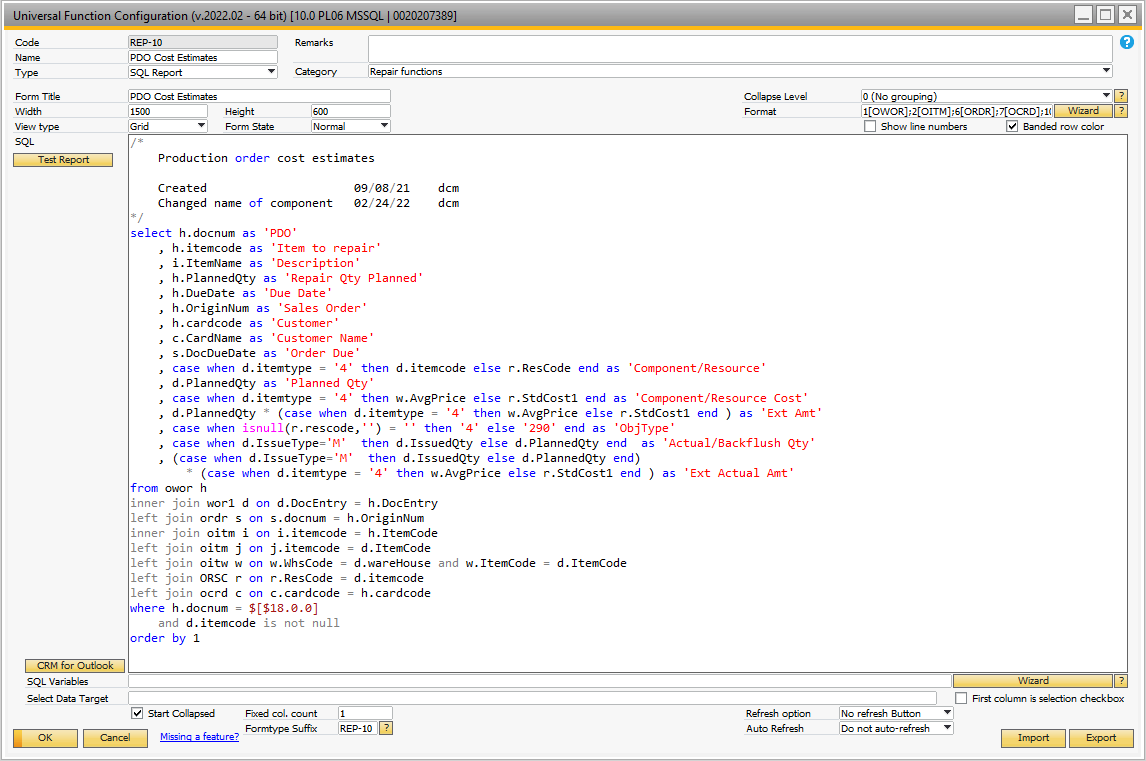
## Figure 1.37 PDO Item Placement Button

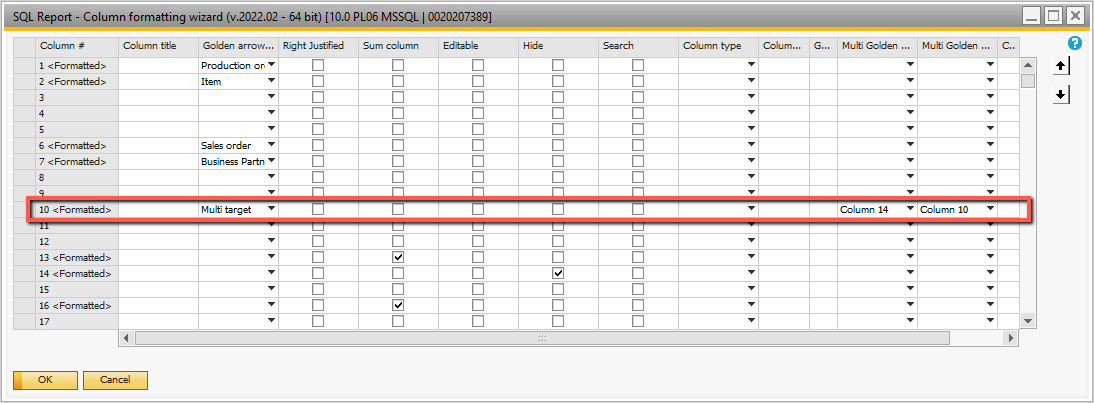
The B1 Validation that is used when this button is clicked is shown in Figure 1.38.



## Figure 1.38 B1 Validation Trigger for Cost Estimate Button on PDO

The UF called by the B1 Validation is shown in Figure 1.39. Note: the data shown in the report includes both items and resources. The golden arrow assignment for the report is controlled by the ‘Multi-target’ setting in the Wizard. This is also shown in Figure 1.39.

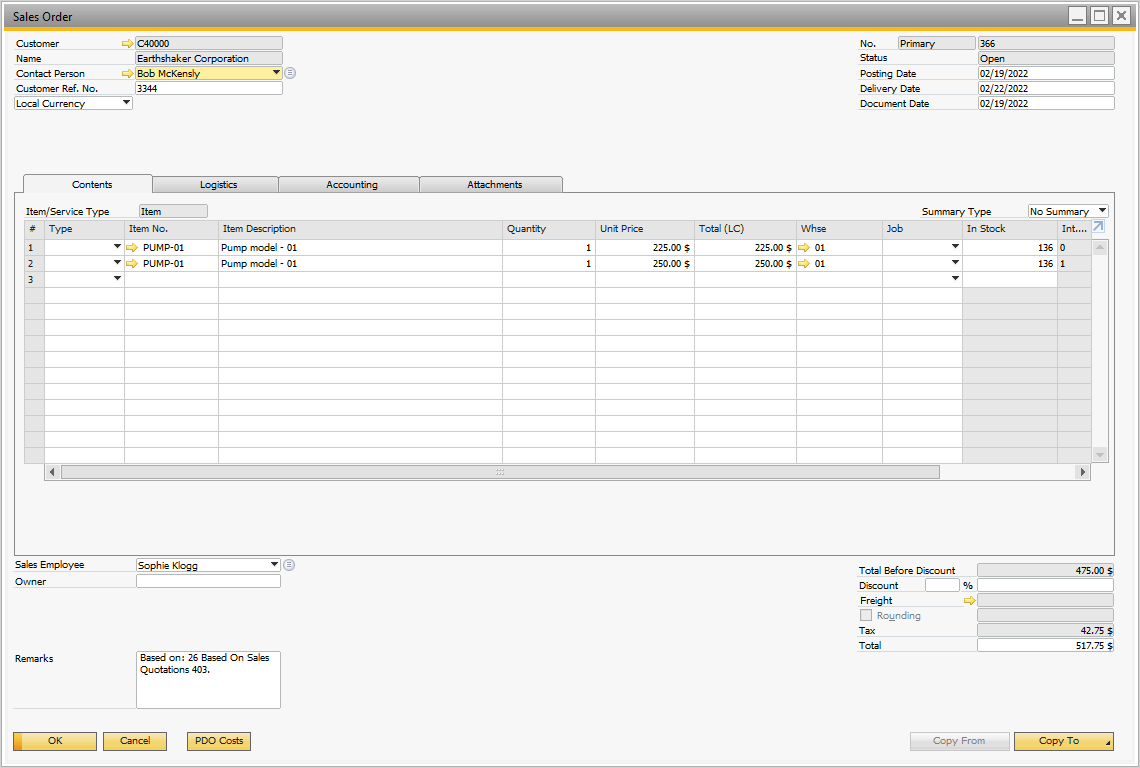




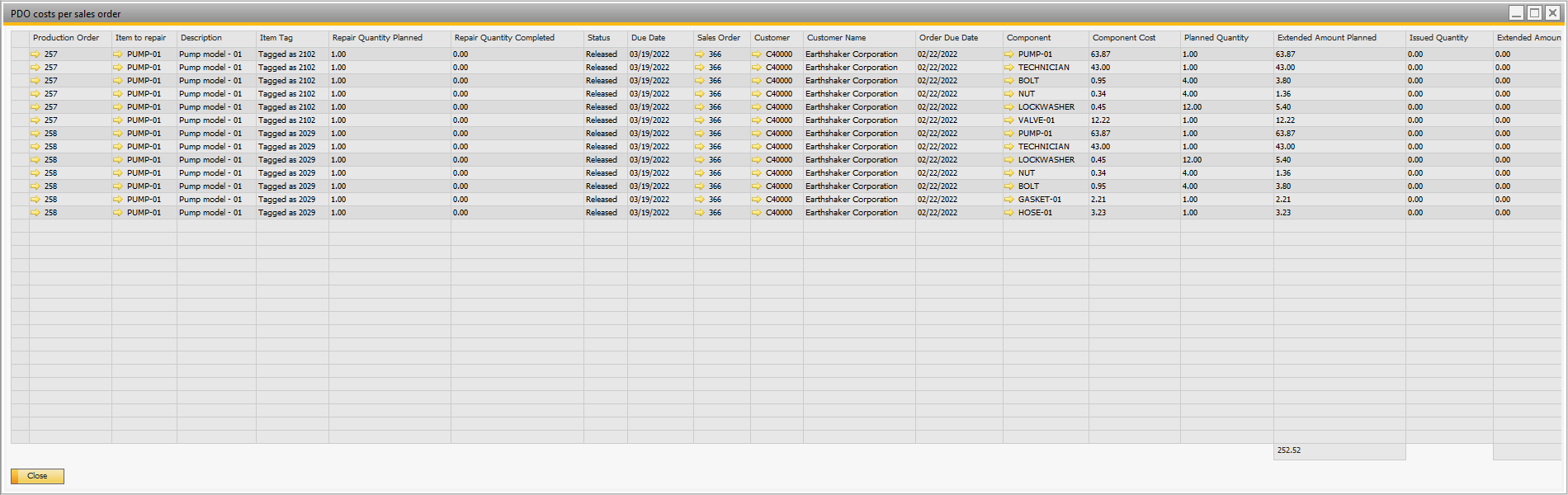
## Figure 1.39 PDO Cost Estimate Query Report

The sales order screen has a button to show all PDO costs related to this sales order. Normally it is just the PDOs that reference the sales order directly, but if a PDO has sub-assemblies that are required and these are linked to the PDO – Sales order, then all of the related PDOs will display.

The sales order screen is shown in Figure 1.40. The report produced when the button is clicked is shown in Figure 1.41.

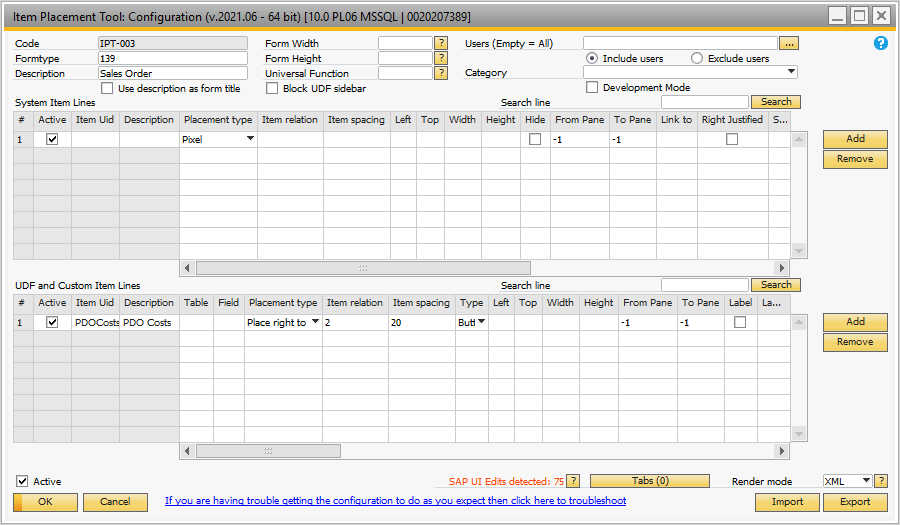


## Figure 1.40 Sales Order with PDO Costs Button

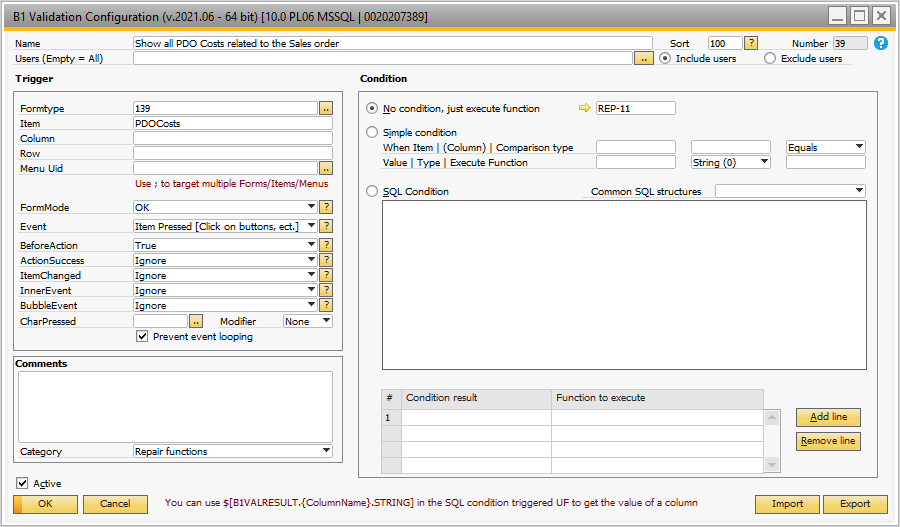


## Figure 1.41 PDO Costs Related to a Sales Order

The item placement screen used to place the button on the sales order is shown in Figure 1.42. The B1 Validation used to process the click of this button is shown in Figure 1.43.

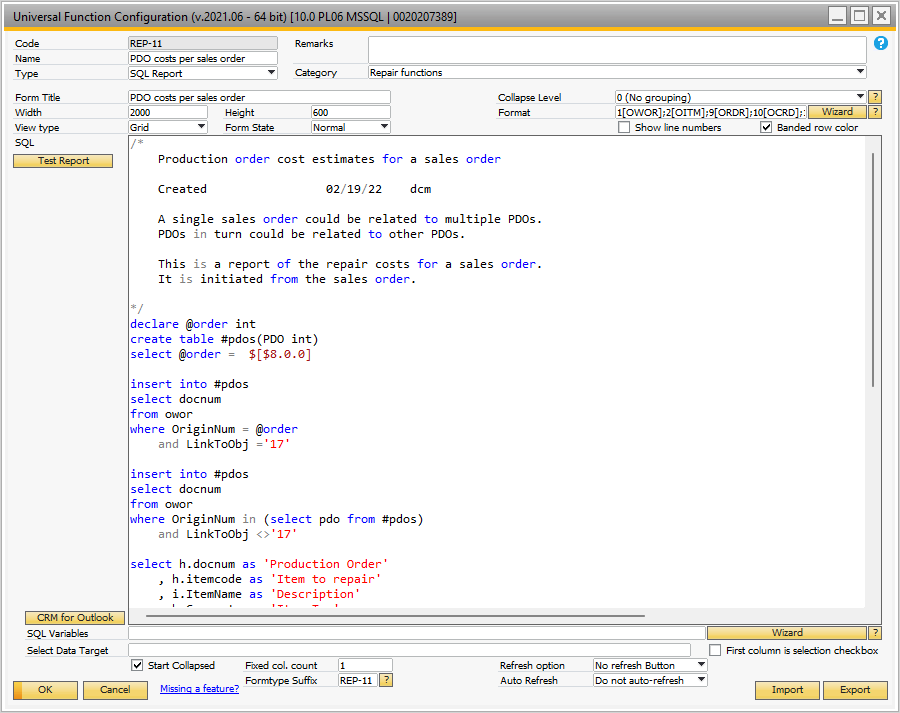


## Figure 1.42 Sales Order PDO Cost Button Placement



## Figure 1.43 B1 Validation for Sales Order PDO Cost Button

The UF REP-11 is the SQL query that displays the costs. This UF is shown in Figure 1.44.



## Figure 1.44 UF Used to Show Sales Order PDO Costs

The SQL in this universal function is shown below.

/\* Production order cost estimates for a sales order Created 02/19/22 dcm A single sales order could be related to multiple PDOs. PDOs in turn could be related to other PDOs. This is a report of the repair costs for a sales order. It is initiated from the sales order.\*/declare @order intcreate table #pdos(PDO int)select @order = $[$8.0.0]insert into #pdosselect docnumfrom owor where OriginNum = @order and LinkToObj ='17'insert into #pdosselect docnumfrom owor where OriginNum in (select pdo from #pdos) and LinkToObj <>'17'select h.docnum as 'Production Order' , h.itemcode as 'Item to repair' , i.ItemName as 'Description' , h.Comments as 'Item Tag' , h.PlannedQty as 'Repair Quantity Planned' , h.CmpltQty as 'Repair Quantity Completed' , case when h.status ='P' then 'Planned' when h.status = 'R' then 'Released' when h.status = 'C' then 'Cancelled' else 'Closed' end as 'Status' , h.DueDate as 'Due Date' , h.OriginNum as 'Sales Order' , h.cardcode as 'Customer' , s.CardName as 'Customer Name' , s.DocDueDate as 'Order Due Date' , case when d.itemtype = '4' then d.itemcode else r.ResCode end as 'Component' , case when d.itemtype = '4' then w.AvgPrice else r.StdCost1 end as 'Component Cost' , d.PlannedQty as 'Planned Quantity' , d.PlannedQty \* (case when d.itemtype = '4' then w.AvgPrice else r.StdCost1 end ) as 'Extended Amount Planned' , d.IssuedQty as 'Issued Quantity' , d.IssuedQty \* (case when d.itemtype = '4' then w.AvgPrice else r.StdCost1 end ) as 'Extended Amount Issued'from owor h inner join wor1 d on d.DocEntry = h.DocEntryleft join ordr s on s.docnum = h.OriginNuminner join oitm i on i.itemcode = h.ItemCodeleft join oitm j on j.itemcode = d.ItemCodeleft join oitw w on w.WhsCode = d.wareHouse and w.ItemCode = d.ItemCodeleft join ORSC r on r.ResCode = d.itemcodewhere h.docnum in (select pdo from #pdos) and h.status <> 'C' -- ignore cancelled PDOs and d.itemcode is not nullorder by 1,2drop table #pdos

# Appendix A

A ‘Configuration Category’ was defined in Boyum to associate all item placement tool, B1 Validations, and Universal Functions as part of this Boyum ‘repair’ add-on.

All of the Boyum functions used in this document are stored in the zipped file included with this documentation.