**BOM Enhancements**

**Mascidon LLC**

**June 2023**

Table of Contents

[BOM Enhancements 5](#_Toc138240193)

[Figure 1.1 Production BOM – SAP 5](#_Toc138240194)

[Figure 1.2 BOM Used In Examples 6](#_Toc138240195)

[Indented BOM 7](#_Toc138240196)

[Figure 2.1 Standard SAP BOM Report 7](#_Toc138240197)

[Figure 2.2 Indented BOM From Mascidon 8](#_Toc138240198)

[Figure 2.3 Modified Indented BOM Report 9](#_Toc138240199)

[Figure 2.4 Adding Gold Arrow to UF BOM-02 10](#_Toc138240200)

[Figure 2.5 Popup Report Question 10](#_Toc138240201)

[Where Used Report 10](#_Toc138240202)

[Figure 2.6 Where Used Indented Report 11](#_Toc138240203)

[Figure 2.7 Where Used Documents Report 11](#_Toc138240204)

[BOM Construction 12](#_Toc138240205)

[Implementation of BOM Approval 12](#_Toc138240206)

[Figure 2.8 Item Property Name Assignment 13](#_Toc138240207)

[Figure 2.10 Visibility of Property 64 Setting 14](#_Toc138240208)

[Figure 2.11 BOM Visibility of Property 64 Setting 14](#_Toc138240209)

[Figure 2.12 Indented BOM – Approval Status Report 15](#_Toc138240210)

[Appendix A – Technical 16](#_Toc138240211)

[SAP File Changes 16](#_Toc138240212)

[Boyum – B1 Validations 17](#_Toc138240213)

[Figure A-1 List of B1 Validations 17](#_Toc138240214)

[BOM Enhancement Queries (Validation) 17](#_Toc138240215)

[Figure A-2 BOM Enhancement Accessed From Item Master 17](#_Toc138240216)

[Figure A-3 BOM-01 UF Decides Which Reports to Run 18](#_Toc138240217)

[Figure A-4 BOM-02 UF Showing Indented BOM 18](#_Toc138240218)

[Figure A-5 BOM-02 Where Used Report 23](#_Toc138240219)

[Figure A-6 Where Used Documents Report 26](#_Toc138240220)

[BOM Where Used and Indented BOM From BOM Form 27](#_Toc138240221)

[Figure A-7 Validation Called from the BOM Form With BOM Reports Button 27](#_Toc138240222)

[Figure A-8 Approval Status Report 27](#_Toc138240223)

[Figure A-9 Approval Process 28](#_Toc138240224)

[BOM Report Access From Forecast 29](#_Toc138240225)

[Figure A-10 Access BOM Report From MRP Forecast Form 29](#_Toc138240226)

[BOM Report from Sales Document – Quote 29](#_Toc138240227)

[Figure A-11 Indented BOM Reporting from Sales Quote 30](#_Toc138240228)

[BOM Report from Sales Document – Sales Order 30](#_Toc138240229)

[Figure A-12 Indented BOM Reporting from Sales Order 30](#_Toc138240230)

[BOM Report from Sales Document – Delivery Order 31](#_Toc138240231)

[Figure A-13 Indented BOM Reporting from Delivery Order 31](#_Toc138240232)

[BOM Report from Sales Document – Invoice 31](#_Toc138240233)

[Figure A-14 Indented BOM Reporting from Invoice 32](#_Toc138240234)

[BOM Reports – from Purchase Order Goods Received 32](#_Toc138240235)

[Figure A-15 Where Used Reporting from PO Goods Received 32](#_Toc138240236)

[BOM Reports – from Purchase Order 33](#_Toc138240237)

[Figure A-16 Where Used Reporting from PO 33](#_Toc138240238)

[Engr Revision Placed on PDO During Add 34](#_Toc138240239)

[Figure A-17 Placing the Engineering Revision on the PDO for the Item Manufactured 34](#_Toc138240240)

[Figure A-18 Macro Placing the Engineering Revision on the PDO 34](#_Toc138240241)

[Set Engineering Revision for Items Being Sold 34](#_Toc138240242)

[Figure A-19 Sales Documents – Setting the Engineering Revision 35](#_Toc138240243)

[Figure A-20 Setting the Engineering Revision on Sales Documents 35](#_Toc138240244)

[Lookup Engineering Revisions for a Customer – Part 35](#_Toc138240245)

[Figure A-21 Show Engineering Revision History By Customer – Part 36](#_Toc138240246)

[Figure A-22 Show Engineering Revision History By Customer – Part 36](#_Toc138240247)

[Figure A-23 Engineering Revision History By Customer – Part 36](#_Toc138240248)

[Resource Lookup 37](#_Toc138240249)

[Figure A-24 Validation Called When Resource Lookup Button Clicked 37](#_Toc138240250)

[Figure A-25 Resource Lookup Report 37](#_Toc138240251)

[Color the item field when property 64 is set – Item Master 37](#_Toc138240252)

[Figure A-26 Validation for Color of Item 38](#_Toc138240253)

[Figure A-27 Set Background Color to Red 38](#_Toc138240254)

[Color the item field when property 64 is set – BOM Form 38](#_Toc138240255)

[Figure A-28 Validation for Color of Item 39](#_Toc138240256)

[Figure A-29 Set Background Color to Red 39](#_Toc138240257)

[Figure A-30 Set Background Color to White 40](#_Toc138240258)

[Item Placement Tool Changes 40](#_Toc138240259)

[Figure A-31 Sales Order Form – Add Button ‘BOM Reports’ 40](#_Toc138240260)

[Figure A-32 Sales Order Quote Form – Add Button ‘BOM Reports’ 41](#_Toc138240261)

[Figure A-33 Production Order Form – Add Engineering Revision UDF 41](#_Toc138240262)

[Figure A-34 Delivery Order Form – Add Button ‘BOM Reports’ 41](#_Toc138240263)

[Figure A-35 Business Partner Form – Add Button ‘Engr Rev Lookup’ 42](#_Toc138240264)

[Figure A-36 Item Master Form – Add Button + UDF 42](#_Toc138240265)

[Figure A-37 BOM Form – Add Button + 2 UDFs 42](#_Toc138240266)

[Figure A-38 Forecast Form – Add Button 43](#_Toc138240267)

[Figure A-39 A/R Invoice Form – Add Button 43](#_Toc138240268)

[Figure A-40 Purchase Order Form – Add Button 43](#_Toc138240269)

[Figure A-41 Purchase Order Goods Receipt Form – Add Button 44](#_Toc138240270)

[Figure A-42 Resource Master Form – Add Button 44](#_Toc138240271)

# BOM Enhancements

The standard SAP BOM provides a BOM report showing all of the indent levels. This report is useful but cannot be extended by the user. It shows the component parts at each level, the quantity required and the price. What if you want to see more detail? Mascidon’s BOM enhancement module provides more detailed BOM reports. Our reports have more information – AND – they can be extended by a user with knowledge of queries / SQL. This extension was developed using SQL and Boyum functionality.

How are BOM’s used within SAP? There are ‘production’, ‘assembly’ and ‘sales’ BOMs. While the enhancements provided work equally well within any BOM type, my concentration is on the ‘Production’ BOM that is used in the manufacturing processes.

These are my assumptions:

* A person or a team of people are responsible to BOM development and maintenance. This is generally delegated to the engineering department.
* Engineering periodically makes improvements to item design – and this may or may not result in BOM changes. At the very least, engineering tracks ‘revisions’ – hopefully tied to CAD drawings or an engineering management software product.

The BOM enhancements are intended to assist with the tracking of the creation and maintenance of BOMs. The BOM enhancements become more useful as your BOMs become more complex. A 10 level BOM requires more effort to implement and maintain than a single level BOM. If making a large piece of capital equipment it may take several weeks to develop the BOM – and the team responsible for this development need to understand how far along in the process they are currently.

Before going into the details of the enhancements, let’s review the standard SAP BOM functionality. The BOM form is shown in Figure 1.1.

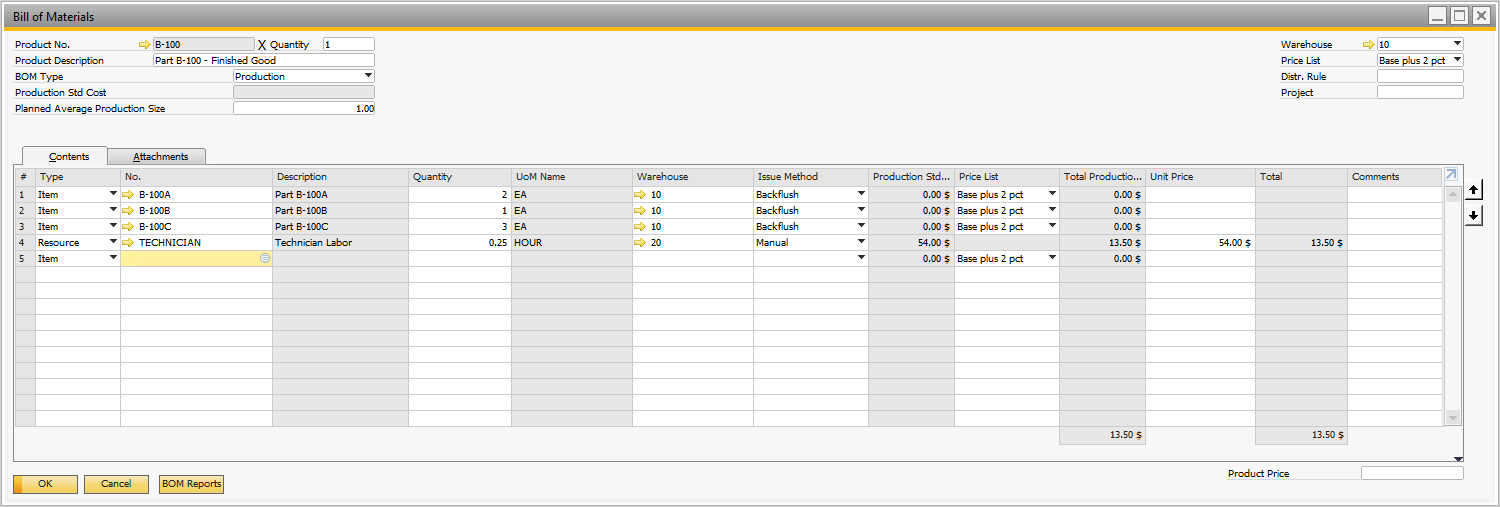


Figure 1.1 Production BOM – SAP

Defining a BOM entails creating the BOM one ‘level’ at a time. The BOM I use for testing is shown in Figure 1.2. It has several features that need to be discussed. The first is that 2 different assemblies used the component part B-400A 🡪 B-300A and B-200F. This is a common happenstance in large BOMs. The second is that 2 manufactured parts – B-200B and B-200C – use the same manufactured component B-300A. Again, this is a common occurrence in large BOMs. It is a 4 level BOM.

**B-100**

**B-200A**

**B-100A**

**B-100B**

**B-100C**

**B-200B**

**B-300A**

**B-200C**

**B-200D**

**B-200E**

**B-200F**

**B-300C**

**B-300B**

**B-400B**

**B-400A**

## Figure 1.2 BOM Used In Examples

BOMs are input into SAP on a level-by-level basis, each assembly is entered separately. In our example BOM, the user enters the BOM with 10 screens, one for each of these assemblies:

* B-100 is the final assembly
* B-100A is a level 1 assembly
* B-100B is a level 1 assembly
* B-100C is a level 1 assembly
* B-200B is a level 2 assembly
* B-200C is a level 2 assembly
* B-200D is a level 2 assembly
* B-200F is a level 2 assembly
* B-300A is a level 3 assembly
* B-300B is a level 3 assembly

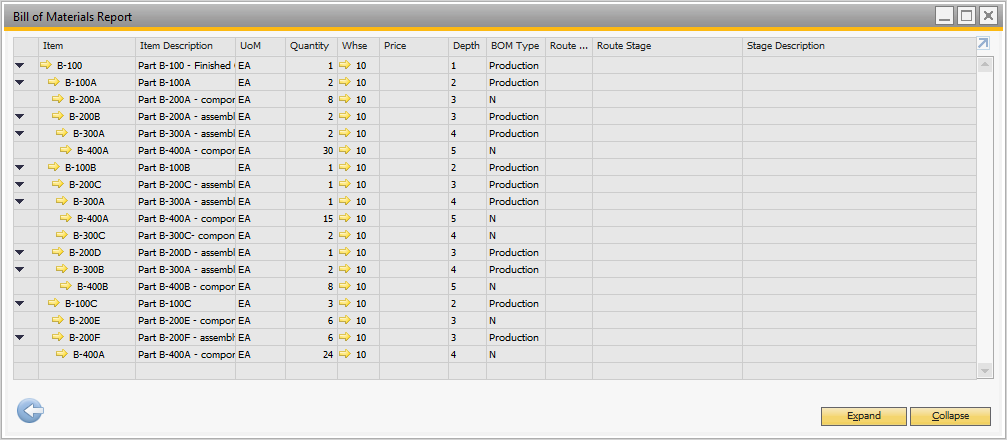
My example is somewhat simplistic. Picture an assembly with 30 or 40 components. Engineering must be careful to properly enter each assembly. They need to know that the BOMs are accurate. One approach would be to ‘Approve’ the BOM on an assembly-by-assembly basis. Our BOM enhancements includes this approval process.

Over time there will be replacement assemblies or components within a BOM . The reason could be better materials, better design, correction of parts design – it does matter - this will occur. Many times the ‘part number’ for the BOM component items doesn’t change – you simply have a newer ‘revision’ of the part. The BOM enhancement process addresses this concern with the inclusion of a revision number for each item. As part of this enhancement, the ‘revision’ number is included in the invoice line item within SAP for each item sold. If a customer calls 2 years later and asks about an item they were sold the customer service person can look up the ‘engineering revision’ for the original part sold.

All of the BOM enhancements provided can be altered by the user – they are available as ‘Boyum’ functions. This means that an indented BOM report or an indented where used report can be modified by the user to meet their needs.

# Indented BOM

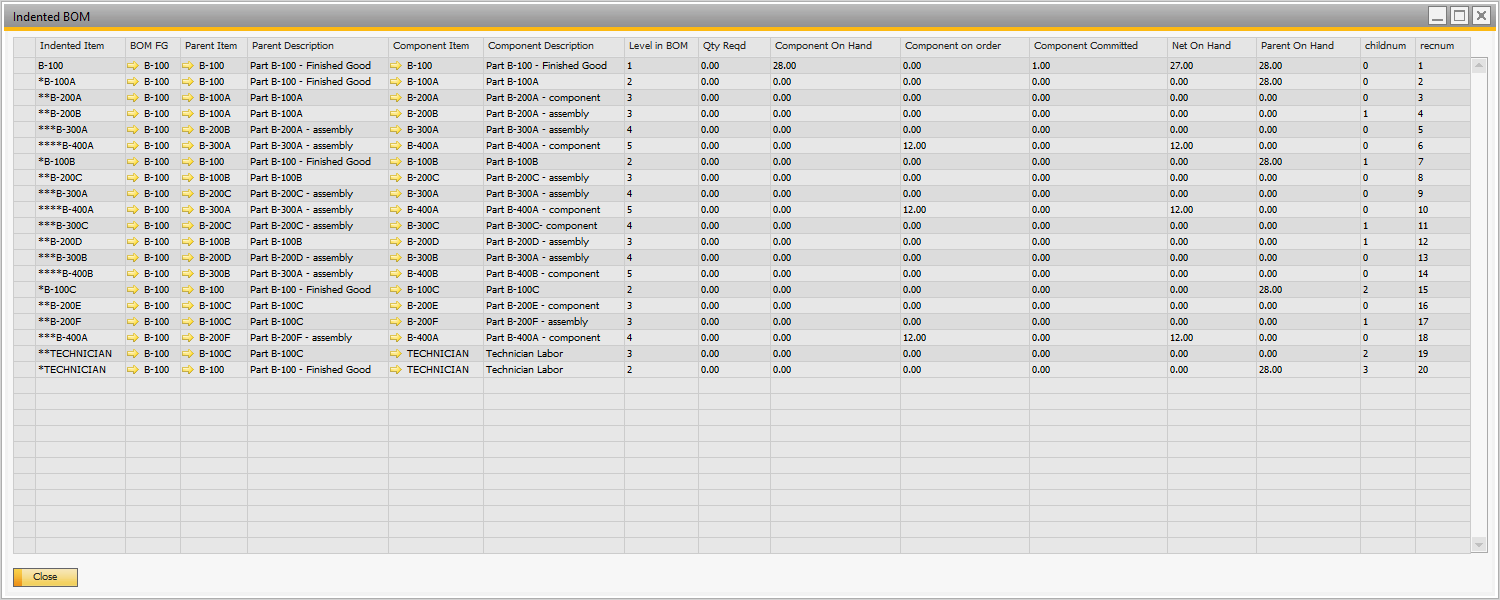
Standard SAP provides an indented BOM as shown in Figure 2.1. The BOM depicted is 4 levels deep and consists of 58 items. Of the 58 items there are 6 subassemblies. The data provided in this report cannot be altered to include additional information (or remove undesired information).



## Figure 2.1 Standard SAP BOM Report

An Indented BOM report is available with Mascidon’s BOM enhancements. It is available from these screens: Item Master; Bill of Materials; Sales Quote; Sales Order; Delivery Order; A/R Invoice; MRP Forecast. Obviously, it is available from anywhere you can drill to the item master screen. Mascidon’s ‘standard ‘ indented BOM report is shown in Figure 2.2.

A similar indented report using Mascidon’s enhancements can be displayed from either the item master screen or the BOM entry screen. This report is shown in Figure 2.2. It has 60 line items – because the 2 resource steps in the BOM are included in the report.



## Figure 2.2 Indented BOM From Mascidon

One of the first things you notice is that Mascidon’s indented BOM includes the ‘Resource’ information. The Boyum Universal Function (SQL) that defines the format of the Indented BOM is shown below (only the display portion of the SQL):

select s.itemdisp as 'Indented Item' , s.fg as 'BOM FG' , s.parent as 'Parent Item' , ii.itemname as 'Parent Description' , s.component as 'Component Item' , case when cc.itemname is null then r.resname else cc.itemname end as 'Component Description' , s.TLevel as 'Level in BOM' , s.qtyreqd as 'Qty Reqd' , isnull(c.onhand,0) as 'Component On Hand' , isnull(c.onorder,0) as 'Component on order' , isnull(c.iscommited,0) as 'Component Committed' , (isnull(c.onhand,0) + isnull(c.onorder,0) - isnull(c.iscommited,0)) as 'Net On Hand' , i.onhand as 'Parent On Hand' , s.childnum , s.recnum from #stack s left join oitw i on i.itemcode = s.parent and i.whscode = @whs left join oitm ii on ii.ItemCode = i.itemcode left join oitw c on c.itemcode = s.component and c.whscode = @whs left join oitm cc on cc.itemcode = c.itemcode left join orsc r on r.rescode = @component order by s.recnum

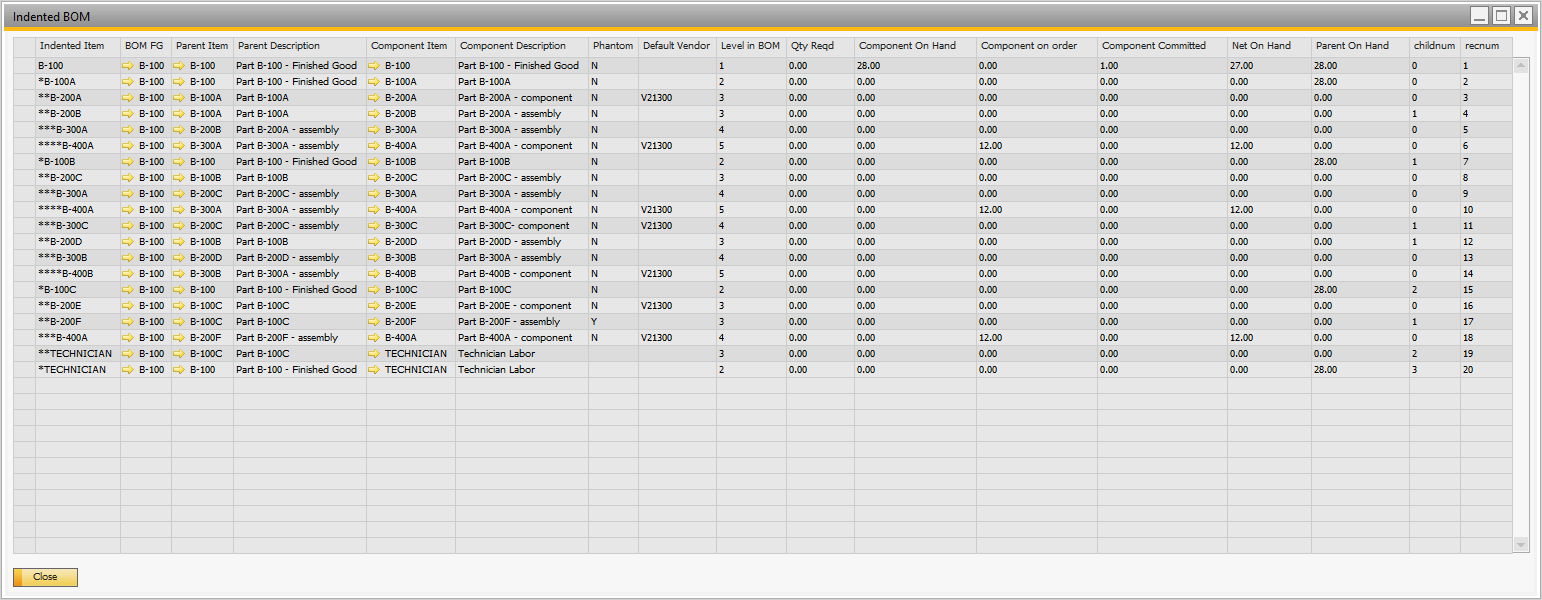
Let’s assume that we should include the inventory flag ‘Phantom’ and the default vendor id for each purchased component. The Universal Function BOM-02 (Indented BOM) would be modified as follows:

select s.itemdisp as 'Indented Item' , s.fg as 'BOM FG' , s.parent as 'Parent Item' , ii.itemname as 'Parent Description' , s.component as 'Component Item' , case when cc.itemname is null then r.resname else cc.itemname end as 'Component Description' **, c.phantom as ‘Phantom’**

**, c.cardcode as ‘Default Vendor’**

, s.TLevel as 'Level in BOM' , s.qtyreqd as 'Qty Reqd' , isnull(c.onhand,0) as 'Component On Hand' , isnull(c.onorder,0) as 'Component on order' , isnull(c.iscommited,0) as 'Component Committed' , (isnull(c.onhand,0) + isnull(c.onorder,0) - isnull(c.iscommited,0)) as 'Net On Hand' , i.onhand as 'Parent On Hand' , s.childnum , s.recnum from #stack s left join oitw i on i.itemcode = s.parent and i.whscode = @whs left join oitm ii on ii.ItemCode = i.itemcode left join oitw c on c.itemcode = s.component and c.whscode = @whs left join oitm cc on cc.itemcode = c.itemcode left join orsc r on r.rescode = @component order by s.recnum

Making this small change to the BOM-02 Universal Function SQL code changes the report to the report shown in Figure 2.3. The items with a default vendor are easily defined. In addition, you can see that item B-200F is a phantom part. Note: you could add a gold arrow to the ‘Default Vendor’ column by changing the BOM-02 Universal Function as shown in Figure 2.4



## Figure 2.3 Modified Indented BOM Report

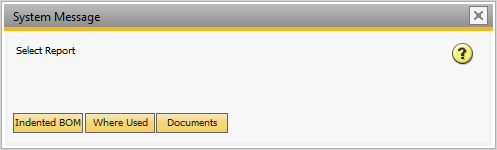




## Figure 2.4 Adding Gold Arrow to UF BOM-02

The changes to the code are simple for anyone familiar with SQL and SAP. That’s not everyone. But you can also contract with Mascidon to modify the SQL reports. All of the reports in this module can be similarly modified.

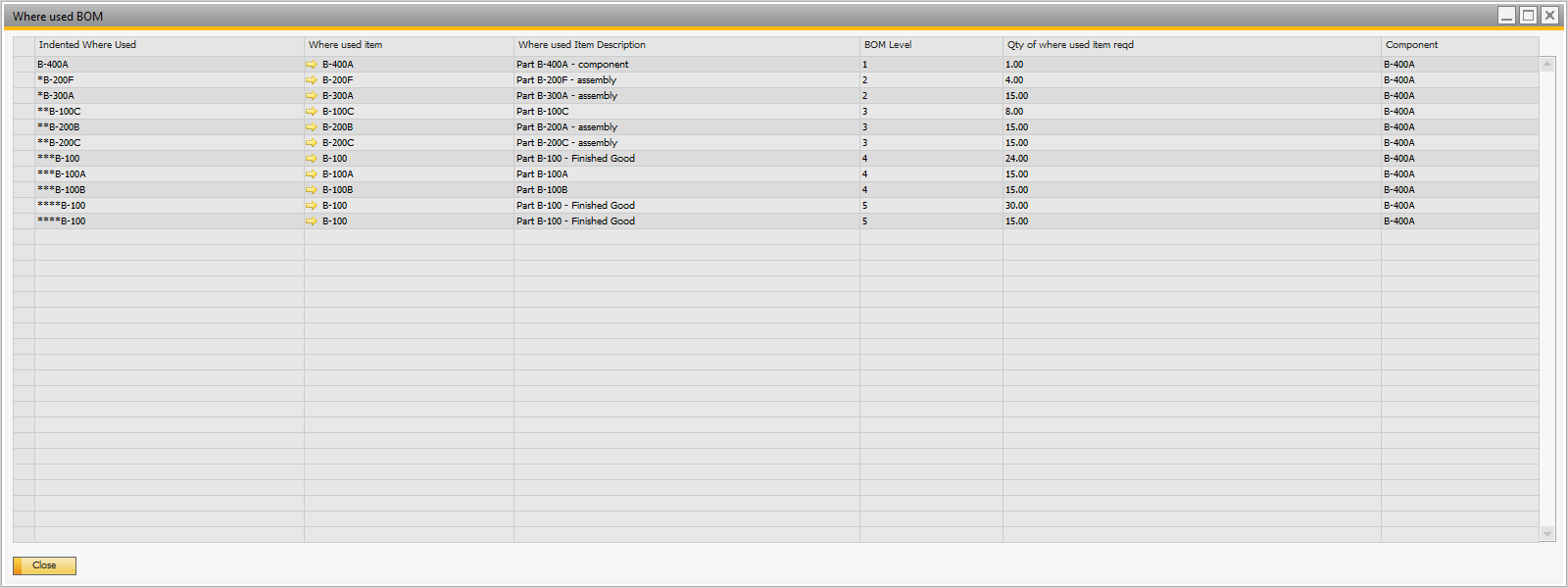
The Indented BOM can be accessed from Item Master, Bill of Materials, Sales Quote, Sales Order, Delivery Order, A/R Invoice, and MRP Forecast by clicking on the button on the bottom of the form ‘BOM Reports’. When accessed from all but the Item Master screen, the Indented BOM report is simply displayed. When accessed from the Item Master screen, the popup shown in Figure 2.5 is presented and the user selects ‘Indented BOM’.



## Figure 2.5 Popup Report Question

# Where Used Report

An indented BOM shows the finished product from the top level to the lowest level. The where used indented reports show the indent from the lowest level in the BOM to the finished goods level. This report is available in 2 formats. The first is a simple ‘where used’ report – an indented BOM from the low level part displayed on the item master screen upwards to associated finished goods. This report is shown in Figure 2.6. One of the main uses of this report is to track down potential production problems. For instance, a customer may send back a product because it has a defect. Your engineering team analyzes the defective part and determines that one of the components has a defect. Then the engineering team displays the where used report. This gives production the information on all higher level assemblies that may have used this defective part.

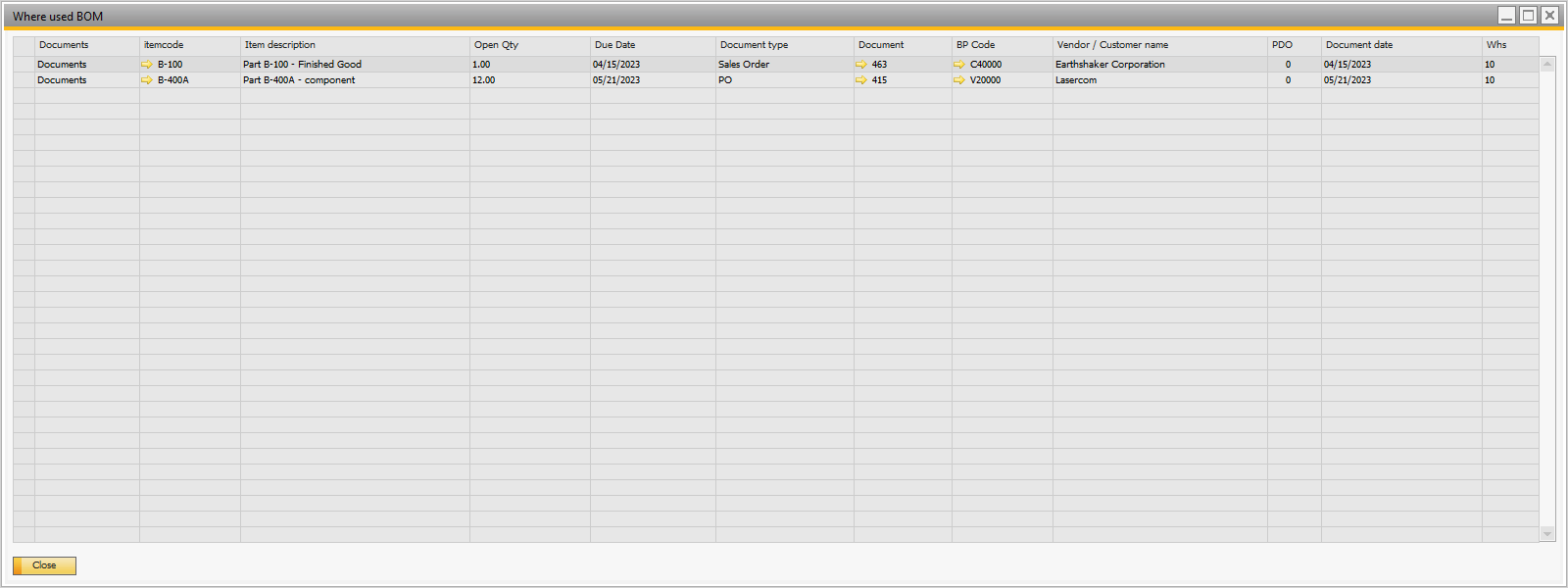


## Figure 2.6 Where Used Indented Report

The second Where-used report is the ‘Where Used Documents’ report. This report uses the data from the Where-Used indented report to show all of the documents related to the parts. This is an important report if you have found a problem with a raw material, component or sub-assembly and now want to determine all manufacturing and finished goods that need to be reviewed. Alternatively, if you have decided to change out a part in the BOM you need to know all of the in process parts that may be affected. There are several related documents that are reported as part of the where used documents report. These are:

* Purchasing Orders
* Sales Orders
* Production Orders

This report is shown in Figure 2.7. The report shows open documents associated with the component part B-400A. This includes a PO for the part itself and a sales order for the finished good using this component.



## Figure 2.7 Where Used Documents Report

The where used reports can be accessed from the item master screen, purchase order screen and the PO goods received screen.

## BOM Construction

For new product design, how does the engineering staff develop the SAP BOM? Let’s assume they have been doing the engineering design using a CAD product and an engineering documentation tool. They have the design of the product completed. At this point they need to determine the ‘parts’ and BOM connections that need to be defined in SAP.

There are decisions to be made. For instance, if the engineering design calls for parts to be stamped in a progressive die, do you really want the ‘intermediate’ parts defined within SAP? i.e. you feed raw materials / components into a progressive die and an intermediate part is manufactured and then immediately used to produce a part that is further processed in the next hit of the progressive die. Defining the intermediate part as a phantom part in SAP is one solution to match the SAP BOM closely to the engineering BOM. However, SAP will never have on hand of this intermediate part and if ever the progressive die process is interrupted, this intermediate part is likely thrown away. So this intermediate part is really not required in SAP – but could be included. Other than that sort of decision, the engineering BOM is used to create the SAP BOM.

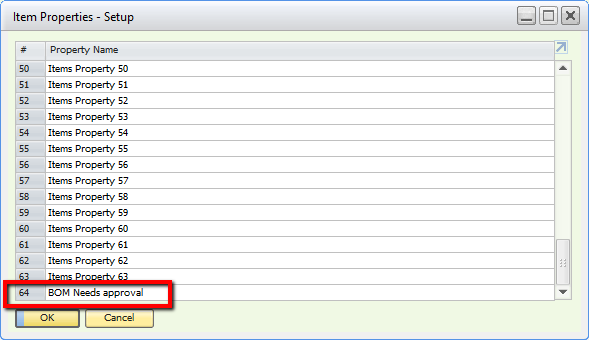
Another decision to be made is – how much use of ‘phantom’ parts should be included? Aside from the example of the progressive die ‘parts’, production reporting is not required for every part. For instance, if manufacturing processes from raw material to finished goods level occurs in a couple of hours – why gather and report production at low levels of assembly? It takes time and effort to collect production counts – and time and effort to create production orders. Decisions on a flat BOM, BOMs that include many phantom parts, and inclusion of resource costs in the BOM need to be made by the engineers, the production managers, and the accounting personnel.

After the decisions on how complex / complete the BOM should be have been made, entering the BOM within SAP can be started from either the top most level of the BOM or from the bottom of the BOM upwards. SAP’s BOMs are defined either way. Let’s assume we have a finished product that has several major sub-assemblies. My suggestion is to have the BOM enhancement from Mascidon show the approval status of the finished product and each of the major sub-assemblies.

Before the BOM can be constructed each of the ‘items’ required for this finished good (as obtained from the engineering BOM data) needs to be entered into the SAP item master. Continuing our example, when the finished goods part and the major sub-assembly parts are created these parts need to be flagged as ‘BOM Approval in-progress’. When each sub-assembly has been entered into SAP and has been reviewed, this flag is turned off. The key is that the approval flag information must be visible to the engineers.

## Implementation of BOM Approval

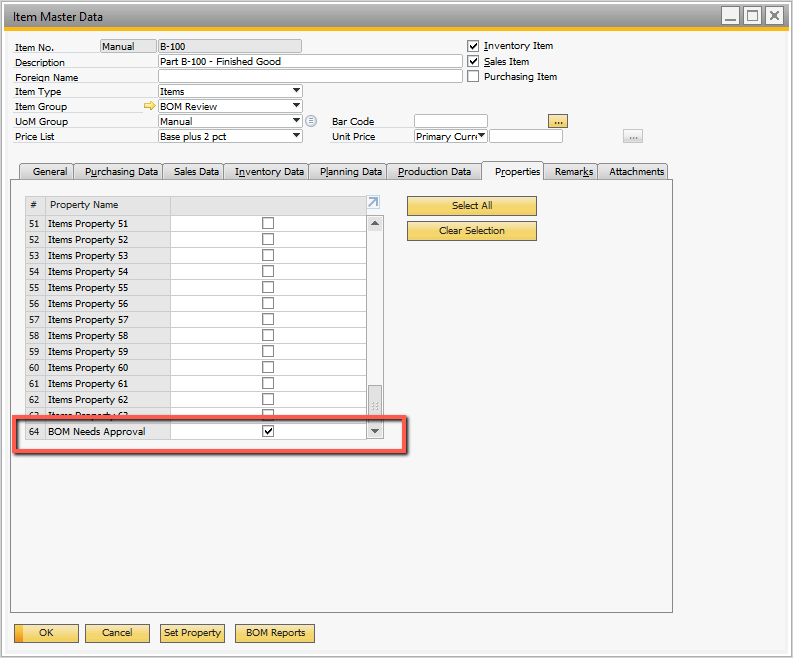
In order to facilitate the BOM approval status a flag needs to be set for the item that is the parent. I have arbitrarily used item property 64 for this purpose. This could be changed if you are already using this particular item property. The first step is to name the item property appropriately. Figure 2.8 shows the Item Property name assignment. To access this screen: Administration 🡪 Setup 🡪 Inventory 🡪 Item Properties. Then navigate to property 64 and change the wording as shown.

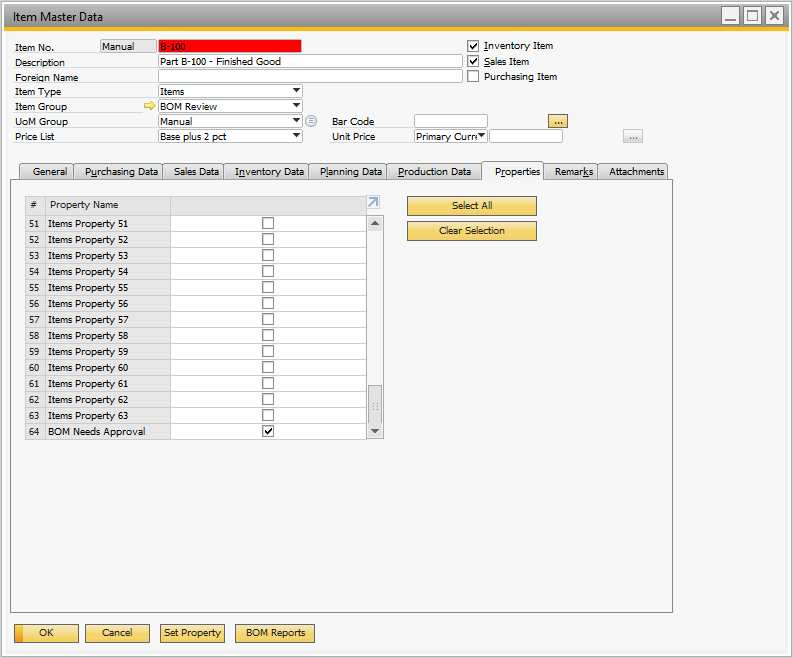


## Figure 2.8 Item Property Name Assignment

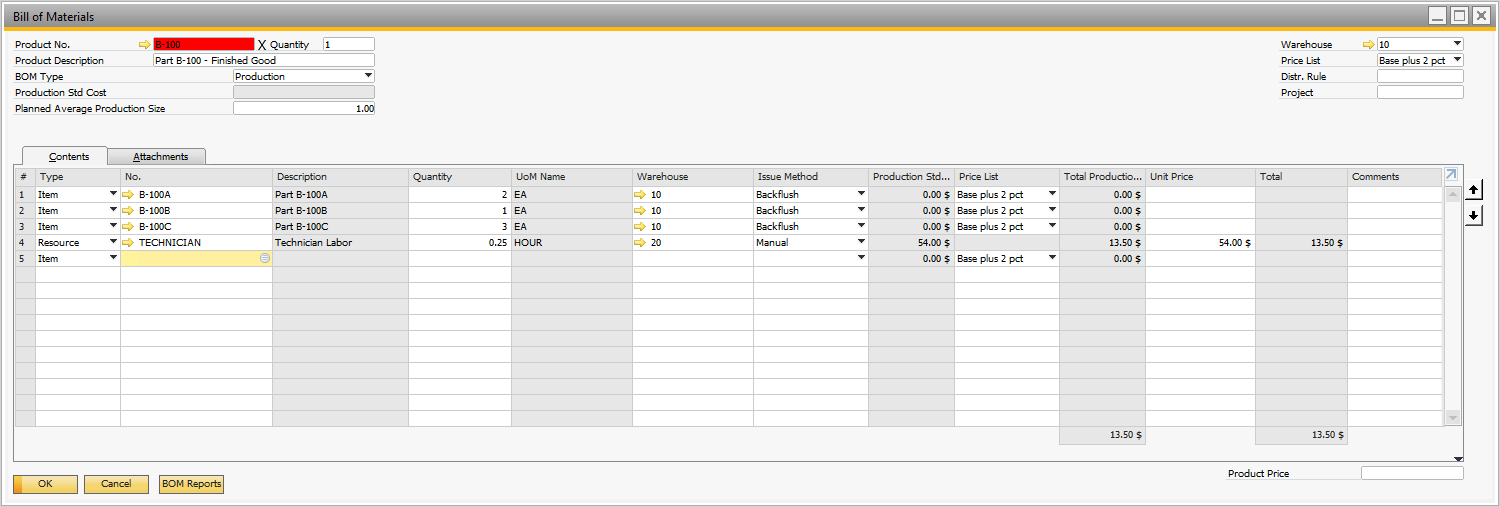
From the item master I have set the property 64 ‘BOM Needs Approval’ for the part B-100 – as shown in Figure 2.9. Then I refresh the screen for this part and the approval requirement is readily apparent because of the background coloring of the item. Refer to Figure 2.10.

In the same manner, the bill of materials screen has this same visibility. Refer to Figure 2.11.

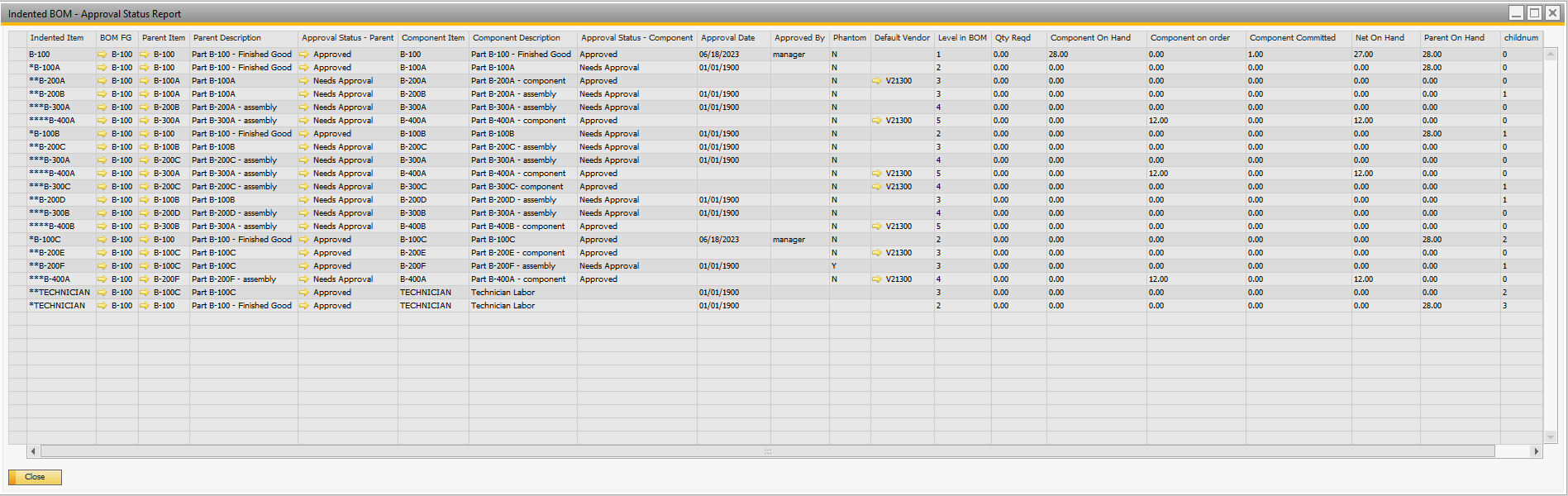
  
Figure 2.9 Setting Property 64



## Figure 2.10 Visibility of Property 64 Setting



## Figure 2.11 BOM Visibility of Property 64 Setting



## Figure 2.12 Indented BOM – Approval Status Report

# Appendix A – Technical

This add-on is implemented using Boyum.

## SAP File Changes

Use standard SAP functionality to add the following fields to the database.

Item Master

* EngrRevision – Engr Revision, Alpha 30 characters

Marketing Documents – Rows

* EngrRevision – Engr Revision, Alpha 30 characters

BOM Title

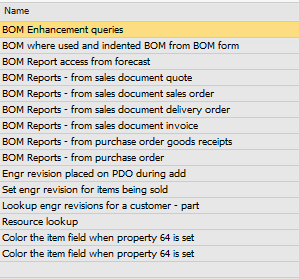
* DateApproved – Date Approved, Date Field
* ApprovedBy – Approved By, Alpha 100 characters

Production Order Title

* EngrRevision – Engr Revision, Alpha 30 characters

# Boyum – B1 Validations

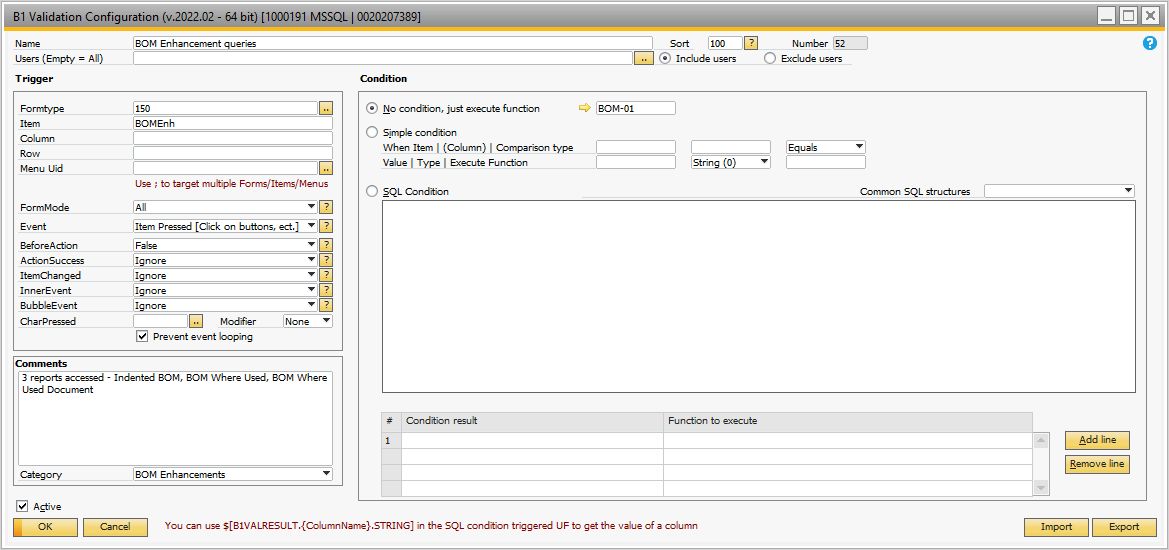
There are 15 B1 Validations associated with this enhancement. Figure A-1 lists these. Each of these is shown in this Appendix along with the associated Universal Functions.



## Figure A-1 List of B1 Validations

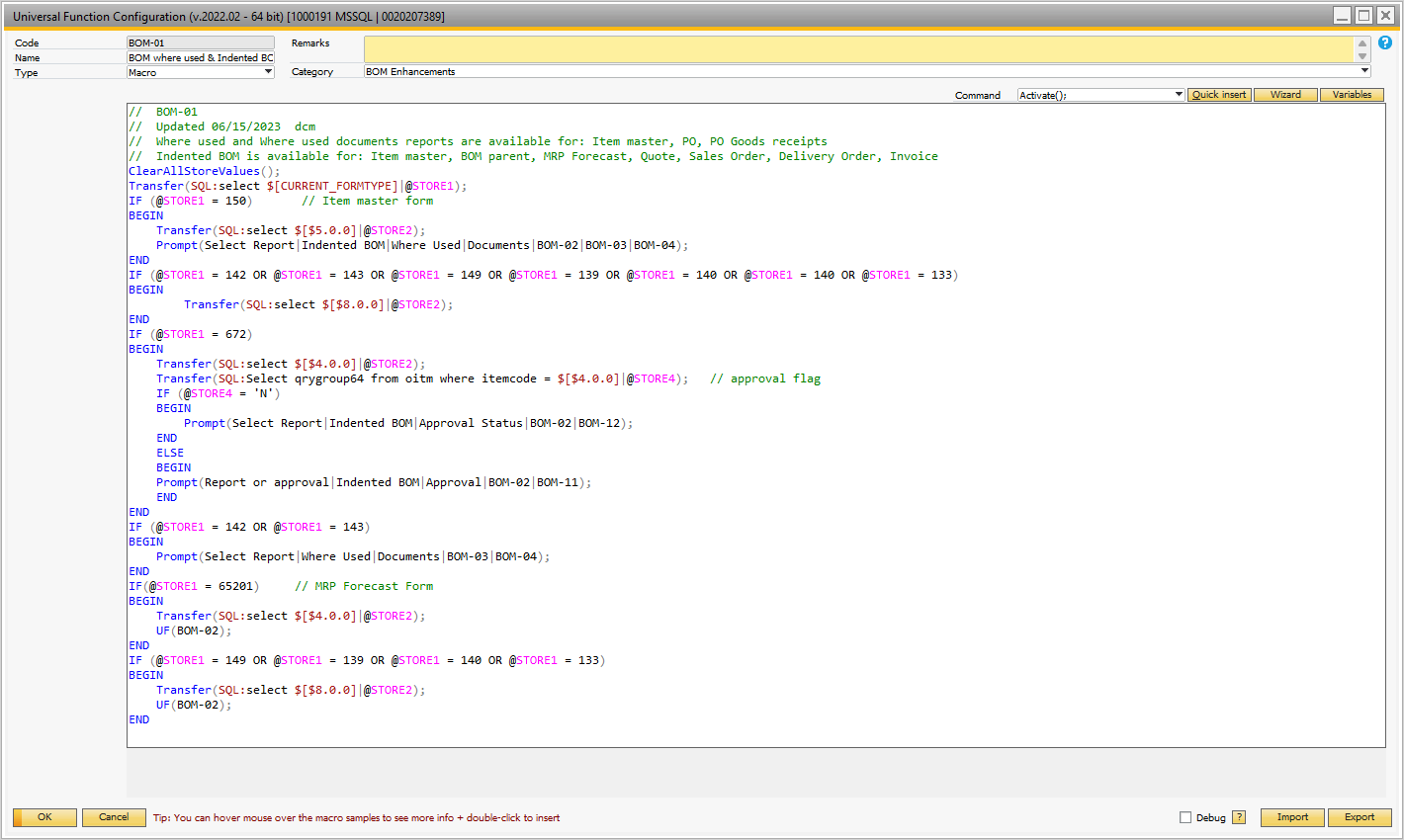
## BOM Enhancement Queries (Validation)

The validation shown in Figure A-2 is called from the item master screen when the button ‘BOM Reports’ is pressed. The Universal Function BOM-01 shown in Figure A-3 is called.

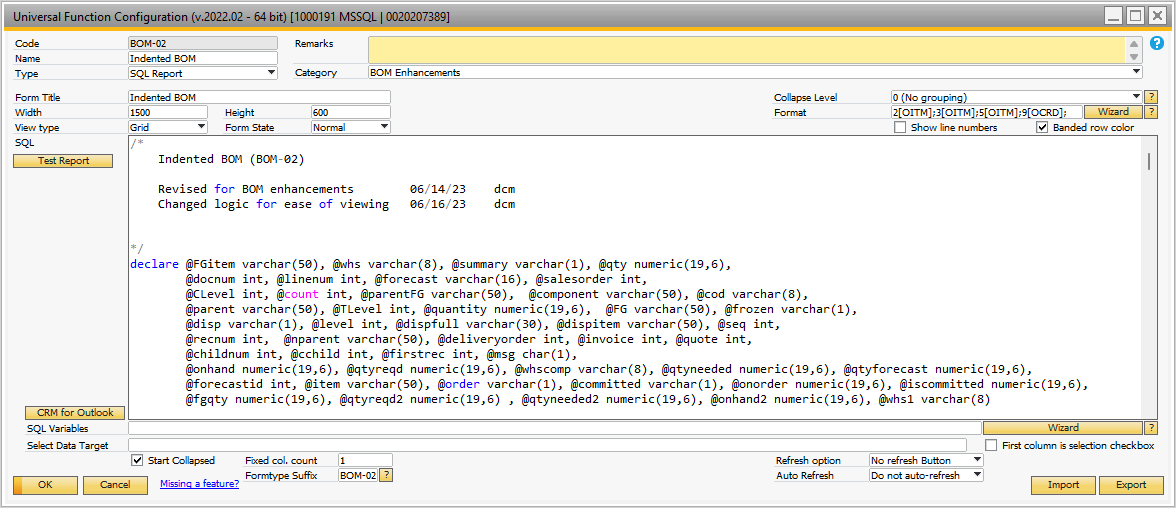


## Figure A-2 BOM Enhancement Accessed From Item Master

When the UF BOM-01 is called from the item master, it prompts the user to decide which of 3 reports to run: Indented BOM; Where Used; or Documents. Depending on the response to the prompt BOM-02, BOM-03 or BOM-04 UF is called. These are shown in Figures A-4 to A-8.



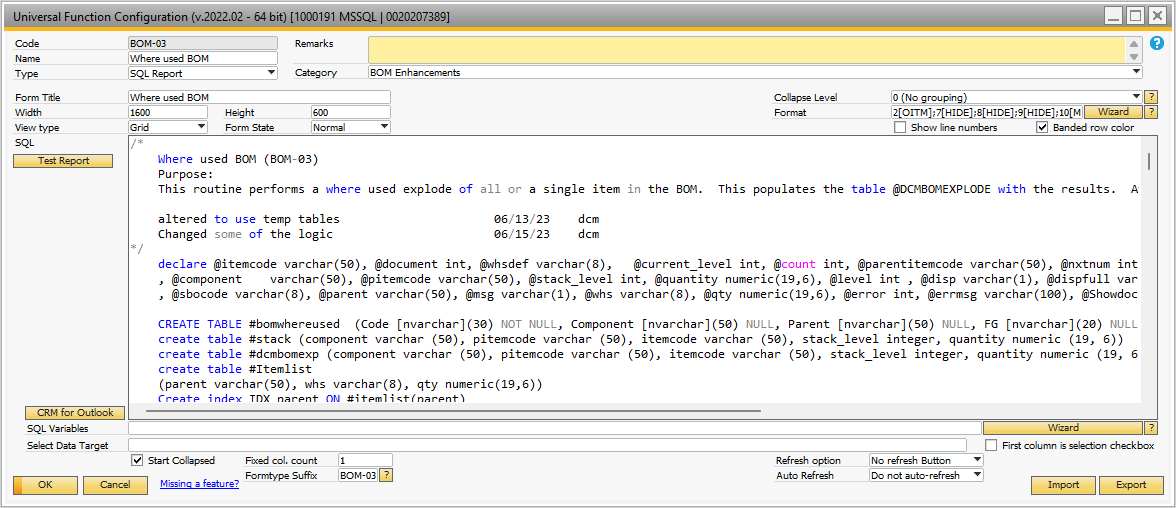
## Figure A-3 BOM-01 UF Decides Which Reports to Run



## Figure A-4 BOM-02 UF Showing Indented BOM

The SQL associated with BOM-02 is shown below:

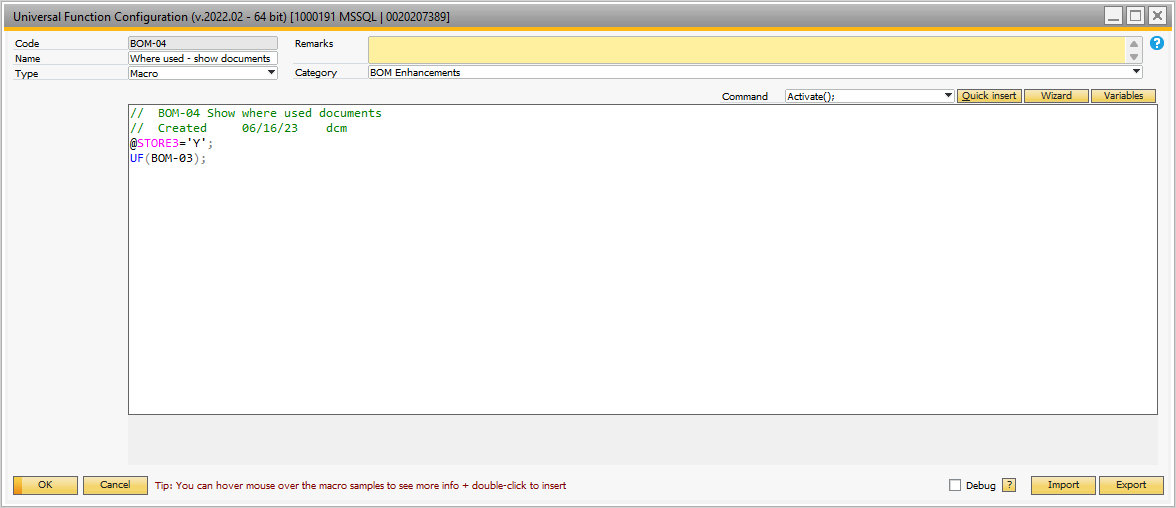
/\* Indented BOM (BOM-02) Revised for BOM enhancements 06/14/23 dcm Changed logic for ease of viewing 06/16/23 dcm\*/declare @FGitem varchar(50), @whs varchar(8), @summary varchar(1), @qty numeric(19,6), @docnum int, @linenum int, @forecast varchar(16), @salesorder int, @CLevel int, @count int, @parentFG varchar(50), @component varchar(50), @cod varchar(8), @parent varchar(50), @TLevel int, @quantity numeric(19,6), @FG varchar(50), @frozen varchar(1), @disp varchar(1), @level int, @dispfull varchar(30), @dispitem varchar(50), @seq int, @recnum int, @nparent varchar(50), @deliveryorder int, @invoice int, @quote int, @childnum int, @cchild int, @firstrec int, @msg char(1), @onhand numeric(19,6), @qtyreqd numeric(19,6), @whscomp varchar(8), @qtyneeded numeric(19,6), @qtyforecast numeric(19,6), @forecastid int, @item varchar(50), @order varchar(1), @committed varchar(1), @onorder numeric(19,6), @iscommitted numeric(19,6), @fgqty numeric(19,6), @qtyreqd2 numeric(19,6) , @qtyneeded2 numeric(19,6), @onhand2 numeric(19,6), @whs1 varchar(8) select @msg = 'N' select @disp = '\*', @level = 0, @seq = 0, @qty = 1 select @order = 'Y', @committed = 'Y' create table #stack ( recnum int, component varchar (50) COLLATE SQL\_Latin1\_General\_CP850\_CI\_AS NOT NULL, parent varchar (50) COLLATE SQL\_Latin1\_General\_CP850\_CI\_AS NOT NULL, FG varchar (50) COLLATE SQL\_Latin1\_General\_CP850\_CI\_AS NOT NULL, TLevel integer, quantity numeric (19, 6), itemdisp varchar(50), sequence int, childnum int, qtyreqd numeric(19,6), qtyreqd2 numeric(19,6), fgqty numeric(19,6) , onhand numeric(19,6), onhand2 numeric(19,6) ) Create index IDX\_Component ON #stack(Component) Create index IDX\_Level\_Component ON #stack(tLevel,Component) delete #stack create table #Itemlist (parent varchar(50), whs varchar(8), qty numeric(19,6)) Create index IDX\_parent ON #itemlist(parent) create table #Onhand (item varchar(50), whs varchar(8), onhand numeric(19,6), onhand2 numeric(19,6)) Create index IDX\_item ON #onhand(item) create table #peggedusage (mfgitem varchar(50), reqd numeric(19,6)) Create index IDX\_mfgitem ON #peggedusage(mfgitem) if '@STORE1' ='150' -- item master begin select @fgitem = '@STORE2' select @whs = DfltWH from oitm where itemcode = @fgitem if isnull(@whs,'') = '' select @whs = dfltwhs from oadm insert into #itemlist values(@fgitem, @whs, 1) end if '@STORE1' ='672' -- BOM begin select @fgitem = '@STORE2' select @whs = DfltWH from oitm where itemcode = @fgitem if isnull(@whs,'') = '' select @whs = dfltwhs from oadm insert into #itemlist values(@fgitem, @whs, 1) end if '@STORE1' ='149' -- Quote begin select @quote ='@STORE2' insert into #itemlist select r.itemcode, r.whscode, r.openqty from qut1 r inner join oitm i on i.itemcode = r.itemcode inner join oqut o on o.docentry = r.docentry where o.docnum = @quote and i.invntitem = 'Y' end if '@STORE1' ='139' -- Sales Order begin select @salesorder = '@STORE2' insert into #itemlist select r.itemcode, r.whscode, r.openqty from rdr1 r inner join oitm i on i.itemcode = r.itemcode inner join ordr o on o.docentry = r.docentry where o.docnum = @salesorder and r.openqty > 0 end if '@STORE1' ='140' -- Delivery Order begin select @deliveryorder = '@STORE2' insert into #itemlist select r.itemcode, r.whscode, r.openqty from dln1 r inner join oitm i on i.itemcode = r.itemcode inner join odln o on o.docentry = r.docentry where o.docnum = @deliveryorder end if '@STORE1' ='133' -- Invoice begin select @invoice = '@STORE2' insert into #itemlist select r.itemcode, r.whscode, r.openqty from inv1 r inner join oitm i on i.itemcode = r.itemcode inner join oinv o on o.docentry = r.docentry where o.docnum = @invoice end if '@STORE1' ='65201' -- Forecast begin select @forecast = '@STORE2' select @forecastid = absid from ofct where code = @forecast declare subloop cursor for select f.itemcode, sum(f.quantity), isnull(i.dfltwh,@whs) from fct1 f inner join oitm i on i.itemcode = f.itemcode where f.absid = @forecastid group by f.itemcode, i.dfltwh open subloop fetch subloop into @item, @qtyforecast, @whs1 while @@fetch\_status = 0 begin insert into #itemlist values (@item, @whs1, @qtyforecast) fetch subloop into @item, @qtyforecast, @whs1 end close subloop deallocate subloop end select @msg = 'N' select @disp = '\*', @level = 0, @seq = 0, @order = 'Y', @committed = 'Y' declare mainloop cursor for select parent, whs, isnull(qty,1) from #Itemlist order by parent desc open mainloop fetch mainloop into @NParent, @whs, @qty while @@fetch\_status = 0 begin -- 4 -- now let's explode the components of the item being sold - @item and then apply the logic to the colors select @FG = @NParent, @fgqty = @qty --delete #stack select @CLevel = 1, @count = 1 select @seq = @seq + 1 -- create a record in the stack for the finished good item -- first step is to store the on hand if (select count(\*) from #onhand where item = @fg and whs = @whs) = 0 begin select @onhand = isnull(onhand,0), @onorder = isnull(onorder,0), @iscommitted = isnull(iscommited,0) from oitw where whscode = @whs and itemcode = @fg select @onhand2 = @onhand if @order = 'Y' select @onhand2 = @onhand + @onorder if @committed = 'Y' and @order = 'Y' select @onhand2 = @onhand - @iscommitted + @onorder if @committed = 'Y' and @order <> 'Y' select @onhand2 = @onhand - @iscommitted insert into #onhand values(@fg, @whs, @onhand, @onhand2) end else select @onhand = onhand, @onhand2 = onhand2 from #onhand where item = @fg and whs = @whs if @onhand >= @qty select @qtyreqd = 0 else select @qtyreqd = @qty - @onhand if @onhand2 >= @qty select @qtyreqd2 = 0 else select @qtyreqd2 = @qty - @onhand2 update #onhand set onhand = onhand - @qty, onhand2 = onhand2 - @qty where item = @fg and whs = @whs update #onhand set onhand = 0 where onhand < 0 update #onhand set onhand2 = 0 where onhand2 < 0 insert into #stack (recnum, component, parent, FG, TLevel, quantity, itemdisp, sequence, childnum, qtyreqd, qtyreqd2, fgqty, onhand, onhand2) values (@seq, @FG, @FG, @FG, @CLevel, 1, @FG, 1, 0, @qtyreqd, @qtyreqd2, @fgqty, @onhand, @onhand2) select @firstrec = @seq if (select count(\*) from itt1 where father = @NParent) = 0 -- no items to explode goto Exit\_Loop select @cchild = 0Get\_Next\_BOM: if @msg = 'Y' begin select @nparent as 'Parent', @cchild as 'Childnum', @clevel as 'Clevel','Get Next BOM' end if (select count(\*) from [DBO].itt1 as bom where bom.father = @NParent and bom.VisOrder = @cchild) = 0 begin goto Step\_Up\_a\_level if @msg = 'Y' begin select 'Not finding the BOM record', @nparent as 'Parent', @cchild as 'Childnum' end goto exit\_loop end select @component = Bom.code, -- child = component @quantity = bom.quantity \* (select top 1 ( isnull(#stack.quantity,0) ) from #stack where #stack.component = @Nparent and #stack.TLevel = @CLevel order by recnum desc), @childnum = bom.VisOrder, @whscomp = bom.warehouse, @qtyneeded = bom.quantity \* (select top 1 ( isnull(#stack.qtyreqd,0) ) from #stack where #stack.component = @Nparent and #stack.TLevel = @CLevel order by recnum desc), @qtyneeded2 = bom.quantity \* (select top 1 ( isnull(#stack.qtyreqd2,0) ) from #stack where #stack.component = @Nparent and #stack.TLevel = @CLevel order by recnum desc) from [DBO].itt1 as bom where bom.father = @NParent and bom.VisOrder = @cchild select @level = @CLevel, @dispfull = '' while @level > 0 begin select @dispfull = @disp + @dispfull select @level = @level - 1 end if @msg = 'Y' begin select 'Get next bom. Level = ', @Clevel as 'CLevel', ' Component: ', @component as Component, ' Qty: ', @quantity as 'Qty', ' Child Seq: ', @childnum as 'Childnum', @cchild as 'CChild' end -- first step is to store the on hand if (select count(\*) from oitm where itemcode = @component) > 0 -- it is an item begin if (select count(\*) from #onhand where item = @component and whs = @whscomp) = 0 begin select @onhand = isnull(onhand,0), @onorder = isnull(onorder,0), @iscommitted = isnull(iscommited,0) from oitw where whscode = @whscomp and itemcode = @component select @onhand2 = @onhand if @order = 'Y' select @onhand2 = @onhand + @onorder if @committed = 'Y' and @order = 'Y' select @onhand2 = @onhand - @iscommitted + @onorder if @committed = 'Y' and @order <> 'Y' select @onhand2 = @onhand - @iscommitted insert into #onhand values(@component, @whscomp, @onhand, @onhand2) end else begin select @onhand = onhand, @onhand2 = onhand2 from #onhand where item = @component and whs = @whscomp end if @onhand >= @qtyneeded select @qtyreqd = 0 else select @qtyreqd = @qtyneeded - @onhand if @onhand2 >= @qtyneeded2 select @qtyreqd2 = 0 else select @qtyreqd2 = @qtyneeded2 - @onhand2 update #onhand set onhand = onhand - @qtyneeded, onhand2 = onhand2 - @qtyneeded2 where item = @component and whs = @whscomp update #onhand set onhand = 0 where onhand < 0 and item = @component update #onhand set onhand2 = 0 where onhand2 < 0 and item = @component end else -- means it is a resource begin select @onhand = 0, @onhand2 = 0, @order = 0, @committed = 0, @iscommitted = 0, @qtyreqd = @qtyneeded end select @seq = @seq + 1 insert into #stack (recnum, component, parent, FG, TLevel, quantity, itemdisp, sequence, childnum, qtyreqd, qtyreqd2, fgqty, onhand, onhand2) values(@seq, @component, @Nparent, @FG, @CLevel + 1, @quantity, @dispfull+@component, 0, @childnum, @qtyreqd, @qtyreqd2, @fgqty, @onhand, @onhand2) -- does the component have children? if (select count(\*) from itt1 where father = @component) > 0 begin select @clevel = @clevel + 1, @cchild = 0, @Nparent = @component goto Get\_Next\_BOM -- yes the component has children end -- no - does the parent have more children? if (select count(\*) from itt1 where father = @nparent and VisOrder > @cchild) > 0 begin -- select @cchild = @cchild + 1 goto Get\_Next\_BOM end -- no the parent does not have any more childrenStep\_up\_a\_level: select top 1 @nparent = parent, @cchild = childnum + 1, @recnum = recnum from #stack where component = @nparent and TLevel = @Clevel order by recnum desc if @msg = 'Y' begin select 'Are we at end? = ', @nparent as 'NParent',@cchild as 'CChild', @recnum as 'Recnum', @firstrec as 'Firstrec', @Clevel as 'CLevel', @component as Component, @quantity as 'Qty', @childnum as 'Childnum' end if @recnum = @firstrec -- back to the beginning begin goto Exit\_Loop end select @Clevel = @clevel - 1 goto Get\_Next\_BOM Exit\_Loop: fetch mainloop into @NParent, @whs, @qty end close mainloop deallocate mainloop select s.itemdisp as 'Indented Item' , s.fg as 'BOM FG' , s.parent as 'Parent Item' , ii.itemname as 'Parent Description' , s.component as 'Component Item' , case when cc.itemname is null then r.resname else cc.itemname end as 'Component Description' , cc.phantom as 'Phantom' , cc.cardcode as 'Default Vendor' , s.TLevel as 'Level in BOM' , s.qtyreqd as 'Qty Reqd' , isnull(c.onhand,0) as 'Component On Hand' , isnull(c.onorder,0) as 'Component on order' , isnull(c.iscommited,0) as 'Component Committed' , (isnull(c.onhand,0) + isnull(c.onorder,0) - isnull(c.iscommited,0)) as 'Net On Hand' , i.onhand as 'Parent On Hand' , s.childnum , s.recnum from #stack s left join oitw i on i.itemcode = s.parent and i.whscode = @whs left join oitm ii on ii.ItemCode = i.itemcode left join oitw c on c.itemcode = s.component and c.whscode = @whs left join oitm cc on cc.itemcode = c.itemcode left join orsc r on r.rescode = @component order by s.recnumdrop table #stackdrop table #itemlistdrop table #onhanddrop table #peggedusage



## Figure A-5 BOM-02 Where Used Report

Note: There are several hidden fields for this report. There are actually 2 possible reports from this UF depending on how it is called. It shows either the ‘Where Used’ report or the ‘Where Used Documents’ report. The SQL associated with the Where Used report is shown below.

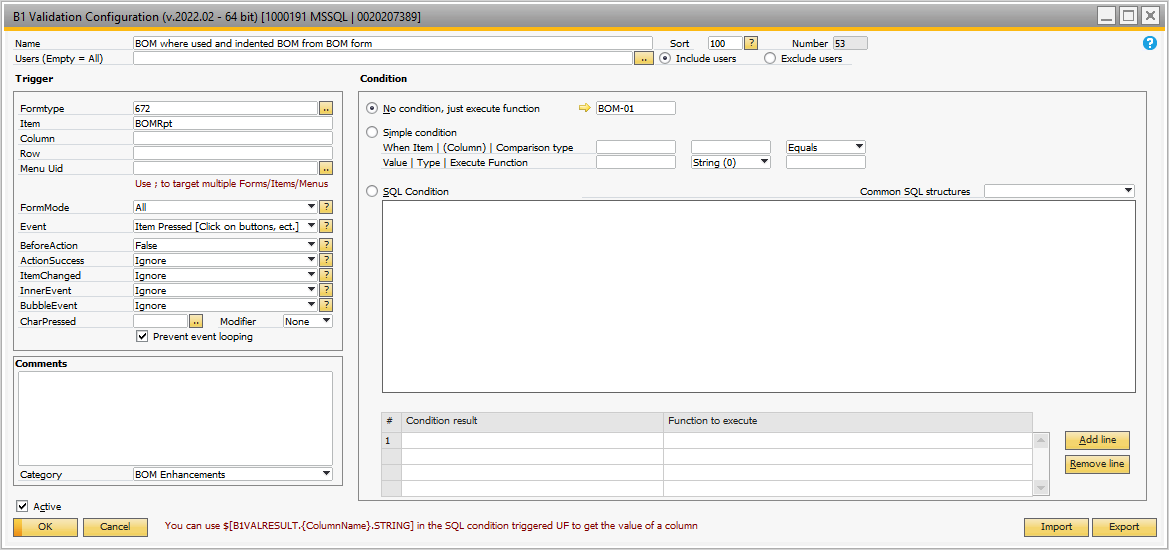
/\* Where used BOM (BOM-03) Purpose: This routine performs a where used explode of all or a single item in the BOM. This populates the table @DCMBOMEXPLODE with the results. After this routine has been run, the BOM Explode queries can be run. altered to use temp tables 06/13/23 dcm Changed some of the logic 06/15/23 dcm\*/ declare @itemcode varchar(50), @document int, @whsdef varchar(8), @current\_level int, @count int, @parentitemcode varchar(50), @nxtnum int , @component varchar(50), @pitemcode varchar(50), @stack\_level int, @quantity numeric(19,6), @level int , @disp varchar(1), @dispfull varchar(70) , @sbocode varchar(8), @parent varchar(50), @msg varchar(1), @whs varchar(8), @qty numeric(19,6), @error int, @errmsg varchar(100), @Showdocuments varchar(1) CREATE TABLE #bomwhereused (Code [nvarchar](30) NOT NULL, Component [nvarchar](50) NULL, Parent [nvarchar](50) NULL, FG [nvarchar](20) NULL, LevelNum [int] NULL, BOMQty [numeric](19, 6) NULL, DisplayItem [nvarchar](70) NULL) create table #stack (component varchar (50), pitemcode varchar (50), itemcode varchar (50), stack\_level integer, quantity numeric (19, 6)) create table #dcmbomexp (component varchar (50), pitemcode varchar (50), itemcode varchar (50), stack\_level integer, quantity numeric (19, 6)) create table #Itemlist (parent varchar(50), whs varchar(8), qty numeric(19,6)) Create index IDX\_parent ON #itemlist(parent) select @whsdef = dfltwhs from oadm select @msg = 'N' select @disp = '\*' -- this is the indent character , @Showdocuments = 'N' IF '@STORE3' = 'Y' begin select @Showdocuments = 'Y' -- show documents end if '@STORE1' ='150' -- item master begin select @itemcode = '@STORE2' select @whs = DfltWH from oitm where itemcode = @itemcode if isnull(@whs,'') = '' select @whs = @whsdef insert into #itemlist select @itemcode, @whs, 1 end if '@STORE1' ='142' -- Purchase Order begin select @document = '@STORE2' insert into #itemlist select d.itemcode, d.whscode, d.quantity from opor h inner join por1 d on d.docentry = h.docentry where h.docnum = @document end if '@STORE1' ='143' -- PO Goods receipt begin select @document = '@STORE2' insert into #itemlist select d.itemcode, d.whscode, d.quantity from opdn h inner join pdn1 d on d.docentry = h.docentry where h.docnum = @document end -- Initialize next num to 0 select @nxtnum = 0 declare mainloop cursor for select parent, whs, isnull(qty,1) from #Itemlist order by parent desc open mainloop fetch mainloop into @itemcode, @whs, @qty while @@fetch\_status = 0 begin delete #stack select @current\_level = 1 insert into #stack values (@itemcode, @itemcode, @itemcode, @current\_level, 1) select @count = isnull(count(1),0) from #stack where stack\_level = @current\_level while @count > 0 begin if @msg = 'Y' select @count as 'Count' declare parentitemcodes cursor for select itemcode from #stack where stack\_level = @current\_level open parentitemcodes fetch parentitemcodes into @parentitemcode while @@fetch\_status = 0 begin insert into #stack select @itemcode, @parentitemcode, bom.father, @current\_level + 1, bom.quantity \* (select sum ( isnull(#stack.quantity,0) ) from #stack where #stack.itemcode = @parentitemcode and #stack.stack\_level = @current\_level) from [DBO].itt1 as bom where bom.code = @parentitemcode fetch parentitemcodes into @parentitemcode end close parentitemcodes deallocate parentitemcodes select @current\_level = @current\_level + 1 select @count = isnull(count(1),0) from #stack where stack\_level = @current\_level end insert into #dcmbomexp (component, pitemcode, itemcode, quantity,stack\_level) select component, pitemcode, itemcode, sum ( quantity ), max ( stack\_level ) from #stack group by component, pitemcode, itemcode order by max ( stack\_level ) declare resultcur cursor for select component, pitemcode, itemcode, quantity, stack\_level from #dcmbomexp open resultcur fetch resultcur into @component, @pitemcode, @itemcode, @quantity, @stack\_level while @@fetch\_status = 0 begin select @nxtnum = @nxtnum + 1 insert into #bomwhereused (code, Component, parent, fg, bomqty, LevelNum) select right(('00000000'+convert(varchar,@nxtnum)),8), @component,@pitemcode,@itemcode,@quantity,@stack\_level fetch resultcur into @component, @pitemcode, @itemcode, @quantity,@stack\_level end close resultcur deallocate resultcur delete #dcmbomexp fetch mainloop into @itemcode, @whs, @qty end close mainloop deallocate mainloop declare storeloop cursor for select s.levelnum, s.fg, s.code from #bomwhereused s open storeloop fetch storeloop into @stack\_level, @parent, @sbocode while @@fetch\_status = 0 begin -- 9 select @level = @stack\_level - 1, @dispfull = '' while @level > 0 begin select @dispfull = @disp + @dispfull select @level = @level - 1 end update #bomwhereused set displayitem = @dispfull + @parent where code = @sbocode fetch storeloop into @stack\_level, @parent, @sbocode end close storeloop deallocate storeloop if @Showdocuments ='N' begin select w.DisplayItem as 'Indented Where Used' , w.FG as 'Where used item' , f.itemname as 'Where used Item Description' , levelnum as 'BOM Level' , BOMQty as 'Qty of where used item reqd' , w.Component , ic.itemname as 'Component Description' , w.Parent , p.itemname as 'Parent Description' from #bomwhereused w inner join oitm ic on ic.itemcode = w.component collate SQL\_Latin1\_General\_CP850\_CI\_AS inner join oitm p on p.itemcode = w.parent collate SQL\_Latin1\_General\_CP850\_CI\_AS inner join oitm f on f.itemcode = w.fg collate SQL\_Latin1\_General\_CP850\_CI\_AS order by w.LevelNum, w.parent end else begin create table #dcmdocuments (Item varchar (50),Itemname varchar (100),Objectdesc varchar(20),Detailtype varchar(20), Document int , Cardcode varchar(20), Cardname varchar(100), PDO int, docdate datetime, OpenQty numeric(19,6),DueDate datetime,Whscode varchar(8) , Docentry int, objecttype varchar(20)) declare doccursor cursor for select distinct fg from #bomwhereused open Doccursor fetch Doccursor into @itemcode while @@fetch\_status = 0 begin -- PO insert #dcmdocuments (Item, Itemname, OpenQty, DueDate, Objectdesc, Detailtype, Document, Cardcode, Cardname, PDO, docdate,Whscode,Docentry,ObjectType ) select @itemcode, d.dscription, d.openqty, d.shipdate, 'PO', '', h.docnum, h.cardcode, h.cardname, 0, h.docdate, d.whscode, h.docentry, '22' from por1 d inner join opor h on h.docentry = d.docentry where h.docstatus = 'O' and h.canceled = 'N' and d.openqty > 0 and d.itemcode = @itemcode -- Sales Orders insert #dcmdocuments (Item, Itemname, OpenQty, DueDate, Objectdesc, Detailtype, Document, Cardcode, Cardname, PDO,docdate, Whscode,Docentry,ObjectType) select @itemcode, d.dscription, d.openqty, d.shipdate, 'Sales Order', '', h.docnum, h.cardcode, h.cardname, 0, h.docdate, d.whscode, h.docentry, '17' from rdr1 d inner join ordr h on h.docentry = d.docentry where h.docstatus = 'O' and h.canceled = 'N' and d.openqty > 0 and d.itemcode = @itemcode -- PDOs insert #dcmdocuments (Item, Itemname, OpenQty, DueDate, Objectdesc, Detailtype, Document, Cardcode, Cardname, PDO, docdate, Whscode,Docentry,ObjectType) select @itemcode, i.itemname, d.plannedqty - d.cmpltqty, d.createdate, 'Production Order', 'Parent', 0, d.cardcode, c.cardname,d.docnum, d.postdate, d.warehouse, d.docentry,'' from owor d inner join oitm i on i.itemcode = d.itemcode left join ocrd c on c.cardcode = d.cardcode where d.status = 'R' and d.cmpltqty < d.plannedqty and d.itemcode = @itemcode fetch Doccursor into @itemcode end close doccursor deallocate doccursor select 'Documents' as 'Documents' , i.itemcode, d.Itemname as 'Item description', d.OpenQty as 'Open Qty', d.DueDate as 'Due Date', d.Objectdesc as 'Document type', d.Detailtype as 'Detail Type', 'Check Document' as 'Check Usage', '' as 'Spacer', d.Document, c.Cardcode as 'BP Code', d.Cardname as 'Vendor / Customer name', d.pdo as 'PDO', d.Docdate as 'Document date', d.Whscode as 'Whs', d.docentry, d.objecttype from #dcmdocuments d inner join oitm i on i.itemcode = item collate SQL\_Latin1\_General\_CP850\_CI\_AS left join ocrd c on c.cardcode = d.cardcode collate SQL\_Latin1\_General\_CP850\_CI\_AS order by d.Objecttype, d.document, d.item for browse drop table #dcmdocuments end drop table #dcmbomexp drop table #stack drop table #bomwhereused drop table #itemlist



## Figure A-6 Where Used Documents Report

Note: this UF sets a ‘Y’ value for the variable @STORE3 and then calls the where-used report. It forces the where used results to display the where-used documents information.

## BOM Where Used and Indented BOM From BOM Form



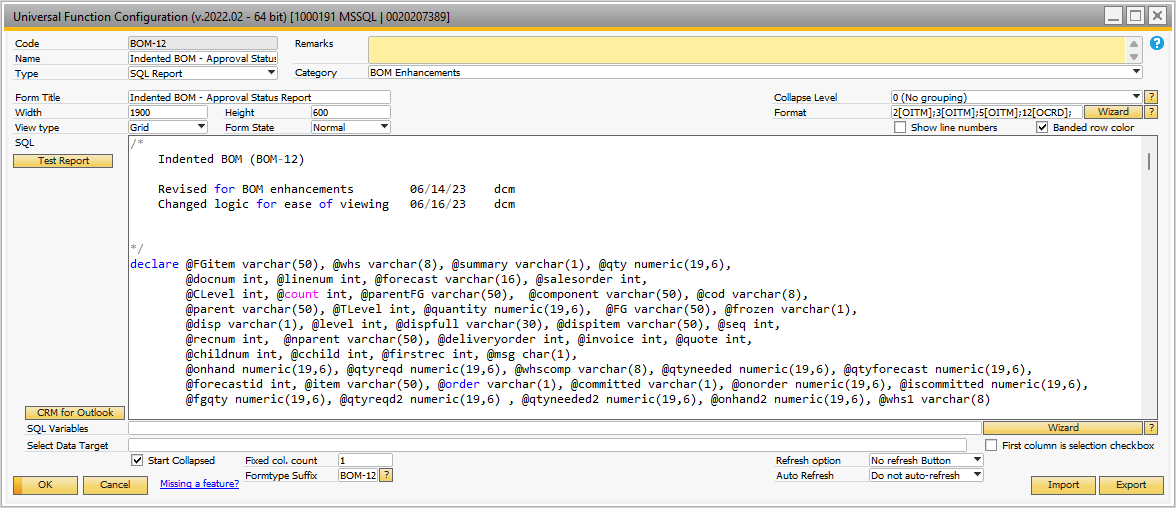
## Figure A-7 Validation Called from the BOM Form With BOM Reports Button

This validation calls the BOM-01 universal function shown in Figure A-3. There are two possible options:

* The BOM parent item approval flag (property 64 in item master) is set to ‘N’ – two reports are presented as options: Indented BOM (BOM-02); or Approval Status (BOM-12) report.
* The BOM parent item approval flag (property 64 in item master) is set to ‘Y’ – 2 options are presented: Indented BOM (BOM-02); or Approval UF (BOM-11). The first is a report and the second updates the approval flags.

BOM-02 was discussed earlier and is the same as called from the item master.

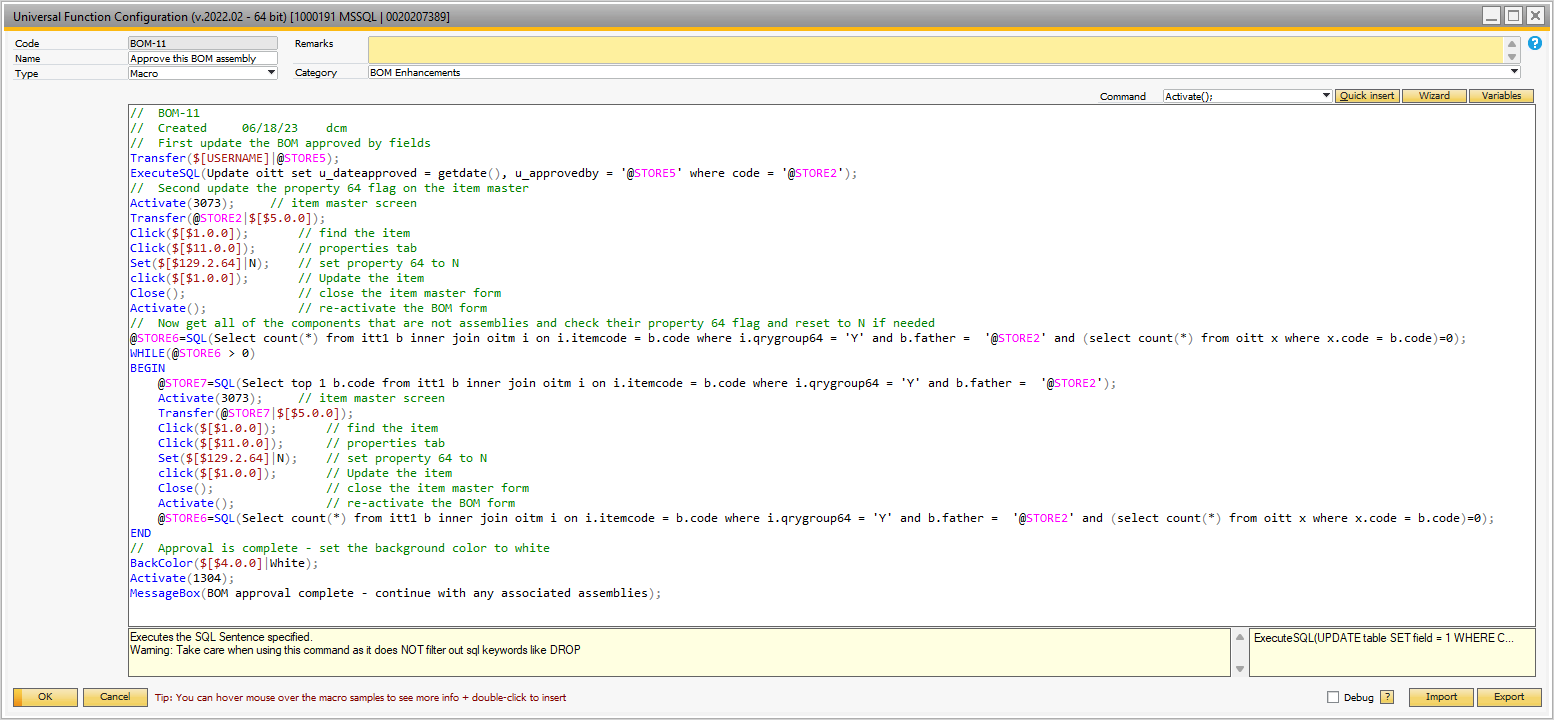
BOM-12 is the approval status report and is shown in Figure A-8.



## Figure A-8 Approval Status Report

This report is similar to the Indented BOM report. The unique SQL portion of the code is shown below.

select s.itemdisp as 'Indented Item' , s.fg as 'BOM FG' , s.parent as 'Parent Item' , ii.itemname as 'Parent Description' , case when ii.qrygroup64 = 'Y' then 'Needs Approval' when ii.itemcode is null then '' else 'Approved' end as 'Approval Status - Parent' , s.component as 'Component Item' , case when cc.itemname is null then r.resname else cc.itemname end as 'Component Description' , case when cc.qrygroup64 = 'Y' then 'Needs Approval' when cc.itemcode is null then '' else 'Approved' end as 'Approval Status - Component' , case when cc.qrygroup64 = 'N' then b.u\_dateapproved else '' end as 'Approval Date' , case when cc.qrygroup64 = 'N' then b.u\_approvedby else '' end as 'Approved By' , cc.phantom as 'Phantom' , cc.cardcode as 'Default Vendor' , s.TLevel as 'Level in BOM' , s.qtyreqd as 'Qty Reqd' , isnull(c.onhand,0) as 'Component On Hand' , isnull(c.onorder,0) as 'Component on order' , isnull(c.iscommited,0) as 'Component Committed' , (isnull(c.onhand,0) + isnull(c.onorder,0) - isnull(c.iscommited,0)) as 'Net On Hand' , i.onhand as 'Parent On Hand' , s.childnum , s.recnum from #stack s left join oitw i on i.itemcode = s.parent and i.whscode = @whs left join oitm ii on ii.ItemCode = i.itemcode left join oitw c on c.itemcode = s.component and c.whscode = @whs left join oitm cc on cc.itemcode = c.itemcode left join orsc r on r.rescode = @component left join oitt b on b.code = cc.itemcode order by s.recnumdrop table #stackdrop table #itemlistdrop table #onhanddrop table #peggedusage

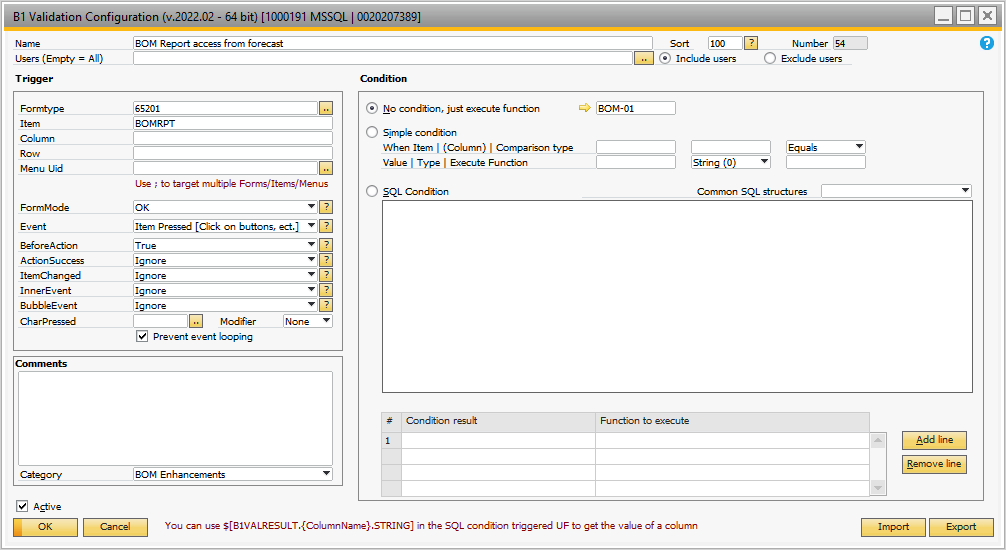


## Figure A-9 Approval Process

The approval process resets the property 64 flag in the item master and then updates the BOM approve by and approval date fields. If the BOM has any components that are not BOMs themselves, then their property flag 64 is also reset. The approval is for this level of the assembly only – you must continue the approval process one assembly at a time. Once this is complete, run the approval status report to confirm that all levels of the BOM have been approved.

## BOM Report Access From Forecast

When the user clicks the ‘BOM Reports’ button on the MRP forecast form, the validation shown in Figure A-10 is invoked. It calls the ‘familiar’ BOM-01 universal function. If you’re getting the idea that this UF is the gatekeeper for the BOM reporting – you’re correct!

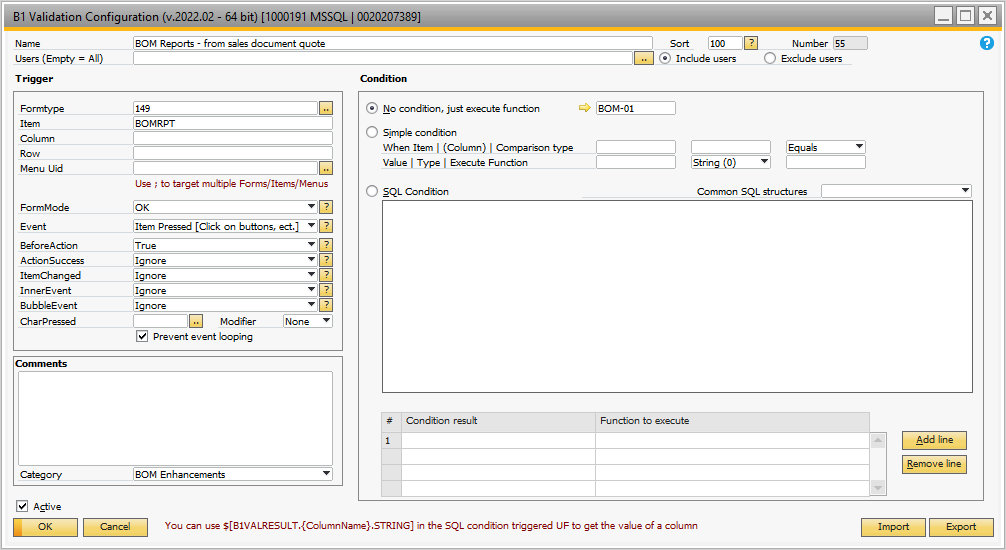


## Figure A-10 Access BOM Report From MRP Forecast Form

Within the BOM-01 universal function the macro stores the forecast id and then initiates the indented BOM report by calling the UF BOM-02. It is passed the originating form – MRP Forecast – in the variable STORE1. The variable STORE2 contains the Forecast id. Within the BOM-02 reporting, the MRP Forecast is used to create a table of all items whose indented BOMs should be reported. i.e. if you had 10 items in the forecast, the indented BOM for each of the 10 items would be reported. Be careful – the forecast could have hundreds of items in the forecast. It would be a messy, long report to show each indented BOM. If there are too many, simply use the gold arrow for the item(s) on the MRP Forecast and run the indented BOM from the item master or BOM form.

## BOM Report from Sales Document – Quote

When the user clicks the ‘BOM Reports’ button on the Sales Quote form, the validation shown in Figure A-11 is invoked. It calls the ‘familiar’ BOM-01 universal function.

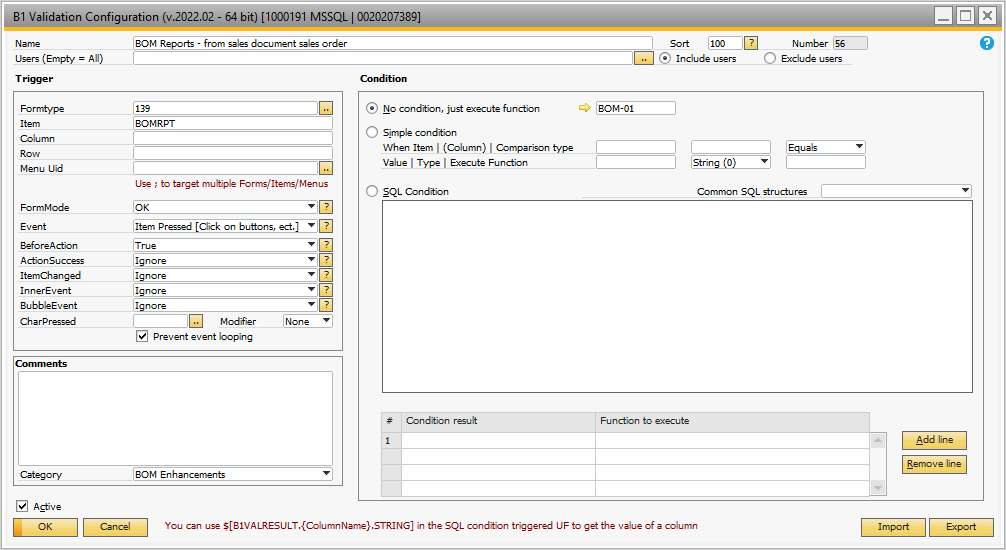


## Figure A-11 Indented BOM Reporting from Sales Quote

Within the BOM-01 universal function the macro stores the sales quote number and then initiates the indented BOM report by calling the UF BOM-02. It is passed the originating form – Sales Quote– in the variable STORE1. The variable STORE2 contains the sales quote number. If the quote contains 5 items, then 5 indented BOMs would be reported.

## BOM Report from Sales Document – Sales Order

When the user clicks the ‘BOM Reports’ button on the Sales Quote form, the validation shown in Figure A-12 is invoked. It calls the ‘familiar’ BOM-01 universal function.

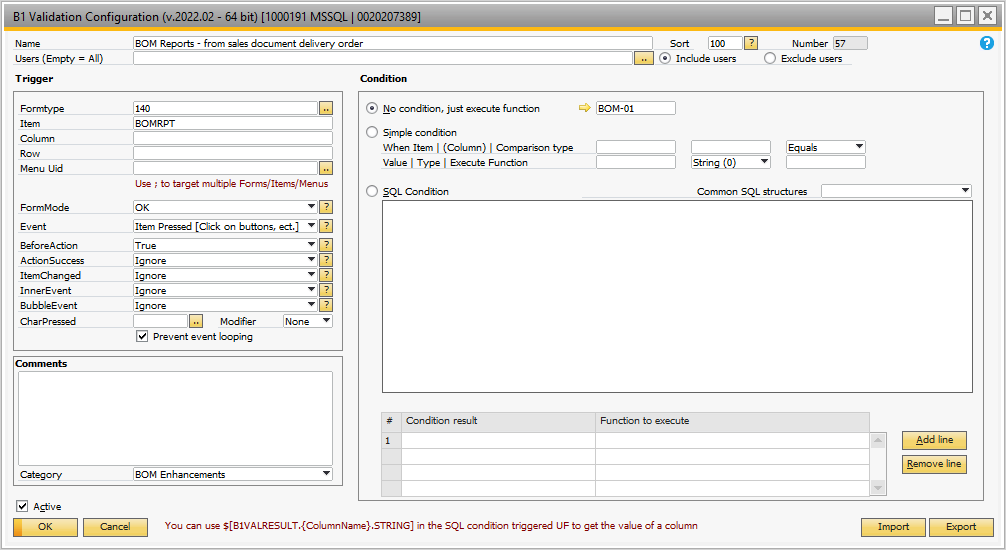


## Figure A-12 Indented BOM Reporting from Sales Order

Within the BOM-01 universal function the macro stores the sales order number and then initiates the indented BOM report by calling the UF BOM-02. It is passed the originating form – Sales Order– in the variable STORE1. The variable STORE2 contains the sales quote number. If the sales order contains 5 items, then 5 indented BOMs would be reported.

## BOM Report from Sales Document – Delivery Order

When the user clicks the ‘BOM Reports’ button on the Sales Quote form, the validation shown in Figure A-13 is invoked. It calls the ‘familiar’ BOM-01 universal function.

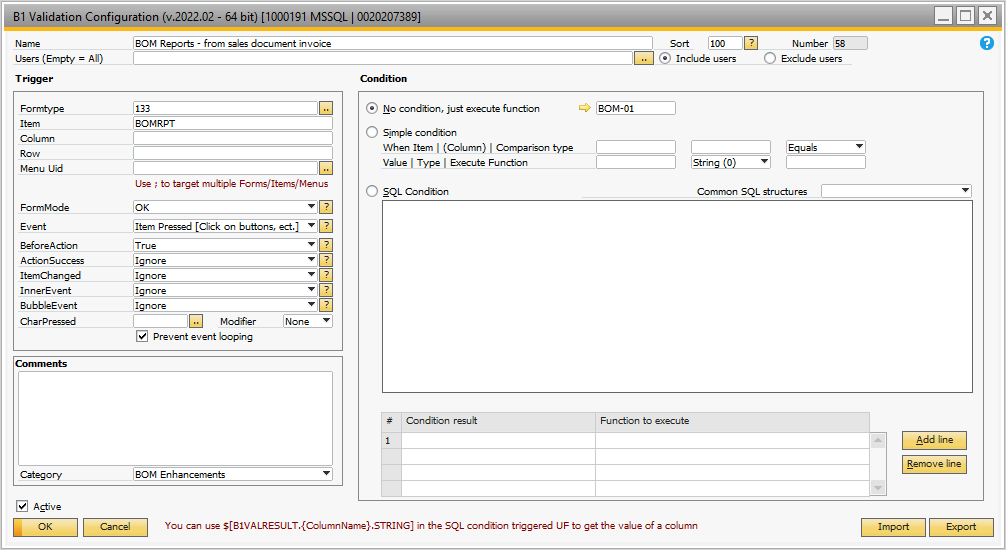


## Figure A-13 Indented BOM Reporting from Delivery Order

Within the BOM-01 universal function the macro stores the delivery order number and then initiates the indented BOM report by calling the UF BOM-02. It is passed the originating form – Delivery Order– in the variable STORE1. The variable STORE2 contains the sales quote number. If the delivery order contains 5 items, then 5 indented BOMs would be reported.

## BOM Report from Sales Document – Invoice

When the user clicks the ‘BOM Reports’ button on the Invoice form, the validation shown in Figure A-14 is invoked. It calls the ‘familiar’ BOM-01 universal function.

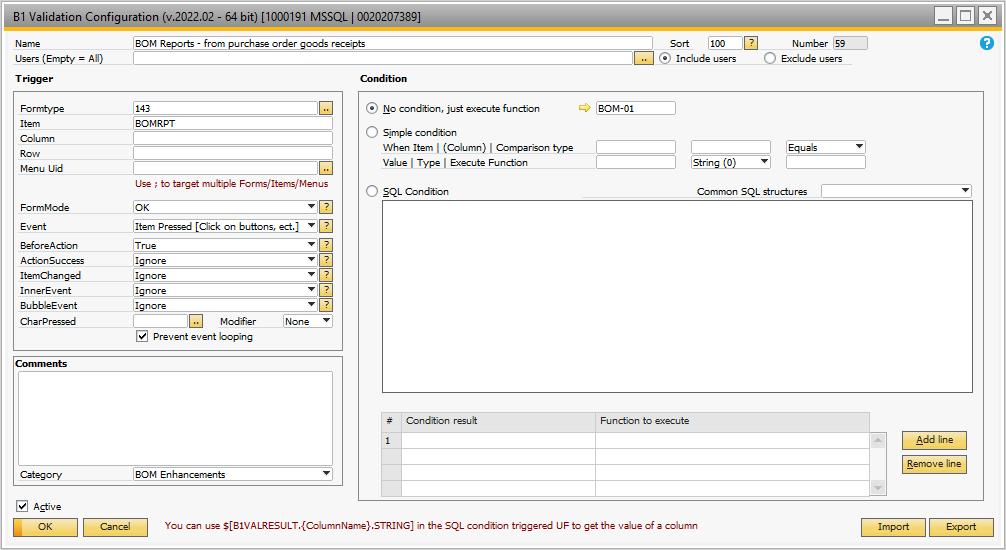


## Figure A-14 Indented BOM Reporting from Invoice

Within the BOM-01 universal function the macro stores the Invoice number and then initiates the indented BOM report by calling the UF BOM-02. It is passed the originating form – Invoice– in the variable STORE1. The variable STORE2 contains the invoice number. If the invoice contains 5 items, then 5 indented BOMs would be reported.

## BOM Reports – from Purchase Order Goods Received

When the user clicks the ‘BOM Reports’ button on the PO Goods received form, the validation shown in Figure A-15 is invoked. It calls the ‘familiar’ BOM-01 universal function.

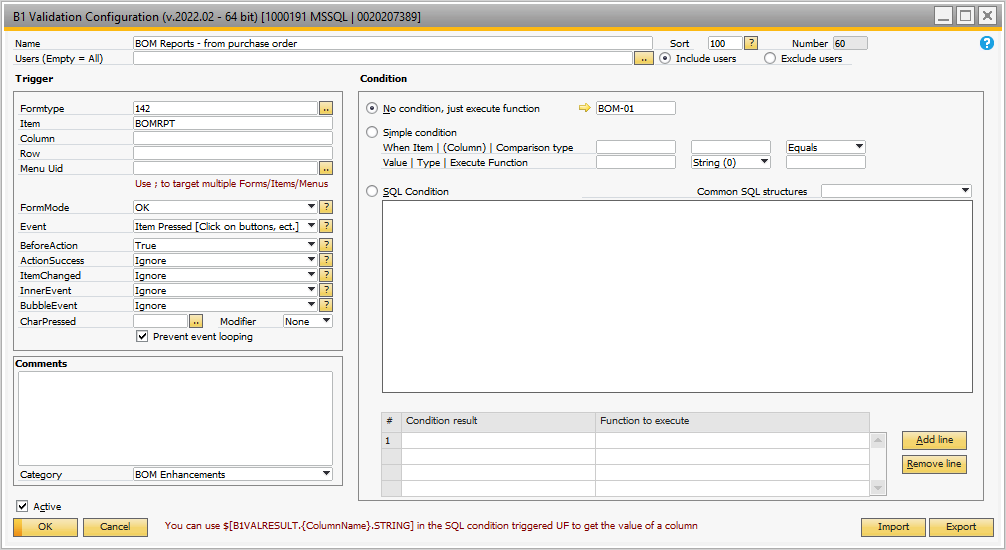


## Figure A-15 Where Used Reporting from PO Goods Received

Within the BOM-01 universal function the macro stores the PO Goods Received number and then shows a popup providing the user the choice of displaying either the Where-Used report or the Where-Used Documents report. These reports are passed the originating form – PO Goods Receipt– in the variable STORE1. The variable STORE2 contains the PO Goods Receipt number. If the PO Goods Receipt contains 5 items, then 5 where used items would be reported.

## BOM Reports – from Purchase Order

When the user clicks the ‘BOM Reports’ button on the PO form, the validation shown in Figure A-16 is invoked. It calls the ‘familiar’ BOM-01 universal function.

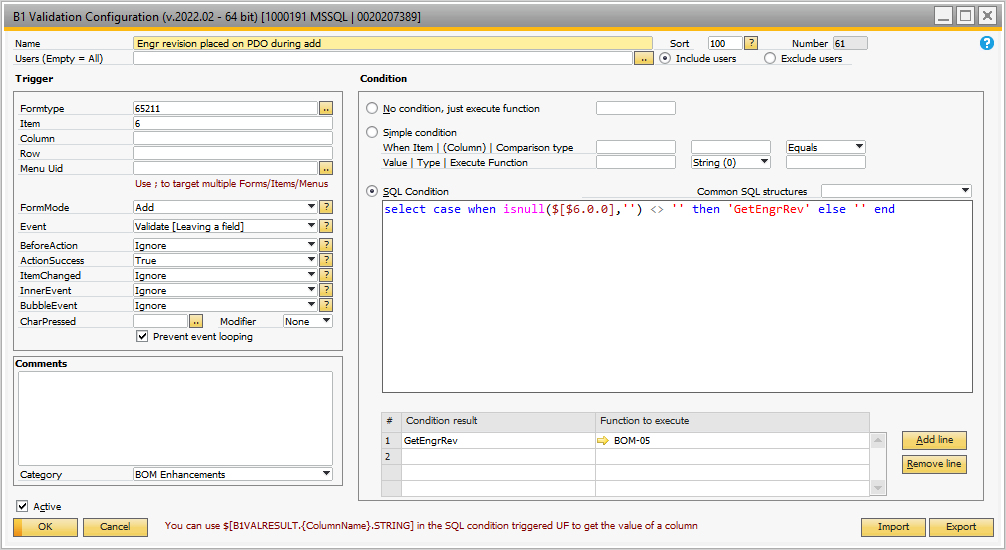


## Figure A-16 Where Used Reporting from PO

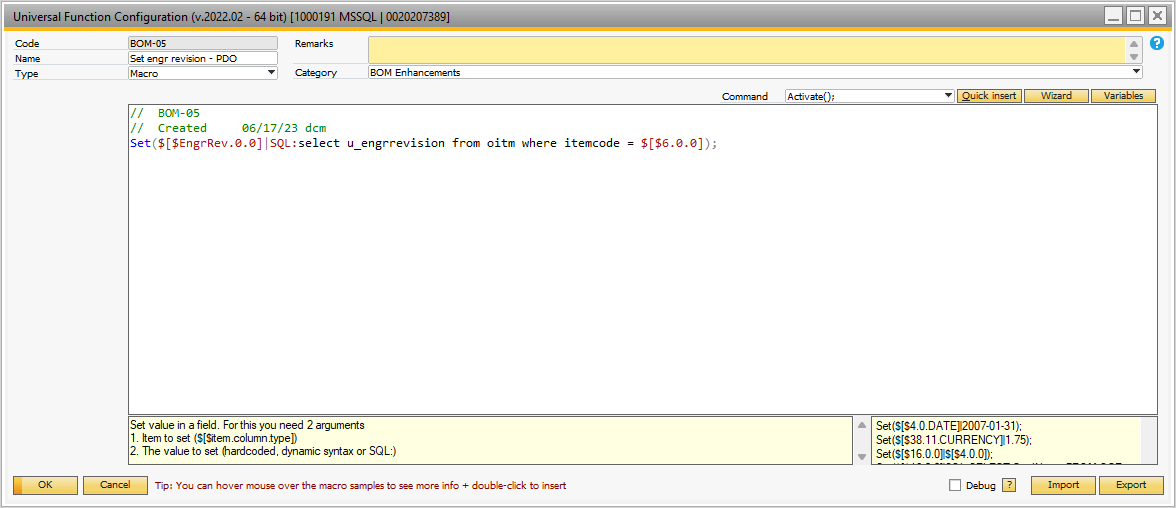
Within the BOM-01 universal function the macro stores the PO number and then shows a popup providing the user the choice of displaying either the Where-Used report or the Where-Used Documents report. These reports are passed the originating form – PO– in the variable STORE1. The variable STORE2 contains the PO number. If the PO contains 5 items, then 5 where used items would be reported.

## Engr Revision Placed on PDO During Add

This B1 Validation places the engineering revision for an item which is stored in the item master on the PDO screen. Figure A-17 shows the B1 Validation. The UF BOM-05 is shown in Figure A-18.



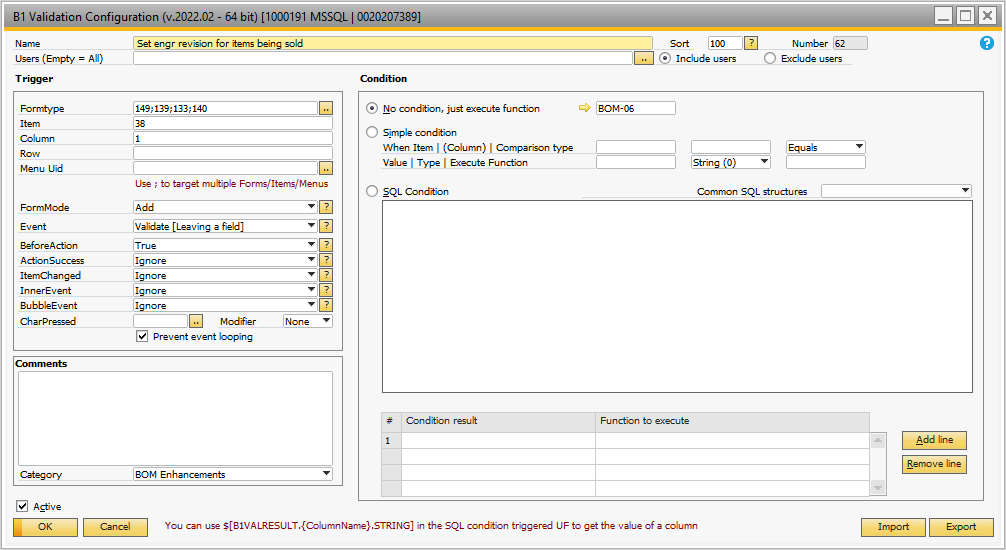
## Figure A-17 Placing the Engineering Revision on the PDO for the Item Manufactured



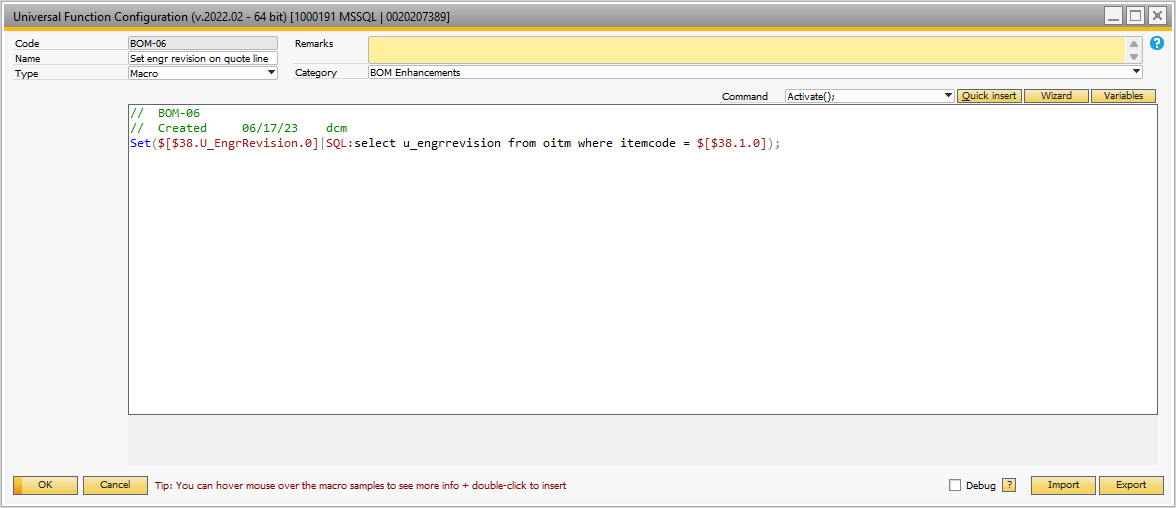
## Figure A-18 Macro Placing the Engineering Revision on the PDO

## Set Engineering Revision for Items Being Sold

The B1 Validation for setting the engineering revision on the sales quote, sales order, delivery order and invoicing forms is shown in Figure A-19. As items are added to these documents the engineering revision is included in the document line item information.



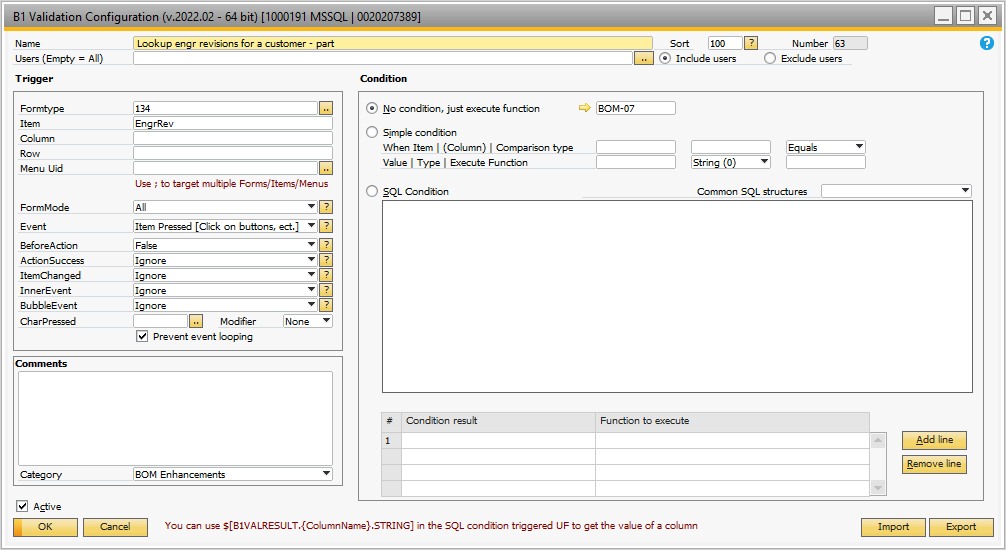
## Figure A-19 Sales Documents – Setting the Engineering Revision



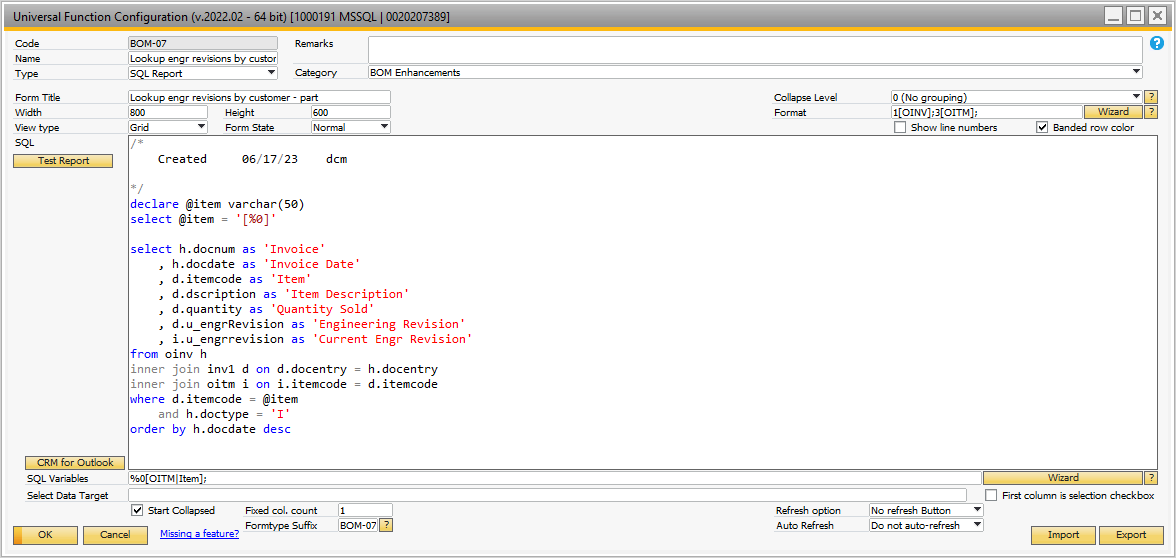
## Figure A-20 Setting the Engineering Revision on Sales Documents

## Lookup Engineering Revisions for a Customer – Part

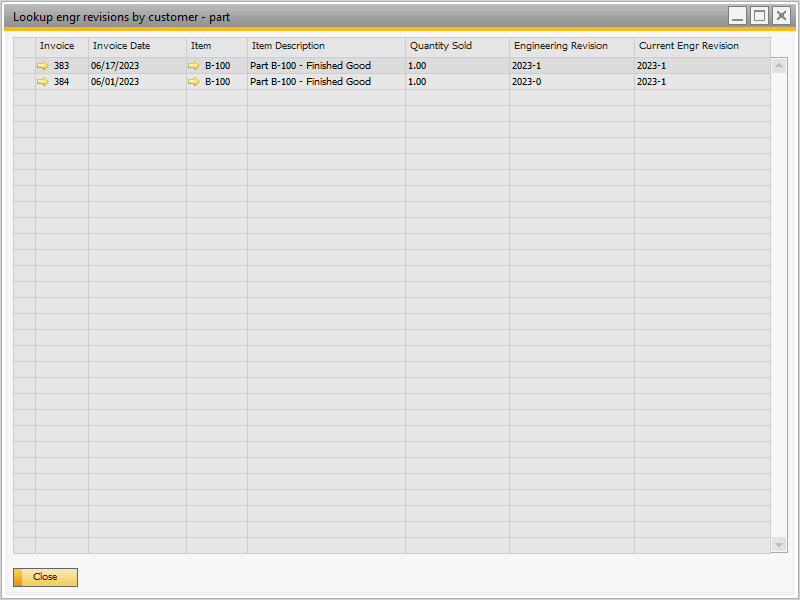
The business reason for maintaining engineering revision information for every sale is so you can readily look up information for customer – part issues – and always get to the correct engineering revision drawings quickly. The B1 Validation that is called when the ‘Engr Rev Lookup’ button is clicked on the business partner master form is shown in Figure A-21. The reporting UF BOM-07 is shown in Figure A-22. The report produced is shown in Figure A-23.



## Figure A-21 Show Engineering Revision History By Customer – Part



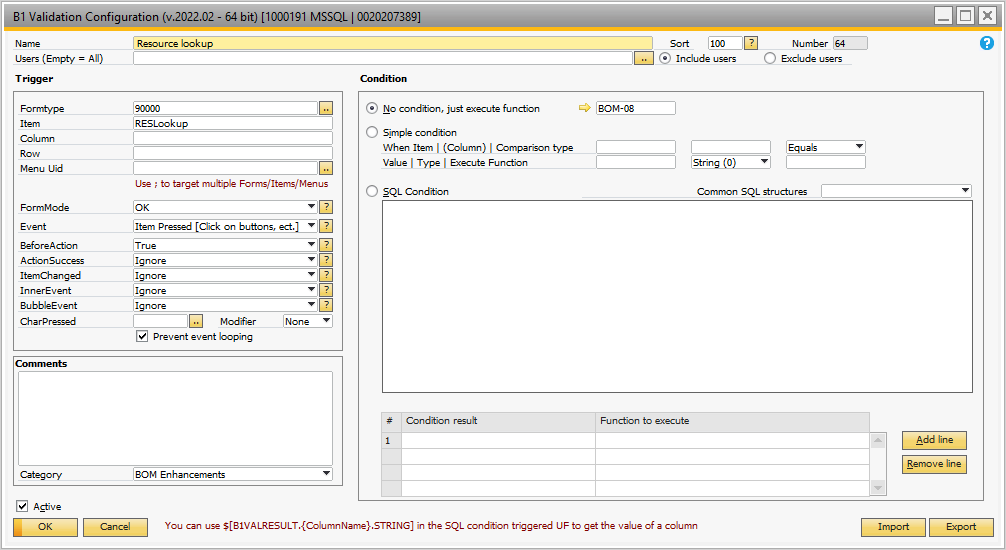
## Figure A-22 Show Engineering Revision History By Customer – Part



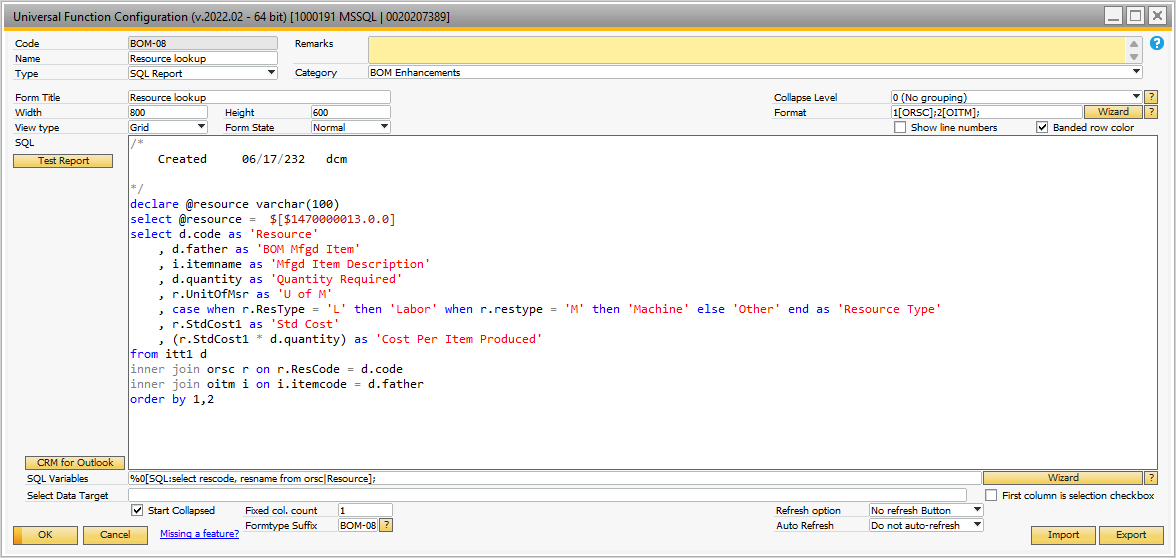
## Figure A-23 Engineering Revision History By Customer – Part

## Resource Lookup

When changing BOMs, the resources often need to be changed as well. The resource lookup provides a means to look at which BOMs are using specific resources. Figure A-24 shows the validation called from the ‘Resource Master’ form when the user clicks on the ‘Resource Lookup Button’. Figure A-25 is the UF BOM-08 which displays the report of resource usage.



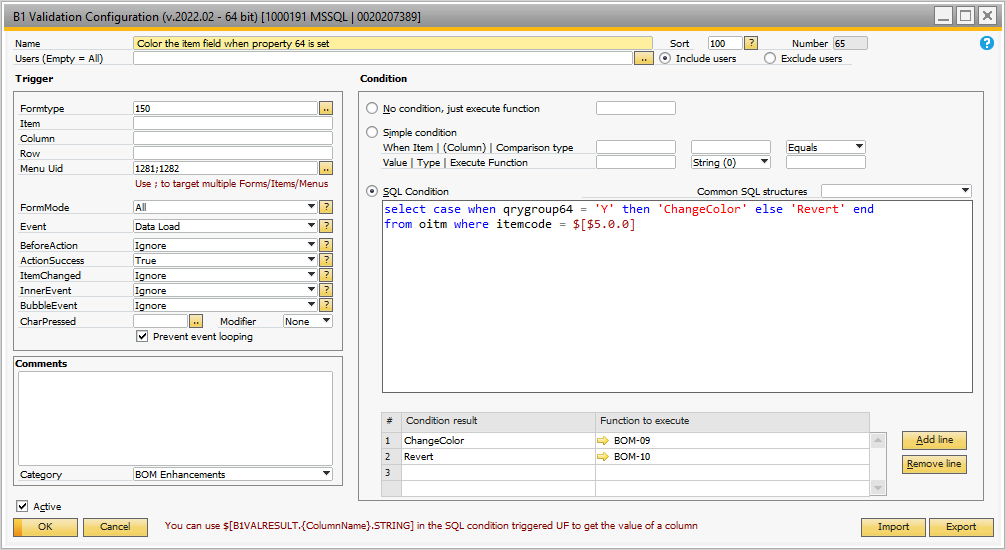
## Figure A-24 Validation Called When Resource Lookup Button Clicked



## Figure A-25 Resource Lookup Report

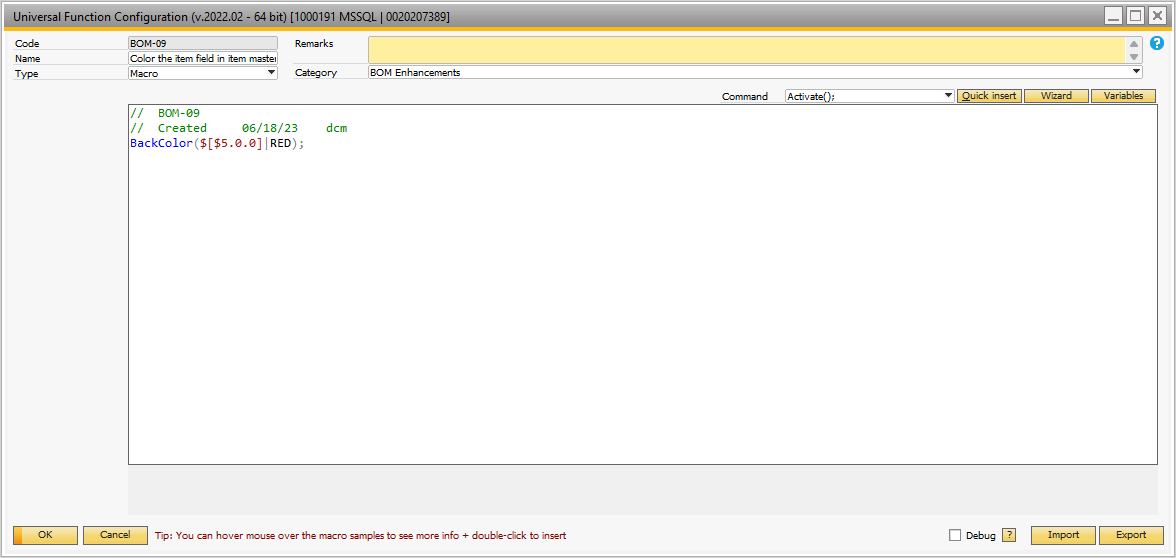
## Color the item field when property 64 is set – Item Master

If the user sets property 64 to ‘checked’ for an item this validation is called when the item is loaded into the item master form. This validation is shown in Figure A-26.



## Figure A-26 Validation for Color of Item

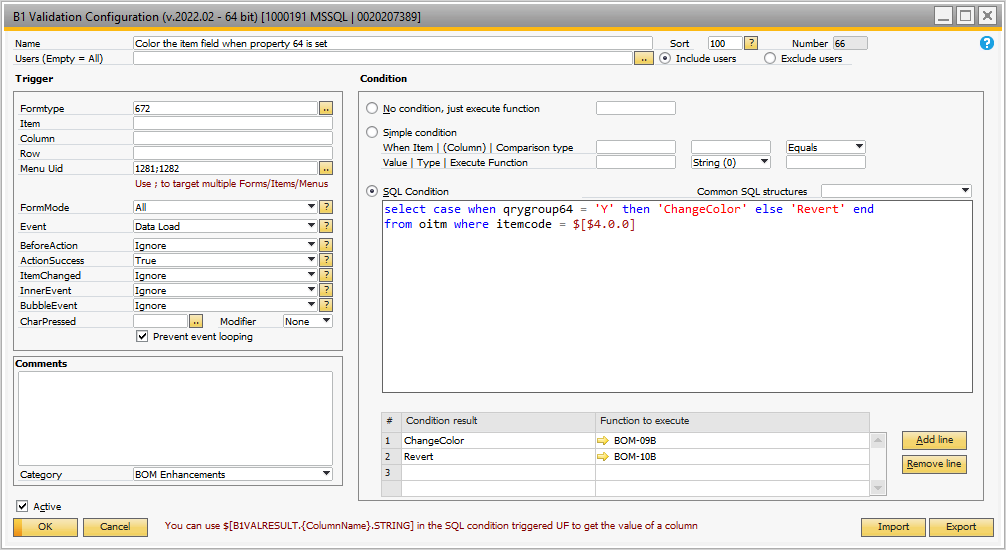
Depending on the value of the property 64 flag, the item background is either set to red or white. The universal functions to set the color are shown in Figures A-27 and A-28.



## Figure A-27 Set Background Color to Red

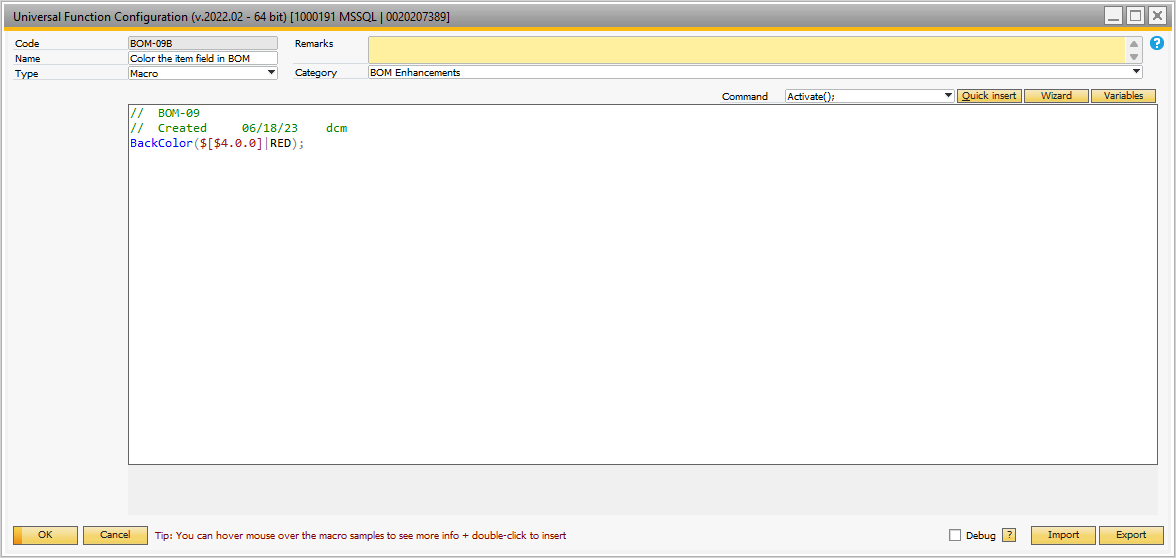
## Color the item field when property 64 is set – BOM Form

If the user sets property 64 to ‘checked’ for an item this validation is called when the item is loaded into the BOM form. This validation is shown in Figure A-28.

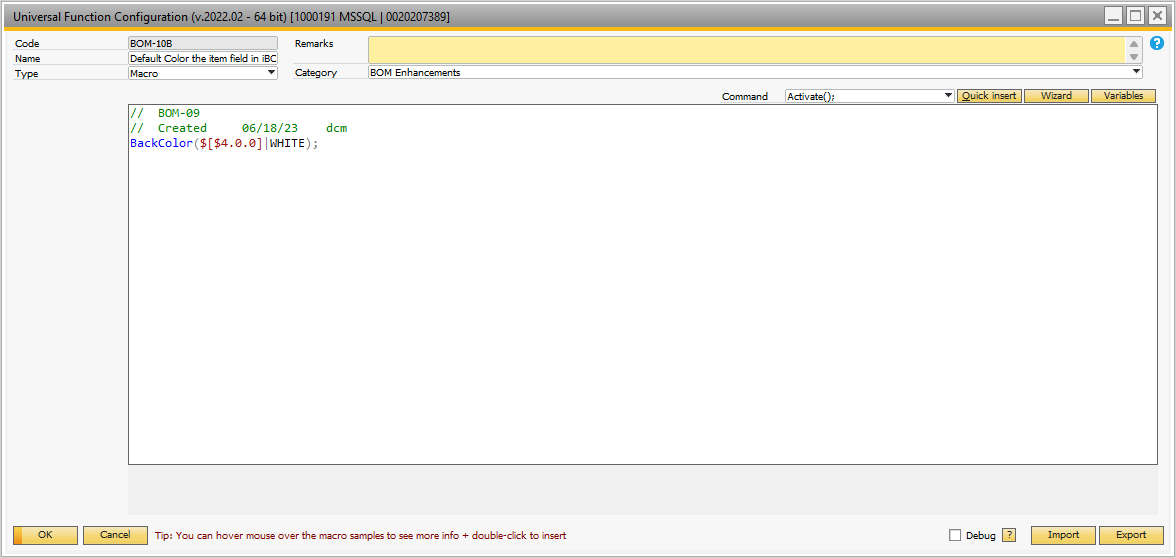


## Figure A-28 Validation for Color of Item

Depending on the value of the property 64 flag, the item background is either set to red or white. The universal functions to set the color are shown in Figures A-29 and A-30.



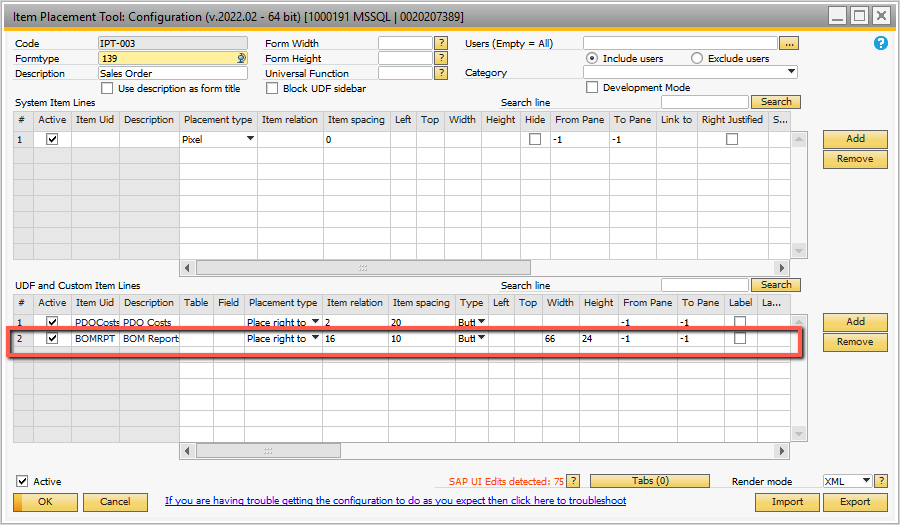
## Figure A-29 Set Background Color to Red



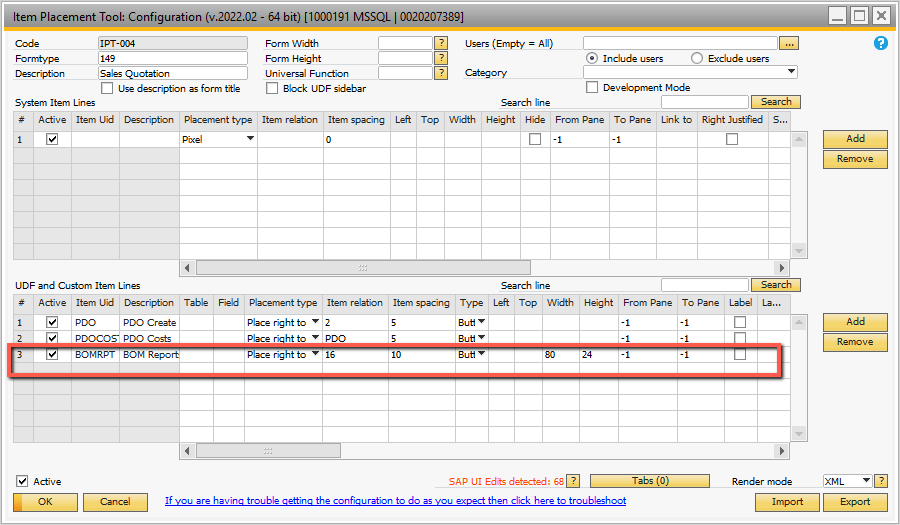
## Figure A-30 Set Background Color to White

# Item Placement Tool Changes

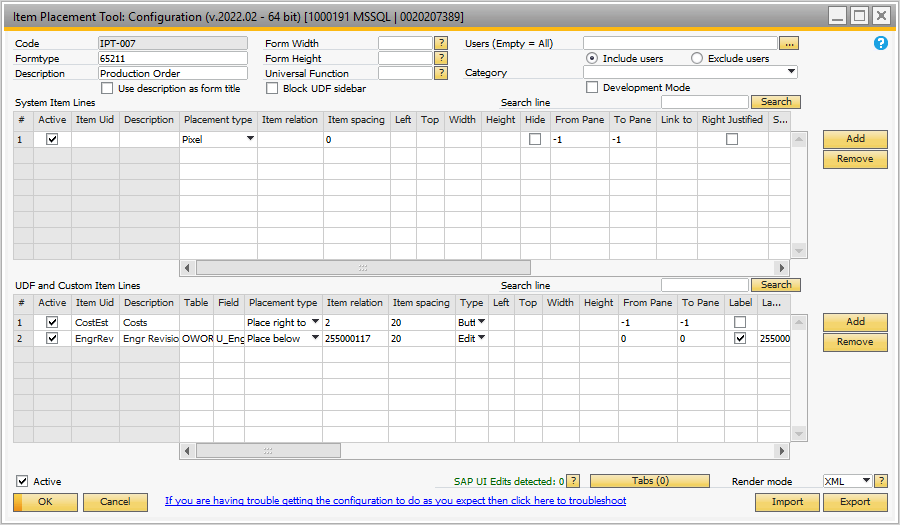
The forms were modified to move user defined fields to the main forms. Buttons were added for the BOM Reports on several forms. These changes are shown below. Note: for most forms the ‘buttons’ added are only active when in ‘Edit OK’ mode.



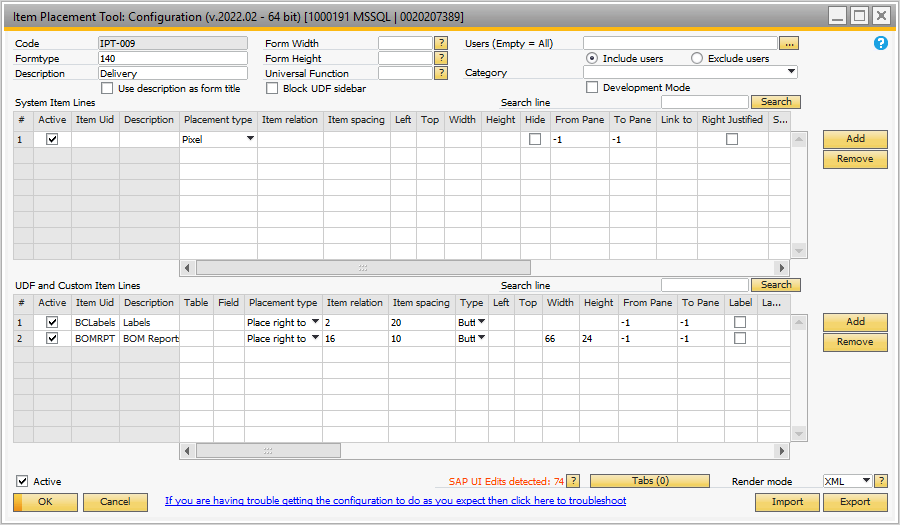
## Figure A-31 Sales Order Form – Add Button ‘BOM Reports’



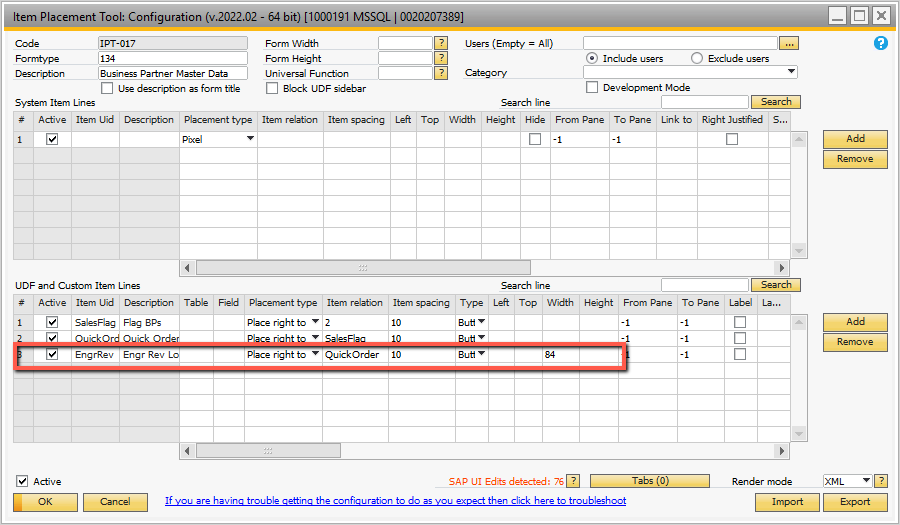
## Figure A-32 Sales Order Quote Form – Add Button ‘BOM Reports’



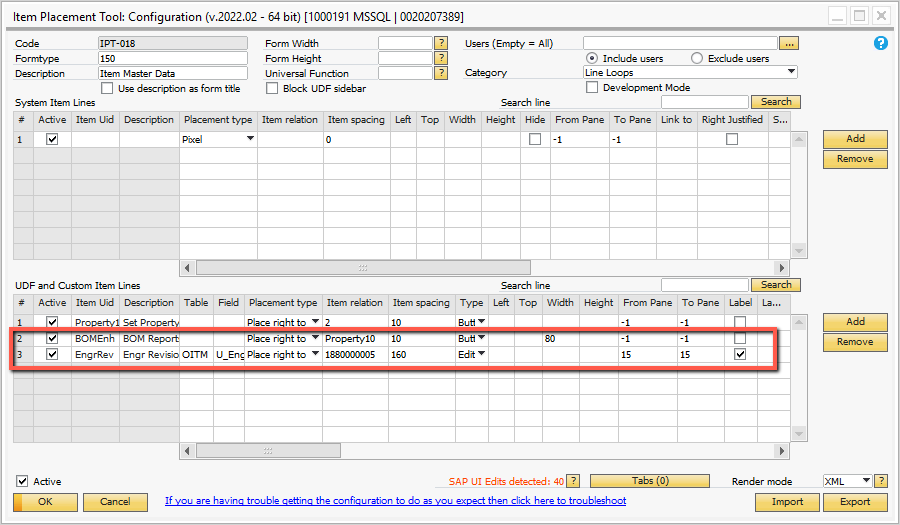
## Figure A-33 Production Order Form – Add Engineering Revision UDF



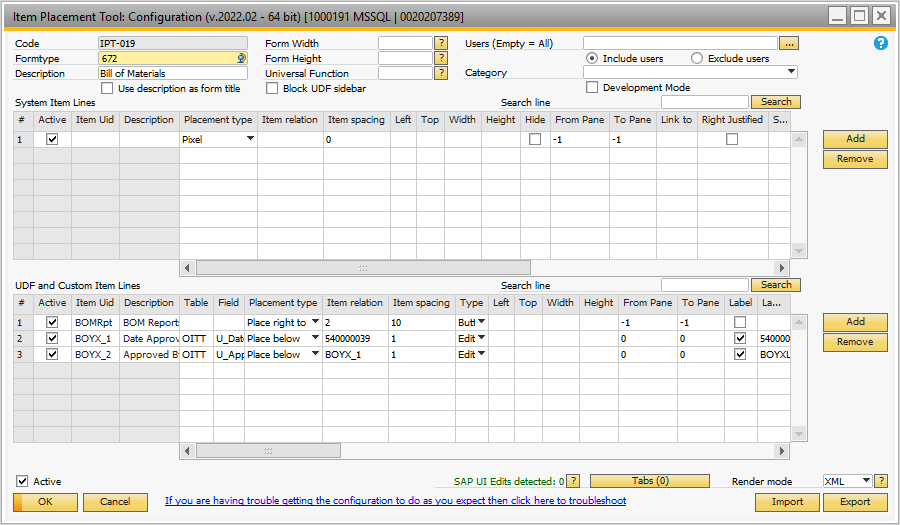
## Figure A-34 Delivery Order Form – Add Button ‘BOM Reports’



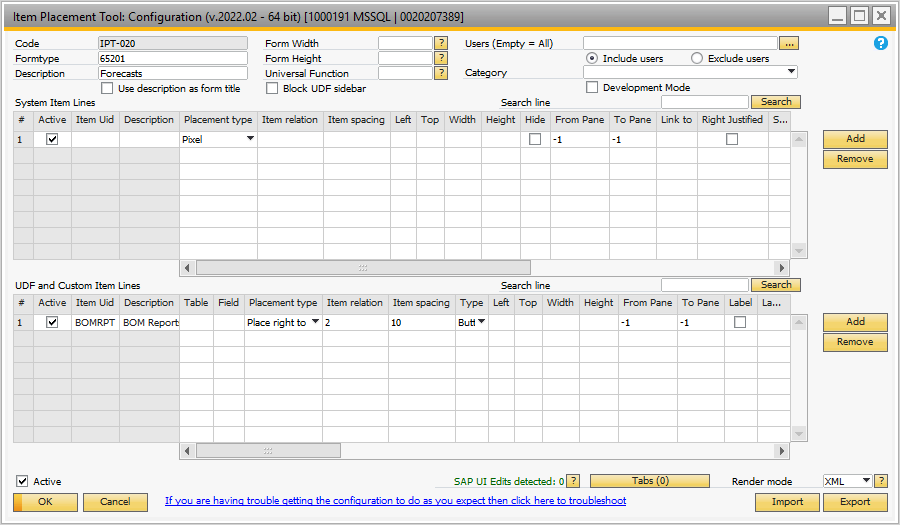
## Figure A-35 Business Partner Form – Add Button ‘Engr Rev Lookup’



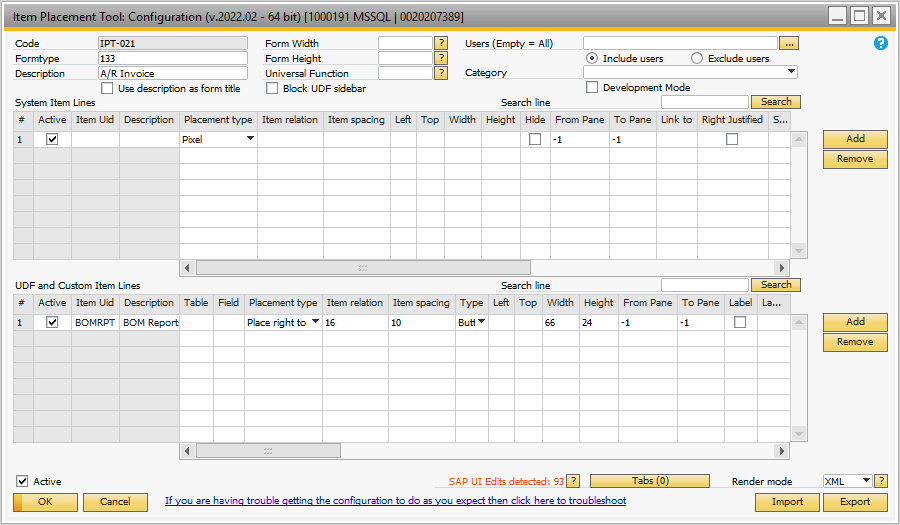
## Figure A-36 Item Master Form – Add Button + UDF



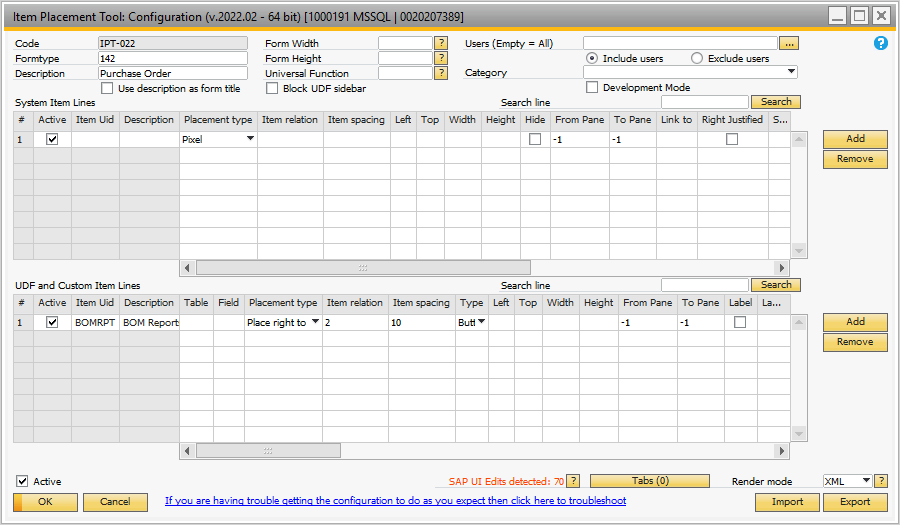
## Figure A-37 BOM Form – Add Button + 2 UDFs



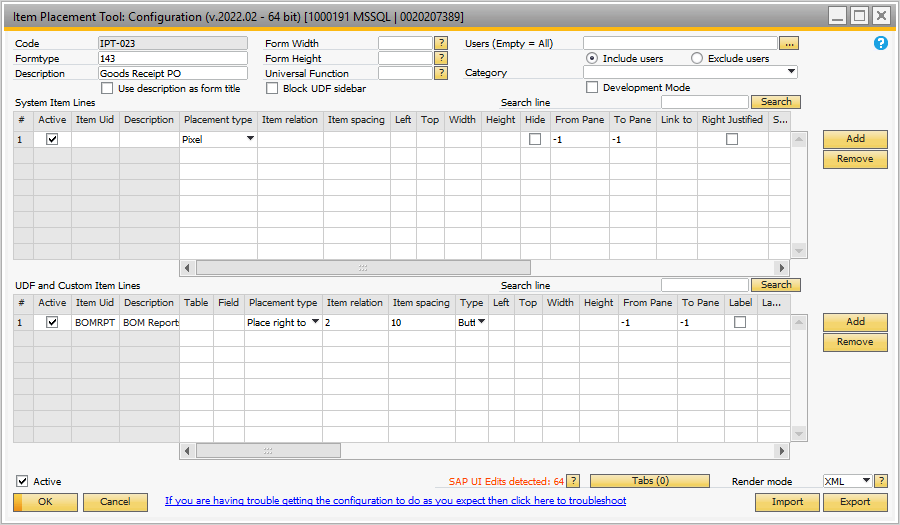
## Figure A-38 Forecast Form – Add Button



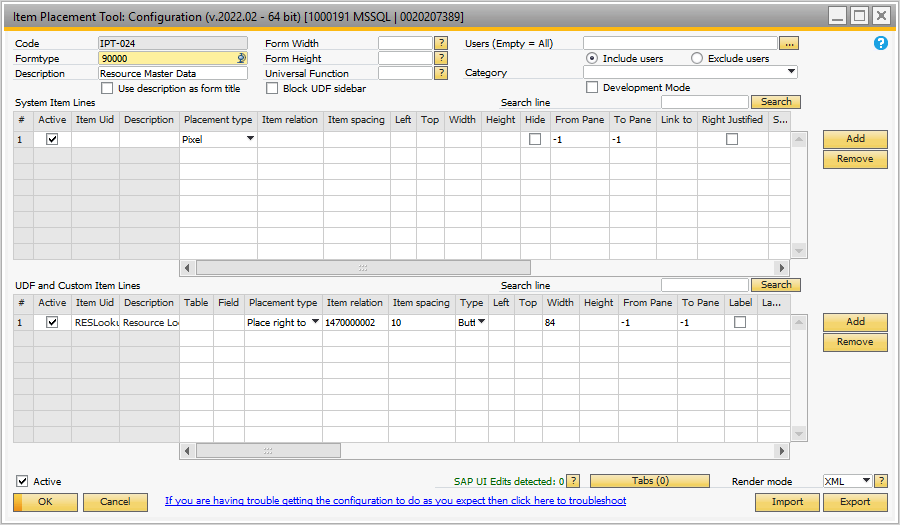
## Figure A-39 A/R Invoice Form – Add Button



## Figure A-40 Purchase Order Form – Add Button



## Figure A-41 Purchase Order Goods Receipt Form – Add Button



## Figure A-42 Resource Master Form – Add Button