Autodesk University 2012

PL2818 – BOMs in the Cloud with Autodesk PLM 360

Hagay Dvir – Autodesk
Kevin Robinson – Autodesk

PL2818 In this class, we will introduce you to the different types of bills of materials (BOMs) and show you how you can use Autodesk PLM 360 to manage them. We will discuss the different types of BOMs, such as engineering, manufacturing, and product, and how the different types will influence your solution. The class will also cover how different types of products, such as engineered-to-order, configured-to-order, or volume manufacturing, affect the BOMs. Finally, we will discuss how to get BOM data into Autodesk PLM 360 and how you can use it once it is there.

Learning Objectives

At the end of this class, you will be able to:

- Explain the difference between engineering and product BOMs and Describe how engineering and product BOMs relate to each other and to other types of BOMs
- Learn how to use PLM 360 to model your BOM requirements
- Manage BOMs in Autodesk PLM 360
- Describe how the product BOM relates to other elements of product lifecycle

About the Speaker

Hagay Dvir is a Product Manager with the PLM 360 product and is responsible for the development of advanced Engineering solutions based on the PLM 360 platform. His past experiences include the implementation of PLM systems in large enterprises. One of the key areas that Hagay worked in is the consumption of Product Design by Manufacturing organization and the development of the Manufacturing Process Plan.

"My main concern is to develop a BOM system that will be simple enough to use by anybody in the organization, yet capable enough to allow companies to represent and consume their BOMs in the most efficient way"

Hagay received his MBA from Michigan State University, Computer Science degree from Tel-Aviv university, and Mechanical Engineering degree from the Technion.

Best Practices for Product LC Management

Yea…, don't expect to see those in my class. The fact of the matter is that I don't think one can define a 'best practice' for almost anything, let along something with so many variables such as a Product. Products can be simple or complicated, may be made in large or small quantities, may be created by a single company or a complex supply chain, and so on. What IS common to almost all products, though, is the need to maintain a list of everything that goes into the product so that the same product can be made again and again in the same way and to the same quality standards even when (especially when!!!) things change.

There are also some common characteristics of products that define common requirements for Bill of Materials solutions for that group of products. For example, high volume products that are configured to order (such as cars and computers) require some configuration abilities in the BOM whereas low volume products that are Engineered to Order need more flexibility in ad-hoc configuration abilities, usually using Rule Systems.

Similarly, companies are not created in a mold. Differences in size, organizational structure and responsibilities, sales strategy, and other factors require different departments in the organization to require different pieces of information at different times and define different work and data flows across the organization.

So I guess there is, actually, a best practice in Product LC Management after all. The best practice is to implement software solutions that are capable of supporting whatever work processes work best for you. This means that the tool must be capable of modeling a wide variety of data and workflow scenarios, while maintaining an easy to use interface that is tailor to the individual needs of each organization within the enterprise.

In this class we will discuss some of the differences in products that require different capabilities from the product, and show some examples of BOM implementations that demonstrate how to start from a given company-specific workflow model and how to implement it in Autodesk PLM 360.

Addendum to class handout material

Prepare Vault data for PLM 360 Import

Hagay Dvir – Autodesk

Purpose of this document

The purpose of this document is to document a repeatable process for getting BOM data exported from Vault ready for import into PLM 360.

Note: I will be using Microsoft Excel 2010 for this purpose.

Background

The BOM imported in PLM 360 is using a somewhat unorthodox import structure that is not directly supported by the Vault product. In both products the methodology used is that of Parent-Child, but whereas Vault identifies a Parent object on each of the Children, PLM 360 requires identifying a Child object on the respective Parent line.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Parent	Number	Title (Iten	Quantity	Category I	Item Desc	Revision	Units	State	Effectivity	Eff. Start	Eff. End	Comment	Date Modified	Last Updated By	
2	<top>	100220	Assy, Cab.	1	Assembly		A	Each	Released	Effective	11/9/2012 3:28			11/9/2012 3:28	Administrator	
3	100220	101684	027.05.01.	2	Part	027.05.01.	A	Each	Released	Effective	11/9/2012 3:28			11/9/2012 3:28	Administrator	
4	100220	101683	027.05.01.	2	Part	027.05.01.	A	Each	Released	Effective	11/9/2012 3:28			11/9/2012 3:28	Administrator	
5	100220	101682	027.34.11.	4	Part	gasket	A	Each	Released	Effective	11/9/2012 3:28			11/9/2012 3:28	Administrator	
6	100220	101681	027.43.11.	2	Part	pin	A	Each	Released	Effective	11/9/2012 3:28			11/9/2012 3:28	Administrator	

Figure 1 - Vault exported structure

This means that each Parent data need to be replicated for each of the Children in the format imported by PLM 360

	A	B	C	D	E	F	G	H	I
1	Parent	Number	vault Item	CAD File	Item Description	Quantity	Category Name	Revision	Units
2	100220	101684	Assembly - 100220 REV:A Eng Release	Assy, Cab.iam	Top Level	2	Assembly	A	Each
3	100220	101683	Assembly - 100220 REV:A Eng Release	Assy, Cab.iam	Top Level	2	Assembly	A	Each
4	100220	101682	Assembly - 100220 REV:A Eng Release	Assy, Cab.iam	Top Level	4	Assembly	A	Each
5	100220	101681	Assembly - 100220 REV:A Eng Release	Assy, Cab.iam	Top Level	2	Assembly	A	Each

Figure 2 - Data ready for PLM 360 Import

Both files have a Parent and a Number field that are the same. The difference is in what's written in the rest of the properties – Vault includes the Child properties whereas PLM 360 requires the Parent properties.

Luckily, Excel provides some easy tools to convert between these formats and this document provides instructions on how to achieve that.

Step 1 – Export data from Vault

When exporting the data, make sure to use the Parent-Child format and to include any properties that you wish to import into PLM 360. Open the file and save it as a native Excel file (XLSX)

Step 2 – Create a new Excel Worksheet

We will use the file created by Vault as the source of data for our import using Excel functions and therefore we don't want to modify the data within the table itself. So first, we'll create a new import table within the same spreadsheet

	A	B	C	D	E	F	G	H	I
1	Parent	Number	vault Item	CAD File	Item Description	Quantity	Category Name	Revision	Units
2									
3									

Figure 3 - Empty Excel Import Table

There are a few key properties that you must include (although you may name them however you wish):

- Parent – is the unique PN of the Parent object
- Number – is the unique PN of the Child object
- Quantity – is the quantity of the Child Object directly under this Parent object

For now, we will simply copy the headers from the original data to the new data and include all the original headers.

Step 3 – Prepare the data

We will be using the data exported from Vault as to lookup the values that we need. Therefore, we need it to be accessible and searchable. To do that, we will give it a name and sort it.

We will be using @VLOOKUP to search data in the table. Since @VLOOKUP only searches data in the first column of a range, and since we need to search data in the 'Number' column, make sure that 'Number' is the first column in the table.

	A	B	C	D	E	F
1	Number	Parent	Title (Item,CO)	Quantity	Category Name	Item Description
2	100220	<top>	Assy, Cab.iam	1	Assembly	My Product
3	101684	100220	027.05.01.048.ipt	2	Part	027.05.01.048
4	101683	100220	027.05.01.048-01.ipt	2	Part	027.05.01.048-01
5	101682	100220	027.34.11.066.ipt	4	Part	gasket
6	101681	100220	027.43.11.037.ipt	2	Part	pin

Figure 4 - Move 'Number' to be the first column

Next we need to give a name to this table. Select the table (without the headers) and give it a name. e.g. 'parts'

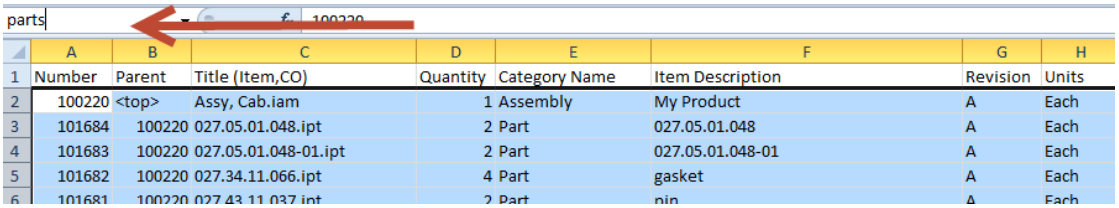


Figure 5 - Name the table

Lastly, we want to easily sort and filter the table. Click anywhere within the source data and choose ‘Filter.

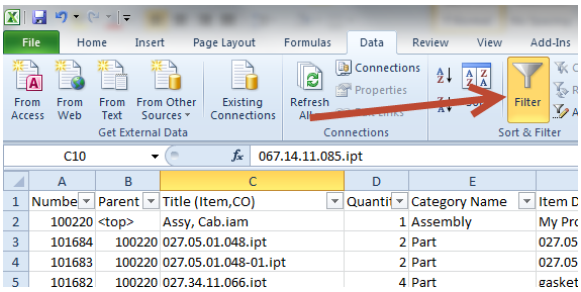


Figure 6 – Auto Filters

And sort the table by the Number field

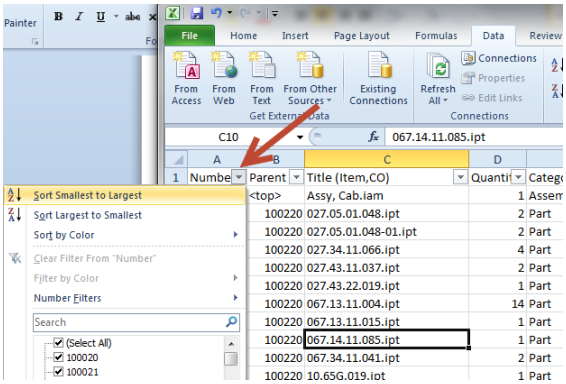


Figure 7 - Sort the Table

Step 4 – Fill in the Relationship section of the import file

We will be using the One-Step method to create the import file (meaning that we will define the Items and their BOM Relationships in a single file). Therefore, we will need an entry in the file for each Item and for each Relationship.

Looking closely at the import table we are creating, we will notice that each line has the following information:

- The Item defined in the line (‘Parent’)

- A relationship to one child (‘Number’)
- Specific properties relevant for the relationship (‘Quantity’)
- Specific properties of the Item (in this case ‘Category Name’, ‘Item Description’, ‘Revision’ and any other columns in the file)

Therefore, to convert from the Vault format to the PLM 360 format, we can keep the columns ‘Parent’, ‘Number’, and ‘Quantity’ and simply replace the rest of the data with information relevant to the Parent rather than the Child

We can start by copying the Parent, Number and Quantity columns from the original exported Excel worksheet into the new worksheet we created. This will become the Relationships section.

	A	B	C	D	E	F	G	H
1	Number	Parent	Title (Item,CO)	Quantity	Category Name	Item Descript	Revision	Units
2	100020	100216		2				
3	100021	100400		1				
4	100022	100214		1				
5	100023	100214		1				
6	100024	100214		1				

Figure 8 - Copy the Parent, Number and Quantity columns

Step 5 – Create the Item section

Now that we have completed the Relationship section, we essentially defined all the Items in the system that have Children in the Source data. What is left is to add a definition for all the Items that don’t have children. We will add those just below the table that we have created.

We can easily find out which items don’t have children by filtering the source data for Parts only.

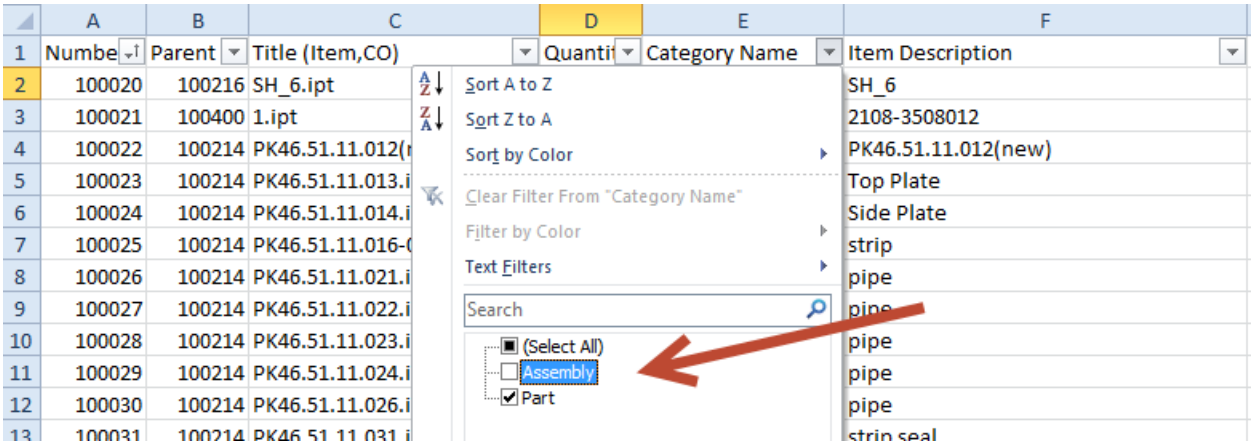
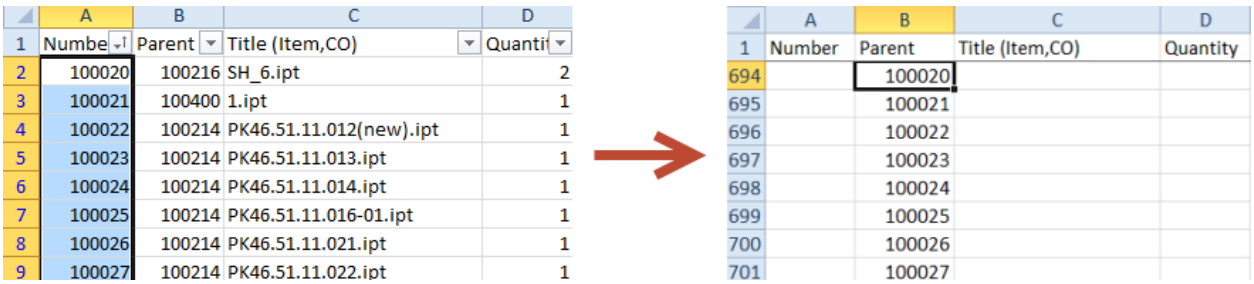


Figure 9 - Filter for Parts

Once we have identified all the children, we can copy the Number column for all the parts to Parent column in the new table. Remember, each line in the Import file represents a Parent. So in order to define an Item that doesn't have children, we define it as a Parent item but do not include any Children.



	A	B	C	D
1	Numbe	Parent	Title (Item,CO)	Quantit
2	100020	100216	SH_6.ipt	2
3	100021	100400	1.ipt	1
4	100022	100214	PK46.51.11.012(new).ipt	1
5	100023	100214	PK46.51.11.013.ipt	1
6	100024	100214	PK46.51.11.014.ipt	1
7	100025	100214	PK46.51.11.016-01.ipt	1
8	100026	100214	PK46.51.11.021.ipt	1
9	100027	100214	PK46.51.11.022.ipt	1

	A	B	C	D
1	Number	Parent	Title (Item,CO)	Quantity
694		100020		
695		100021		
696		100022		
697		100023		
698		100024		
699		100025		
700		100026		
701		100027		

Figure 10 - Copy Items to new table

Leave a blank line between the Relationship section and the Items section for now. It will help select the right datasets while we are working on this section and we can remove it later on.

Since these items belong to many different parents in the Source data, it is possible that we have duplicates here. We need to get rid of the duplicates by using Excel's built in 'Remove Duplicates' functionality.

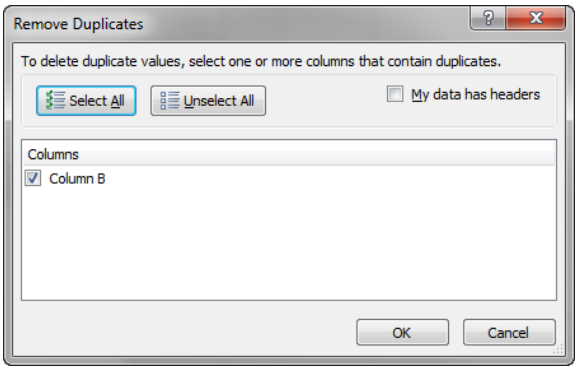
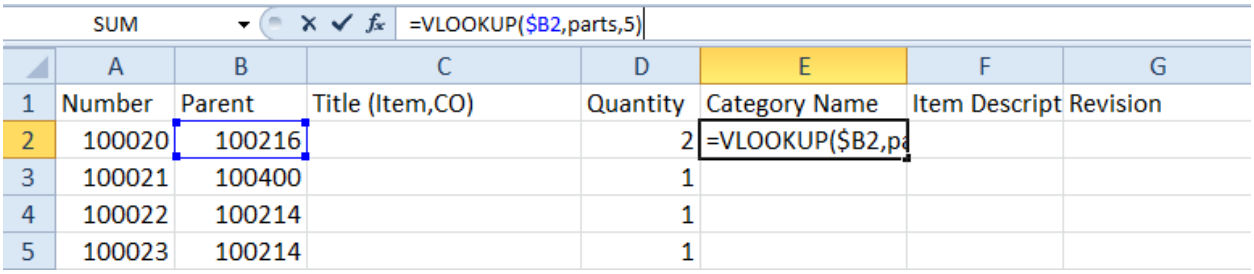


Figure 11 - Remove Duplicates

We can now delete the empty line left between the sections.

Step 6 – Fill in the blanks

The rest of the properties should be coming from the Parent object, not the Child. But we can simply lookup the values in the original table using the 'Parent' value as a key from the new table, and matching it to the 'Number' key in the parts table.

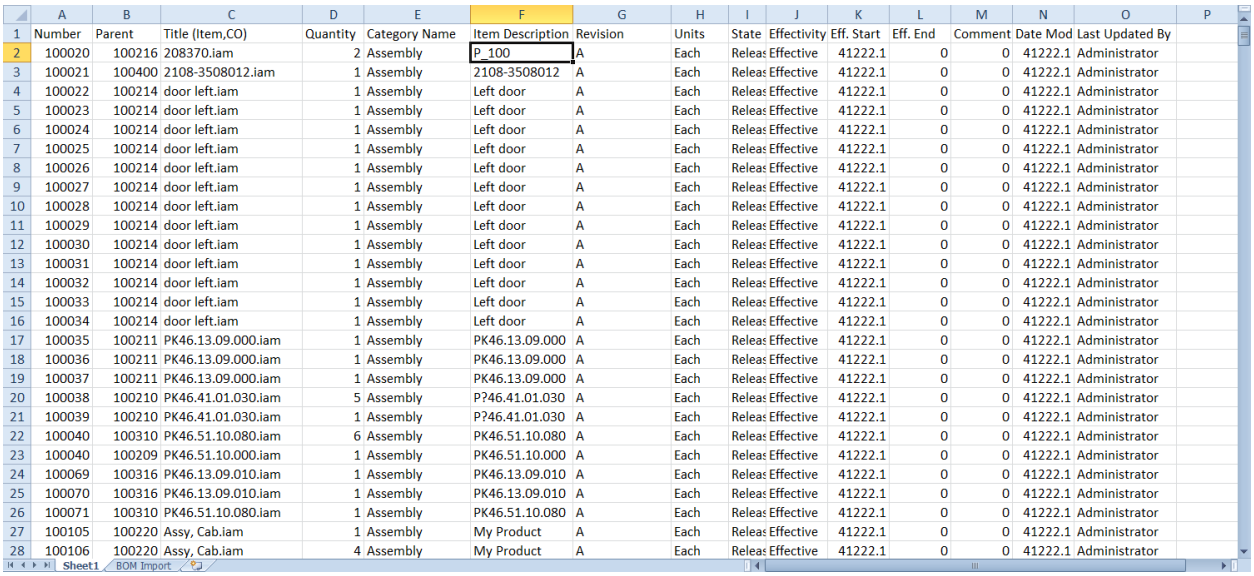


SUM =VLOOKUP($B2,parts,5)

	A	B	C	D	E	F	G
1	Number	Parent	Title (Item,CO)	Quantity	Category Name	Item Descript	Revision
2	100020	100216		2	=VLOOKUP($B2,pa		
3	100021	100400		1			
4	100022	100214		1			
5	100023	100214		1			

Figure 12 - Lookup Data

No matter which column data we are filling in, we always want to use the Parent as the key. The column number to pull the data from (5 in this case) will be the same as the column number in the new area. So we will use 6 for F, 7 for G, etc. Here is what the filled in table looks like:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Number	Parent	Title (Item,CO)	Quantity	Category Name	Item Description	Revision	Units	State	Effectivity	Eff. Start	Eff. End	Comment	Date Mod	Last Updated By	
2	100020	100216	208370.iam	2	Assembly	P_100	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
3	100021	100400	2108-3508012.iam	1	Assembly	2108-3508012	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
4	100022	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
5	100023	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
6	100024	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
7	100025	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
8	100026	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
9	100027	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
10	100028	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
11	100029	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
12	100030	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
13	100031	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
14	100032	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
15	100033	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
16	100034	100214	door left.iam	1	Assembly	Left door	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
17	100035	100211	PK46.13.09.000.iam	1	Assembly	PK46.13.09.000	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
18	100036	100211	PK46.13.09.000.iam	1	Assembly	PK46.13.09.000	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
19	100037	100211	PK46.13.09.000.iam	1	Assembly	PK46.13.09.000	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
20	100038	100210	PK46.41.01.030.iam	5	Assembly	P?46.41.01.030	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
21	100039	100210	PK46.41.01.030.iam	1	Assembly	P?46.41.01.030	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
22	100040	100310	PK46.51.10.080.iam	6	Assembly	PK46.51.10.080	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
23	100040	100209	PK46.51.10.000.iam	1	Assembly	PK46.51.10.000	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
24	100069	100316	PK46.13.09.010.iam	1	Assembly	PK46.13.09.010	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
25	100070	100316	PK46.13.09.010.iam	1	Assembly	PK46.13.09.010	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
26	100071	100310	PK46.51.10.080.iam	1	Assembly	PK46.51.10.080	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
27	100105	100220	Assy, Cab.iam	1	Assembly	My Product	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	
28	100106	100220	Assy, Cab.iam	4	Assembly	My Product	A	Each	Releas	Effective	41222.1	0	0	41222.1	Administrator	

Sheet1 BOM Import

Figure 13 - Completed Relationship section

Before we move on to defining the Item section, we need to make sure that changes to the Source data will not affect the results. To do that, we will simply use Excel's Copy-Paste as Values.

Copy the entire table, right click on the top-left cell, and choose 'Paste as Values'

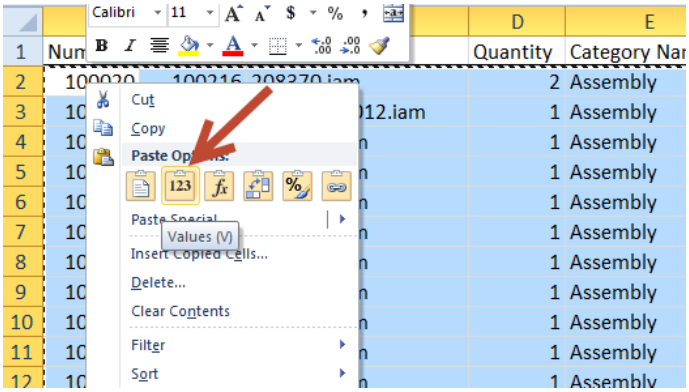


Figure 14 - Past as Values

Step 7 – Adding a link to the Vault import data

This step is only required if you are using the Vault Integration tool and would like each item in the BOM to have a Pointer ('Linking Pick List') to the respective Vault item. To do that, we will need to construct the 'Item Descriptor' (PLM 360 terminology for an identifier of an item in the system) as a field for the import.

The default Item Descriptor for Vault imported items looks like the following. You will need to change the formulas in the Excel file to match what the descriptor looks like in your tenant if it is different

[Type] – [PN] REV:[REV] [Lifecycle State]

All of this information is already available to us in the Excel file and from our knowledge of the Vault integration, so we can easily construct a formula to create another column in the excel file that will hold the descriptor

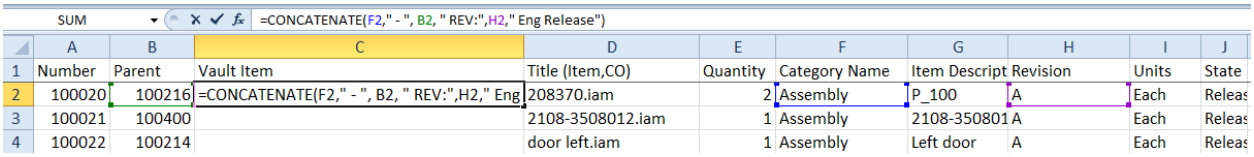
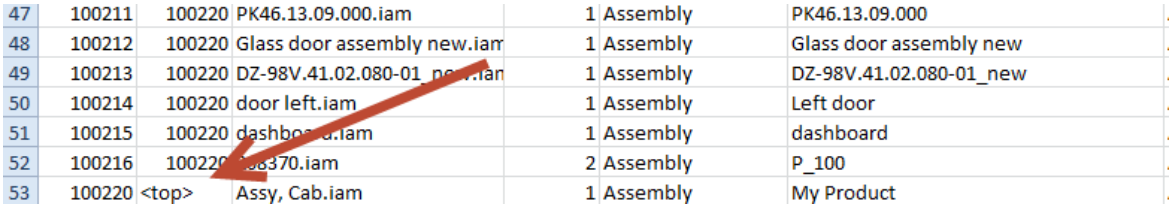


Figure 15 - Constructing a Descriptor

And we copy the formula to the rest of the table.

Step 8 – Cleanup

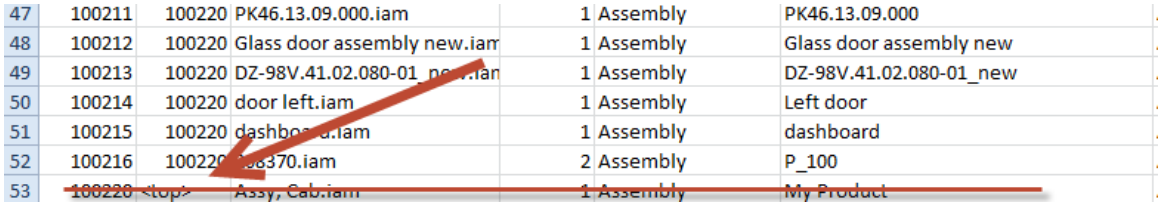
Find the line in the new table that represents the Top assembly



47	100211	100220	PK46.13.09.000.iam	1	Assembly	PK46.13.09.000
48	100212	100220	Glass door assembly new.iam	1	Assembly	Glass door assembly new
49	100213	100220	DZ-98V.41.02.080-01_new.iam	1	Assembly	DZ-98V.41.02.080-01_new
50	100214	100220	door left.iam	1	Assembly	Left door
51	100215	100220	dashboard.iam	1	Assembly	dashboard
52	100216	100220	98370.iam	2	Assembly	P_100
53	100220	<top>	Assy, Cab.iam	1	Assembly	My Product

Figure 16 - Top Assembly

This is required when exporting data from Vault since each line represent a Child object. But for PLM 360 every line represents a Parent and therefore this line is redundant. The details of item 100220 will be included in lines 51,52,54 and all the other Children



47	100211	100220	PK46.13.09.000.iam	1	Assembly	PK46.13.09.000
48	100212	100220	Glass door assembly new.iam	1	Assembly	Glass door assembly new
49	100213	100220	DZ-98V.41.02.080-01_new.iam	1	Assembly	DZ-98V.41.02.080-01_new
50	100214	100220	door left.iam	1	Assembly	Left door
51	100215	100220	dashboard.iam	1	Assembly	dashboard
52	100216	100220	98370.iam	2	Assembly	P_100
53	100220	<top>	Assy, Cab.iam	1	Assembly	My Product

Figure 17 - Remove the Top product from the table

Step 9 – Prepare a Workspace for the Import

We need to make sure we have Properties available to host all the data we want to extract from the Excel file. This step is outside the scope of this document.

Step 10 – Import

Finally, we get to import the data to PLM. First we need to create a new import project. Since we are using the one-step import, select the Parent Child BOM import type

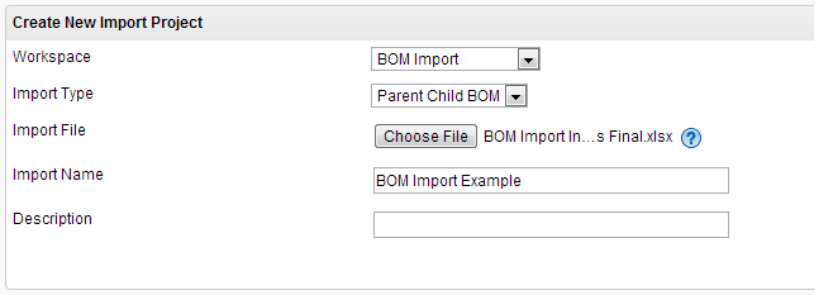


Figure 18 - Import Project

Critically important to this import is to identify the Parent PN and the Child PN. Make sure sure to check ‘Match as BOM Parent’ for the Parent column, and ‘Match as BOM Child’ for the child PN column. Other than that, choose to match any additional properties required.

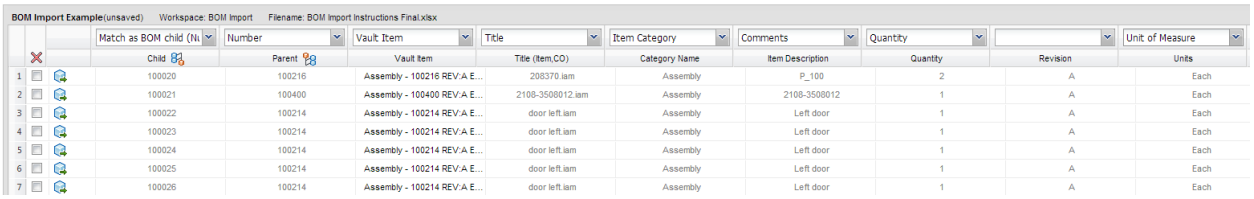


Figure 19 – Match properties

After matching the properties, we need to define the import settings. The options are pretty straight forward and self-explanatory. I recommend using the strict enforcement of validations and constraints first to help identify any issues.

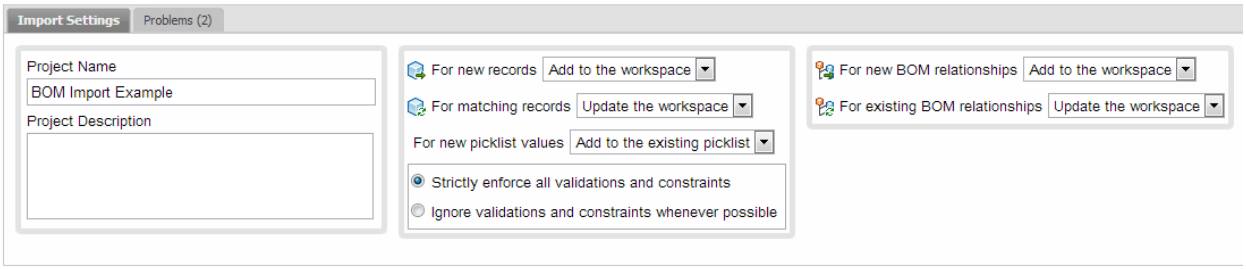


Figure 20 - Import Settings

Save, and make sure no problems exist. Solve any issues identified.

Then, click Run.