Pack Product into Packages

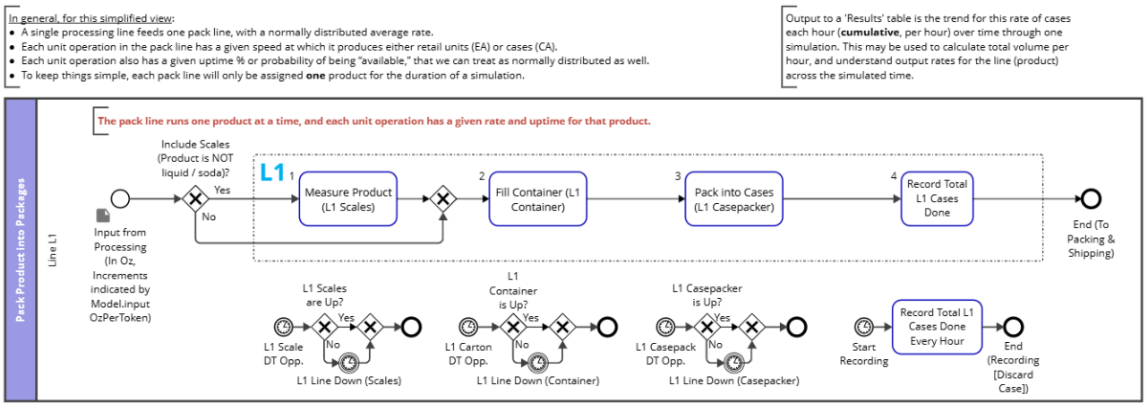


Table of Contents

Overview

We will look at the business situation driving the need for the simulation model, and the high-level design of the model itself.

Business Situation

Your project is to model your product packaging operation (it's left up to you to imagine what the product might be; soda, gumballs, widgets, etc. It's initially configured for a soda bottling / packaging line).

You're tasked with understanding how different levels of product input will impact throughput / output on the line, including how any unplanned downtime might affect the operation of the line.

Model Design

The model has an interesting structure. There are parallel processes in this system, where one impacts the other: packaging of products, and downtime of the line:

- Packaging Product:
  - The various components of the line (e.g., the Container Machine(s)) are used to put the product into retail containers
  - The containers are packaged into cases.
  - Again, this could be bottles of soda that are 12, 20, or 68 ounces and sold in 6, 12, etc. packs. Or this could be containers of gumballs that are sold in packs of 6, 12, etc. with varying weights.
  - The line is configurable for ounces per container, containers per case, and speeds of handling each. It's configurable for liquid product (skipping weighing product on Scales) before placing in containers.
- Downtime and Maintaining the Line:
  - When the line goes down unexpectedly, we must get the line running again ASAP, by fixing any issues due to the downtime.
  - The model is initially configured with one Mean Time to Recovery (MTTR) for all steps, and the input spreadsheet (loaded into a table variable) gives probabilities of a failure occurring.

The Process Model

In this section, we'll walk through the model and review the modeling techniques employed.

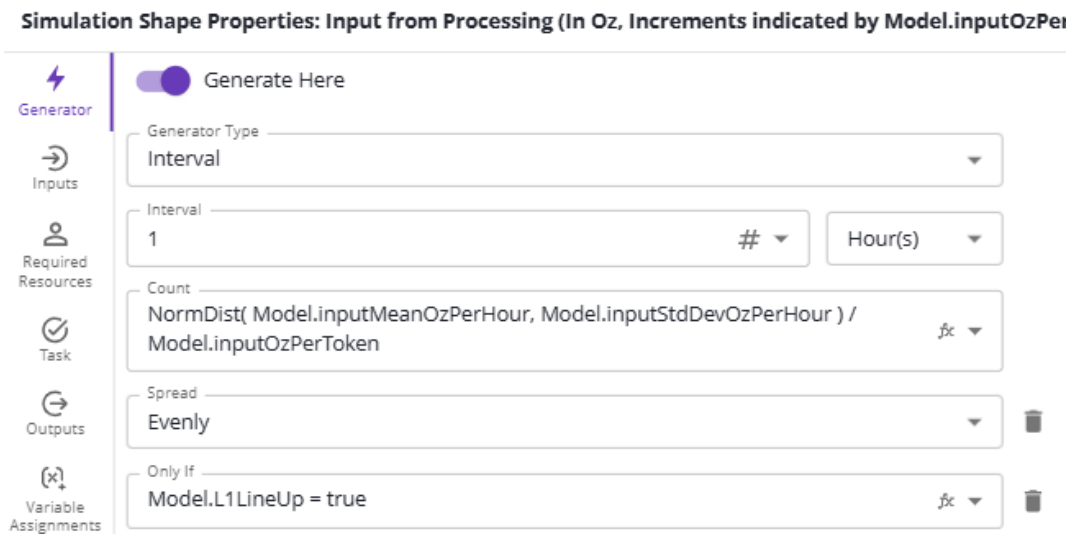
- Controlling Incoming Product Rate, by tying a *Generator* that introduces 'Cases' (or tokens) to the equipment needed to package the product. We'll also review *Variables* that help control the *Generator*, and other behavior in the model.
- Controlling Product Flow, *Task* Durations, and *Resource* Use, to ensure that each step operates on the correct number of tokens (through *Input* ('Gate') conditions), takes the expected amount of time, using any Resources that we have defined.
- Managing Unplanned Equipment Maintenance, by controlling downtime of the line in the processes parallel to the packaging products process. This is done through *Generators, Task Durations, Outputs* conditions, and through use of *Resources*.
- Recording Total Cases Produced, Every Hour, by writing the count to a Table, which we can then look at in the simulation results after a complete simulation has been run.

Controlling Incoming Product Rate

Products are introduced to the process at the start Event ("Input from Processing…") via a *Generator*. To review how this is implemented:

1. Click the Start shape.
2. Click the Simulation Properties button 
3. Click the Generator tab / page of the Simulation Shape Properties dialog box (the dialog box is non-modal; you may click on various objects and the information will automatically be saved or updated as appropriate):

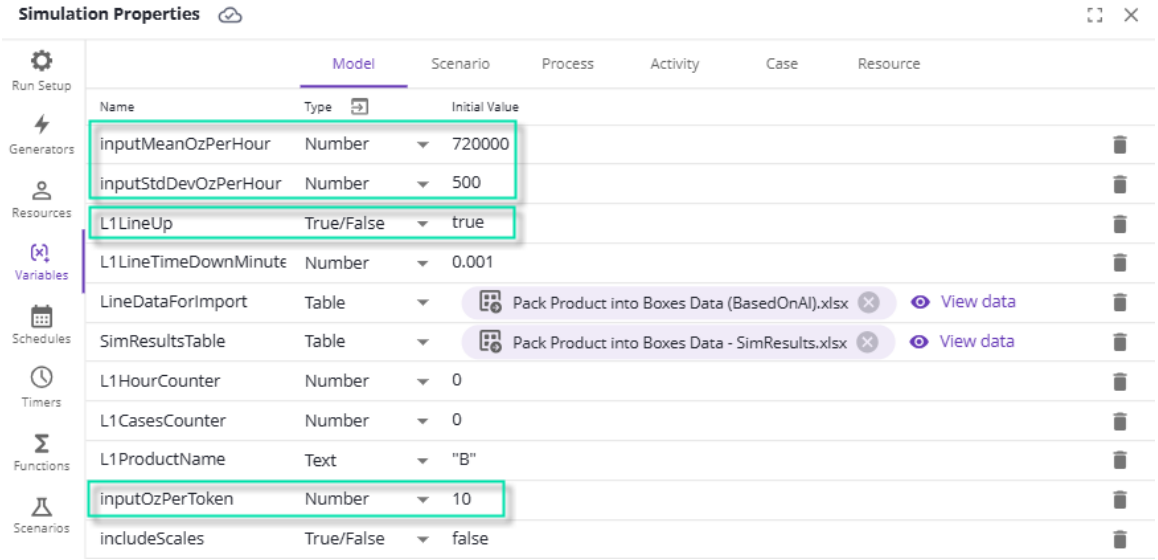
4. This displays the Generator information.



5. We have a Generator that evenly spreads the tokens (or 'Cases'), each representing the number of ounces of product indicated by a Model variable ("inputOzPerToken"), throughout each hour. The number of tokens is determined by a 'Normal' distribution (or 'bell curve'), with the Mean and Standard Deviation as specified in Model scope variables, and modified by variable that represents the number of ounces per token. Let's look at those Model variables next.

Model Variables Controlling the Generator

6. Click somewhere 'empty' in your diagramming canvas, where you won't select any object. When the Simulation Properties dialog box is already open, this displays the 'setup' or 'environment' for the entire simulation model.
7. Click on the 'Variables' tab/page of the properties dialog box



8. We can see that the "inputMeanOzPerHour" and "inputStdDevOzPerHour" variables have numeric values that will be used in our normal distribution. For example, a

mean of 720000, with a standard deviation of 500, tokens (ounces of product) per hour.

9. We can also see that the “inputOzPerToken” is set. For example, a value of 10 indicates that each token (case) represents 10 ounces of product.
10. Returning to the Generator’s ‘Count’ field, we can see and interpret the expression:
    a. NormDist(Model.inputMeanOzPerHour, Model.inputStdDevOzPerHour) / Model.inputOzPerToken
    b. Substituting values from the example above: NormDist(720000,500) / 10
    c. If we assume that the mean value of 720000 is picked, then we have 720000 / 10, or 72000 tokens per hour being generated.
11. We also see the “L1LineUp” Boolean (True/False) variable. This is used to indicate when there is unplanned downtime. If the line goes down, this variable is assigned the value ‘false’, and that suppresses the generation of tokens until the expression becomes true again. We will later review how this variable is set and used.
12. Note that there are other variables defined and set with initial values here. Of key interest are the ‘Table’ variables, and there are other variables that control the model in different places, such as the ‘L1ProductName’ variable. We will review the use of these variables later as well.

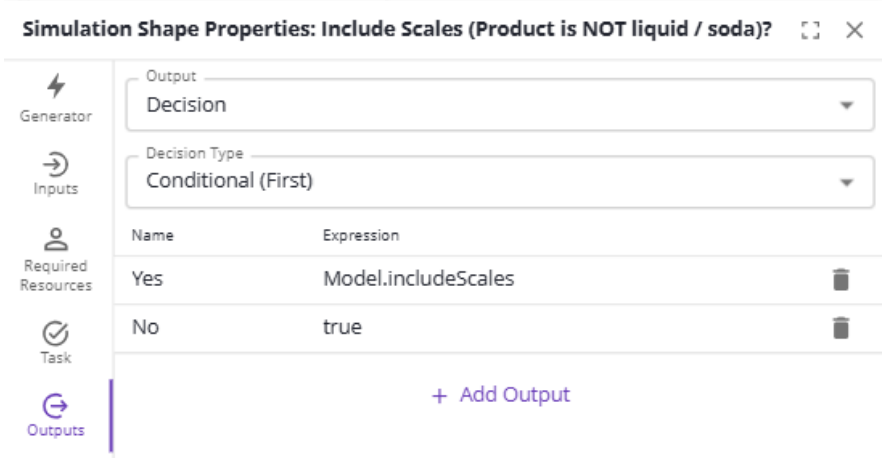
Controlling Product Flow, Task Durations, and Resource Use

We’ll next look at how we control the flow of Product through the line (*Outputs* on Gateways), *Task* durations, and use *Resources* in the model.

Controlling Product Flow with Gateways

13. Click on the Gateway labeled “Include Scales (Product is NOT liquid / soda)?”, to display the Simulation Properties dialog box for that step.
14. Click on the ‘Outputs’ tab/page of the properties dialog box, if it’s not already chosen

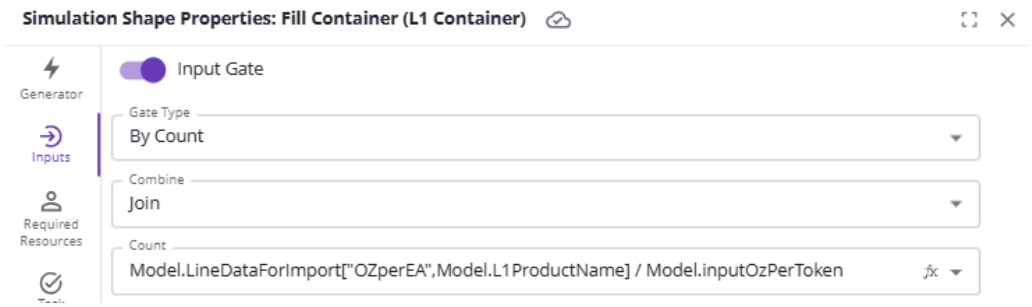
15. We can see that the ‘Model’ scope variable named ‘includeScales’ controls the decision. If it’s ‘true,’ then the ‘Yes’ path is taken. Otherwise, the ‘No’ path is taken, as its condition is always true (though won’t be evaluated unless the Yes condition is false).



Controlling Collection of Tokens through Input ('Gate') Conditions

16. Click the 'Fill Container (L1 Container)' Activity, and ensure the 'Inputs' page is selected

17. You can see that we are 'Joining' tokens together, 'By Count,' and the Count is controlled by an expression that uses Model scope variables:



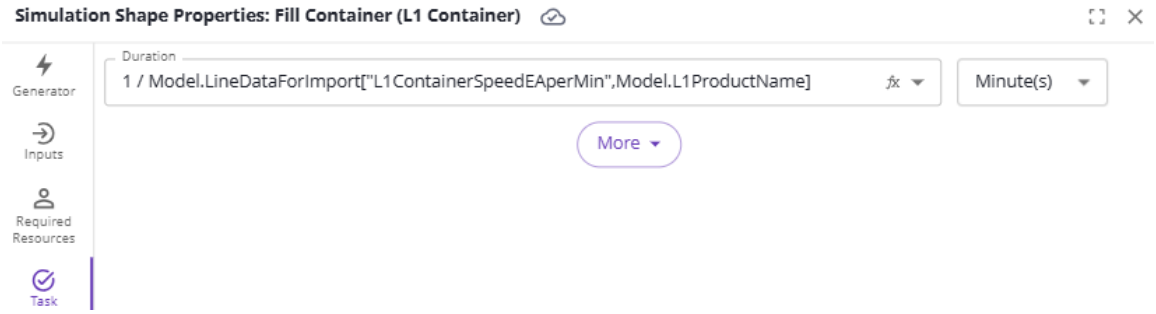
Note: We know that this looks complex! Stay with us; we'll explain!

- Let's interpret this expression on the Inputs page. What we are doing here is indicating that each container for the product is composed of a certain number of ounces of product; each token at this point represents the number of ounces of product indicated by a variable. We are looking up the number of ounces (OZ) per each container (EA) in a Table variable, and then dividing by the number of ounces represented by each token:
    - So, for example:
        - Model.LineDataForImport["OZperEA",Model.L1ProductName] / Model.inputOzPerToken
        - "20 / 10" =
        - 2 tokens per container.

Let's look at another similar expression, in the duration of this task, and then we'll look at how the 'Table' variables work.

Controlling Task Durations

18. Keep the 'Fill Container (L1 Container)' activity selected, and ensure the 'Task' page is selected in the Simulation Shape Properties dialog box.



Note: We know that this looks complex! Stay with us; we'll explain!

- Let's interpret this expression on the Task page. What we are doing here is indicating that each container of product (each token at this point represents a container of product, as we've joined enough tokens to represent an entire container's worth of product) takes a fraction of a minute, based on a formula:
    - 1 / Model.LineDataForImport["L1ContainerSpeedEAperMin", Model.L1ProductName]
    - The ratio of 1 divided by the following: 'Container Speed per Minute' for the product we're operating on.
    - So, for example:
        - "1 / 600" =
        - 0.0016 minutes (or about 0.1 seconds) per container.

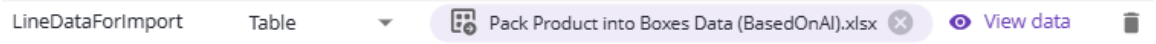
Where did we get the meaning of the expression, and these numbers, from? We need to look at variables again, this time looking into one of our 'Table' variables.

Reviewing Table Variables

Table variables are like having a spreadsheet of values available to you by simply referencing one variable name. You indicate a single value (like a 'cell' in a spreadsheet) by indicating which specific row and then column of data you want.

19. Click somewhere 'empty' in your diagramming canvas, where you won't select any object, to display the Simulation Properties dialog box.
20. Click on the 'Variables' tab/page of the properties dialog box

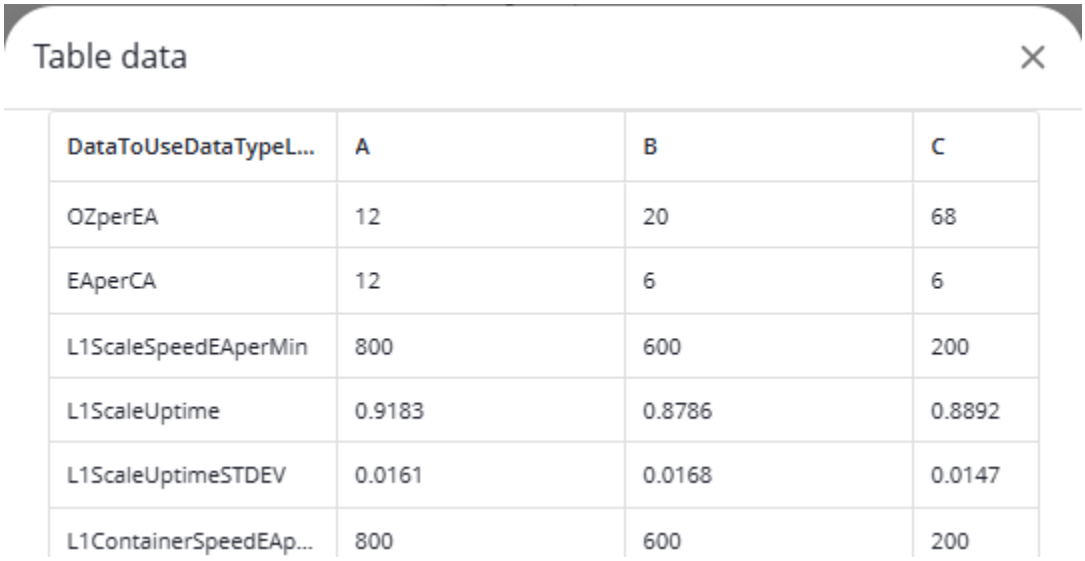
21. Find the "LineDataForImport" Table variable.



22. Click the 'View data' button.

View data

A window displays, showing the table data we have loaded into this variable.



DataToUseDataTypeL...	A	B	C
OZperEA	12	20	68
EAperCA	12	6	6
L1ScaleSpeedEAperMin	800	600	200
L1ScaleUptime	0.9183	0.8786	0.8892
L1ScaleUptimeSTDEV	0.0161	0.0168	0.0147
L1ContainerSpeedEAp...	800	600	200

As you can see, we have 3 columns of data, one column for each product that runs on the line. We can control the behavior of the model simply by changing data values in the table, re-uploading the table from our source file (e.g., a Microsoft® Excel® file), and having the model refer to these values generically through the Table. So, let's look again at the Duration expression we looked at earlier:

1 /

Model.LineDataForImport["L1ContainerSpeedEAperMin",Model.L1ProductName]

So, if we look at the Model scope variable "LineDataForImport", and we want to look into row "L1ScaleSpeedEAperMin" and column indicated by the 'regular' (non-Table) Model variable "L1ProductName". This allows us to change the "L1ProductName" variable value to choose an entirely different column of data, and to get different behavior from the model based on that.

If we assume that "L1ProductName" has a value of "B" (column B), then we have the following:

Model.LineDataForImport["L1ScaleSpeedEAperMin", "B"]

We can see that "L1ScaleSpeedEAperMin" for column "B" is 600. So we have 1/600, or 0.1, Minutes for our Task Duration.

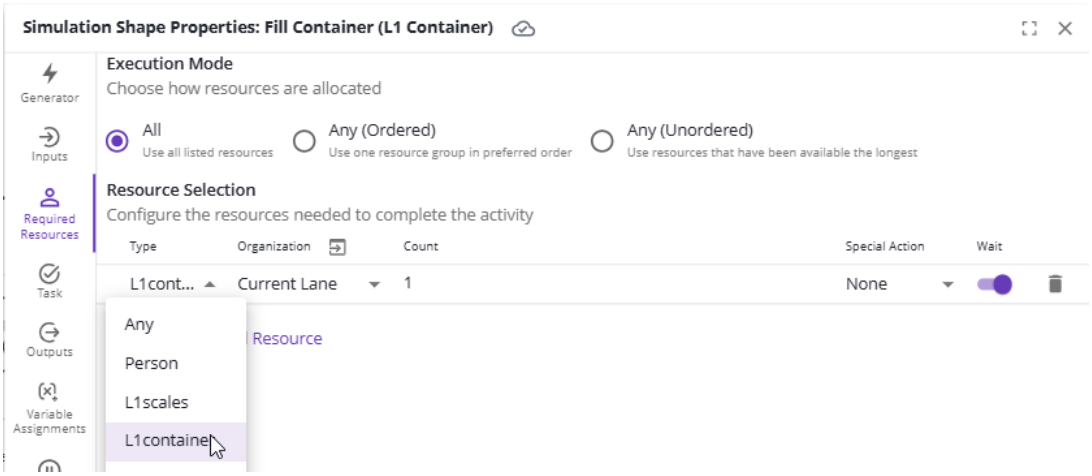
23. Click the 'X' to close the 'Table data' window.

Resources and Resource Use

We need resources to do work within the process.

24. Click the 'Fill Container (L1 Container)' Activity, and ensure the 'Required Resources' page is selected

25. You can see that we have specified we need 1 of the “L1container” resources available to the current Lane in the diagram:

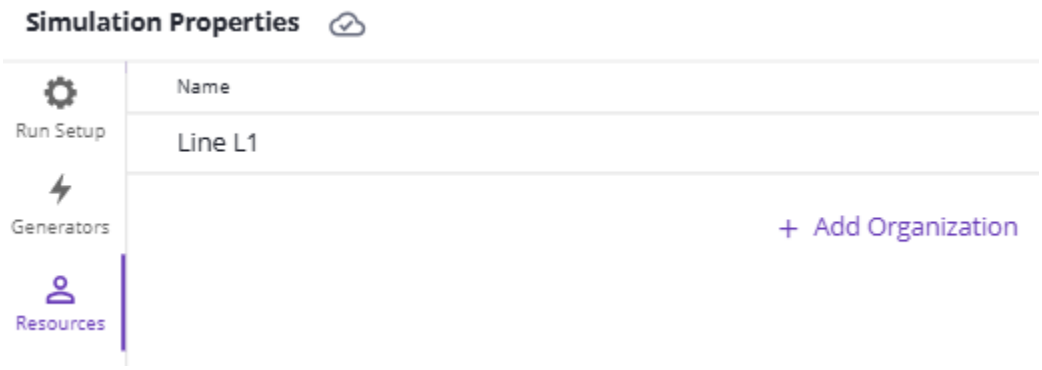


We are indicating that each token-- that represents the amount of product (e.g., 20 ounces of liquid to fill a single bottle of soda, when we are running product/column ‘B’ from our Table)-- takes a single ‘container’ machine (or slot in the container machine, that can use one slot at a time) for Line 1.

Where did we define what resources are available for this Lane? That’s in the Simulation Properties, which we will look at next.

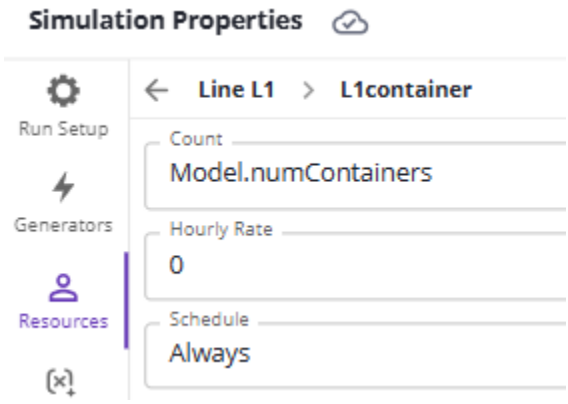
26. Click somewhere ‘empty’ in your diagramming canvas, where you won’t select any object, to display the Simulation Properties dialog box.
27. Click on the ‘Resources’ tab/page of the properties dialog box

28. We see a list of ‘Organizations,’ listed by Lane in the diagram. There is only 1 Lane: “LineL1”.



29. Click on the “Line L1” Lane/Organization; the resources (resource types) defined for this Organization (Lane) are shown.

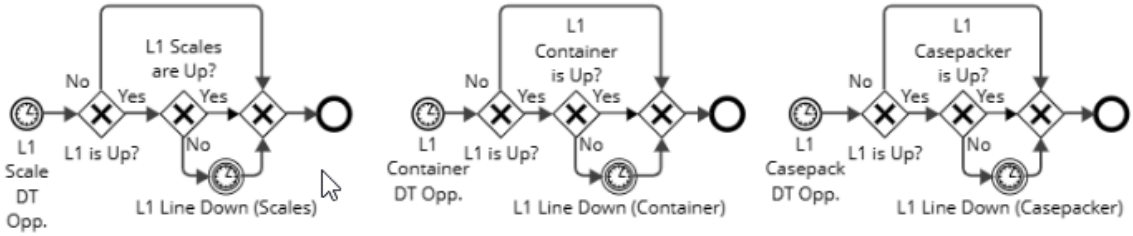
30. Click on the “L1container” resource, to see its definition:



We can see that we have a resource that is ‘Always’ available (on a schedule of ‘Always’, which is 24 hours per day, 7 days per week), there is no ‘Hourly Rate’ cost associated with the resource, and we have the number of resources indicated by the Model variable “numContainers” This allows us to control the number of resources we have from our Variables list; which makes it a ‘control panel’ of sorts, to control many aspects of the simulations that we will run.

Managing Unplanned Equipment Maintenance

The maintenance, or downtime, for the production equipment-- Scales, Container machine(s), and/or Case Packer(s)-- is handled through the use of independently-fed start points for unscheduled downtime. These start points are the “DT Opp.” Timer Events shown underneath the production Activities:



Generating Downtime Requests

The start activities for each downtime Even are connected to independent Generators. The unscheduled maintenance occurs on a probability basis (as opposed to a 'Mean Time Between Failures" or MTBF basis). To review how the unscheduled downtime is controlled:

1. Click on the Timer Event labeled "L1 Scale DT Opp.", and then the Simulation Properties button 
2. Click the Generator tab / page of the Simulation Shape Properties dialog box 

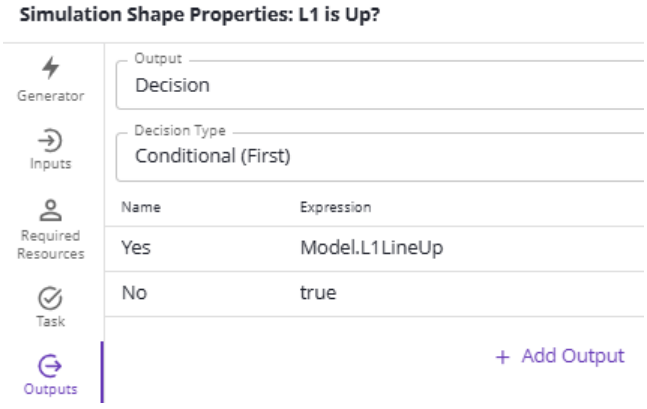
3. This displays the Generator information.

Note that we randomly distribute 12 possible downtime events throughout each hour. This number of possible downtime events, and the spread, could of course be adjusted, and/or controlled by variables.

In addition, notice that-- for the Scales specifically-- we have used the 'Only If' condition to only generate downtime requests if we are including the use of the Scales in the model.

Controlling Downtime Requests

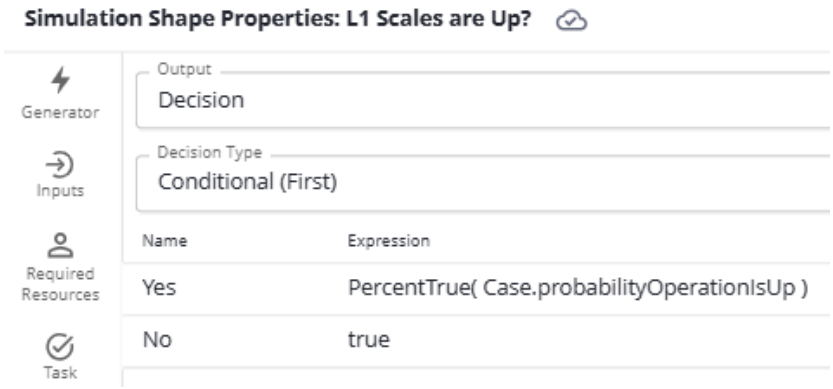
4. Now click on the "L1 is Up?" Gateway (decision) and choose the Outputs page.



Note that if the Line is not currently up, we will ignore this downtime request, as this element of the line is not likely to go down (or go down again) while some portion of the line is down. This may introduce slightly less downtime than expected, so this portion of the model could be removed if that's a concern.

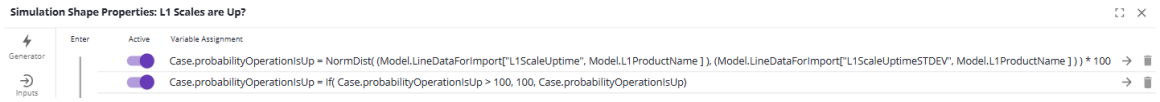
Let's assume that the line is up, and look at the next shape along the 'Yes' path.

5. Click on the "L1 Scales are Up?" Gateway, and click on the Outputs page, where you can see we control the behavior of the decision with use of the 'PercentTrue' function used with a variable:



6. Choose the Variable Assignments page.

7. This displays variable assignments. For this step of the process, we can see assignments when the token 'Enters' the step (before it processes Inputs, etc.).



8. Let's review the assignment to the Case-scope variable 'probabilityOperationIsUp':
    a. First, we assign a 'Normal Distribution' probability based on the 'Mean' and 'Standard Deviation' as indicated in our table:
        i. The formula given is

        NormDist(

        (Model.LineDataForImport["L1ScaleUptime", Model.L1ProductName]), (Model.LineDataForImport["L1ScaleUptimeSTDEV", Model.L1ProductName])

) * 100

        ii. If we look up the 'Uptime' and 'UptimeSTDEV' values in our table, we see 0.8786, 0.0168 respectively. So, the formula is then:

        NormDist(0.8786,0.0168) * 100

        iii. If we assume that the mean value of 0.8786 was returned from the NormDist function, then we have:

        0.8786 * 100 = 87.86

b. Second, we test to see if the value returned was over 100 (we cannot have a probability of something being true over 100%, so we need to test for this), and if it is we cap the value at 100:

   i. The formula given is:

      If(

      Case.probabilityOperationIsUp > 100,

      100,

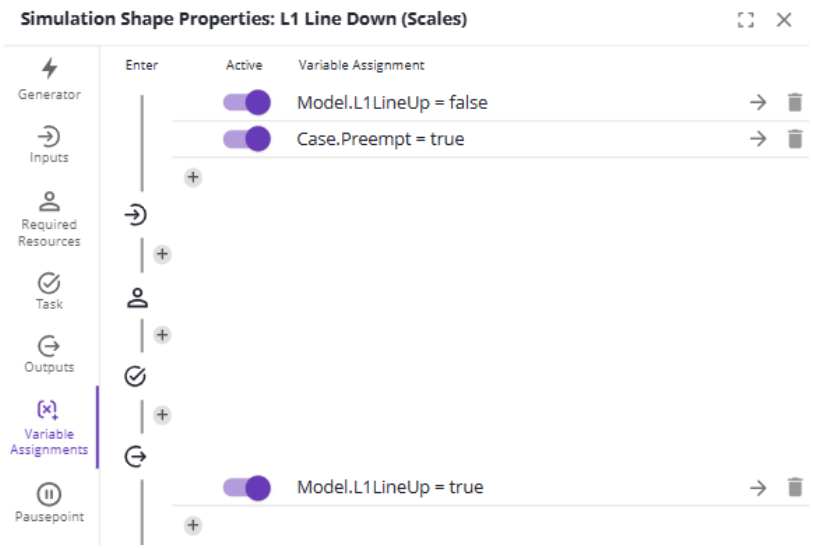
      Case.probabilityOperationIsUp

)

   ii. This becomes, for example, the following:

      If(86.87 > 100, 100, 86.87) = 86.87

9. Click on the Timer shape labeled “L1 Line Down (Scales)” and click on the ‘Variable Assignments’ page.

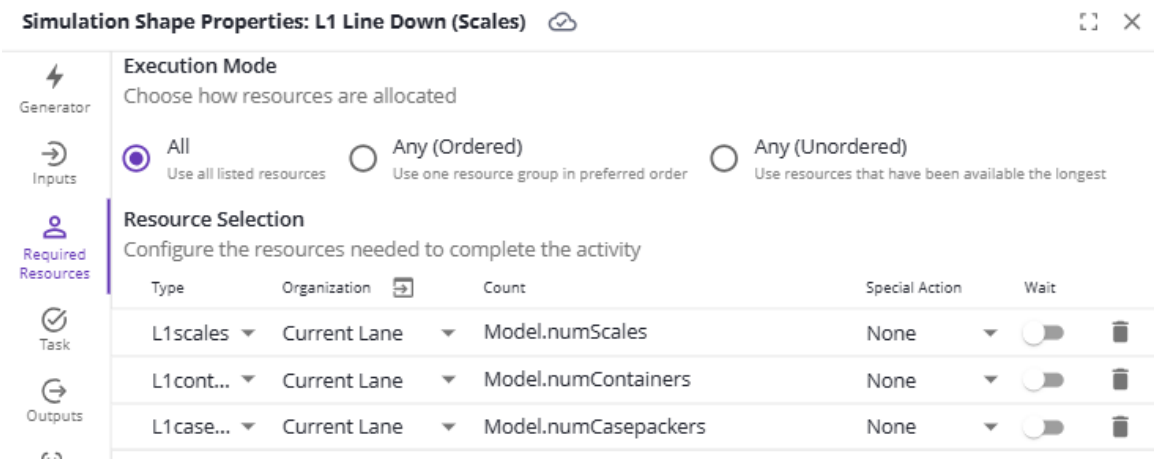


   a. We set the L1LineUp to false, to indicate that the line is down.

   b. We set ‘Case’ scope variable on the token named Preempt to true. Preempt is a special built-in variable that indicates this token should, if possible, preemptively take resources away from other (non-preemptive) tokens.

   c. When the step is finished (after we have acquired our required resources, and used them for the specified Task Duration, both of which are covered next), we will mark the line ‘up’ (L1LineUp to true) again.

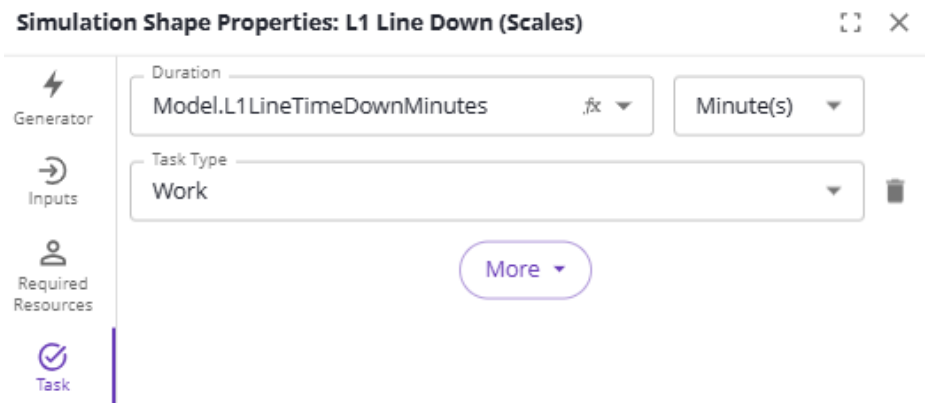
10. Click on the ‘Required Resources’ page.



You can see that we require resources for Scales, Container machine(s), and Case Packer(s) by setting the count for *each* downtime token to the number of resources we created, by using the Model-scope variables used in the Resources definition we reviewed earlier. This forces that if one part of the line goes down, the entire line goes down, and all components of the line go down. You could, of course, only take parts of the line down by modifying this behavior.

Remember that the token is 'Preemptive' at this point, so it will 'steal' resources from other, non-Preemptive tokens for the resources it requires.

11. Click on the Task page. We can see we're controlling the duration of the L1 Line Down by the Model variable "L1LineTimeDownMinutes". So, the line will be down for the number of minutes specified in the variable.



Note that all downtime steps use the same value and would not have to; you could control how long each element of the line is down by having separate variables for each piece of equipment (each component of the line) that is used.

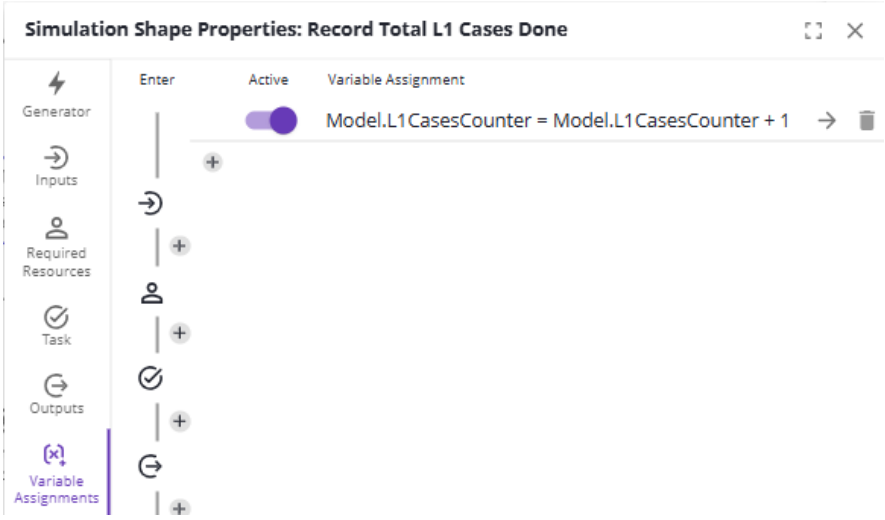
12. Click the 'Done' button, to close the Simulation Properties dialog box.

Recording Total Cases Produced, Every Hour

The simulation results will, by default, tell us how many total tokens (or 'Cases') completed the model by the end of the simulation run. While we could use logging of tokens ('Cases'), we want a simpler summary in Table format. We can use variables to track how many tokens (in our situation, cases of product that were packed by the 'Case Packer' machine) complete the simulation each hour and write the variable values to a Table that we can look at later.

Recording Total Cases Packed

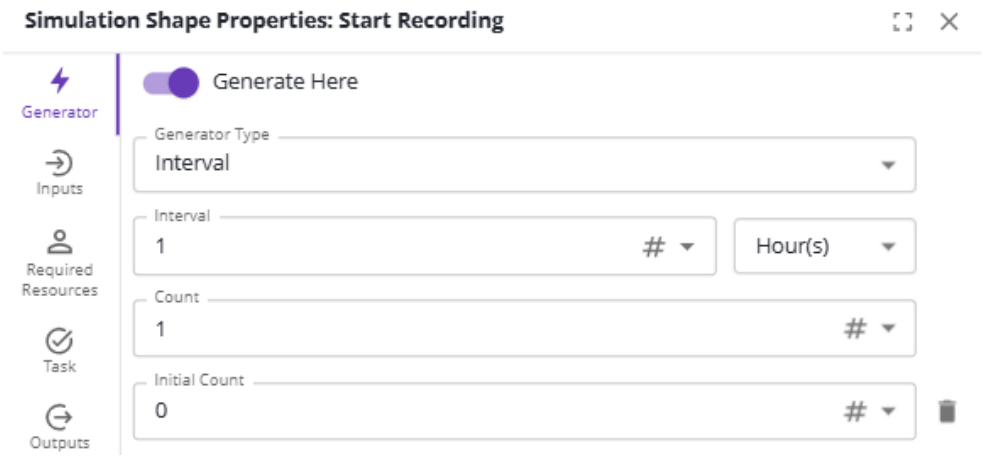
1. Click on the shape labeled “Record Total L1 Cases Done” and click on the ‘Variable Assignments’ page.



We can see that, every time a token (representing a ‘case’ packed by the Case Packer) arrives, we increase the total count of tokens completed by 1.

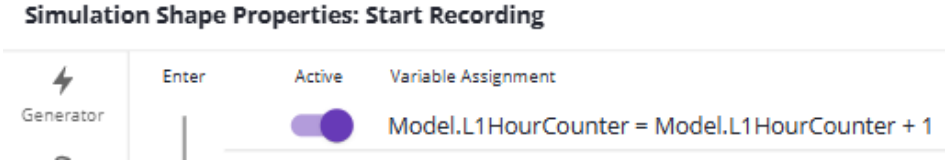
Now, we’ll look at how we record this amount every hour, to a Table.

2. Click on the Timer Event labeled “Start Recording” and click on the Generator page.

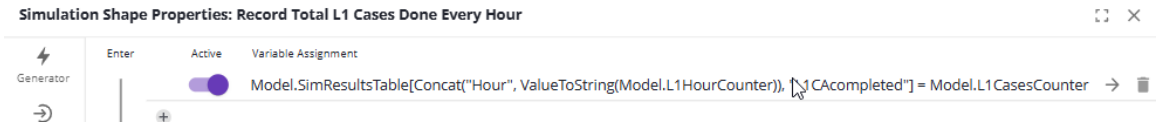


We generate a token every hour, except not initially when simulation starts, so that we can record what is going on at that time.

3. Click on the ‘Variable Assignments’ page, and you can see we increment a Model variable that counts each hour.



4. Click on the “Record Total L1 Cases Done Every Hour” step and choose the ‘Variable Assignments’ page.



   a. Let’s look at the expression shown:
   Model.SimResultsTable[Concat("Hour", ValueToString(Model.L1HourCounter)), "L1CAcompleted"] = Model.L1CasesCounter

   b. We want to write the value of the ‘L1CasesCounter’ variable (how many cases were packed by the Case Packer) to the Model-scope Table type variable ‘SimResultsTable’. We’ll write that into the row and column specified by the following:

      i. Row: A concatenation of the word “Hour” with the string equivalent of what hour it is, as indicated by the “L1HourCounter” variable set on the prior step. For example, “Hour5”

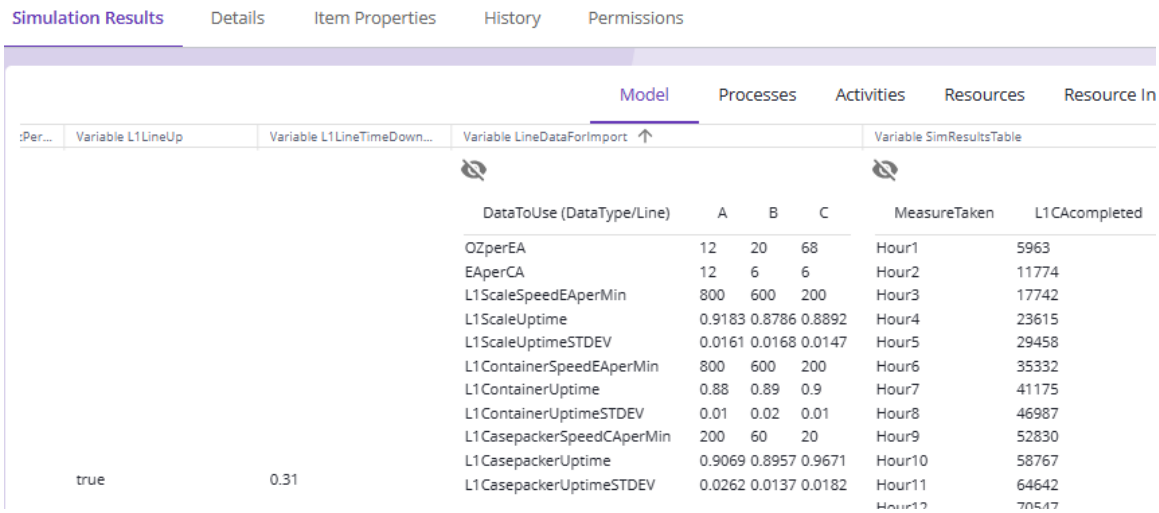
      ii. Column: The column specified by the string “L1CAcompleted” (the only other column besides the row labels column).

We’ll look at how to see the values of this Table variable when we look at Simulation Results.

Simulation Results (Report)

When you've run a complete simulation, the Results window (the simulation 'report') will display.

One of the key results we're interested in is our Table of the number of tokens completed (cases packed) per hour. We can see the final values of our Table variables in the Model tab of the Simulation Results.



There are other key statistics for the line you might look at. For example, which Activities or Resources have waiting tokens, statistics such as wait queue sizes or non-zero waiting times.

Potential Experiments & Model Enhancements

Potential Experiments

There are experiments one might run, or enhancements to the model that can be made. Examples are:

1. What if there wasn't any unplanned downtime in any given 24-hour period, e.g. it only happened infrequently? You can test this by changing the 'L1LineTimeDownMinutes' Model variable to zero (0).
2. What if we replaced the container filling machine with a newer version that could do 800 (vs. 600) containers per minute? Or some other rate? Does it increase production significantly? You can do this by modifying the number in the input file used by the 'LineDataForImport' Model variable and re-uploading that file.
3. What if we add another container machine to fill containers in parallel? You can test this by changing the 'numContainers' Model variable from 1 to 2. Does that increase production significantly? How does it compare with a faster container filling machine?

Potential Model Enhancements

The model could be enhanced in various ways, to allow for other experimentation, to make the model more realistic, etc. For example:

1. Adjust downtime to be more realistic for a soda bottling line. For example:
    a. Typical MTBF for modern rotary bottle fillers (20 oz lines): 250 to 1,000 hours between unplanned failures (Well-maintained, high-end equipment: 600–1,000+ hours. Older or heavily used lines: closer to 250–500 hours).
    b. Typical MTTR would be 30 to 120 minutes (0.5 to 2 hours).
2. What if a 'Kanban' type system of single-piece flow is used? For example, the Container machine would shut down if the case packer didn't have capacity, or the incoming product (or scales) shut down if the container machine didn't have capacity? What size of product queue (e.g., unfilled soda bottle 'containers') would be needed? Instead of implementing a queue of unfilled containers, would adding another container filling machine make sense?