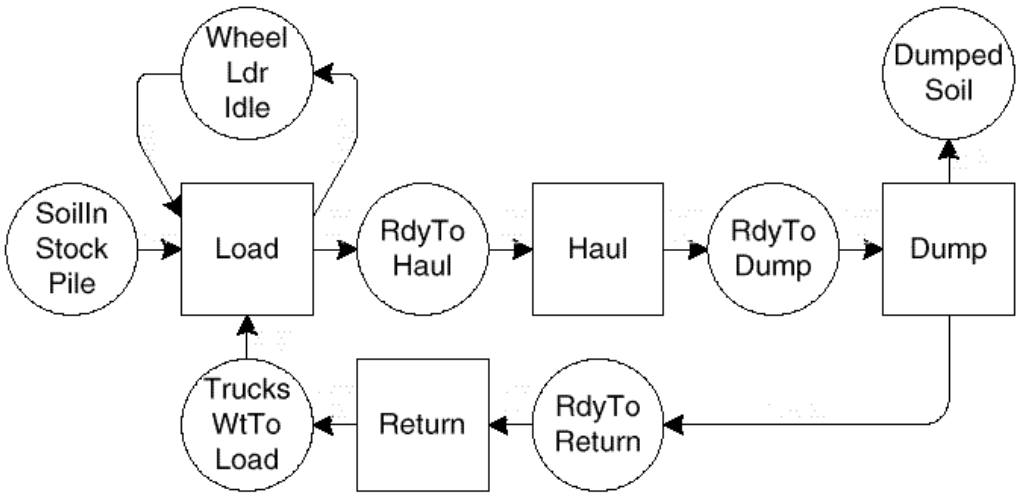
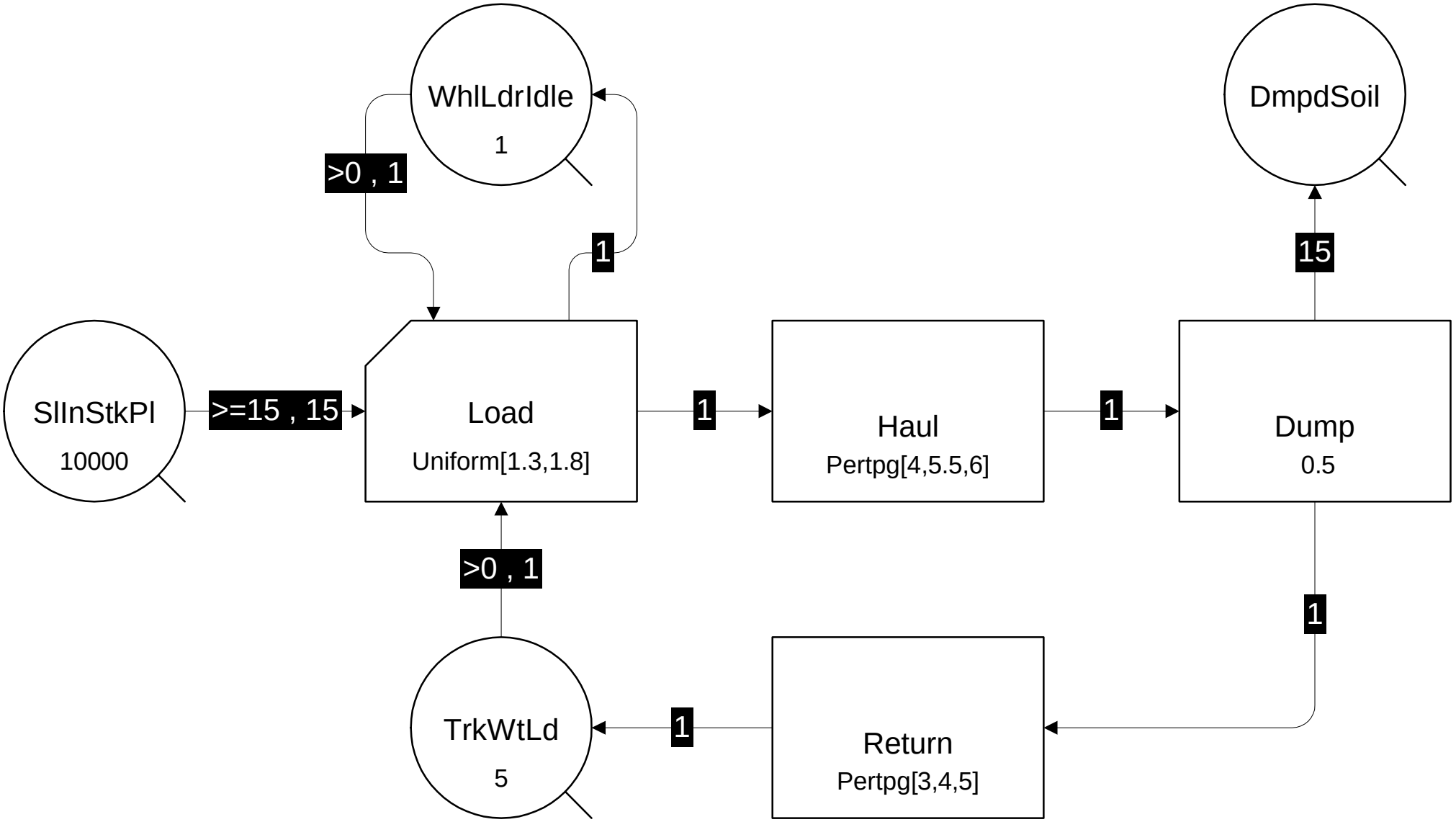
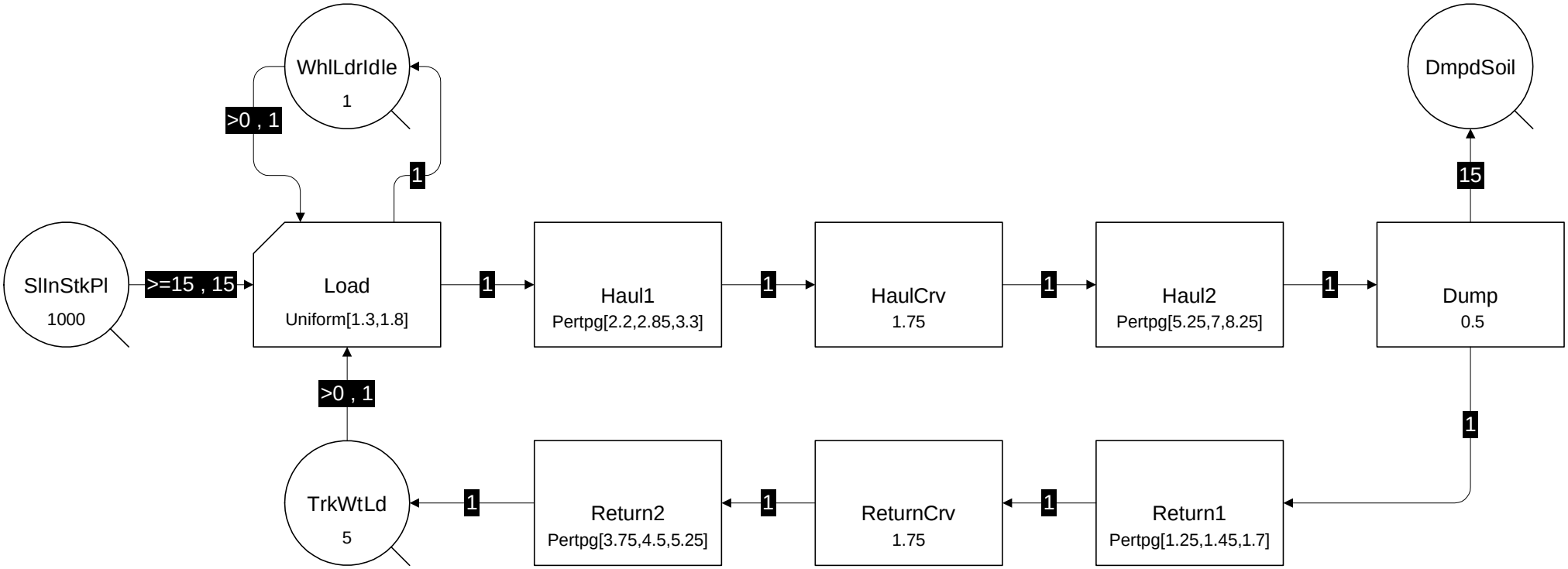
Conditions Needed to Start	Activity	Outcome of Activity
Wheel loader idle at source. Empty truck waiting to load. Enough soil in stockpile.	Load	Wheel loader idle at source. Loaded truck ready to haul.
Loaded truck ready to haul.	Haul	Loaded truck ready to dump.
Loaded truck ready to dump.	Dump	Dumped soil. Empty truck ready to return.
Empty truck ready to return.	Return	Empty truck waiting to load.



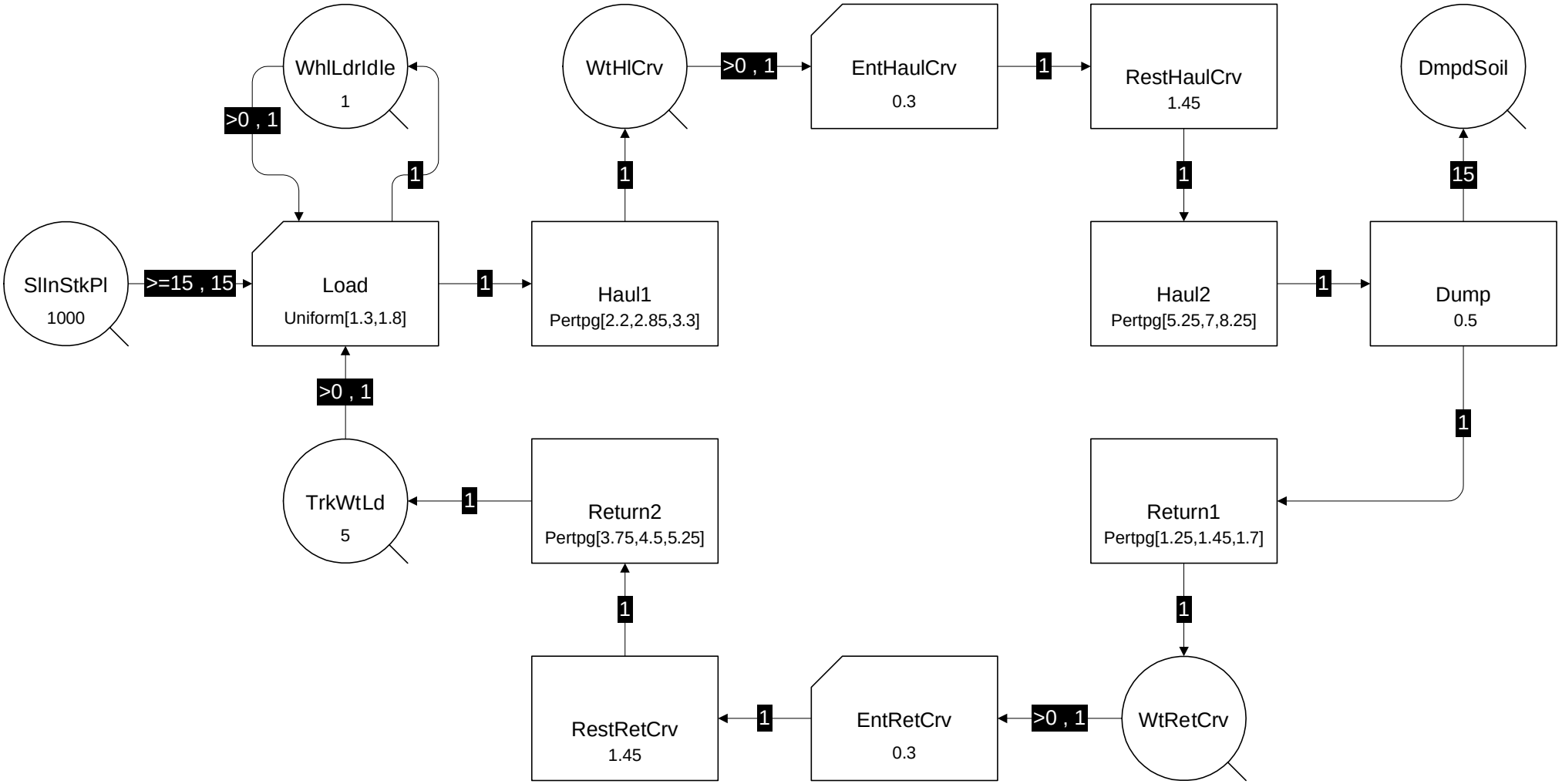
This is a classic model for the movement of 10000 m3 of soil using 15 m3 trucks and a wheel loader. The loading time is that required for the wheel loader to completely fill a truck. There is no restriction for dumping, which can take place immediately after hauling.



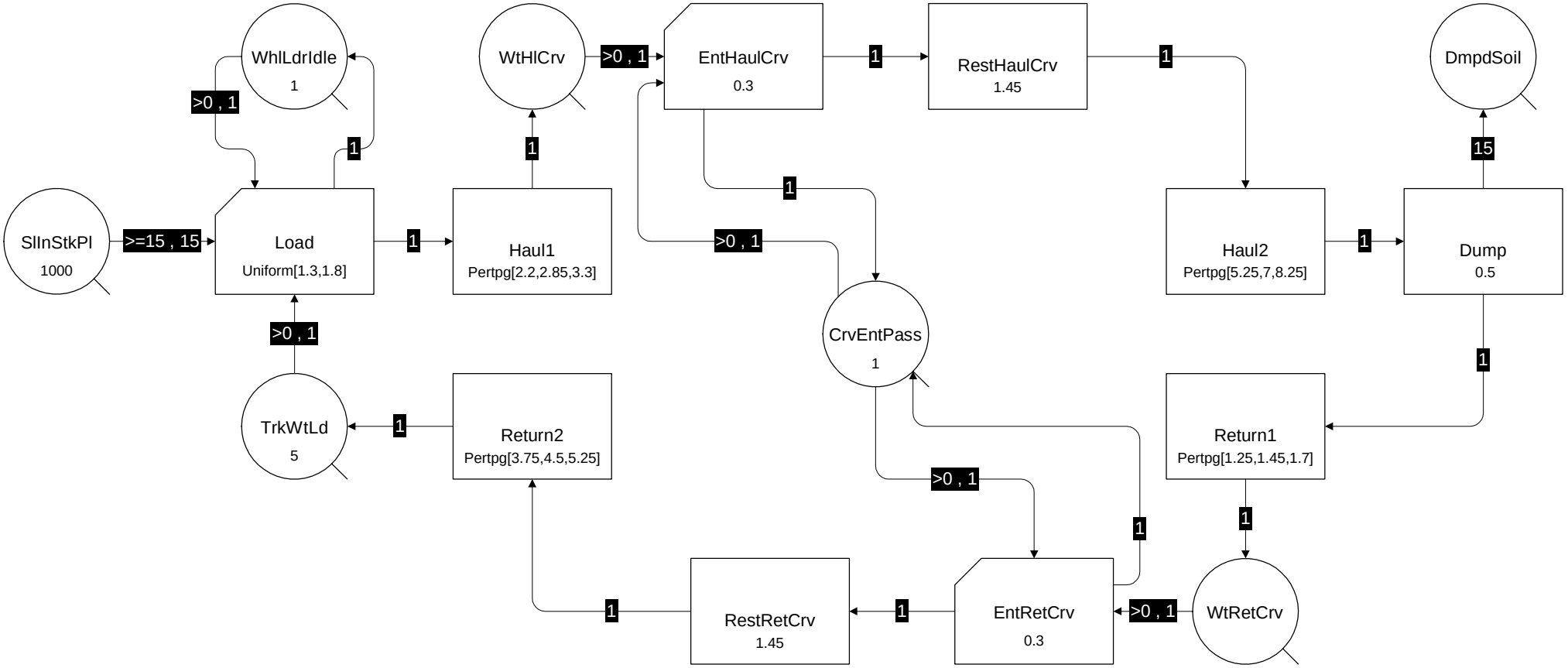
In this model the haul road has been broken into three parts. Haul now consists of a Haul1, HaulCrv, and Haul2. Similarly, Return now consists of Return1, Return Crv, and Return2. HaulCrv and ReturnCrv are for crossing a narrow, curved segment. This segment will eventually allow traffic in only one direction with no passing (single file loaded or empty traffic but not both at the same time). This is the first modification in a series of steps required to implemen t this.



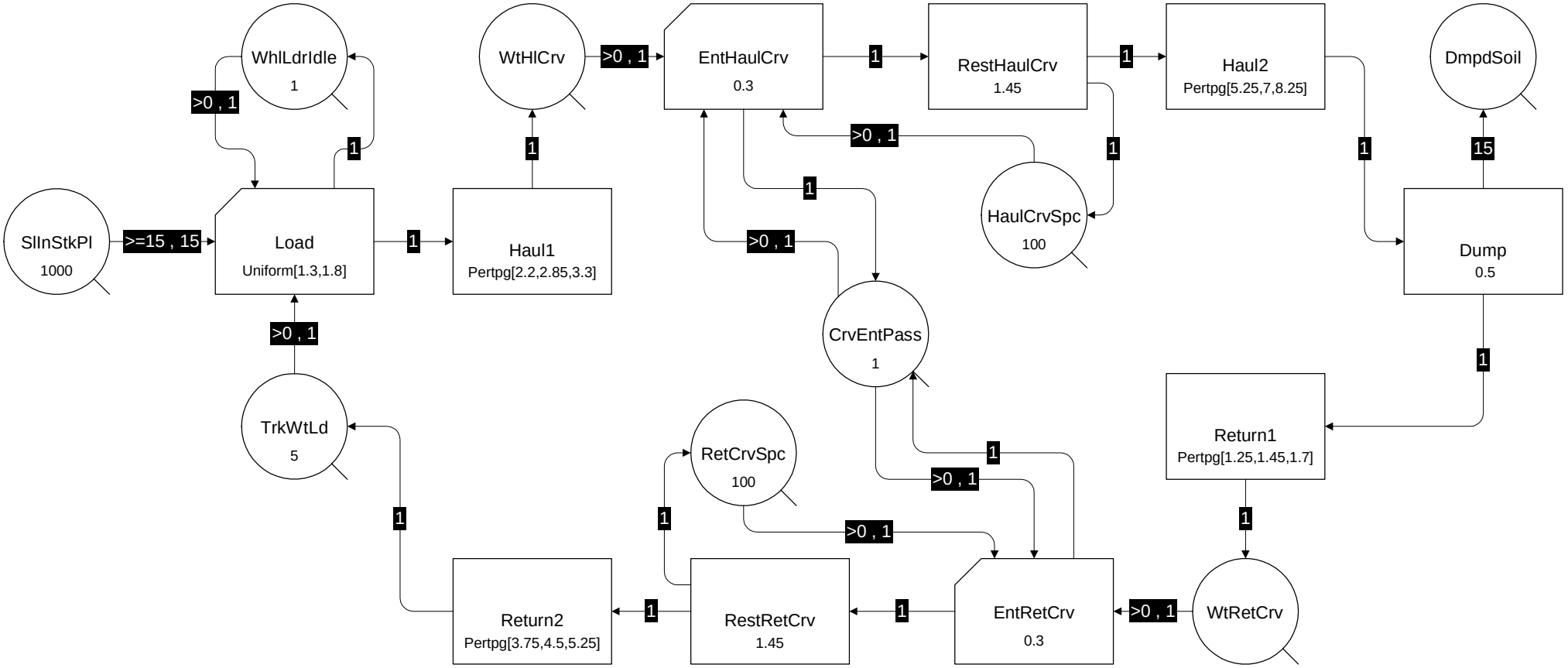
This model is functionally equivalent to the previous model. The haul and return over the narrow curve has been broken up into two parts, and preceded by re spective queues. WtHlCrv and WtRetCrv are conceptually necessary since crossing over the narrow segment is not an immediate consequence of arriving at the segment. A truck may need to wait for traffic to switch to its direction or for the truck ahead to make space. The crossing of the curve itself has been broken up into two par ts, one to represent the entry into the curve and the other to represent the traversal of the remainder. This is done in preparation to recognize that although more than one truck can traverse the curver at the same time, only one can be entering it.



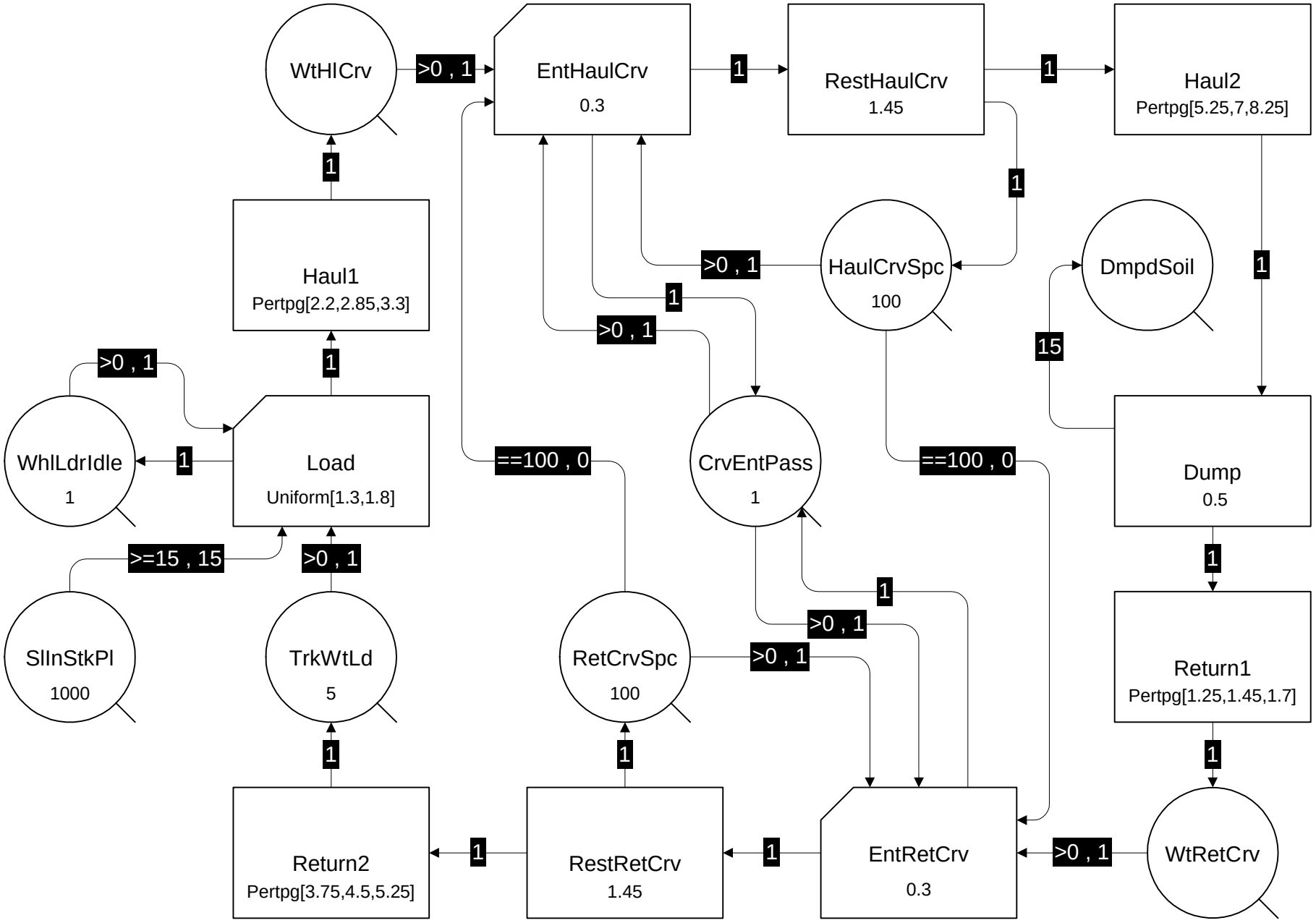
In this model, the entry of trucks into the curve has been forced to be sequential by the introduction of CrvEntPass, which is initialized with only one resource. Now only one truck can enter the curve at a time (from either side). This model still does not prevent head-on collisions.



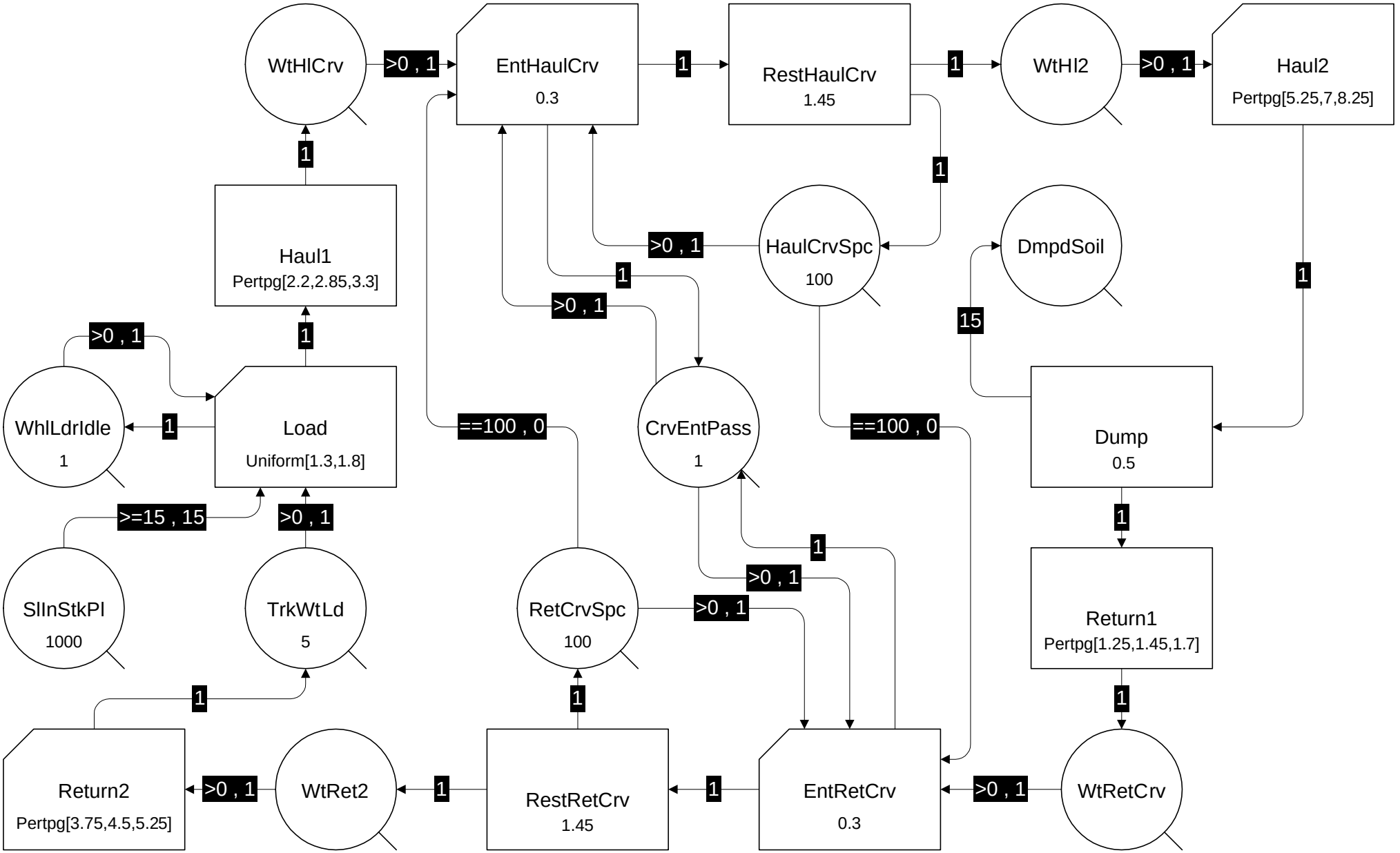
In this model, the HaulCrvSpc and RetCrvSpc queues have been added to provide dynamic information about the number of trucks that are traversing the curve while hauling or returning. Each of the queues is initialized with an arbitrarily large number (100) of resources. Whenever a truck enters the curve, a resource in the corresponding queue is consumed (making its content less than 100). Whenever a resource finishes traversing the curve, a resource is added to the corresponding queue (incrementing its content, perhaps to 100 if it is the last truck traversing the curve). In this model this information is not used, but it is necessary for the subsequent step.



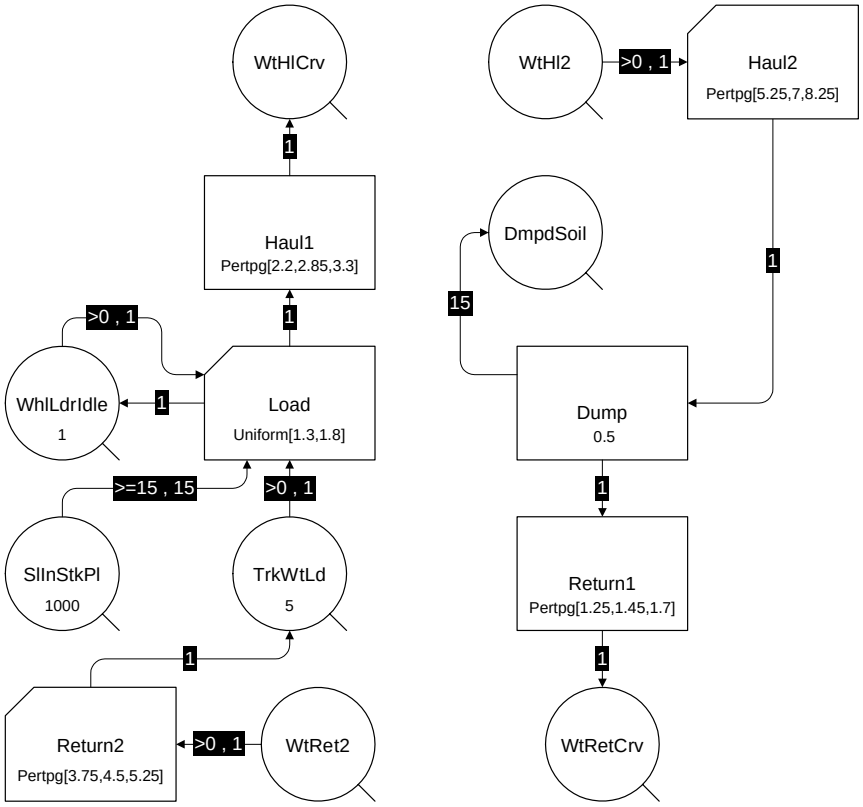
In this model, the nodes have been rearranged and the information provided by the contents of HaulCrvSpc and RetCrvSpc has been used to implement the single-lane unidirectional logic of the curve. This is done by the links with inscription "==100,0" that connect HaulCrvSpc to EntRetCrv and RetCrvSpc to EntHaulCrv. Essentially, a hauling truck is allowed to enter the curve only when there are no returning trucks in it (i.e., the RetCrvSpc queue contains 100 resources). The same applies to returning trucks.

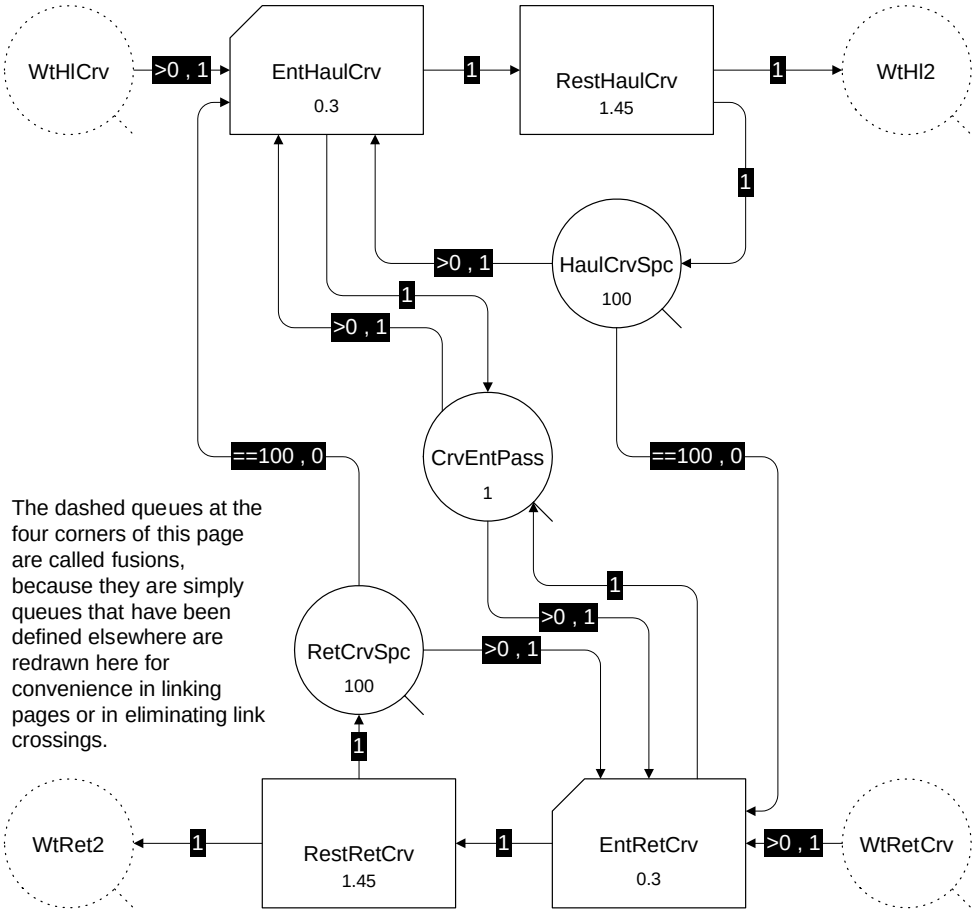


This model has turned the Haul2 and Return2 normal activities into combi activities by the introduction of WtHl2 and WtRet2. Conceptually, the model is the same as before. This has been done, however, to enable the partitioning of this model into two separate pages.



Here, the model has been broken up into two pages. One page (the 2nd page) with the logic of the curve and another page with the rest of the model (this page). This page is linked to the next one by the WtHlCrv, WtHl2,WtRetCrv and WtRet2 queues (fusions of these queues appear in the next page).

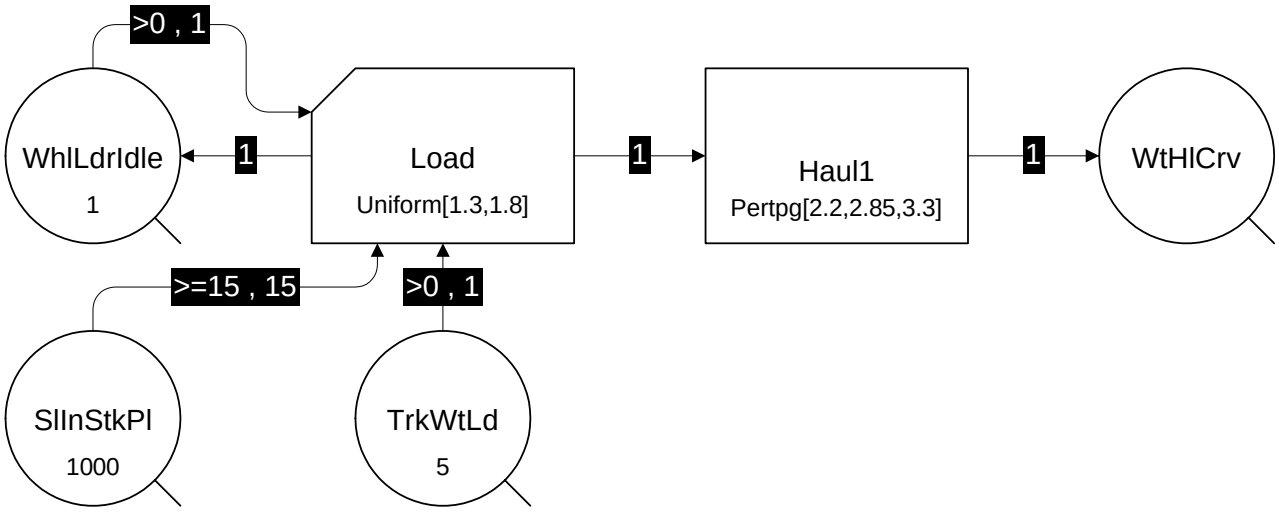




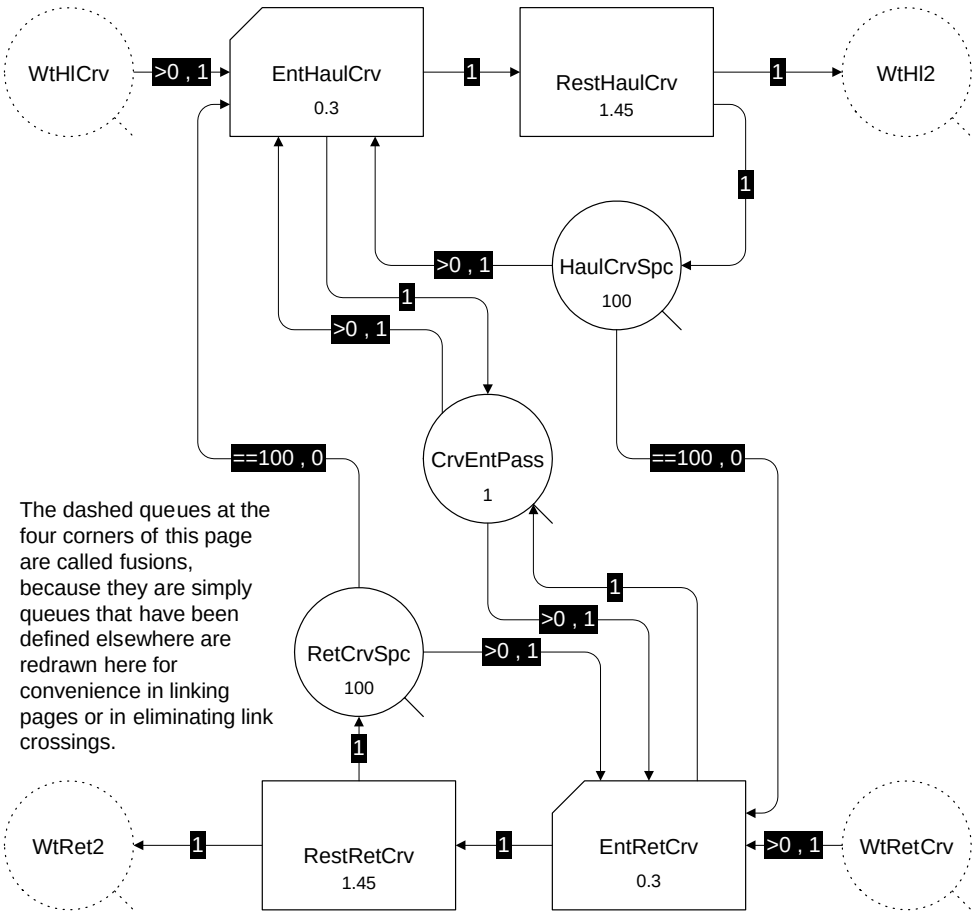
The dashed queues at the four corners of this page are called fusions, because they are simply queues that have been defined elsewhere are redrawn here for convenience in linking pages or in eliminating link crossings.

EZStrobe Tutorial Model 9 Page 1/3

This portion of the model, the first of three pages, contains the loading and hauling to the curve. It links to other parts of the model by fusions of the WtHlCrv and TrkWtLd queues.

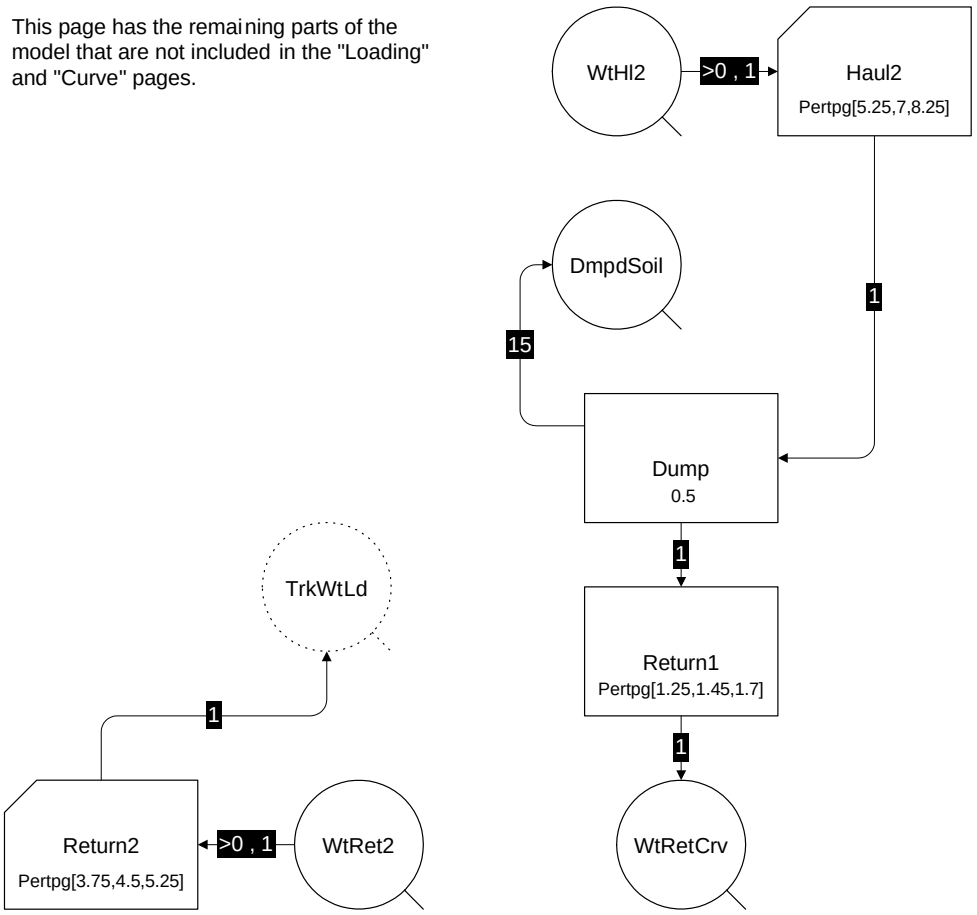


EZStrobe Tutorial Model 9 Page 2/3



The dashed queues at the four corners of this page are called fusions, because they are simply queues that have been defined elsewhere are redrawn here for convenience in linking pages or in eliminating link crossings.

This page has the remaining parts of the model that are not included in the "Loading" and "Curve" pages.

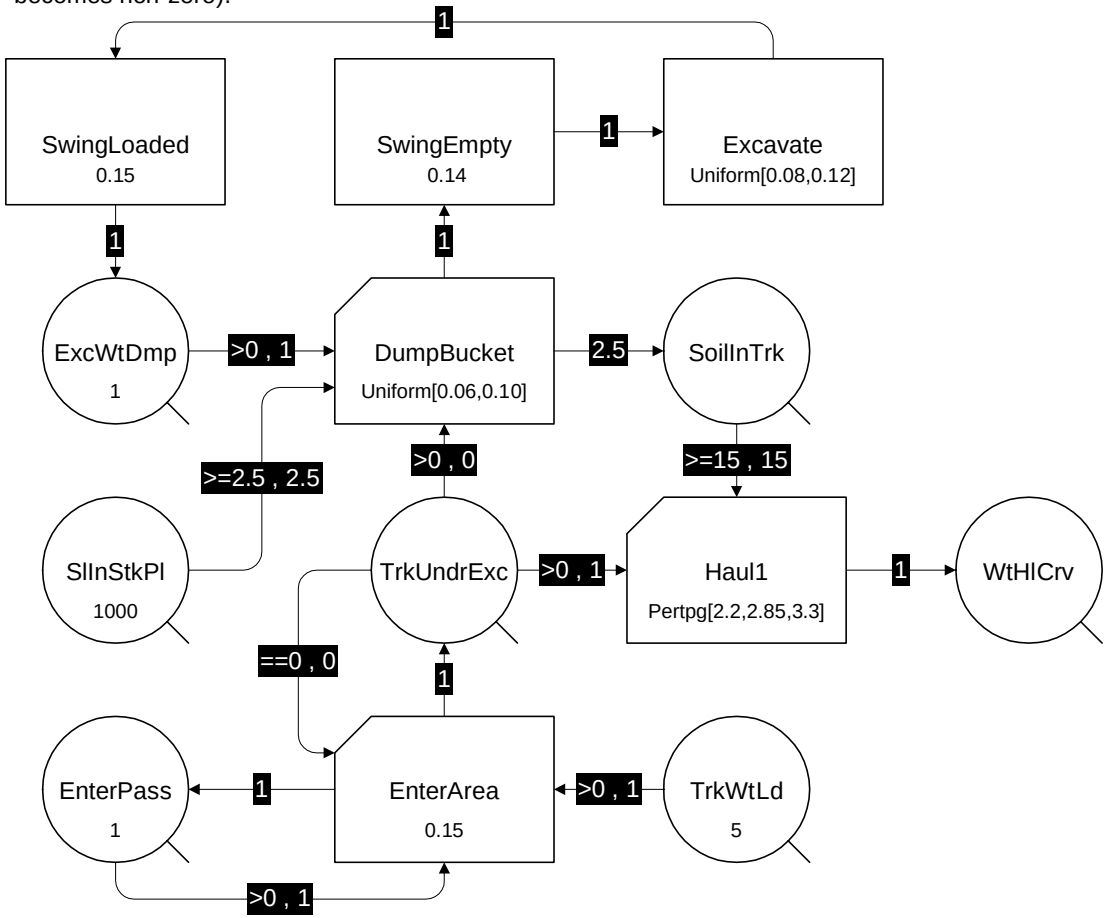


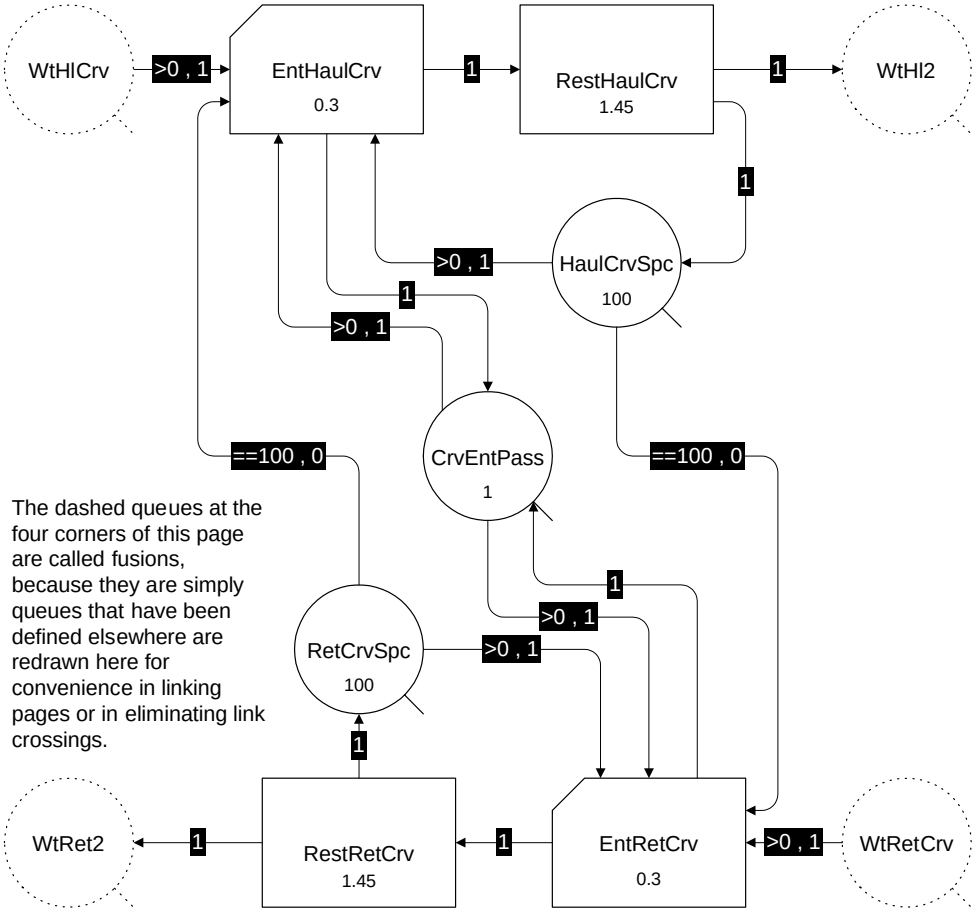
Here, the loading has been modeled in much more detail, representing the complete cycle of the excavator (switch from a wheel loader). The excavator's dump to the truck, empty swing, actual digging, and loaded swing are modeled explicitly. In this model the excavator will wait for a truck with its bucket ready to dump if it has time to do so after a previous truck has left. TrkUnderExc contains a truck while it is being loaded, hence the average waiting time there is the time required to load 15 m3 of soil into a truck with a 2.5 m3 excavator.

Notice the link from TrkUnderExc to EnterArea. Its function is to allow EnterArea to start only if there is no truck under the excavator. Under these circumstances it can obviously not remove a truck from TrkUnderExc regardless of the number placed after the comma in the link (because none are available). A zero is used to make this explicit.

Also notice the link from TrkUnderExc to DmpBucket. It indicates that there must be a truck in TrkUnderExc in order for the excavator to dump its bucket, but it also indicates that when DmpBucket starts it should not remove a truck from TrkUnderExc.

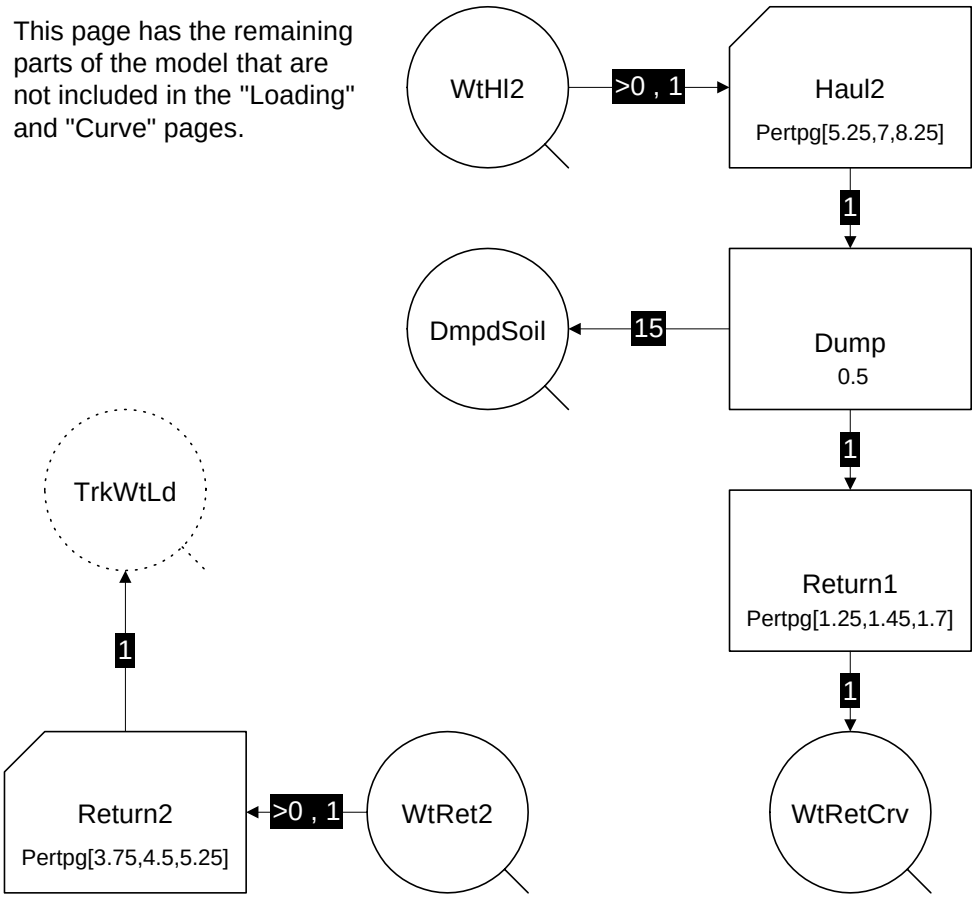
The EnterPass queue is necesary to prevent more a truck from entering the load area while another truck is entering (it is only after a truck completes its entry that the content of TrkUnderExc becomes non-zero).





The dashed queues at the four corners of this page are called fusions, because they are simply queues that have been defined elsewhere are redrawn here for convenience in linking pages or in eliminating link crossings.

This page has the remaining parts of the model that are not included in the "Loading" and "Curve" pages.



SoilAmt	Amount of soil in m3	10000
ExcCap	Excavator capacity in m3	2.5
TruckCap	Truck capacity in m3	15
nTrucks	Number of trucks	5
TrckCst	Truck cost ($/hr)	48
ExcCst	Excavator cost ($/hr)	65
OHCst	Overhead cost ($/hr)	75

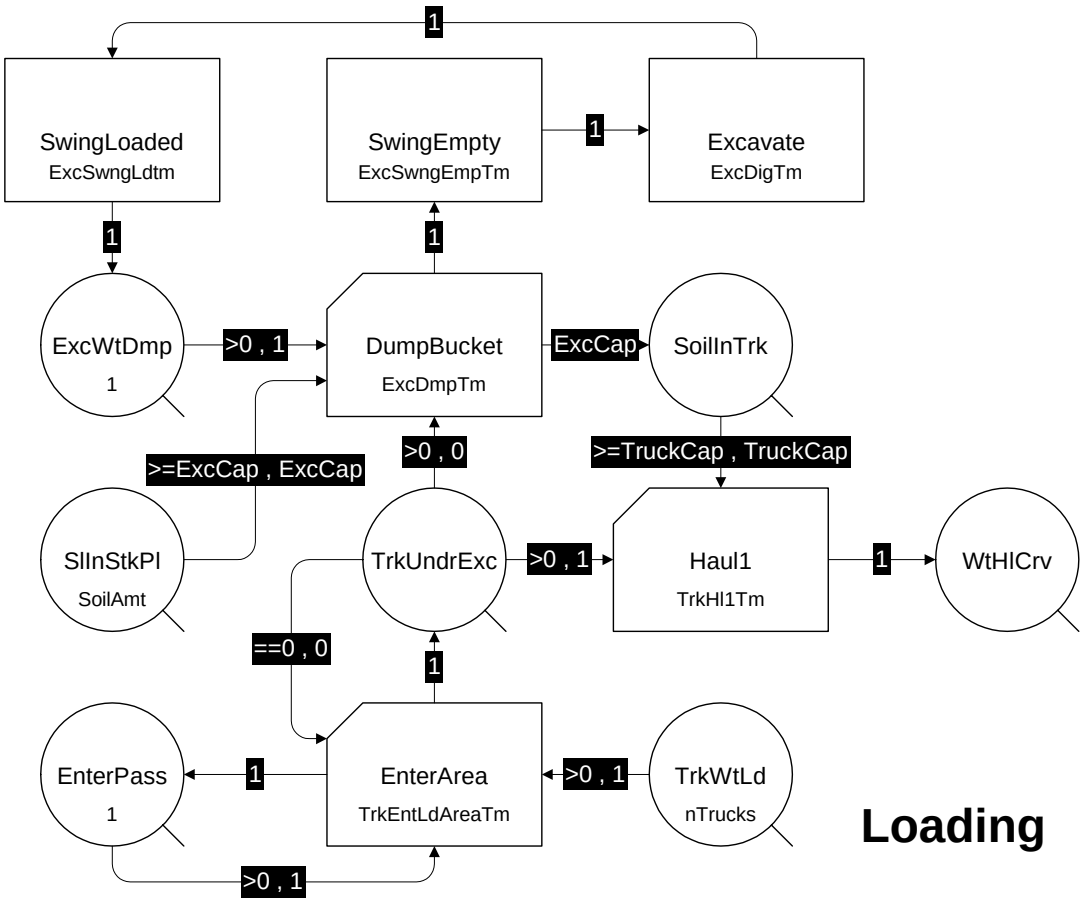
This model has been parameterized so that inputs are defined in one place and accessed symbolically throughout the model.

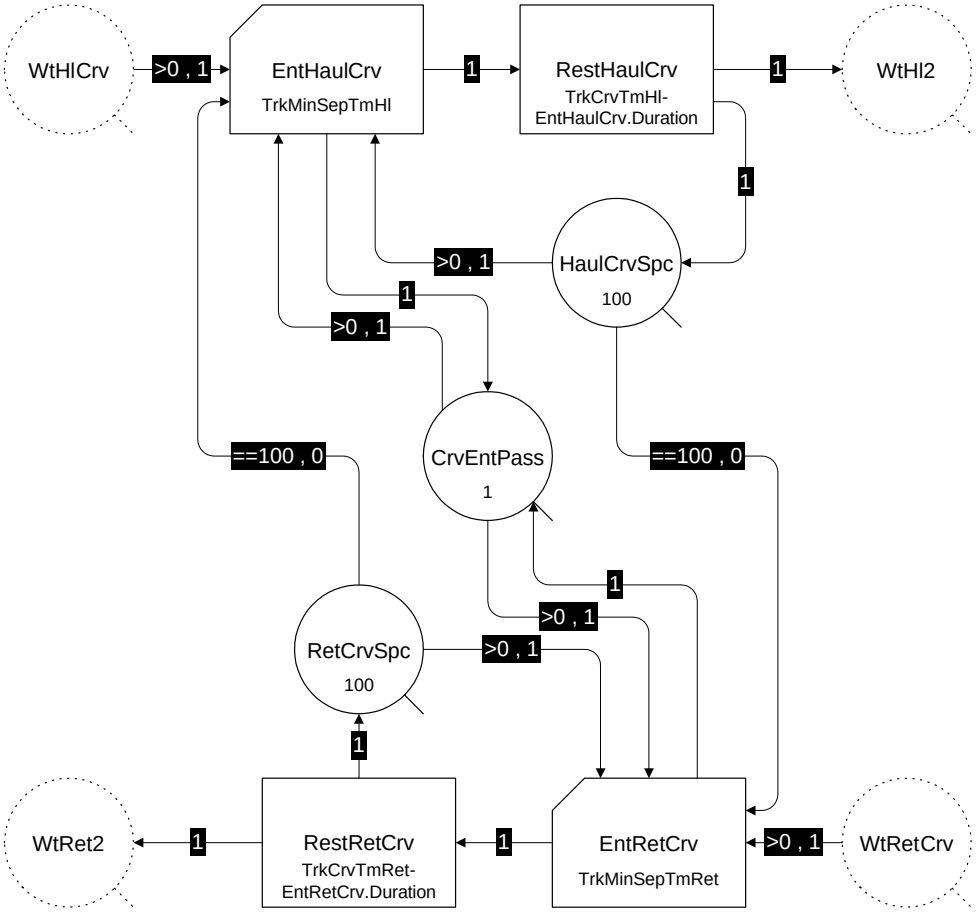
ExcDmpTm	Excavator dumping time	Uniform[0.06,0.10]
ExcSwngEmpTm	Excavator swing empty time	0.14
ExcDigTm	Excavator digging time	Uniform[0.08,0.12]
ExcSwngLdtm	Excavator swing loaded time	0.15

TrkEntLdAreaTm	Truck entry to load area time	0.15
TrkHl1Tm	Truck time for first haul	Pertpg[2.2,2.85,3.3]
TrkMinSepTmHl	Truck minimum separation time for hauling	0.3
TrkCrvTmHl	Truck time to traverse curve loaded	1.75
TrkHl2Tm	Truck time for second haul	Pertpg[5.25,7,8.25]
TrkDmpTm	Truck time to dump	0.5
TrkRet1Tm	Truck time for first return	Pertpg[1.25,1.45,1.7]
TrkMinSepTmRet	Truck minimum separation time returning	0.3
TrkCrvTmRet	Truck time to traverse curve returning	1.75
TrkRet2Tm	Truck time for second return	Pertpg[3.75,4.5,5.25]

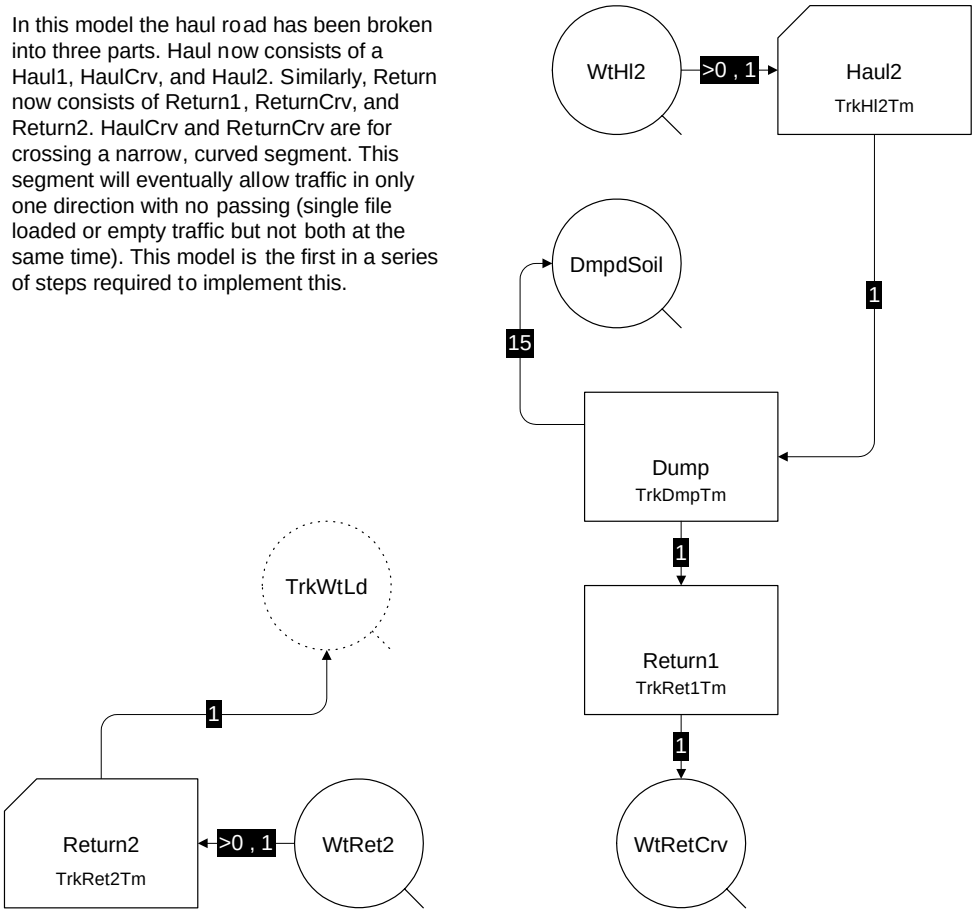
HourlyCst	Hourly cost	OHCst+ExcCst+nTrucks*TrckCst
Hrs	Hrs needed to move material	SimTime/60
ProdRate	Production rate (m3/hr)	DmpdSoil.CurCount/Hrs
UnitCst	Unit cost ($/m3)	HourlyCst/ProdRate

Important results have been extracted from the model and displayed explicitly.
This model is ready to be published in the web.





In this model the haul road has been broken into three parts. Haul now consists of a Haul1, HaulCrv, and Haul2. Similarly, Return now consists of Return1, ReturnCrv, and Return2. HaulCrv and ReturnCrv are for crossing a narrow, curved segment. This segment will eventually allow traffic in only one direction with no passing (single file loaded or empty traffic but not both at the same time). This model is the first in a series of steps required to implement this.



EZStrobe User's Guide

To run EZStrobe you can either select EZStrobe when you launch Visio and it asks you to select a drawing template, or you can use the menu sequence File -> New -> EZStrobe from within Visio. Create your model by dragging and dropping modeling elements from the stencil. The Stencil, EZStrobe.vss, is a window with green background that lies along the left edge of the Visio workspace. Never open the EZStrobe Stencil directly (only the template, EZStrobe.vst).

When you drag Combis, Normals, and Queues from the stencil unto the drawing, a dialog box appears where you can supply the properties of the node. You can later edit these properties by double-clicking over the node. If dialog boxes do not appear upon drag or double-click, it is due to a faulty installation -- please try re-installing or contact the author if problems persist.

After you drag a Link, its ends must be connected to nodes. Not all parts of nodes can be targets of connections, only certain points along their perimeter. You can tell that a Link end is over a connectable portion of a node if while moving the Link's end, it shows a bold square around the grabbed end. A proper connection will be red when the link is selected.

Edit model options and run the model by selecting from the menu that pops up when you right-click over an empty area of the page.

Meaningful dynamic information exposed by EZStrobe via system-defined and system-maintained variables. In the following tables, the word *Activity* (in italics) should be replaced by the name of an actual Activity in the model; the word *Queue* (in italics) should be replaced by the name of an actual Queue in the model.

Global Variables (accessible all the time):

Variable Form	Description
SimTime	The current value of the simulation clock
Activity.AveDur	The average value of the duration of the instances of *Activity*
Activity.AveInter	The average inter-instantiation time between instances of *Activity*
Activity.CurInst	The current number of instances of *Activity*
Activity. InContext	Returns 1 if an instance of *Activity* is being created or terminated, and 0 otherwise
Activity.FirstStart	The time at which the first instance of *Activity* started
Activity.LastStart	The time at which the last instance of *Activity* started
Activity.MaxDur	The maximum value of the durations of the instances of *Activity*
Activity.MaxInter	The maximum of the inter-instantiation times between instances of *Activity*
Activity.MinDur	The minimum value of the durations of the instances of *Activity*
Activity.MinInter	The minimum of the inter-instantiation times between instances of *Activity*
Activity.SDDur	The standard deviation of the durations of the instances of *Activity*
Activity.SDInter	The standard deviation of the inter-instantiation times between instances of *Activity*
Activity.TotInst	The total number of instances of *Activity* that have been created
Queue.LastAmtReceived	The amount of resource that last entered *Queue*
Queue.AveCount	The time-weighted average of the content of *Queue*
Queue.AveWait	The average waiting time for resources that have entered *Queue*
Queue.CurCount	The current content of *Queue*
Queue.MaxCount	The maximum content experienced by *Queue*
Queue.MinCount	The minimum content experienced by *Queue*
Queue.SDCount	The time-weighted standard deviation of content experienced by *Queue*
Queue.TotCount	The total amount of resource that has entered *Queue*

Instance Variables (accessible only when an instance of Activity is starting or terminating):

Variable Form	Description
Activity.Duration	The duration of the instance of *Activity* that is starting or ending
Activity.Instance	The instance number of the instance of *Activity* that is starting or ending

Function Prototype	Function Description
Abs[val]	Absolute value of val
Acos[val]	ArcCosine of val
Asin[val]	ArcSine of val
Atan[val]	ArcTangent of val
AtanXdivY[x,y]	ArcTangent of x/y
AveCount[queue]	Average content of queue
AveDur[activity]	Average duration of activity
AveInter[activity]	Average time between succesive starts of activity
AveWait[queue]	Average wait at queue
Beta[a,b]	Sample from a unit Beta distribution
Confidence[SD,level,nSamples]	Half width of a confidence interval
Cos[val]	Cosine of val
Cosh[val]	Hyperbolic cosine of val
CurCount[queue]	Current content of queue
CurInst[activity]	Current number of instances of activity
Duration[activity]	Duration of the instance in context of activity
Erlang[order,mean]	Sample from an Erlang distribution
Exp[val]	Exponential function of val
Exponential[mean]	Sample from an Exponential distribution
FirstStart[activity]	Time at which the first instance of activity started
Gamma[a,b]	Sample from a Gamma distribution
InContext[Activity]	Indicates whether there is an instance in context of the activity
Instance[activity]	Instance number of the instance in context of activity
Int[val]	Integer part of val
LastAmtReceived[Queue]	Returns the last amount received by GenQueue
LastRnd[]	Retrieve the last number returned by Rnd[]
LastStart[activity]	Time at which the last instance of activity started
Ln[val]	Natural logarithm of val
Log[val]	Base 10 logarithm of val
Max[val1,val2]	Maximum of val1 and val2

Function Prototype	Function Description
MaxCount[queue]	Maximum content of queue
MaxDur[activity]	Maximum duration of activity
MaxInter[activity]	Maximum time between succesive starts of activity
Min[val1,val2]	Minimum of val1 and val2
MinCount[queue]	Minimum content of queue
MinDur[activity]	Minimum duration of activity
MinInter[activity]	Minimum time between succesive starts of activity
Mod[val,div]	Remainder of val / div
Normal[mean,sd]	Sample from a Normal distribution
NormalInv[mean,stdev,cumulative]	Inverse of the Normal distribution
Pert[p0,Mode,p100]	Sample from a Beta distribution
Pertpg[p5,Mode,p95]	Sample from a Beta distribution
Rnd[]	Sample a number uniformly distributed between 0 and 1
Round[expression,decimals]	Round expression to the specified number of decimal places (which can be negative)
ScaledBeta[low,high,a,b]	Sample from a scaled Beta distribution
SDCount[queue]	Standard deviation of content of queue
SDDur[activity]	Standard deviation of duration of activity
SDInter[activity]	Standard deviation of time between succesive starts of activity
Sin[val]	Sine of val
Sinh[val]	Hyperbolic sine of val
Sqrt[val]	Square root of val
StdNormalInv[Cumulative]	Inverse of the Standard Normal distribution
Tan[val]	Tangent of val
Tanh[val]	Hyperbolic tangent of val
tInv[alpha,degFreedom]	Inverse of the t Distribution
TotCount[queue]	Total content of queue
TotInst[activity]	Total number of instances of activity
Triangular[low,mode,high]	Sample from a triangular distribution
Uniform[low,high]	Sample from a Uniform distribution