User Manual

Software Version: 2020-11
September 17, 2020

Table of Contents

1 Introduction

JaamSim (Java Animation Modelling and Simulation) is a discrete-event simulation software package that includes a drag-and-drop graphical user interface, 3D graphics, and a full set of built-in objects for model building. It is object oriented, extremely fast, and scalable to the largest of applications. Windows, Linux, and OSX are all supported.

JaamSim is free open-source software, licensed under Apache 2.0. The latest version of the software and manuals can be downloaded from the JaamSim website: www.jaamsim.com. The source code is published on GitHub: www.github.com/jaamsim/jaamsim. Presentations and tutorials for JaamSim can be found by following the Videos link on the JaamSim website.

JaamSim provides all the key functions needed for any simulation model:

- Controls for launching and manipulating simulation runs;
- Drag-and-drop user interface;
- Interactive 3D graphics;
- Input and output processing; and
- Model development tools and editors.

JaamSim also provides a full suite of built-in objects for model building, including:

- Objects for process flow type models (servers, queues, etc.);
- Objects for modelling continuous processes (integrator, PID controller, etc.
- Text objects for labelling and documentation;
- Graphs for visualizing simulation outputs;
- Probability distributions for random sampling; and
- Graphical objects for background maps and logos.

Advanced users can create additional palettes of application-specific objects. A separate Programming Manual can be downloaded from the JaamSim website.

2 Installing and Running JaamSim

2.1 System Requirements

JaamSim runs under Windows, Mac OS X, and Linux on most modern computers. Any computer with an Intel Core i3, i5, and i7 series processor is sufficient for JaamSim (second generation "Sandy Bridge" processor and later).

2.2 Installing Java

JaamSim requires a recent (Version 8 or later) installation of the Java Runtime Environment (JRE) or Java Development Kit (JDK), available for download from Oracle at www.java.com.

Starting with Update 211 released on April 16, 2019, Oracle requires commercial users to pay a license fee. Commercial users can obtain a free version of Java from AdoptOpenJDK.net. The preferred version is 'OpenJDK 8 (LTS)' with the 'HotSpot' virtual machine.

The default JRE/JDK for Windows computers is the 32-bit version, in which case the 32-bit version of JaamSim must be used.

For computers running 64-bit Windows, the 64-bit version of JaamSim offers significantly improved performance. This version requires the 64-bit version of the JRE/JDK. Both the 32-bit and 64-bit JRE/JDK can be installed on the same computer.

2.3 Installing JaamSim

JaamSim consists of a single executable that can be copied directly to the user's computer. No special installation program is required. Copy the JaamSim executable file to a working directory, such as the directory that will contain the model input files.

Three versions of the JaamSim executable are available for each release:

- JaamSimYYYY-NN.exe (the 64-bit executable for Windows)
- JaamSimYYYY-NN_x86.exe (the 32-bit executable for Windows)
- JaamSimYYYY-NN.jar (the executable jar file for Windows, Mac OS X, and Linux)

JaamSim releases are denoted by the year of release (YYYY) and by release number (NN) within the year.

2.4 Installation Problems

2.4.1 No Graphics

If JaamSim does not run correctly on your computer, the most likely cause is an older graphics driver. Updating the driver to the latest version normally will solve the problem. If a suitable driver is not available, add the tab -sg (Safe Graphics) to the command line. This tag instructs JaamSim to use only the simplest OpenGL graphics commands, which may solve the problem.

2.4.2 Distorted Text

Distorted text in the View windows is caused by 'Conservative Morphological Anti-Aliasing'. To turn this setting off, open the 'Intel HD Graphics Control Panel' by right clicking on the Windows desktop and selecting 'Graphics Properties'. Select '3D' and then set the 'Conservative Morphological Anti-Aliasing' to 'Turn Off'.

2.4.3 Mac Computers with the Retina Monitor

Mac computers with the Retina monitor require the use of the "Display Menu" app that can be downloaded from the Apple App Store. This app compensates for the non-standard way in which Apple uses the higher resolution of the Retina monitor. If JaamSim is launched without this app, it will not be possible to select objects using the mouse.

2.4.4 Ubuntu Linux

The Intel graphics driver for Ubuntu has a long-standing bug that affects JaamSim. It can be avoided by entering the following command before launching JaamSim:

```
export LIBGL_ALWAYS_SOFTWARE=1
```

This command instructs Ubuntu to use software rendering, which, although slower, is sufficient for most models.

2.5 Running JaamSim from the GUI

JaamSim can be launched by double-clicking on the executable file.

2.6 Running JaamSim from the Command Line

JaamSim can also be launched, configured and started automatically from the command line or a batch file using the command:

```
JaamSim.exe config1.cfg -tags
```

or, when using the .jar file:

```
java -jar JaamSim.jar config1.cfg -tags
```

Here, config1.cfg is the name of the input file to be loaded and -tags are the optional tags for the run. Multiple tags must be separated by a space. The following tags are supported:

Table 2-1 Batch Mode Run Tags

Tag	Description
-b *or* -batch	Starts the simulation immediately after the input file has been read, and exits when the run has completed. This tag is useful for batch file execution.
-m *or* -minimize	Minimizes the graphical user interface, allowing the simulation to run slightly faster when visualizations are not required (for instance, in overnight simulation runs).

Tag	Description
-s *or* -script	Directs JaamSim to accept configuration file inputs piped to JaamSim through standard-in and to direct its outputs specified by the RunOutputList keyword to standard-out. The .jar file (jaamsim.jar) must be used with this feature, not the executable (jaamsim.exe).
-h *or* -headless	Runs JaamSim without the graphical user interface so that it can be executed on a server that has no graphics capability. Batch mode (-b) is set automatically with this option.
-sg *or* -safe_graphics	Instructs JaamSim to use only the simplest OpenGL commands for compatibility with older graphics processors.

It is also possible to load two or more configuration files into a single model using the following command:

```
JaamSim.exe config1.cfg config2.cfg -tags
```

or,

```
java -jar JaamSim.jar config1.cfg config2.cfg -tags
```

Multiple simulation runs can be executed one after the other by using a batch file that contains a series of these commands. For example, a batch file containing the following two lines would execute two runs: run1.cfg and run2.cfg:

```
JaamSim.exe run1.cfg -b
JaamSim.exe run2.cfg -b
```

Note that the batch file and input configuration files must be in the same directory for this example to work.

JaamSim can be interfaced with other software packages using the -s (script) tag. For example, the following command instructs JaamSim to load the configuration file config.cfg and then accept additional configuration file inputs from program1. The outputs specified by the RunOutputList keyword for Simulation are then directed as inputs to program2.

```
program1.exe | java -jar JaamSim.jar config.cfg -s -b | program2.exe
```

At present, the script tag is supported only for the .jar file version of JaamSim - it does not work with the .exe file version.

Note that “java -jar” must be used in this command for standard-in and standard-out to be connected correctly to JaamSim. Without the "java -jar" portion of the command, the java virtual machine sets both standard-in and standard-out to null.

3 Getting Started

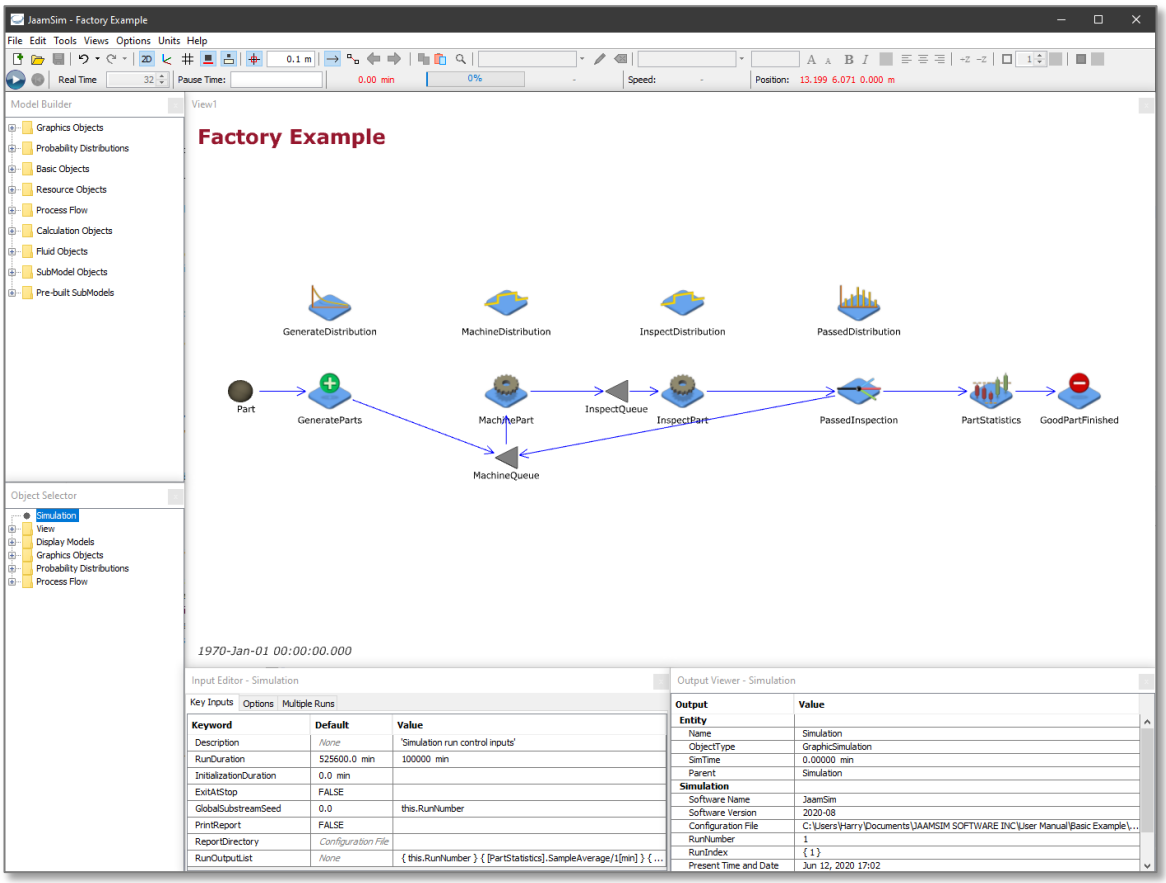
3.1 Basic Example

In this section, you will be guided through building a model of a simple factory that is used by the book 'Simulation Modeling and Analysis'[1] to compare a number of simulation packages. The model is defined as follows:

- Unfinished parts arrive to the factory with exponential inter-arrival times having a mean of 1 minute.
- The arriving parts are first sent to a machining centre which has a processing time that is uniformly distributed on the interval [0.65, 0.70] minutes.
- The machined parts are then inspected with a processing time that is uniformly distributed on the interval [0.75, 0.80] minutes.
- The inspected parts have a 90% probability of being 'good' and are sent to shipping. The remaining 10% are 'bad' and sent back to the machining centre for rework.

The finished JaamSim model is shown below.

Figure 3-1 Factory Example



[1] Simulation Modeling and Analysis, 5th Edition, Averill M. Law

The primary output for the model is the average time in system for the parts. The simulation model will be executed for 10 replications of 100,000 minutes each.

The model is constructed in the following steps:

1) Creating model objects
2) Connecting the objects
3) Adding probability distributions
4) Collecting statistics
5) Running the model
6) Obtaining output reports
7) Performing multiple replications
8) Obtaining the times in each state
9) Reading model inputs from a data file
10) Running multiple scenarios

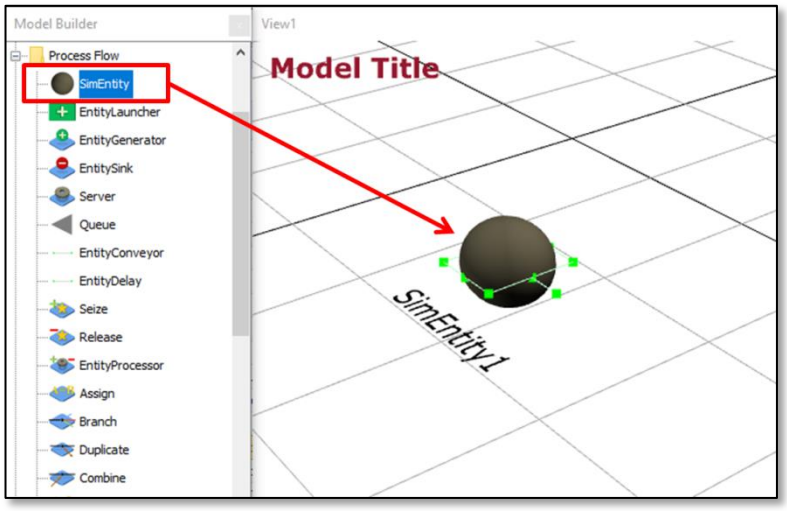
3.1.1 Creating Model Objects

After launching JaamSim, the following windows will appear:

- Control Panel – provides a number of run control features;
- View Window – displays a graphical representation of the model;
- Model Builder – offers a selection of objects that can be added to the model;
- Object Selector – lists the objects present in the model;
- Input Editor – allows for editing of keywords for a selected object; and
- Output Viewer – displays outputs for a selected object.

In the Model Builder, expand the Process Flow palette and then drag-and-drop a SimEntity into the View Window (View1). This creates a SimEntity object with a default name (SimEntity1) and shape (Sphere) and automatically selects it, denoted by green highlighting as shown below.

Figure 3-2 SimEntity Dragged to the View Window



This object will serve as the prototype for entities that will be processed in the model.

The name for SimEntity1 can be changed by clicking on SimEntity1 in the Object Selector and pressing F2. An easier way to change the name is to double click on the name in the 3D view window and edit it in place. The new name is accepted by pressing the Enter key or by clicking elsewhere in the view window. Note that an object name can be any alphanumeric text that does not include spaces or most symbols. Use either of these methods to change the name of SimEntity1 to 'Part'.

Table 3-1 SimEntity Object to Create

Model Builder Palette	Object Type	Name
Process Flow	SimEntity	Part

Since the model graphics will be 2D, aside from the Part entity, click the 2D button on the Control Panel, so that the view becomes bird's eye. The coordinate axes are not needed in the 2D view and can be turned off by clicking the 'Show Axes' button.

Now change the title of the model by double clicking on the text 'Model Title' in the top left of the view window and editing the text to read 'Factory Example'. The edits are accepted by pressing the 'Enter' key or by clicking elsewhere in the model.

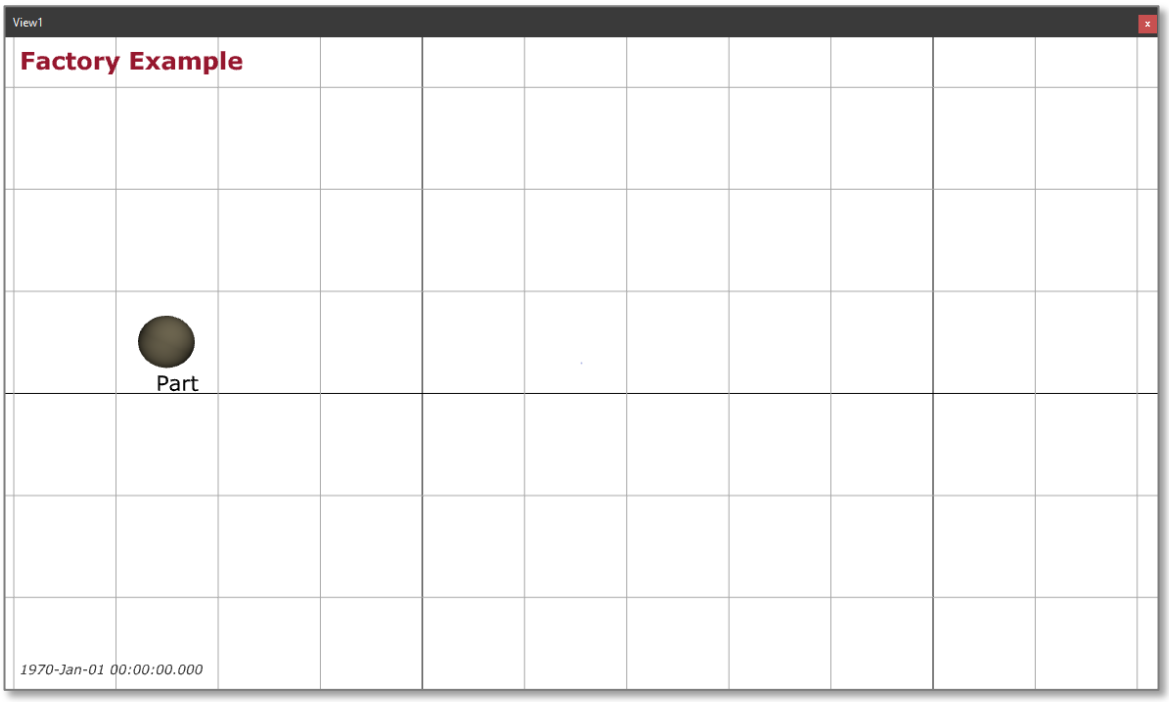
Adjust the viewing position using the actions listed below.

Table 3-2 View Adjustments

Action	Description
Left Click + Drag	Pan in the xy-plane
Scroll Wheel	Zoom in or out

After adjusting the view position and zoom, the View window should appear similar the following figure.

Figure 3-3 Configured View Window



Now create and rename the objects listed below:

Table 3-3 Objects to Create

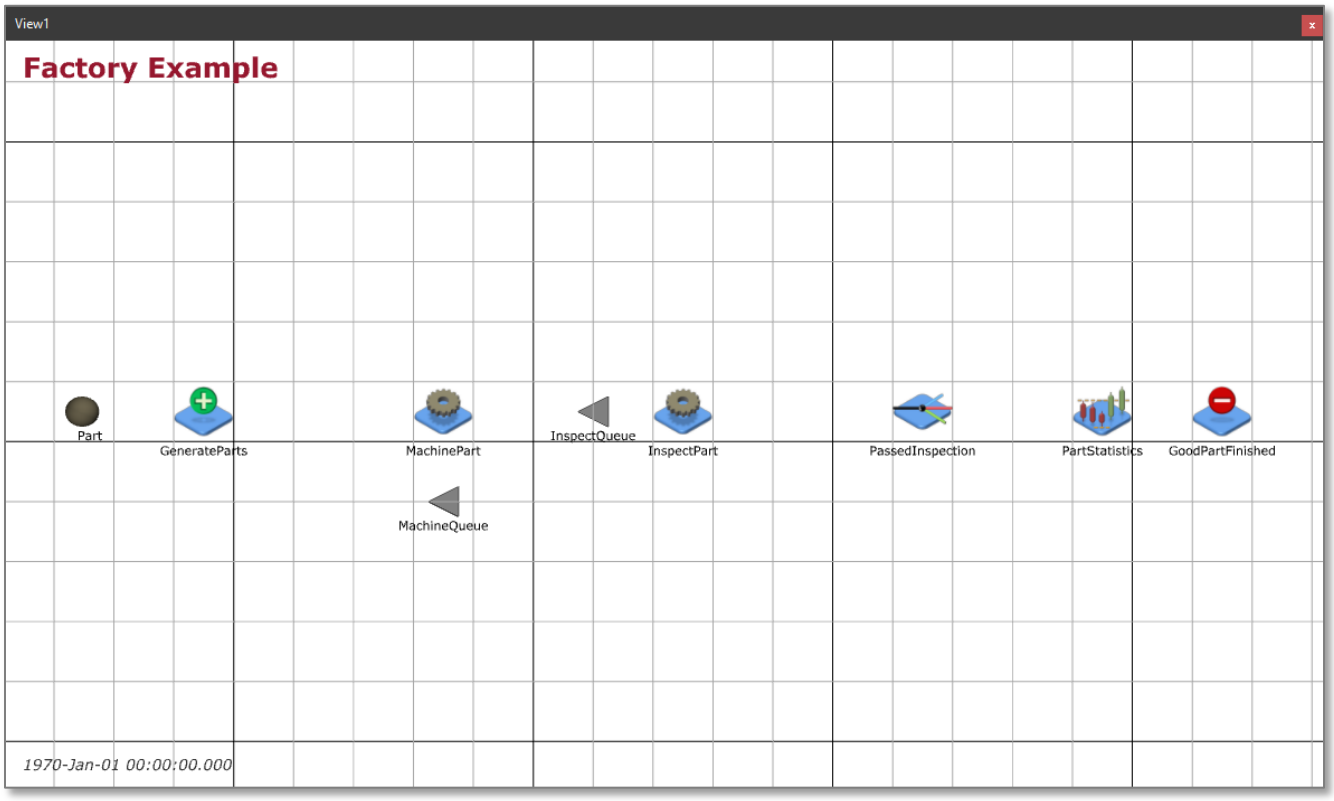
Model Builder Palette	Object Type	Name
Process Flow	EntityGenerator	GenerateParts
Process Flow	Server	MachinePart
Process Flow	Server	InspectPart
Process Flow	Queue	MachineQueue
Process Flow	Queue	InspectQueue
Process Flow	Branch	PassedInspection
Process Flow	Statistics	PartStatistics
Process Flow	EntitySink	GoodPartFinished

An object can be moved by selecting it and then using Control + Left Drag or by using the arrow keys. The position of a label relative to its object can be changed using the same method.

Multiple objects can be selected by holding down the Control key and left clicking on each object. Once selected, the objects can be moved as a group using Control + Left Drag.

Position the objects as shown below.

Figure 3-4 Screenshot of Step 1



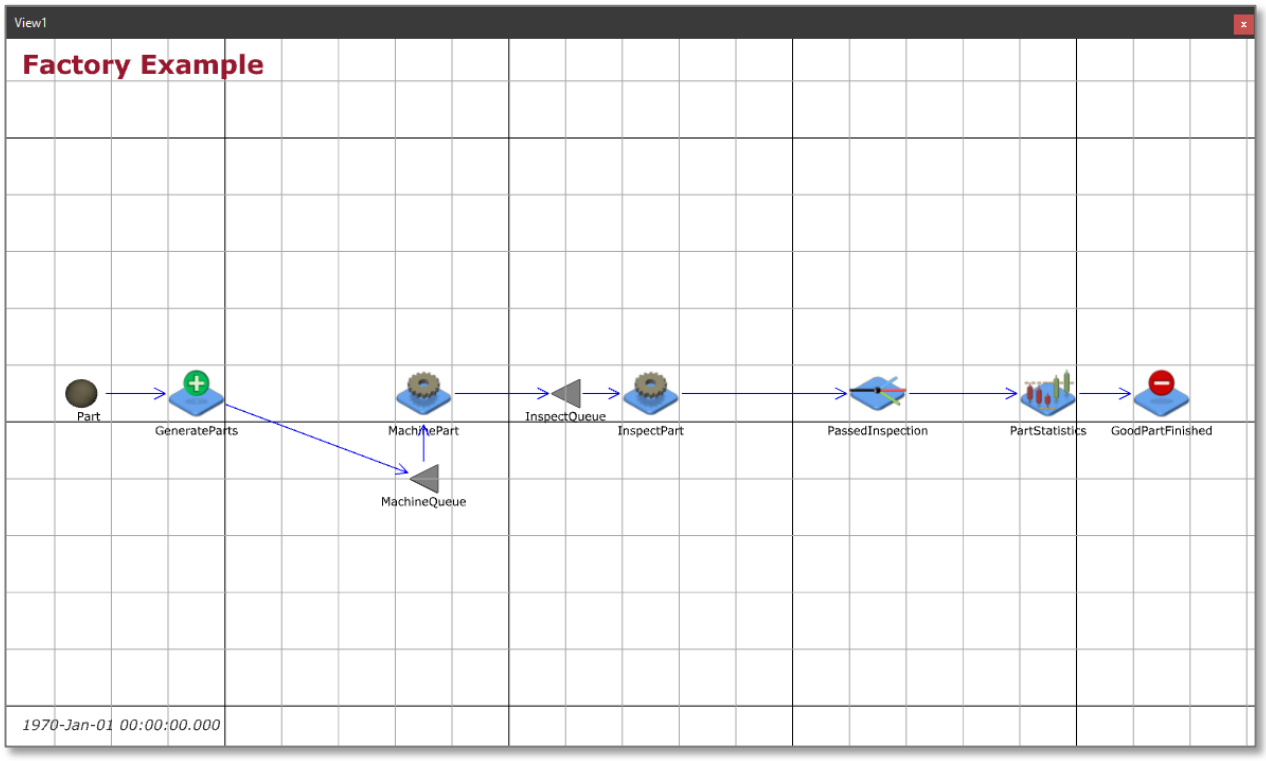
Save your work in progress by clicking the 'Save' button and choosing an appropriate name and folder for your model. It is good practice to save your model frequently as it is constructed. Note that there is no auto-save feature in JaamSim.

3.1.2 Connecting the Objects

In this step, the objects are connected so that the parts flow from one object to the next. This can be done using the Input Editor to set the inputs show below, but an easier method is to use the 'Create Entity Links' button. Select this button and then click on the objects in the following sequence: Part, GenerateParts, MachineQueue, MachinePart, InspectQueue, InspectPart, PassedInspection, PartStatistics, and GoodPartFinished.

If you make a mistake, you can interrupt the connection process by clicking on the background, and start again with the next object to be connected. When finished, click on the 'Create Entity Links' button again to de-activate it.

Figure 3-5 Screenshot of Step 2



The blue arrows indicate the flow of generated entities through the model. The arrows can be turned on and off using the 'Show Entity Flow' button. This button was selected automatically when the 'Create Entity Links' button was clicked.

You can confirm the connections by checking the relevant inputs using Input Editor. This editor provides the normal method for entering model inputs. Inputs are identified by keywords and organized by topic into separate tabs. An input value can be entered by clicking the on entry in the Value column of the editor. A dropdown menu of options is provided for most keywords.

Figure 3-6 Input Editor

Input Editor - GenerateParts

Key Inputs | Options | Thresholds | Maintenance | Format | Graphics

Keyword	Default	Value
Description	*None*	
NextComponent	*None*	MachineQueue
FirstArrivalTime	0.0 min	
InterArrivalTime	0.01666666666...	
EntitiesPerArrival	1.0	
PrototypeEntity	*None*	Part
BaseName	*Generator Name*	
MaxNumber	*Infinity*	

The following inputs were set above using the 'Create Entity Links' button. Any discrepancies can be corrected by revising the values directly in the Input Editor.

Table 3-4 Object Connection Inputs

Object	Keyword	Value
GenerateParts	PrototypeEntity	Part
GenerateParts	NextComponent	MachineQueue
MachinePart	WaitQueue	MachineQueue
MachinePart	NextComponent	InspectQueue
InspectPart	WaitQueue	InspectQueue
InspectPart	NextComponent	PassedInspection
PassedInspection	NextComponentList	GoodPartFinished

The PrototypeEntity input tells the EntityGenerator named 'GenerateParts' to make copies of the SimEntity named 'Part'. The NextComponent input tells each object where to send the generated Parts after it has finished its processing. The WaitQueue input tells the Servers named 'MachinePart' and 'InspectPart' to store their waiting Parts in the Queues named 'MachineQueue' and 'InspectQueue', respectively.

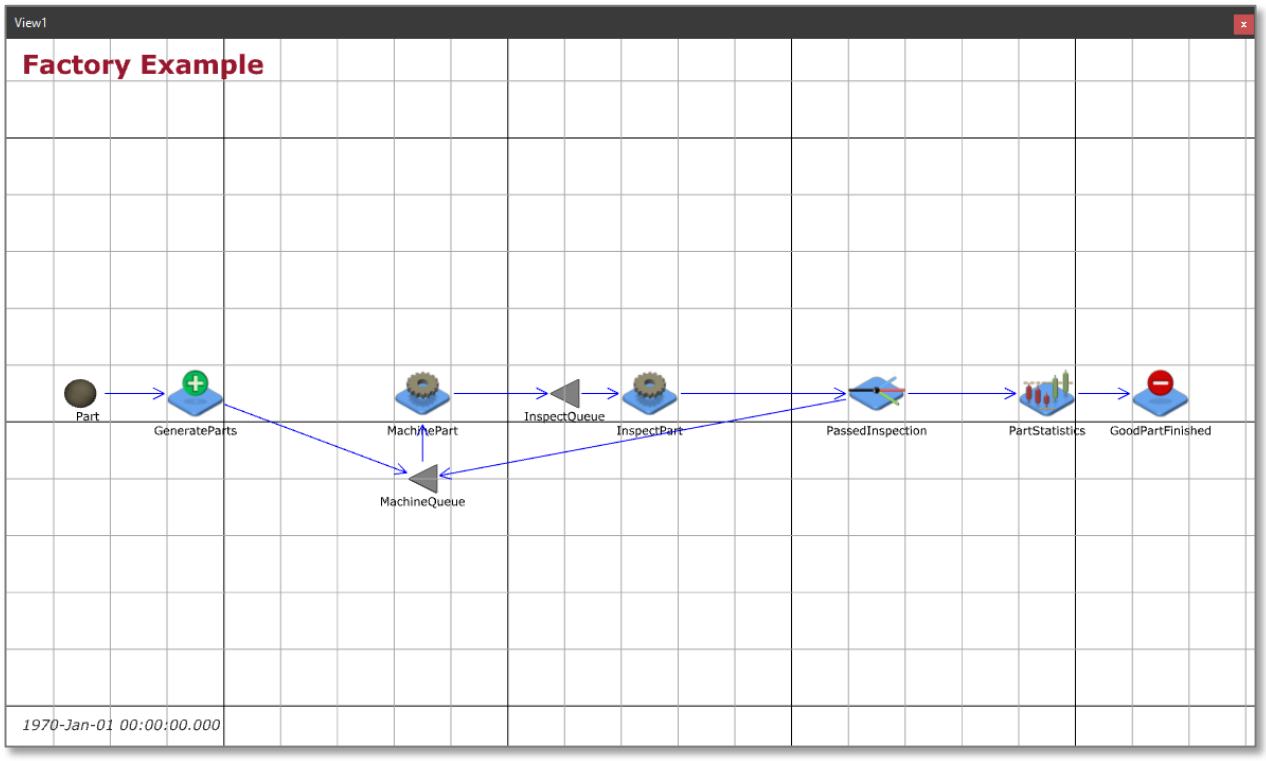
The last step in connecting the model is to set the destination for the parts that failed inspection. This is done using the Input Editor to set the NextComponentList input for the PassedInspection object as follows.

Table 3-5 Rework Connection

Object	Keyword	Value
PassedInspection	NextComponentList	PartStatistics MachineQueue

This input gives the Branch object named 'PassedInspection' the choice of two destinations for the entities it processes. The model should now appear as follows.

Figure 3-7 Screenshot of Step 3



Save your work in progress by clicking the 'Save' button .

3.1.3 Adding Probability Distributions

The factory model uses the following probability distribution objects to generate the following durations and inspection results:

- Exponential distribution for the inter-arrival times for the arrival of unfinished parts
- Uniform distributions for the duration of the machining and inspection operations
- Discrete distribution to choose whether or not a part has passed inspection

Open the Probability Distributions palette in the Model Builder and drag and drop these objects as follows.

Table 3-6 Probability Distributions

Model Builder Palette	Object Type	Name
Probability Distributions	ExponentialDistribution	GenerateDistribution
Probability Distributions	UniformDistribution	MachineDistribution
Probability Distributions	UniformDistribution	InspectDistribution
Probability Distributions	DiscreteDistribution	PassedDistribution

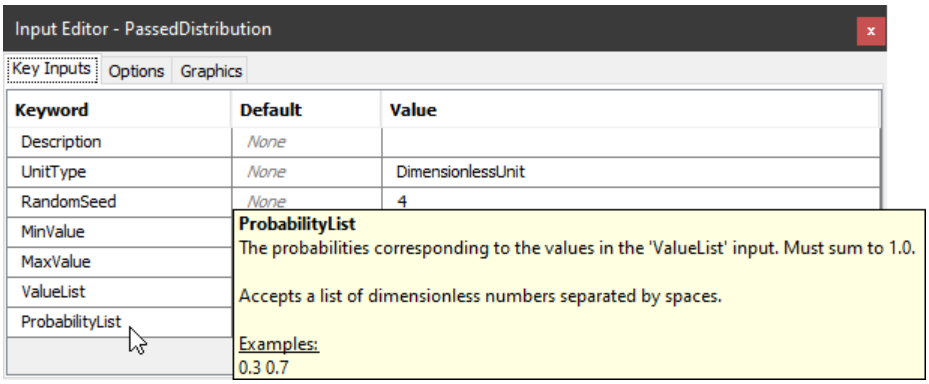
Now use the Input Editor to enter the following inputs for the distribution objects.

Table 3-7 Probability Distribution Inputs

Object	Keyword	Value
GenerateDistribution	UnitType	TimeUnit
GenerateDistribution	Mean	1 min
MachineDistribution	UnitType	TimeUnit
MachineDistribution	MinValue	0.65 min
MachineDistribution	MaxValue	0.70 min
InspectDistribution	UnitType	TimeUnit
InspectDistribution	MinValue	0.75 min
InspectDistribution	MaxValue	0.80 min
PassedDistribution	UnitType	DimensionlessUnit
PassedDistribution	ValueList	1 2
PassedDistribution	ProbabilityList	0.9 0.1

These inputs follow from the problem definition given at the start of this section. You can verify the meaning of each input using the pop-up information given when the mouse is hovered over a keyword name in the Input Editor. For example, the following pop-up appears for the ProbabilityList keyword for the PassedDistribution object.

Figure 3-8 Keyword Description Pop-up



The last step is to use these distributions in the appropriate places in the model by setting the following inputs using the Input Editor. These inputs can be set by clicking the keyword's dropdown menu button in the Input Editor and selecting the entry from a list of possibilities. The dropdown menu button appears when the entry in Value column is clicked.

Table 3-8 Using the Probability Distributions

Object	Keyword	Value
GenerateParts	InterArrivalTime	GenerateDistribution
MachinePart	ServiceTime	MachineDistribution
InspectPart	ServiceTime	InspectDistribution
PassedInspection	Choice	PassedDistribution

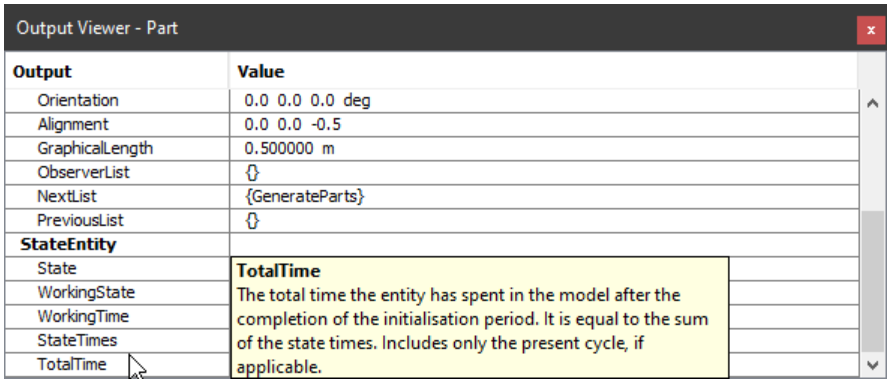
Now that all the objects have been positioned satisfactorily, you can turn off the background grid by clicking the 'Show Grid' button to unselect it.

Save your work in progress by clicking the 'Save' button.

3.1.4 Collecting Statistics

One of the objectives of the model is to determine the average time in system for the finished parts. This task is carried out by the PartStatistics object which collects a sample value from each entity it receives. The Output Viewer shows the values that are available for each object. Clicking on the Part object and examining the pop-up descriptions for each output shows that the TotalTime output provides the needed value.

Figure 3-9 Output Description Pop-up



Enter the following inputs to the PartStatistics object to collect the TotalTime value for each Part object that is received.

Table 3-9 Collecting the Values for TotalTime

Object	Keyword	Value
PartStatistics	UnitType	TimeUnit
PartStatistics	SampleValue	`this.obj.TotalTime`

The UnitType input is required because the TotalTime output returns a value with the units of time. The input `this.obj.TotalTime` is an example of an expression. It is interpreted as follows:

- `this` – the PartStatistics object itself
- `this.obj` – the output obj for the PartStatistics object, which returns the entity that the PartStatistics object has received.

- `this.obj.TotalTime` – the output TotalTime for the entity that the PartStatistics object has received.

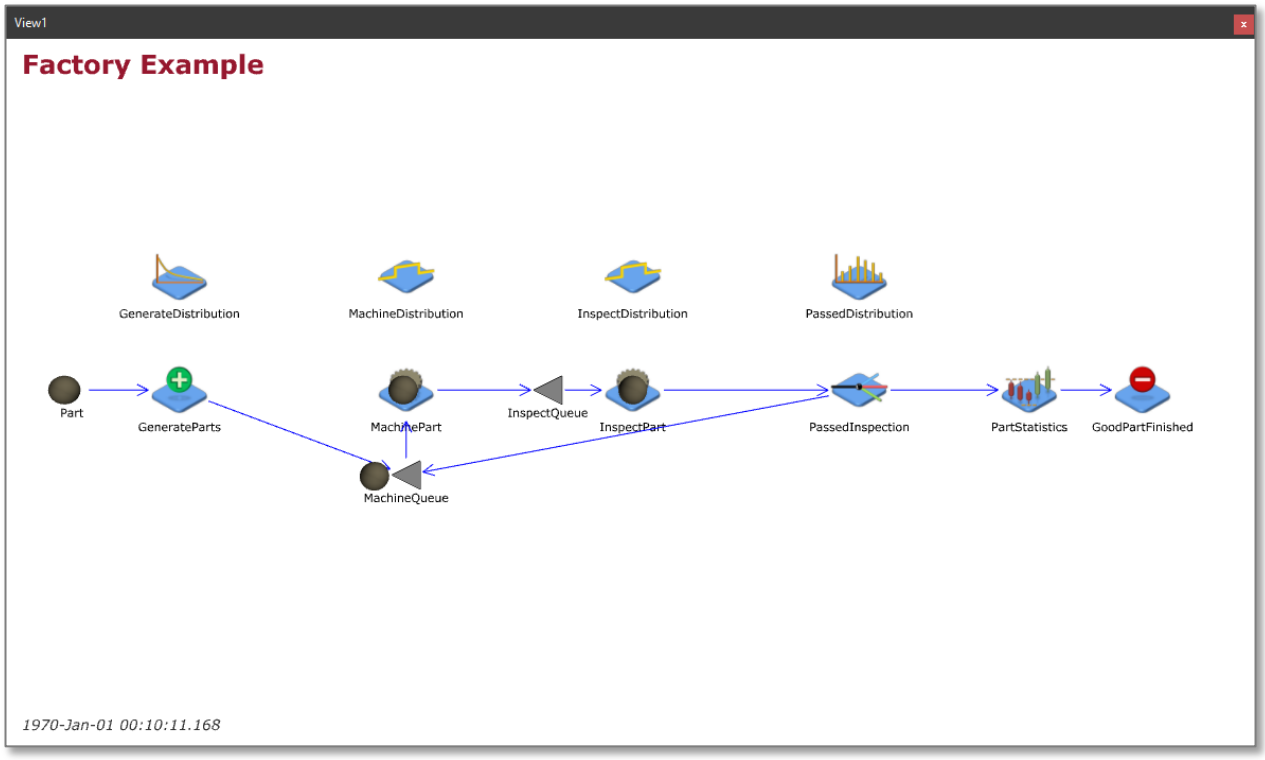
JaamSim has a sophisticated expression system that can manipulate numbers with or without units, objects, strings, arrays, hash maps, and lambda functions.

Save your work in progress by clicking the 'Save' button .

3.1.5 Running the Model

Press the 'Run' button to start the simulation. Note the digital clock display at the bottom left corner of the View window is advancing, but the execution speed is very slow, with only 1 second of simulated time per second of real (wall-clock) time. This occurs because, by default, the 'Real Time' button Real Time in the Control Panel is depressed. This setting restricts the execution speed to the value given by the 'Speed Multiplier' control 1, shown on the right of the 'Real Time' button. Click the up button until the Speed Multiplier display shows 32, indicating that 32 seconds of simulated time are executed in 1 second of wall-clock time. Parts should now be seen making their way through the model, which should appear similar to the following screenshot.

Figure 3-10 Screenshot of Running Model



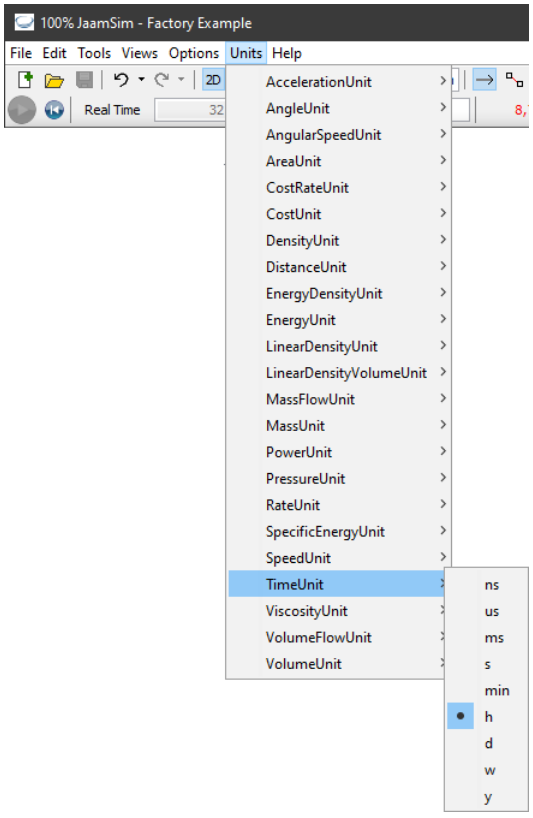
The model can be paused by clicking on the 'Pause' button . Note that this button toggles between 'Run' and 'Pause' each time it is clicked. Useful keyboard shortcuts are the space bar for pausing and resuming the model, and the '>' and '<' keys for increasing and decreasing its execution speed.

Now, click the 'Real Time' button Real Time to allow the model to execute as fast as possible. In this mode, the remainder of the 8760 hours of simulated time (the default duration is one year) is completed in only a few seconds. Clicking on the GoodPartsFinished object and examining the NumberAdded output shows that a total of about 524,467 parts were processed during the run.

The model can be re-initialized to zero simulation time by clicking the 'Reset' button . Note that if the model is re-run, the same 524,467 parts will be processed in exactly the same times.

The average time in system can be obtained by clicking on the PartStatistics object and examining the SampleAverage output, which shows that on average each part required 0.0750045 hours. Since the processing times were entered in minutes, it would convenient to see all the time outputs in this unit. To make this change click on the 'Units' entry in the menu bar and select 'TimeUnit' and 'min'.

Figure 3-11 Setting the Unit for Time



With this change, the Output Viewer shows the SampleAverage output as 4.50027 minutes.

3.1.6 Obtaining Output Reports

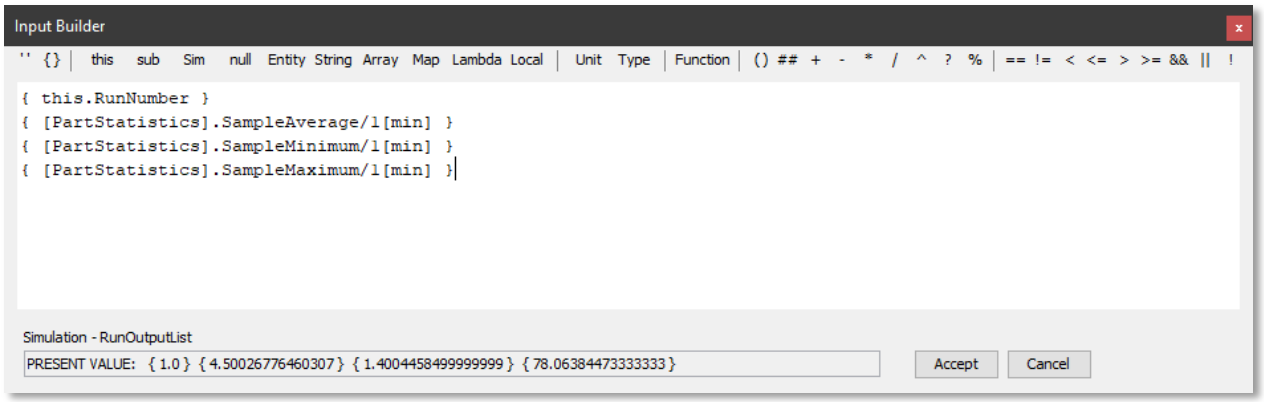
Run control type inputs are handled by the Simulation object, which appears at top of the Object Selector. This object has no graphics so it does not appear as an object in the View window. Two of its keywords are RunDuration and InitializationDuration which determine the duration and the initialization (warm-up) period for each simulation run. These inputs have been left at their default values of 1 year and zero seconds, respectively.

The Simulation object also controls the output reports produced for each simulation run. There are two types of reports. The first option is the main report which includes all the relevant outputs from each object in the model. This report can be obtained by setting the Simulation object's PrintReport keyword to TRUE. The report is given the same name as the input file, but with the extension changed to .rep. It is printed to the same folder as the input file or to the folder specified by the input to the ReportDirectory keyword. This report is useful when testing or verifying a model, but it provides too much information and is inconvenient when multiple runs are to be performed.

The second and more commonly used option is to report a selected number of outputs in tabular format with each run appearing on a separate row. This report is obtained by specifying the desired outputs in the input to the Simulation object's RunOutputList keyword. The input to this keyword is a list of expressions enclosed by curly braces. At the end of each run, the expressions are evaluated and the results printed as a single row of tab-delimited values. Normally, the expressions return either strings or dimensionless numbers to facilitate import into a spreadsheet.

It is best to use the Input Builder to prepare this input. It can be opened by clicking on the Value cell for the RunOutputList keyword in the Input Editor and selecting the dropdown menu button. Although the input text is returned on a single line, the Input Builder allows the user to enter the expressions on separate lines. The following figure shows the Input Builder entry for the Factory Model.

Figure 3-12 Input to the RunOutputList Keyword



This input specifies four columns of outputs in the report:

- Column 1 – simulation run number
- Column 2 – average time in system in minutes
- Column 3 – minimum time in system in minutes
- Column 4 – maximum time in system in minutes

Note how the SampleAverage, SampleMinimum, and SampleMaximum outputs were converted to dimensionless values by dividing the times by 1 minute. If they were not converted, then the value would be recorded as 4.50027[min] or 0.0750045[h], depending on the TimeUnit selection, instead of 4.50027.

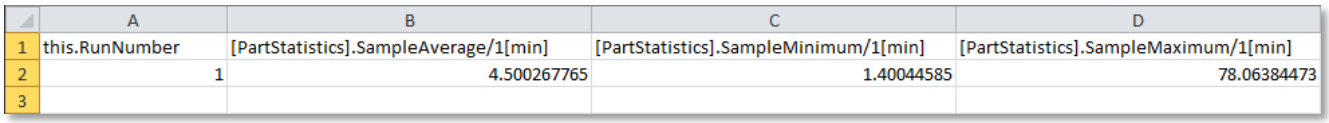
The values for the expressions at the present simulation time are shown at the bottom of the Input Builder. If there is an error in one of the expressions, it is treated as a literal string and returned as such.

The Input Builder includes an auto-completion feature for both object names and output names which helps to reduce input errors.

The Custom Output Report is given the same name as the input file, but with the extension changed to .dat. It is printed to the same directory as the input file or to the folder specified by the input to the ReportDirectory keyword. The file is tab delimited which means that it can be imported directly to

Excel or another spreadsheet program. The following figure shows the output report for the Factory Model after opening it in Excel.

Figure 3-13 Custom Output Report



	A	B	C	D
1	this.RunNumber	[PartStatistics].SampleAverage/1[min]	[PartStatistics].SampleMinimum/1[min]	[PartStatistics].SampleMaximum/1[min]
2	1	4.500267765	1.40044585	78.06384473
3				

The expressions entered to the CustomOutputList are reproduced in the first line of the report as column headings.

3.1.7 Performing Multiple Replications

Multiple replications of the model can be executed with different random seeds using the Simulation object keywords found on the 'Multiple Runs' tab in the Input Editor. Setting the RunIndexDefinitionList keyword to 10 defines a single 'run index' that starts with the value 1 and increments with each run until it reaches 10. It is possible to define multiple run indices, but this is not required for this example. Setting the StartingRunNumber to 1 and the EndingRunNumber to 10 indicates that 10 runs will be performed starting with run number 1.

The last step is to set the GlobalSubstreamSeed input (on the Key Inputs tab) to the expression `this.RunNumber`, so that the global seed for each run is its run number. Without this input, all ten runs would have exactly the same results. These inputs are summarized in the following table.

Table 3-10 Multiple Replications

Object	Keyword	Value
Simulation	RunIndexDefinitionList	`10`
Simulation	StartingRunNumber	`1`
Simulation	EndingRunNumber	`10`
Simulation	GlobalSubstreamSeed	`this.RunNumber`
Simulation	RunDuration	`100000 min`
Simulation	InitializationDuration	`0 min`

The last two entries in the table set the duration of each run to 100,000 minutes and the warm-up period to 0 minutes.

To execute these runs, set the 'Real Time' button Real Time to off, so that the runs execute a fast as possible, click the 'Reset' button to set the simulation time to zero (if necessary), and click the 'Run' button. The ten simulation runs will be completed automatically, one after another.

The custom output report opened in Excel is shown in the following figure.

Figure 3-14 Custom Output Report – 10 Replications

	A	B	C	D
1	this.RunNumber	[PartStatistics].SampleAverage/1[min]	[PartStatistics].SampleMinimum/1[min]	[PartStatistics].SampleMaximum/1[min]
2	1	4.708511346	1.400912467	70.80303455
3	2	4.463609171	1.400644817	65.96299783
4	3	4.702000828	1.40032885	69.05701017
5	4	4.352200762	1.40019555	54.5309457
6	5	4.424242865	1.400363733	73.58991478
7	6	4.614879786	1.40114495	78.32881068
8	7	4.582902384	1.400896867	63.78610323
9	8	4.77549793	1.400371233	75.5928532
10	9	4.49451842	1.400738867	88.873011
11	10	4.504425088	1.400307733	67.93048023
12	Average	4.562278858		
13	Standard Deviation	0.137607217		
14	SD of the Mean	0.043515223		

The cells shown in yellow were added in Excel to show the mean time in system for the ten replications and to the estimate standard deviation for the mean value. The results show that the parts spend an average of 4.56 minutes in the system and that the two standard deviation uncertainty in this value is +/- 0.09 minutes (i.e. 2 x 0.0435 minutes).

3.1.8 Obtaining the Times in each State

A powerful feature of JaamSim is the ability to assign states to an object and to track the total time the object spends in each state. States such as `"Idle"` and `"Working"` are assigned automatically to a permanent object such as a Server. A generated object such as SimEntity can be assigned states using the StateAssignment keyword, which is available for the objects in the Process Flow palette such as Server, Queue, etc. The specified state is assigned to the SimEntity when it is received by the process flow object. The input to the StateAssignment can be any alphanumeric name chosen by the user provided that it does not include any spaces or symbols.

The state times recorded by each object can be accessed using its StateTimes output, which returns a hash map of state times. For example, the total time spent by Server1 in its `"Idle"` state is returned by the expression `'[Server1].StateTimes("Idle")'`. This output is sufficient for permanent objects, but for generated objects such as SimEntity that are destroyed before the end of the simulation run, it is necessary to aggregate the total time spent by all the SimEntities during the run. This function is provided by the same Statistics object that was used to collect the time in system data in Section 3.1.4. It is activated by setting the RecordEntityStateTimes keyword to `TRUE`.

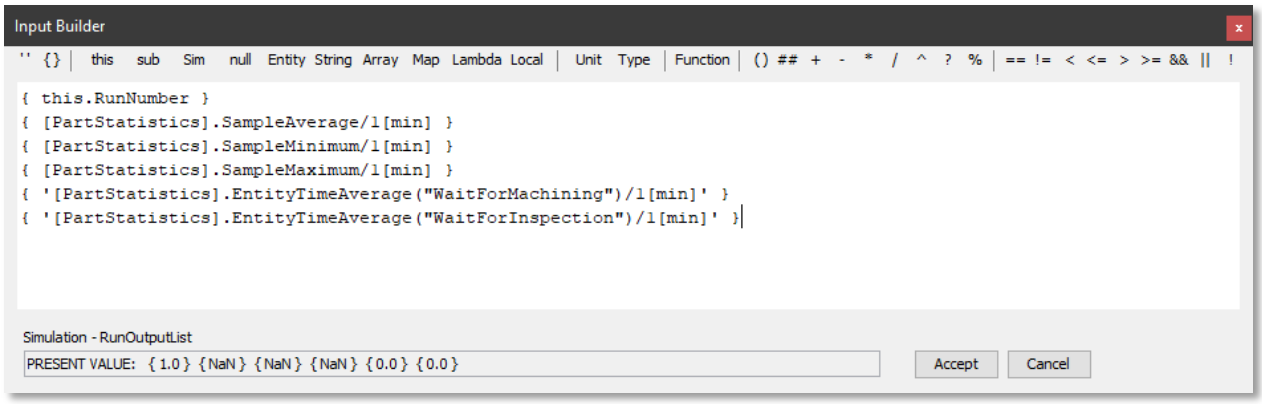
The following table summarizes the additional inputs required to collect time in state data for the simulated parts.

Table 3-11 Times in States Inputs

Object	Keyword	Value
MachineQueue	StateAssignment	`WaitForMachining`
MachinePart	StateAssignment	`Machining`
InspectQueue	StateAssignment	`WaitForInspection`
InspectPart	StateAssignment	`Inspection`
PartStatistics	RecordEntityStateTimes	`TRUE`

The aggregate average times for the `"WaitForMachining"` and `"WaitForInspection"` states can be added to the custom output report by modifying the input to the CustomOutputList keyword for the Simulation object as follows.

Figure 3-15 Input to the RunOutputList Keyword



The following figure shows the resulting custom output report opened in Excel.

Figure 3-16 Custom Output Report – Times in States

	A	B	C	D	E	F
1	this.RunNumber	[PartStatistics].SampleAverage/1[min]	[PartStatistics].SampleMinimum/1[min]	[PartStatistics].SampleMaximum/1[min]	[PartStatistics].EntityTimeAverage("WaitForMachining")/1[min]	[PartStatistics].EntityTimeAverage("WaitForInspection")/1[min]
2	1	4.708511346	1.400912467	70.80303455	1.180470258	1.916466271
3	2	4.463609171	1.400644817	65.96299783	1.1432122	1.70827747
4	3	4.702000828	1.40032885	69.05701017	1.194668588	1.894897089
5	4	4.352200762	1.40019555	54.5309457	1.133049593	1.608939814
6	5	4.424242865	1.400363733	73.58991478	1.132869236	1.682566709
7	6	4.614879786	1.40114495	78.32881068	1.164721462	1.839396845
8	7	4.582902384	1.400896867	63.78610323	1.181788224	1.787244305
9	8	4.77549793	1.400371233	75.5928532	1.209468932	1.953377583
10	9	4.49451842	1.400738867	88.873011	1.151815423	1.732734899
11	10	4.504425088	1.400307733	67.93048023	1.159046087	1.734881433
12	Average	4.562278858			1.165111	1.785878242
13	Standard Deviation	0.137607217			0.02606838	0.112194141
14	SD of the Mean	0.043515223			0.008243545	0.035478903

3.1.9 Reading Model Inputs from a Data File

It is easier to use a model when its inputs are entered in a data file. This can be accomplished using a FileToMatrix object to read the data file and to organize the inputs for use in the model. The data file is prepared in Excel and saved as a tab-delimited text file. Once it is saved in this format, it can be re-opened in Excel by right-clicking on the text file and selecting Open with > Microsoft Excel.

The following figure shows the format of the input file.

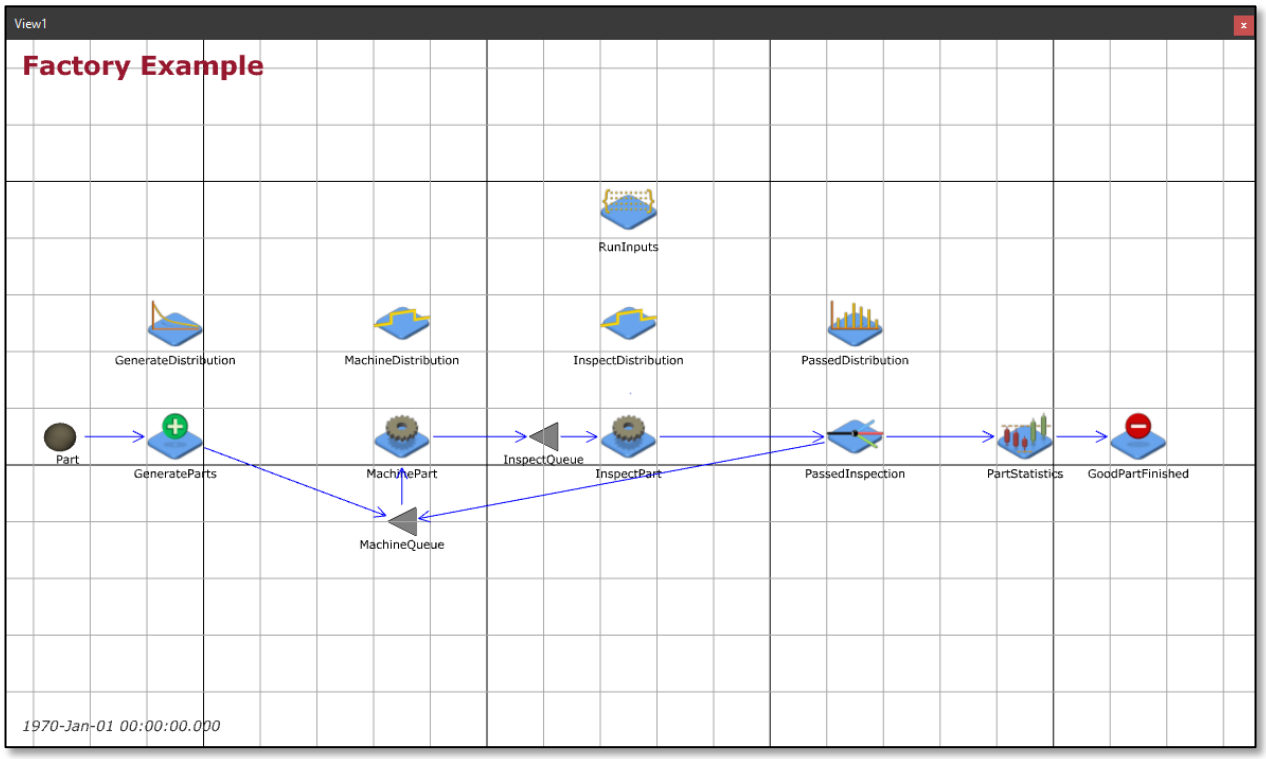
Figure 3-17 Input File

	A	B	C	D	E
1	#				
2	# FACTORY MODEL				
3	#				
4	#	Machining Time	Machining Time	Inspection Time	Inspection Time
5	# Avg. IAT	Minimum Duration	Maximum Duration	Minimum Duration	Maximum Duration
6	# (minutes)	(minutes)	(minutes)	(minutes)	(minutes)
7	1.00	0.65	0.70	0.75	0.80

The first six lines beginning with a hash symbol (#) are comments that document how the file is organised. Note that the data values in the row 7 are entered without units to be compatible with Excel.

The following figure shows the revised Factory model:

Figure 3-18 Factory Model with Data File



The only new element is the FileToMatrix object 'RunInputs'.

Table 3-12 Data File Objects to Create

Model Builder Palette	Object Type	Name
Basic Objects	FileToMatrix	RunInputs

After dragging and dropping the FileToMatrix object and renaming it, the first step is to set its `DataFile` keyword to the input file. At this point, you could use its Value output directly in the inputs to the various objects. For example, the Mean keyword for the GenerateDistribution could be set to the expression `'[RunInputs].Value(1)(1) * 1[min]'`. The trouble with this approach is that the input to the GenerateDistribution depends on the way the input file is organized.

A better approach is to define a series of custom outputs for RunInputs that can be used in the inputs to other objects. A custom output can be used like an attribute, but has the significant difference that its present value is determined by an expression that is evaluated on demand. In contrast, the value for an attribute is set at the start of the simulation and can only be changed by an Assign object.

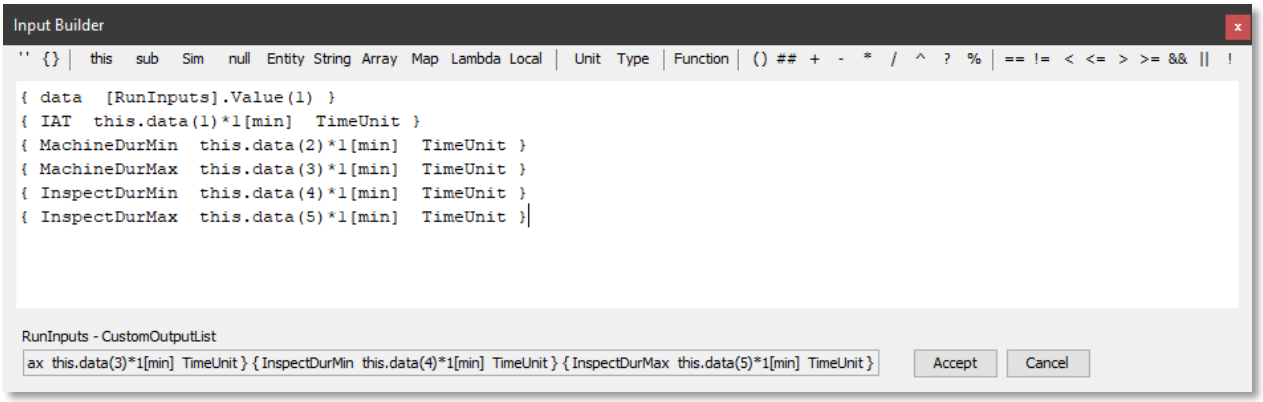
Custom outputs for an object are defined by its CustomOutputList keyword. It is best to use the Input Builder to construct this input. The first custom output to define is the data output, which returns an array of the parameters in the file. It is defined by the following input to the CustomOutputList keyword:

```
{ data  '[RunInputs].Value(1) }
```

This input consists of the name of the output (data) followed by an expression that returns its present value. In this case, the present value is the array `{1.0, 0.65, 0.7, 0.75, 0.8}` that is taken from the input file.

The custom output data must be defined first because it is used to define the other custom outputs. The following figure shows the entry in the Input Builder used to define the remaining custom outputs:

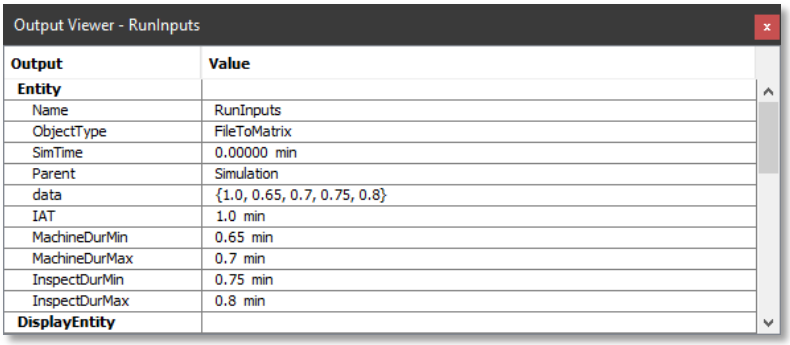
Figure 3-19 Custom Output Definitions



Note that the dimensionless values from the input file are multiplied by `1[min]` to give the custom outputs the units of minutes.

The present value for a custom output is shown in the Output Viewer under the 'Entity' heading, just above any attribute values. The following figure shows the values for the custom outputs displayed by the Output Viewer:

Figure 3-20 Custom Output Values in the Output Viewer



The last step is to use the custom outputs in the inputs to the model's objects. The following table shows the complete set of inputs that are changed.

Table 3-13 Data File Inputs

Object	Keyword	Value
RunInputs	DataFile	`'Factory Model Inputs.txt'`

RunInputs	CustomOutputList	`{ data [RunInputs].Value(1) }`
		`{ IAT this.data(1)*1[min] TimeUnit }`
		`{ MachineDurMin this.data(2)*1[min] TimeUnit }`
		`{ MachineDurMax this.data(3)*1[min] TimeUnit }`
		`{ InspectDurMin this.data(4)*1[min] TimeUnit }`
		`{ InspectDurMax this.data(5)*1[min] TimeUnit }`

GenerateDistribution	Mean	`[RunInputs].IAT`
MachineDistribution	MinValue	`[RunInputs].MachineDurMin`
MachineDistribution	MaxValue	`[RunInputs].MachineDurMax`
InspectDistribution	MinValue	`[RunInputs].InspectDurMin`
InspectDistribution	MaxValue	`[RunInputs].InspectDurMax`

With these changes the values entering in the input file are passed to the objects in the model. There should be no changes to the output report when the model is executed.

3.1.10 Running Multiple Scenarios

Once the model has been set up to read its input data from a file, it is simple to perform a series of scenarios, each with a specified number of replications. The scenarios are entered as rows of input parameters in the data file. For this example, four scenarios will be executed with the following inter-arrival times for the parts: 1.2 minutes, 1.1 minutes, 1.0 minutes, and 0.9 minutes. The following figure shows the input file for these scenarios:

Figure 3-21 Input File for Scenarios

	A	B	C	D	E
1	#				
2	# FACTORY MODEL				
3	#				
4	#	Machining Time	Machining Time	Inspection Time	Inspection Time
5	# Avg. IAT	Minimum Duration	Maximum Duration	Minimum Duration	Maximum Duration
6	# (minutes)	(minutes)	(minutes)	(minutes)	(minutes)
7	1.2	0.65	0.7	0.75	0.8
8	1.1	0.65	0.7	0.75	0.8
9	1.0	0.65	0.7	0.75	0.8
10	0.9	0.65	0.7	0.75	0.8

To run the scenarios it is necessary to modify the inputs for the Simulation object to perform 40 simulation runs, i.e. 4 scenarios x 10 replications per scenario. This is done by defining two 'run indices' using the RunIndexDefinitionList keyword. The input `4 10` defines two run indices: the first with the range 1 to 4 (scenario number) and the second with the range 1 to 10 (replication number). With this input, the first ten runs are the replications for scenario 1, the second ten runs are the replications for scenario 2, and so on.

The RunIndex output for the Simulation object returns an array containing the run index values for the present simulation run. For example, during run 23 the RunIndex output returns `{ 2, 3 }`, i.e. the third replication for the second scenario.

The RunIndex output makes it easy to prepare the inputs needed to perform the scenarios:

- First, set the GlobalSubstreamSeed input for the Simulation object to the expression `this.RunIndex(2)`, which returns the replication number.
- Second set the custom output data for the RunInputs object to the expression `[RunInputs].Value([Simulation].RunIndex(1))`, which returns the array of input parameters for the row in the matrix given by the scenario number.

No other changes to the RunInputs object are required. The following table lists all the input changes that were required to model the scenarios.

Table 3-14 Scenarios Inputs

Object	Keyword	Value
Simulation	RunIndexDefinitionList	`4 10`
Simulation	EndingRunNumber	`40`
Simulation	GlobalSubstreamSeed	`this.RunIndex(2)`

RunInputs	CustomOutputList	`{data [RunInputs].Value([Simulation].RunIndex(1))}`
		`{ IAT this.data(1)*1[min] TimeUnit }`
		`{ MachineDurMin this.data(2)*1[min] TimeUnit }`
		`{ MachineDurMax this.data(3)*1[min] TimeUnit }`
		`{ InspectDurMin this.data(4)*1[min] TimeUnit }`
		`{ InspectDurMax this.data(5)*1[min] TimeUnit }`

With these changes, all 40 simulations runs will be performed one after another when the Run button is clicked.

3.2 Graphical Enhancements to the Basic Example

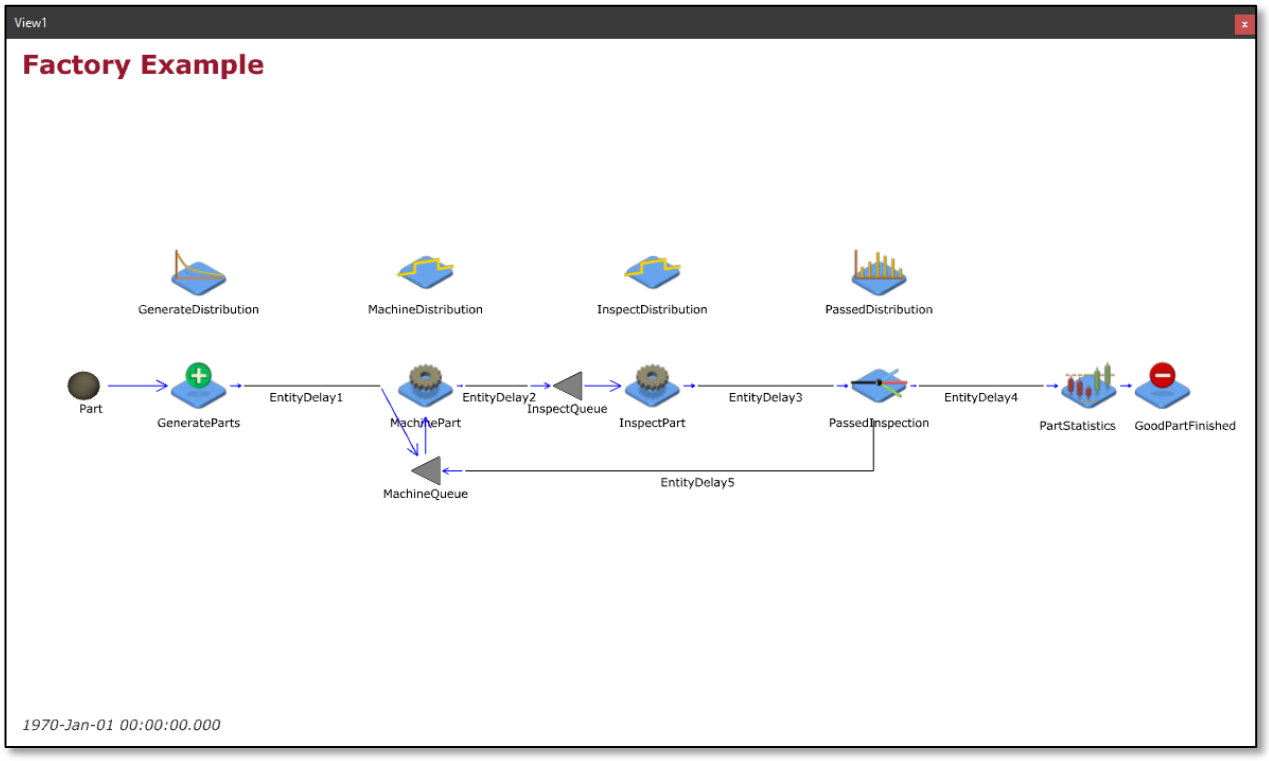
The following sections will guide you through some of the enhancements that can be made to the factory model.

3.2.1 Adding Travel Times

Most systems require travel times between the various processes. Even if these times are negligible, adding them to the model can make it much easier to follow the individual entities as they move from process to process.

A fixed travel time can be modelled by either an EntityDelay or an EntityConveyor object. For the factory model, we will use EntityDelay objects. The following figure shows the finished model.

Figure 3-22 Factory Model with Travel Times



First, drag and drop the five EntityDelay objects into the model.

Table 3-15 Travel Time Objects to Create

Model Builder Palette	Object Type	Name
Process Flow	EntityDelay	EntityDelay1
Process Flow	EntityDelay	EntityDelay2
Process Flow	EntityDelay	EntityDelay3
Process Flow	EntityDelay	EntityDelay4
Process Flow	EntityDelay	EntityDelay5

An EntityDelay is displayed as a polyline type object that contains nodes which can be dragged individually to achieve the desired path. Drag the end nodes into the positions shown above, taking care that the direction of travel for EntityDelay5 is correct. Note that when a polyline object is selected, the first node is blue, the last node is yellow, and intermediate nodes are green. The intermediate node required for EntityDelay5 can be added by right clicking on the line and selecting 'Add Node' from the context menu.

Once the EntityDelay objects are positioned correctly, you can drag their labels into better positions.

The next step is to use the 'Create Entity Links' button  to connect the objects into the correct sequence. This can be done in the normal way for all the objects except the Branch object. For that object it is best to use the Input Editor to set its NextComponentList input.

Table 3-16 Travel Time Connection Inputs

Object	Keyword	Value
GenerateParts	NextComponent	EntityDelay1
EntityDelay1	NextComponent	MachineQueue
MachinePart	NextComponent	EntityDelay2
EntityDelay2	NextComponent	InspectQueue
InspectPart	NextComponent	EntityDelay3
EntityDelay3	NextComponent	PassedInspection
PassedInspection	NextComponentList	EntityDelay4 EntityDelay5
EntityDelay4	NextComponent	GoodPartFinished
EntityDelay5	NextComponent	MachineQueue

The last step is to assign a value to the Duration inputs to the EntityDelay objects. Assign a value of 5 seconds to these inputs.

Table 3-17 Travel Time Inputs

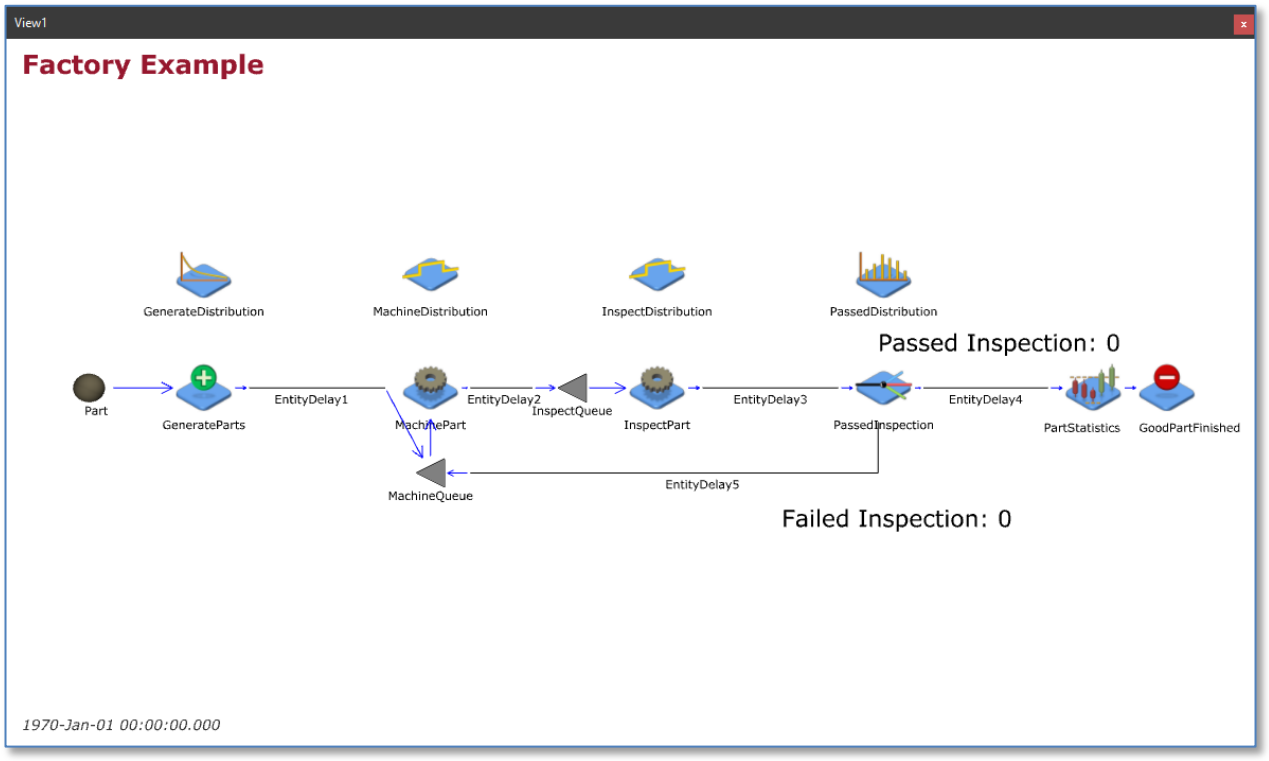
Object	Keyword	Value
EntityGenerator	Duration	5 s
EntityDelay2	Duration	5 s
EntityDelay3	Duration	5 s
EntityDelay4	Duration	5 s
EntityDelay5	Duration	5 s

Now run the model with the 'Real Time' button Real Time pressed and with 'Speed Multiplier' set to 32. You will see the parts object move smoothly from process to process.

3.2.2 Displaying the Number of Parts

A useful enhancement to the model is to display the number of parts that have passed and failed inspection as the simulation runs. These displays can be added using Text objects. The following figure shows the finished model.

Figure 3-23 Factory Model with Number of Parts Displays



Start by dragging and dropping a Text object into the position for the 'Failed Inspection' display. The 'Passed Inspection' display will be created later by copying and modifying the first Text object.

Table 3-18 Number of Parts Objects to Create

Model Builder Palette	Object Type	Name
Graphical Objects	Text	Text1

With Text1 selected, click the 'Align Left' button to left justify the text at its screen position.

The 'Failed Inspection' display is set by entering the following inputs to Text1.

Table 3-19 Number of Parts Inputs – Failed Inspection

Object	Keyword	Value
Text1	Format	`'Failed Inspection: %,.0f'`
Text1	DataSource	`[EntityDelay5].NumberAdded`

The Format input uses the Java format string `%,.0f` to specify that a floating point number is to be displayed with zero decimal digits and to include the comma separator. The DataSource input specifies the value that will be displayed. Use the Input Builder to prepare this input.

Now run the model to confirm that the new Text objects works correctly. Once you are satisfied that it is correct, right click and chose 'Copy' from the context menu. Then, right click where the 'Passed Inspection' display is to appear and select 'Paste' from the context menu. Alternatively, you can use the 'Copy' and 'Paste' buttons or the keyboard shortcuts Ctrl+C and Ctrl+V.

The last step is to edit the following inputs for the copied Text object.

Table 3-20 Number of Parts Inputs – Passed Inspection

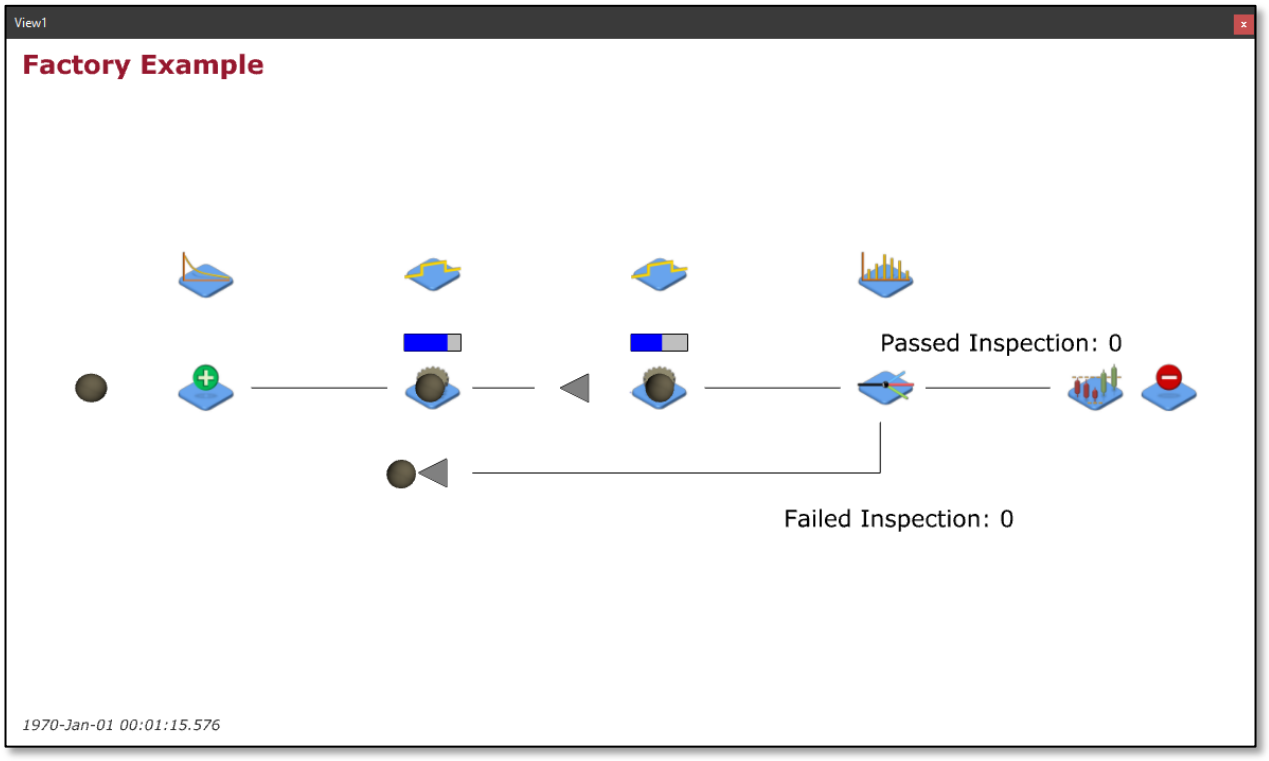
Object	Keyword	Value
Text1_Copy1	Format	`'Passed Inspection: %,.0f'`
Text1_Copy1	DataSource	`[EntityDelay4].NumberAdded`

The model should now display the number of parts that passed and failed inspection.

3.2.3 Adding Bar Gauges to the Servers

Often it is helpful to show how close a Server is to completing its work on the part it is processing. The following figure shows the Factory Model will BarGauge objects attached to the MachinePart and InspectPart servers.

Figure 3-24 Factory Model with Progress Bar Gauges



First, drag and drop a BarGauge object from the Graphics Objects palette to a position above the MachineParts server.

Table 3-21 Bar Gauge Objects to Create

Model Builder Palette	Object Type	Name
Graphical Objects	BarGauge	BarGauge1

With the BarGauge selected, use the mouse to rotate the BarGauge 90 degrees to a horizontal position and to reduce its width.

The quantity displayed by the BarGauge is set using its DataSource input. The following table shows the input changes for the BarGauge, including the inputs for Size and Orientation that were made using the mouse.

Table 3-22 Bar Gauge Inputs

Object	Keyword	Value
BarGauge1	DataSource	`[MachinePart].FractionCompleted`
BarGauge1	Size	`0.3 1 0 m`
BarGauge1	Orientation	`0.0 0.0 -90.0 deg`

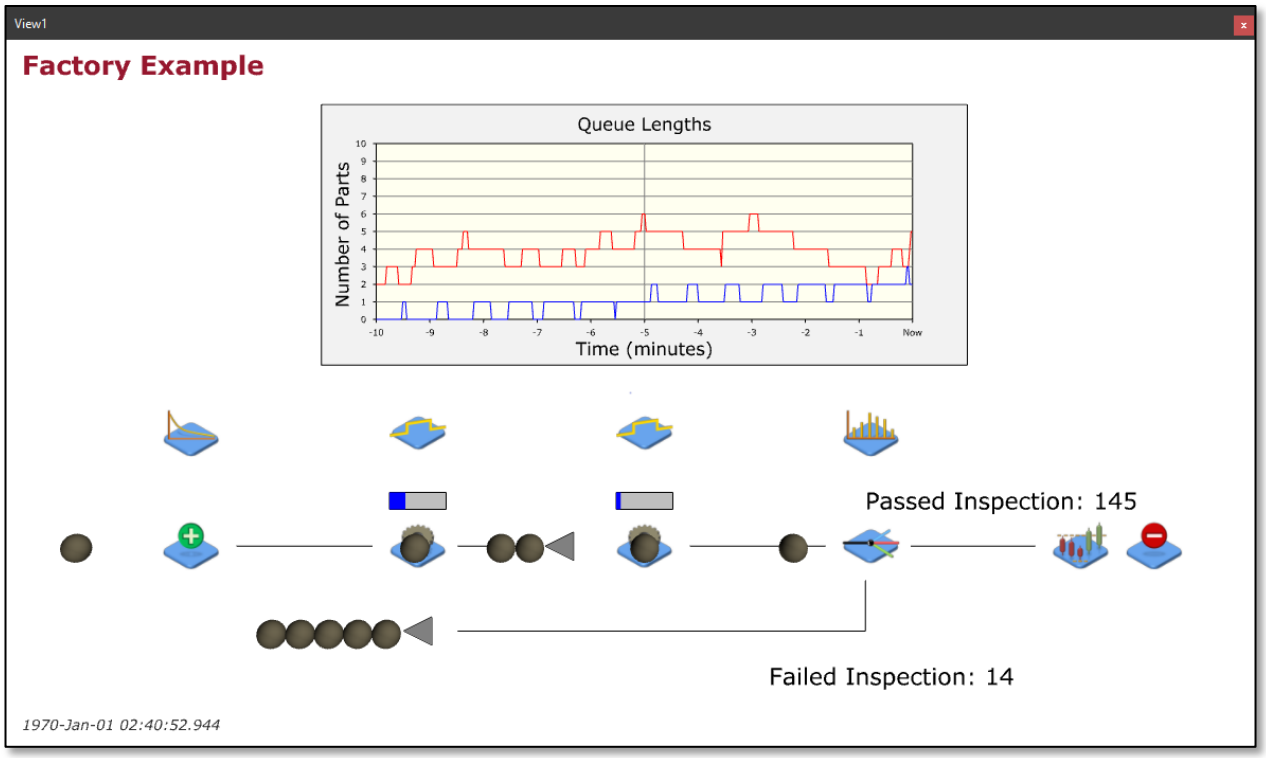
Run the model to confirm that the BarGauge operates correctly.

Now make a copy of the BarGauge and position it next to the InspectParts server. Revise its DataSource input to refer to InspectPart instead of MachinePart.

3.2.4 Adding a Graph

A Graph object can be used to show how a system performs over time. The following figure shows a graph that displays the number of parts in each queue. The red line shows the number of parts waiting for machining while the blue line shows the number waiting for inspection. The graph shows how the queue lengths varied over the previous 10 minutes. The right hand side of the graph shows the present queue lengths at the x-axis label of "Now". The left hand side of the graph shows the queue lengths at time Now – 10 minutes.

Figure 3-25 Factory Model with Graph



The graph is added to the Factory model by dragging and dropping a Graph object from the Graphical Objects palette to the location shown in the above figure.

Table 3-23 Graph Objects to Create

Model Builder Palette	Object Type	Name
Graphical Objects	Graph	Graph1

Use the mouse to resize the graph to something similar to the one shown in the figure.

Now set the following inputs for the graph which appear on the Key Inputs tab in the Input Editor:

- The UnitType input specifies the units for the quantities to be graphed and must be entered before any other input. All the quantities graphed on primary y-axis must be of this unit type. If required, a quantity with a different unit type can be graphed on a secondary y-axis using the SecondaryUnitType and SecondaryDataSource inputs. For this example, the graph shows the queue lengths, which are dimensionless quantities. Therefore, the UnitType input is set to `DimensionlessUnit`.

- The DataSource input specifies the quantities to be graphed, given by a list of expressions. Each expression is enclosed by curly braces and returns a number with the specified unit type. The input can be entered in a single line directly into the Input Editor field, but it is much easier to use the Input Builder which allows the expressions to be entered on separate lines and provides auto-completion when entering the object and output names.

- The LineColours input specifies the colours for the two lines. Each colour is enclosed by curly braces to allow the option of specifying RGB values. Normally, it is best to use one of the standard web colours (X11) which can be entered by name.

- The NumberOfPoints input specifies the number times the expression is sampled over the duration of the graph. The default value of 100 corresponds to a sampling interval of 10 minutes / 100 = 0.1 minute. This value is adequate, but the graph is easier to interpret if the number of points is increased to 400, corresponding to a sampling interval of 0.025 minutes. Note that specifying an excessive number of points can limit the execution speed of a model.

Table 3-24 Bar Gauge Inputs – Key Inputs Tab

Object	Keyword	Value
Graph1	Title	`'Queue Lengths'`
Graph1	NumberOfPoints	`400`
Graph1	UnitType	`DimensionlessUnit`
Graph1	DataSource	`{[MachineQueue].QueueLength}` `{[InspectQueue].QueueLength}`
Graph1	LineColours	`{ red } { blue }`

The next step is to set the inputs for the graph's x-axis which appear on the X-Axis tab:

- The XAxisStart and XAxisInterval inputs specify the time shown on the left hand side of the x-axis and the interval between tick marks. For this graph, the left hand side of the x-axis corresponds to the present simulation time ("Now") minus 10 minutes and there is 1 minute

between tick marks. Therefore the XAxisStart and XAxisInterval inputs are `-10 min` and `1 min`, respectively. Note that the time on right hand side of the x-axis is automatically zero ("Now").

- The XAxisUnit input specifies the unit in which the x-axis labels are to be shown. The input of `min` indicates that they are to be displayed in minutes instead of the default time unit of hours.

- The XLines input specifies the positions for the vertical grid lines on the graph. The input of `-5 min` indicates a single grid line at the x-axis value of -5 minutes.

Table 3-25 Bar Gauge Inputs – X-Axis Tab

Object	Keyword	Value
Graph1	XAxisTitle	`'Time (minutes)'`
Graph1	XAxisUnit	`min`
Graph1	XAxisStart	`-10 min`
Graph1	XAxisInterval	`1 min`
Graph1	XLines	`-5 min`

The final step is to set the inputs for the graph's y-axis which appear on the Y-Axis tab:

- The YAxisStart and YAxisEnd inputs specify the initial and final values shown on the y-axis. Since the UnitType input was set to `DimensionlessUnit`, these inputs are entered without units. The input of `10` to YAxisEnd indicates that the y-axis ends at 10 parts. The YAxisStart input is left at its default value of zero.

- The YAxisLabelFormat input specifies the Java format used for the y-axis labels. The input `%.0f` indicates a floating point format with zero decimal digits.

- The YAxisUnit input specifies the unit in which the y-axis labels are to be shown. Since the UnitType input was set to `DimensionlessUnit`, there are no units for the y-axis and this input can be left blank.

- The YLines input specifies the positions of the horizontal grid lines on the graph. The input specifies lines at queue lengths of 1 – 9.

Table 3-26 Bar Gauge Inputs – Y-Axis Tab

Object	Keyword	Value
Graph1	YAxisTitle	`'Number of Parts'`
Graph1	YAxisEnd	`10`
Graph1	YAxisLabelFormat	`%.0f`
Graph1	YLines	`1 2 3 4 5 6 7 8 9`

Run the model to confirm that the above inputs were entered correctly.

3.3 Additional Model Features

3.3.1 Variable Arrival Rate

Real systems rarely experience long periods of operation with a constant arrival rate for the incoming customers, parts, etc. More typically, the average arrival rate varies with both the time of day and the day of week. Although it is tempting to use the usual ExponentialDistribution object with a variable input to the Mean keyword, this method is incorrect and often results in dramatically wrong results.

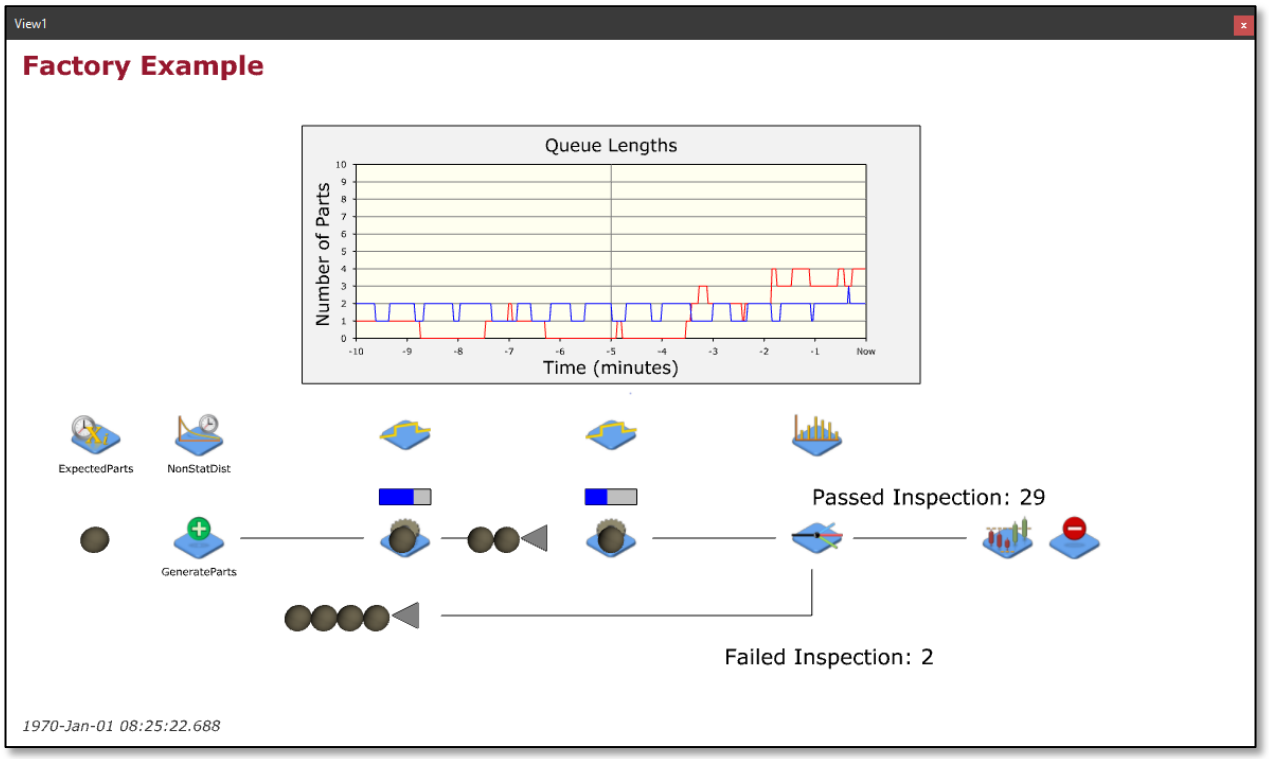
The correct way to model this situation is to use the NonStatExponentialDist object which performs the calculations for the non-stationary exponential distribution, and a TimeSeries object that specifies the total number of arrivals that will occur on average by the each time in the time series.

For the example, assume that parts arrive only between 8am and 5pm each day, and that on average they arrive every 0.8 minutes from 8am to noon and every 1.25 minutes from noon to 5pm. At these rates, an average of 240 / 0.8 = 300 parts arrive between 8am and noon, and an average of 300 / 1.25 = 240 parts arrive between noon and 5pm. On average, a total of 300 + 240 = 540 parts arrive each day by 5pm. The Value input for the TimeSeries is therefore:

```
{0 h 0} {8 h 0} {12 h 300} {17 h 540} {24 h 540}
```

The following figure shows the modified Factory Model in operation.

Figure 3-26 Variable Arrival Rate



The changes required for this version of the model are given in the following tables:

Table 3-27 Variable Arrival Rate Objects to Create

Model Builder Palette	Object Type	Name
Basic Objects	TimeSeries	ExpectedParts
Basic Objects	NonStatExponentialDist	NonStatDist

Table 3-28 Variable Arrival Rate Inputs

Object	Keyword	Value
ExpectedParts	UnitType	`DimensionlessUnit`
ExpectedParts	Value	`{0 h 0} {8 h 0} {12 h 300} {17 h 540} {24 h 540}`
ExpectedParts	CycleTime	`24 h`
NonStatDist	ExpectedArrivals	`ExpectedParts`
GenerateParts	FirstArrivalTime	`NonStatDist`
GenerateParts	InterArrivalTime	`NonStatDist`

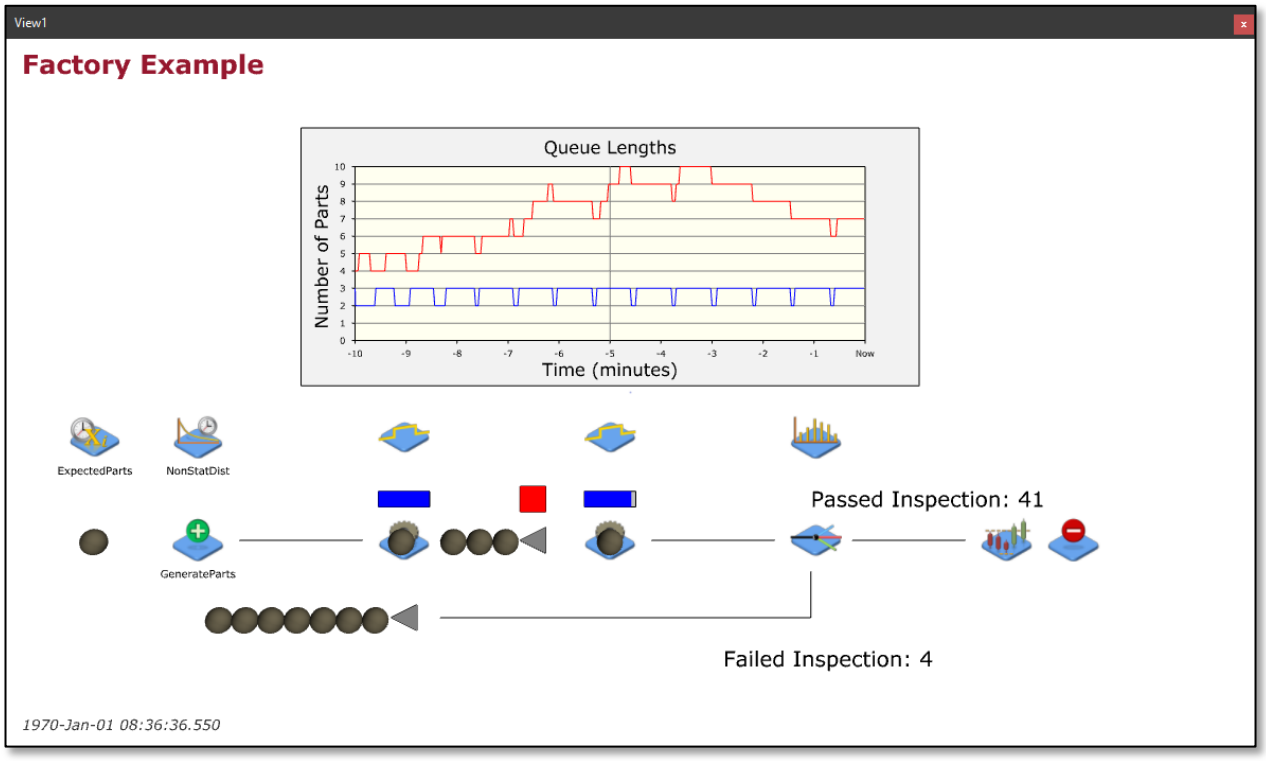
The unused GenerateDistribution object can be deleted once these changes have been made. Note that the CycleTime input instructs the TimeSeries to repeat the average number of arrivals during each 24 hour period.

3.3.2 Queue Length Limit

Another common modelling requirement is to stop upstream processes when a queue reaches a maximum size. For example, the Factory Model can be modified to place a limit of 3 parts on the InspectQueue. JaamSim uses an ExpressionThreshold object to monitor the length of the InspectQueue and the ReleaseThresholdList input for the MachineParts server to stop it from releasing a part to the InspectionQueue when the queue length limit is reached.

The follow figure shows the Factory Model with the InspectionQueue limited to a maximum of three parts.

Figure 3-27 Queue Length Limit



The colour of the ExpressionThreshold changes between green and red to indicate whether it is open or closed.

The changes required for this version of the model are given in the following tables:

Table 3-29 Queue Length Limit Objects to Create

Model Builder Palette	Object Type	Name
Basic Objects	ExpressionThreshold	QueueThreshold

Table 3-30 Queue Length Limit Inputs

Object	Keyword	Value
QueueThreshold	OpenCondition	`'[InspectQueue].QueueLength + [EntityDelay2].NumberInProgress < 3'`
QueueThreshold	InitialOpenValue	`TRUE`
MachinePart	ReleaseThresholdList	`QueueThreshold`

Note that it is necessary for the OpenCondition input to include the number of parts that are in transit between the MachinePart server and the InspectQueue. If EntityDelay2 is replaced by an EntityConveyor, it would be possible to stop both the MachinePart server and the EntityConveyor when the QueueThreshold closes. In that situation, the OpenCondition would exclude the number of parts that are in transit.

The MachinePart server is set to the state `"Stopped"` whenever one or more of its threshold inputs prevent it from working. The amount time spent in this state can be obtained using its StateTimes output, e.g. `[MachinePart].StateTimes("Stopped")`.

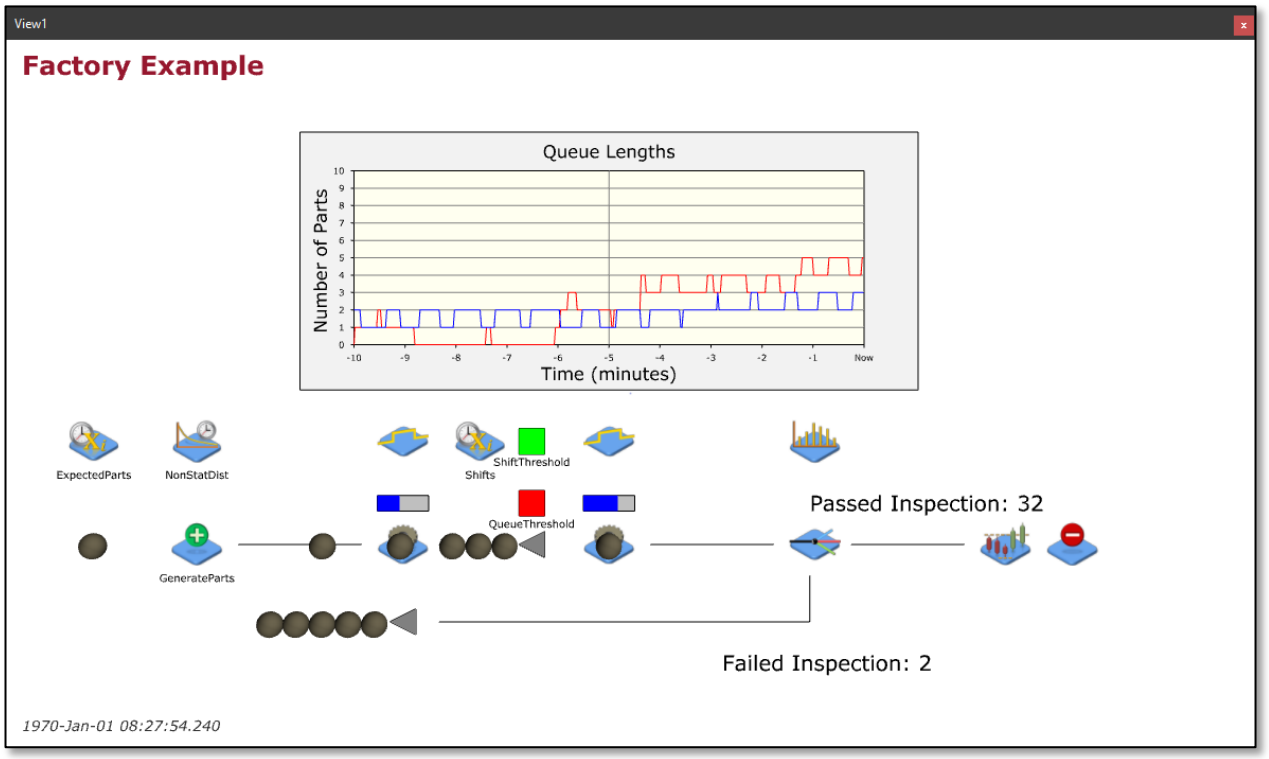
3.3.3 Working Hours

Although parts arrive between 8am and 5pm, the working hours for the operation could start and end at different times. Suppose that the machining and inspection operations work between 8am and 5:15pm, with a break between noon and 12:30pm. The modification can be implemented using a TimeSeries object to specify the working hours, a TimeSeriesThreshold object to detect the closures, and the ImmediateThresholdList input for the MachinePart and InspectPart servers to stop operations in the middle of work, if necessary.

It would be possible to use an ExpressionThreshold instead of a TimeSeriesThreshold to stop the two servers, but it is better to use a TimeSeriesThreshold because it is significantly more efficient. The difference is that a TimeSeriesThreshold can determine exactly when its next state change will occur while an ExpressionThreshold must test its expression frequently to determine whether its state has changed.

The follow figure shows the Factory Model with the shifts described above.

Figure 3-28 Factor Model with Shifts



The changes required for this version of the model are given in the following tables:

Table 3-31 Shifts Objects to Create

Model Builder Palette	Object Type	Name
Basic Objects	TimeSeries	Shifts
Basic Objects	TimeSeriesThreshold	ShiftThreshold

Table 3-32 Shifts Inputs

Object	Keyword	Value
Shifts	UnitType	DimensionlessUnit
Shifts	Value	{0 h 0} {8 h 1} {12 h 0} {12.5 h 1} {17.25 h 0}
Shifts	CycleTime	24 h
ShiftThreshold	UnitType	DimensionlessUnit
ShiftThreshold	TimeSeries	Shifts
ShiftThreshold	MinOpenLimit	0.9
MachinePart	ImmediateThresholdList	ShiftThreshold
InspectPart	ImmediateThresholdList	ShiftThreshold

The states for the MachinePart and InspectPart servers are set to "Stopped" when the ShiftThreshold is closed. This is the same state for MachinePart that is set when the QueueThreshold is closed. The times for the two conditions can be separated by checking the StateTimes output for the TimeSeriesThreshold whose state is set to "Open" and "Closed" as appropriate.

3.3.4 Maintenance and Breakdowns

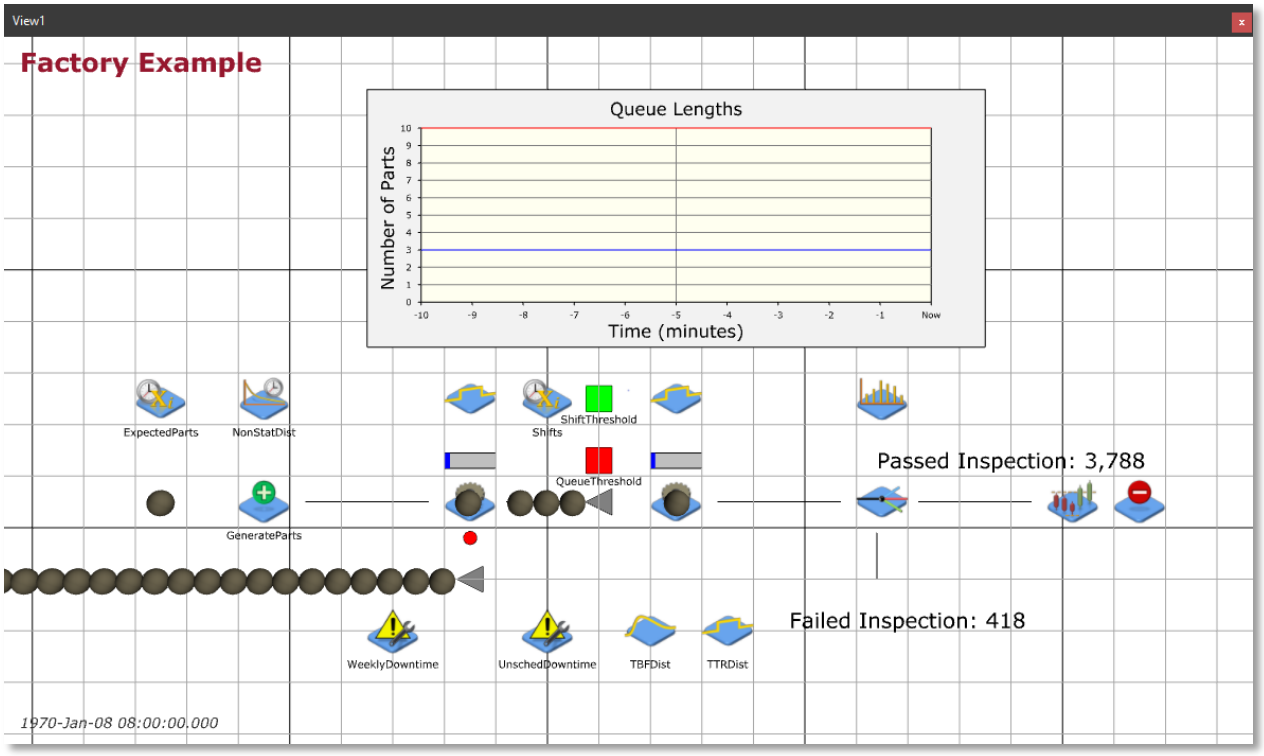
Most equipment requires planned maintenance and experiences breakdowns. Assume that one hour of scheduled maintenance is performed on the machining operation each week starting at 8am. Also assume that the operation is subject to random failures with a mean time between failures (MTTF) of 100 operating hours and a mean time to repair (MTTR) of 1 hour.

The maintenance and breakdowns can be added to the Factory Model using two DowntimeEntities and the ForcedMaintenanceList and ImmediateBreakdownList keywords for the MachinePart server. Probability distributions are used to generate the random times between failures and times to repair. A BooleanIndicator is used to display whether the MachinePart server is available.

Times to repair are assumed to be uniformly distributed between 0.5 hours and 1.5 hours. Times between failures are assumed to follow a Weibull distribution with a shape parameter of 2 and a scale parameter of 113 hours, resulting in a mean value of 100 hours.

The follow figure shows the Factory Model with the maintenance and breakdowns described above.

Figure 3-29 Factory Model with Maintenance and Breakdowns



The changes required for this version of the model are given in the following tables:

Table 3-33 Maintenance and Breakdowns Objects to Create

Model Builder Palette	Object Type	Name
Basic Objects	DowntimeEntity	WeeklyDowntime
Basic Objects	DowntimeEntity	UnschedDowntime
Probability Distributions	WeibullDistribution	TBFDist
Probability Distributions	UniformDistribution	TTRDist
Graphics Objects	BooleanIndicator	BooleanIndicator1

Table 3-34 Maintenance and Breakdowns Inputs

Object	Keyword	Value
WeeklyDowntime	FirstDowntime	`176 h`
WeeklyDowntime	Interval	`1 w`
WeeklyDowntime	Duration	`1 h`
TBFDist	UnitType	`TimeUnit`
TBFDist	Scale	`113 h`
TBFDist	Shape	`2`
TTRDist	UnitType	`TimeUnit`
TTRDist	MinValue	`0.5 h`
TTRDist	MaxValue	`1.5 h`
UnschedDowntime	IntervalWorkingEntity	`MachinePart`
UnschedDowntime	Interval	`TBFDist`
UnschedDowntime	Duration	`TTRDist`
MachinePart	ForcedMaintenanceList	`WeeklyDowntime`
MachinePart	ImmediateBreakdownList	`UnschedDowntime`
BooleanIndicator1	DataSource	`'[WeeklyDowntime].WorkingState && [UnschedDowntime].WorkingState'`

You can verify that the maintenance inputs work correctly by setting the Pause Time input on the Control Panel to 176 h (the time of the first maintenance event) and running the model to that simulation time. The model will pause before the events at 176 hours have been performed, so it is necessary to resume the model to confirm that the colour of the BooleanIndicator turns red at this time and that it returns to green at 177 hours.

It is more difficult to verify the breakdown inputs because the first breakdown occurs at a randomly selected number of working hours for the MachinePart server. The server's output WorkingTime shows its present operating time. Start the model briefly and record the Value output for the UnschedDowntime object. This value is the operating time that was selected randomly for the first breakdown. The easiest way to find the simulation time at which this operating time is reached is to monitor the UnschedDowntime output StartTime which returns the time at which the last downtime event began. Stop the model as soon as this output become non-zero and record the time. Now set the Control Panel's Pause Time to this value and restart the model. Once the model reaches this time, resume the model at a slow execution speed to verify that the breakdown occurs at the specified operating time for the server.

The MachinePart server offers a number of options for how maintenance and breakdown events are to be handled:

- An 'Immediate' maintenance/breakdown event is performed as soon as the event occurs, interrupting any work that is being performed.
- A 'Forced' maintenance/breakdown event is performed when any work being performed is finished.

- An 'Opportunistic' maintenance/breakdown event is when the server's queue is empty and it has no work to perform.

The choice of the ForceMaintenanceList and ImmediateBreakdownList inputs mean that a maintenance event is performed when the server finishes its present task, while a breakdown event interrupts its present task.

Note that it is possible for a server to have many separate types of maintenance and breakdown events, allowing 'Reliability, Availability and Maintainability' (RAM) type analyses to be performed.

The MachinePart server's state is set to `"Breakdown"` for the duration of a breakdown event and to `"Maintenance"` for the duration of a maintenance event.

4 Graphical User Interface

The graphical user interface (GUI) consists of the Control Panel, one or more View windows, the Model Builder, the Object Selector, the Input Editor, and the Output Viewer.

4.1 Mouse Actions

4.1.1 Moving the Model

A model's appearance in a View window is determined by the position of the view's camera and the direction in which it points. The basic actions with the mouse are zooming the camera in and out using the scroll wheel, panning the camera using left drag, and orbiting the camera using right drag.

Table 4-1 View Camera Controls

Mouse/Keyboard Action	Effect
Left Click	Selects the point of interest. The point on the surface of the object under the cursor becomes the point of interest. If no object is under the cursor, the point on the xy-plane is used.
Scroll Wheel	Zooms the camera in or out. The camera is moved towards or away from the point of interest. One click moves the camera 10% closer to or farther away from the point of interest.
Left Drag	Pans the xy-plane. The camera is moved in the xy-plane from its present position. The point of interest is reset to the cursor position.
Shift + Left Drag	Pans the z-axis. The camera is moved along the z-axis from its present position. The point of interest is reset to the cursor position.
Right Drag	Orbits the camera. The camera orbits left/right and up/down around the point of interest.
Shift + Right Drag	Look around. The camera looks left/right and up/down. The point of interest is unchanged.

If no object is selected, the camera can also be moved using the arrow keys or the WASD keys.

4.1.2 Moving and Resizing Objects

An individual object can be moved by left clicking on the object to select it and then using control + left drag to reposition it. Multiple objects can be moved by selecting them one at a time using control + left click and them moving them as a group using control + left drag.

By default, dragging an object moves it in the xy-plane. An object can be moved in the z-direction by holding down both the Control and Shift keys and dragging the object up and down.

The size and orientation of a selected object can be manipulated using Ctrl + left drag on a corner of the object or on its rotation handle.

Table 4-2 Moving and Resizing Objects

Mouse/Keyboard Action	Effect
Left Click	Selects the object. The object under the cursor is selected for input/output viewing and for repositioning, resizing, or rotating. The selected object is indicated by a green rectangle bordering it. Green handles allow the object to be resized or rotated.
Ctrl + Left Click	Selects multiple objects. Adds the object to a list of selected objects.
Ctrl + Left Drag	Moves the object(s) parallel to the xy-plane, while holding its z-coordinate constant.
Shift + Ctrl + Left Drag	Moves the object(s) parallel to the z-axis, while holding its x-coordinate and y-coordinate constant.
Ctrl + Left Drag on a Handle	Resizes or rotates the object. Ignored it multiple objects are selected.

The selected object can also be moved using the arrow keys or the WASD keys.

Polyline objects such as Arrow can be re-shaped using the mouse. When selected, these types of objects display coloured handles at each node in the polyline. The node at the start of the line is coloured blue, the one at the end is coloured yellow, and intermediate nodes are coloured green.

Table 4-3 Reshaping Polyline Objects

Mouse/Keyboard Action	Effect
Ctrl + Left Drag on a node	Moves location of the node in the xy-plane.
Shift + Ctrl + Left Drag on a node	Moves the location of the node along the z-axis.
Alt + Ctrl + Left Click on the Line	Adds a new node to the line.
Shift +Alt + Ctrl + Left Click on an intermediate node	Deletes the node.

Adding and removing nodes can also be done using the right click (context) menu.

4.2 Context Menu

Right-clicking displays a list of all objects under the cursor. Selecting one of these objects displays the context menu for the selected object shown in the following table. If only one object is under the cursor, the context menu is displayed right away. If no object is under the cursor, the context menu is displayed to allow the possibility of pasting an entity at that location.

Table 4-4 Context Menu Entries

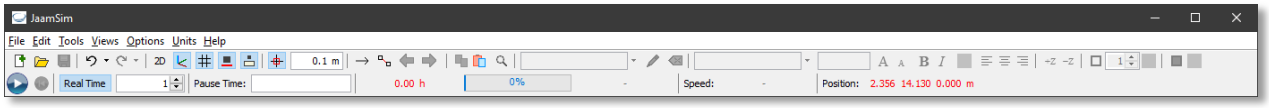
Menu Item	Description
Input Editor	Selects the object and opens its Input Editor window. Information for the selected object is shown in Input Editor, Output Viewer, and the Property Viewer.
Output Viewer	Selects the object and opens its Output Viewer window. Information for the selected object is shown in Input Editor, Output Viewer, and the Property Viewer.

Menu Item	Description
Property Viewer	Selects the object and opens its Property Viewer window. Information for the selected object is shown in Input Editor, Output Viewer, and the Property Viewer.
Copy	Copies the selected object to the clipboard.
Paste	Pastes a copy of an object from the clipboard to the present mouse location.
Delete	Deletes the selected object.
Change Graphics	Opens a dialog box to select a new DisplayModel graphical representation for the selected object.
Show Label	Creates an EntityLabel object that displays the present name of the selected object. The object's name can be changed by double-clicking on the EntityLabel and editing the displayed name.
Set RelativeEntity	Allows the position of the object to be set relative to the position of a second object. If the second object is moved, the first object moves with it.
Set Region	Allows the position of the object to be set in a local coordinate system defined by the Region. If the region is moved or rotated, the objects in the Region move with it.
Centre in View	Centres the current View on the selected object.
Add Node	Adds a new node to the selected polyline-type object at the present mouse location.
Delete Node	Deletes the node at the present mouse location from the selected polyline-type object.
Split	Divides the selected polyline-type object into two separate objects at the present mouse position.

4.3 Control Panel

The Control Panel provides a number of run controls and output displays to monitor and control the progress of a simulation run.

Figure 4-1 JaamSim Control Panel



The Control Panel is divided into three rows, consisting of the Menu Bar, the Button Bar, and the Run Control Bar:

4.3.1 Menu Bar

File

The File entry displays a menu with actions related to saving and loading model input configuration files.

Table 4-5 File Menu

Menu Entry		Description
	New	Starts a new model.
	Open	Opens a model.
	Save	Saves the present model.
	Save As…	Saves the current model as a new input configuration file.
	Import…	Imports one or more 3D models or images. Creates both the ColladaModels/ImageModels containing the graphics and the corresponding DisplayEntities.
	Print Input Report	Prints the present inputs in a standard file format (.inp).
	Exit	Closes all windows and exits JaamSim.

Edit

The Edit entry displays a menu with actions related to editing a model.

Table 4-6 Edit Menu

	Menu Entry	Description
	Undo	Reverses the last change made to the model.
	Redo	Re-performs the last change to the model that was undone.
	Copy	Copies the selected object to the clipboard.
	Paste	Pastes a copy of an object from the clipboard to the location of the most recent mouse click.
	Delete	Deletes the selected object.
	Find	Searches for an object with a given name.

Tools

The Tools entry displays a menu that provides options for showing the windows of the JaamSim graphical user interface:

Table 4-7 Tools Menu

Menu Entry		Description
	Show Basic Tools	Shows the four main tools: Model Builder, Object Selector, Input Editor, and Output Viewer.
	Close All Tools	Closes all of the tools that are open.
	Model Builder	Drag and drop creation and placement of simulation objects.
	Object Selector	Tree listing of all objects in the present simulation model.
	Input Editor	View and edit keyword inputs for the selected simulation object.
	Output Viewer	Output values for the selected object.
	Property Viewer	Detailed list of the internal properties of the selected object.
	Log Viewer	Console for viewing input warnings and error messages.
	Event Viewer	Displays the future events that have been scheduled and enables the user to execute events one at a time. Also displays the processing time spent by each type of event.
	Reset Positions and Sizes	Resets the positions and sizes of the tool windows to their default values.

The Property Viewer is typically used by programmers who are developing and debugging JaamSim applications, and so its usage is beyond the scope of this manual.

Views

The Views entry displays a menu containing a list of currently defined View windows and provides the ability to create new Views. A View window shows a graphical 3D representation of the model.

Table 4-8 Views Menu

Menu Entry		Description
	View1	Opens the window for View1, the default View. Does nothing if the window has already been opened.
	Define New View	Creates a new View object and displays its window.
	Reset Positions and Sizes	Resets the positions and sizes of the view windows to their default values.

Options

The Options entry in the menu bar contains the following entries:

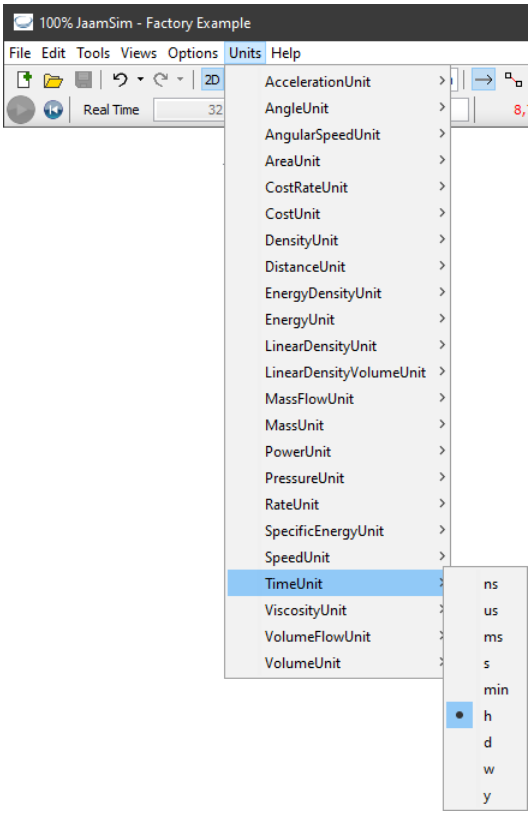
Table 4-9 Options Menu

Menu Entry		Description
	Always On Top	If checked, the control panel will always remain on top of other windows.
	Graphic Debug Info	If checked, information on video memory usage and rendering time will be shown as an overlay on the View windows.

Units

The Units entry in the menu bar is used to set the units in which to display model outputs in the Control Panel, Output Viewer, and output reports. The default values in the Input Editor will also be displayed in the selected unit.

Figure 4-2 Units Menu



For example, if you want simulation time to be displayed in seconds instead of hours, select TimeUnit from the submenu of unit types and click on the entry labelled 's'. With this change, the Control Panel will display simulation time in seconds and every output in the Output Viewer with units of time will be displayed in seconds.

Note that the choice of output unit has no effect on the internal calculations in a simulation model. Furthermore, model inputs can always be entered in any valid unit.

Help

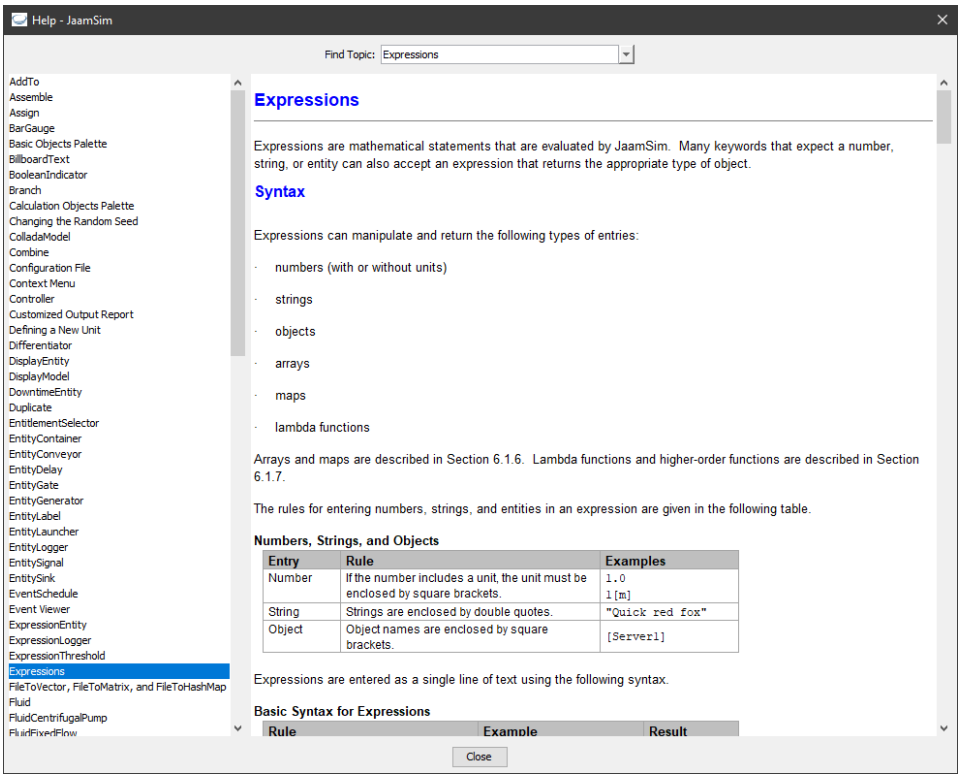
The Help entry in the menu bar contains the following entries:

Table 4-10 Options Menu

Menu Entry		Description
	About	Shows the software version number and copyright notice.
	Help	Displays the Help tool.

The Help tool provides documentation for all the features in JaamSim. The Help tool can be launched by pressing F1.

Figure 4-3 Help Tool



4.3.2 Button Bar

The left side of the Button Bar contains controls for basic model operations such as opening and saving a model. The right side contains controls for formatting the selected object.

Basic Model Operations

Table 4-11 Basic Model Operations

Button Bar Item		Description
	New	Starts a new model.
	Open …	Opens a model.
	Save	Saves the present model.
	Undo	Reverses the last change made to the model.
	Redo	Re-performs the last change to the model that was undone.
	2D View	Sets the camera position to show a bird-eye view of the 3D scene.
	Show Axes	Shows the unit vectors for x-, y-, and z-axes.
	Show Grid	Shows the coordinate grid on the xy-plane.
	Show Labels	Displays the label for every entity in the model.
	Show SubModels	Displays the components of each sub-model.
	Snap to Grid	During repositioning, objects are forced to the nearest grid point.
	Snap Grid Spacing	Distance between adjacent grid points, e.g. 0.1 m, 10 km, etc.
	Show Entity Flow	When selected, arrows are shown between objects to indicate the flow of entities.
	Create Entity Links	When selected, entities are linked when selection is changed. For example, left-clicking on a Server and then on an Assign object, will set the NextComponent input for the Server to the Assign object.
	Previous	Selects the previous object in the chain of linked objects.
	Next	Selects the next object in the chain of linked objects.
	Copy	Copies the selected object to the clipboard.
	Paste	Pastes a copy of an object from the clipboard to the location of the most recent mouse click.
	Entity Finder	Searches for an object with a given name.

Formatting Buttons

Table 4-12 Formatting

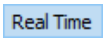
Button Bar Item		Description
	DisplayModel	Sets the default appearance of an entity. A DisplayModel is analogous to a text style in a word processor.
	Edit DisplayModel	Selects the present DisplayModel so that its inputs can be edited.
	Clear Formatting	Resets the format inputs for the selected Entity or DisplayModel to their default values.
	Font	Sets the font for the text.
	Text Height	Sets the height for the text, e.g. 0.1 m, 200 cm, etc.
	Larger Text	Increases the text height to the next higher multiple of the snap grid spacing.
	Smaller Text	Decreases the text height to the next smaller multiple of the snap grid spacing.
	Bold	Makes the text bold.
	Italic	Italicizes the text.
	Font Colour	Sets the colour of the text.
	Align Left	Aligns the text to the left margin.
	Align Centre	Centres the text between the right and left margins.
	Align Right	Aligns the text to the right margin.
	Move Up	Increases the selected object's z-coordinate by one hundredth of the snap-grid spacing. By moving the object closer to the camera, it will appear on top of other objects with smaller z-coordinates.
	Move Down	Decreases the selected object's z-coordinate by one hundredth of the snap-grid spacing. By moving the object farther from the camera, it will appear below other objects with larger z-coordinates.
	Show Outline	Shows the outline.
	Line Width	Sets the width of the line in pixels.
	Line Colour	Sets the colour of the line.
	Show Fill	Fills the entity with the selected colour.
	Fill Colour	Sets the colour of the fill.

4.3.3 Run Control Bar

The Run Control Bar contains controls for starting, pausing, and resetting a simulation model and for controlling its execution speed and displaying its status.

Run Controls

Table 4-13 Run Controls

Run Control Bar Item		Description
	Run/Pause	Starts, pauses, and resumes the simulation run.
	Reset	Clears any generated entities and sets the simulation time to zero.
	Real Time Mode	When selected, the simulation runs at a fixed multiple of wall clock time.
	Speed Multiplier	Target ratio of simulation time to wall clock time when Real Time mode is selected.
	Pause Time	The time at which the software automatically pauses the simulation. Accepts numbers with time units (e.g. 500 h, 1 y) or date/time format (`hh:mm:ss.s` or `'YYYY-MM-DD hh:mm:ss.s'`).

The following keyboard shortcuts are provided for controlling a simulation run.

Table 4-14 Keyboard Shortcuts for Run Control

Keystroke	Action
Space bar	Starts, pauses, and resumes the simulation run.
>	Increases the real time speed multiplier by a factor of 2.
<	Decreases the real time speed multiplier by a factor of 2.

It is not necessary to press the shift key when using the '>' and '<' keys to control the speed multiplier.

Run Status

The right side of the tool bar consists of indicators to illustrate the progress and status of the simulation run:

Table 4-15 Run Status

Tool Bar Item	Description
Simulation Time	The present simulation time.
Run Progress	Percentage of the present simulation run that has been completed.
Remaining Time	The remaining time required to complete the present simulation run.
Achieved Speed Multiplier	The ratio of elapsed simulated time to elapsed wall-clock time that was achieved.
Cursor Position	Location of the cursor in the xy-plane.

4.4 View Window

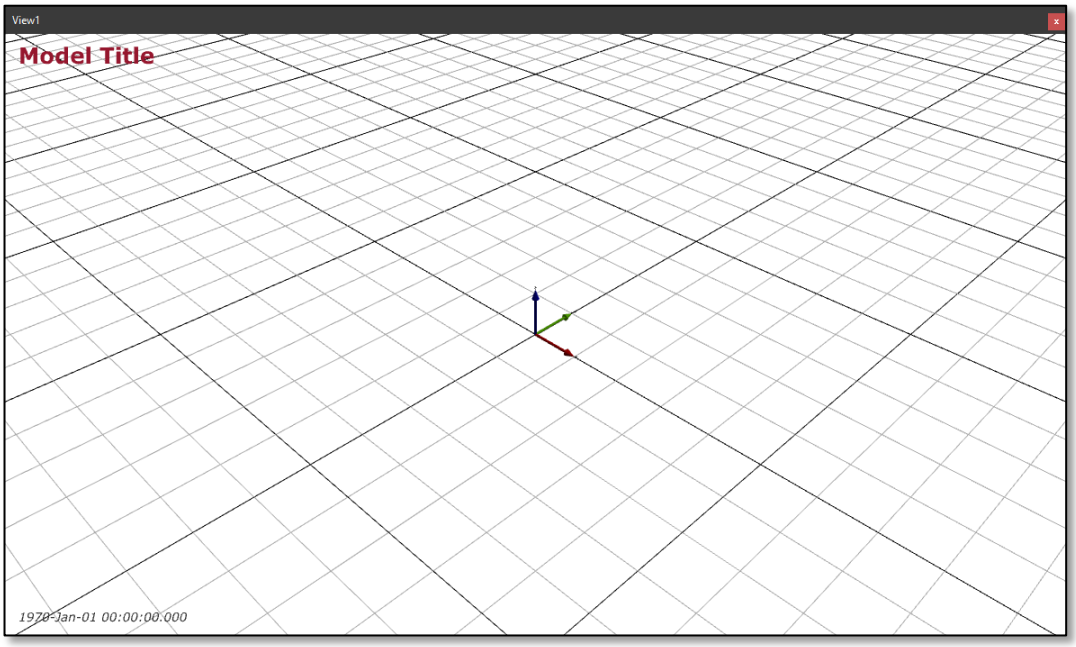
A View window displays a graphical representation of the simulation. Model objects are dragged and dropped into a View window using the Model Builder tool.

Multiple View windows can be defined that depict different portions of the 3d model space.

A new View can be created by clicking on the 'Views' item in the Menu bar. An existing View can be selected for editing by clicking on its entry in the Object Selector. Views can be copied, pasted, renamed, or deleted in the same way as other objects in the Object Selector.

The following screenshot shows the default View window.

Figure 4-4 Default View Window



View windows are created with a number of default graphical objects:

- XY-Grid – a static grid on the xy-plane
- XYZ-Axis – coordinate vectors located at the origin
- Title – overlay text where a title for the simulation can be entered
- Clock – overlay clock showing simulation time in YYYY-MM-DD format

The XY-Grid and XYZ-Axis are DisplayEntity objects with the Movable keyword is set to FALSE so that they cannot be moved accidentally and do not respond to mouse clicks. They can be turned on or off using the 'Show Grid' and 'Show Axes' buttons.

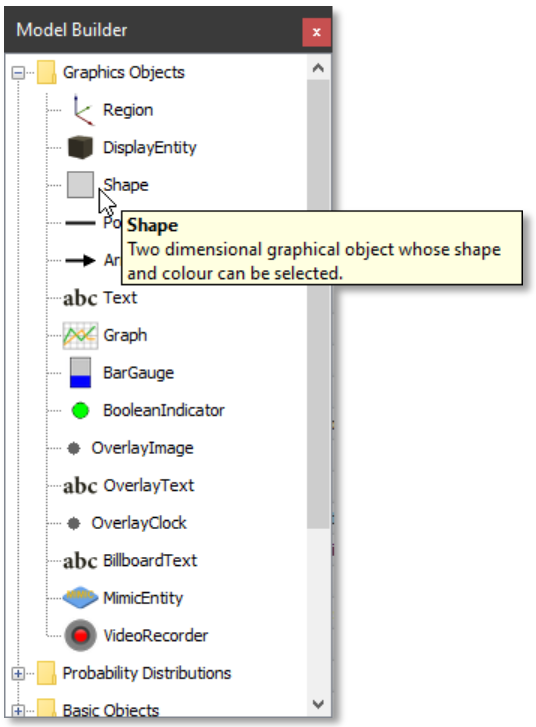
The Title and Clock objects overlay objects (OverlayText and OverlayClock, respectively) that appear in a fixed position on the View window and are not part of the 3D scene. The Title can be modified by double clicking on the text and editing it in place. Both objects can be moved using Ctrl + Left Drag or deleted using the Delete key.

4.5 Model Builder

The Model Builder provides palettes of objects that can be dragged and dropped to construct a new model or to modify an existing one.

The following figure shows the Model Builder with the Graphics Objects palette expanded.

Figure 4-5 Model Builder



Pop-up descriptions can be shown by hovering the mouse over each object in the Model Builder. More information can be obtained about an object by left clicking on it and pressing F1 to launch the Help window for that object.

Some objects, such as DisplayModels and Units, do not have a graphical representation and cannot be dragged and dropped by the user. These objects appear in the Object Selector, but not in the Model Builder.

An object created by drag and drop is assigned a name by appending a number to the name of the object type. For example, the first DisplayEntity that is dragged and dropped will be named DisplayEntity1. The second one will be named DisplayEntity2, and so on.

If the 'Show Labels' button is set, then an EntityLabel object displaying the object's name will be created below the dragged and dropped object.

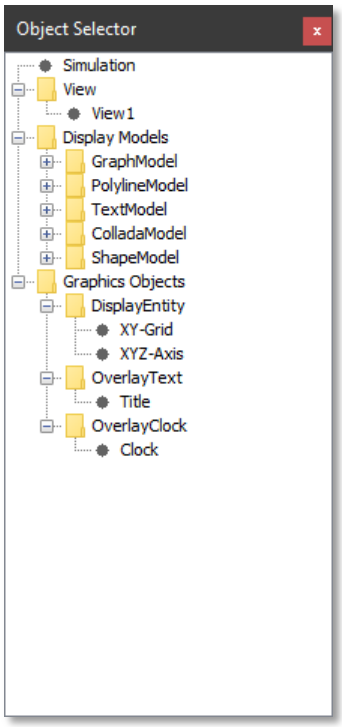
An object can be renamed by double-clicking on the name displayed by its EntityLabel and editing it in place. The new name is accepted by pressing the Enter key or by clicking elsewhere in the view window. An object name can be any alphanumeric text that does not include spaces, tabs, single or double quotes, square brackets, hash characters, or periods.

When an entity is dragged and dropped on top of a region, the entity's Region input is set to that region.

4.6 Object Selector

The Object Selector contains all objects that have been created for the present model, including ones that were created automatically by JaamSim. Objects are grouped according to their palette and type in a tree format that mirrors the structure of the Model Builder. A specific object can be selected either by clicking its node in the Object Selector or by clicking it in a View window.

Figure 4-6 Object Selector



Objects that have been created using the Model Builder can be renamed or deleted using the Object Selector. Once an object has been selected, it can be renamed by pressing F2 or by left-clicking on its highlighted entry in the Object Selector, similar to the convention in Windows. A selected object can be deleted by pressing the Delete key, or by right-clicking in either a View window or the Object Selector and selecting Delete.

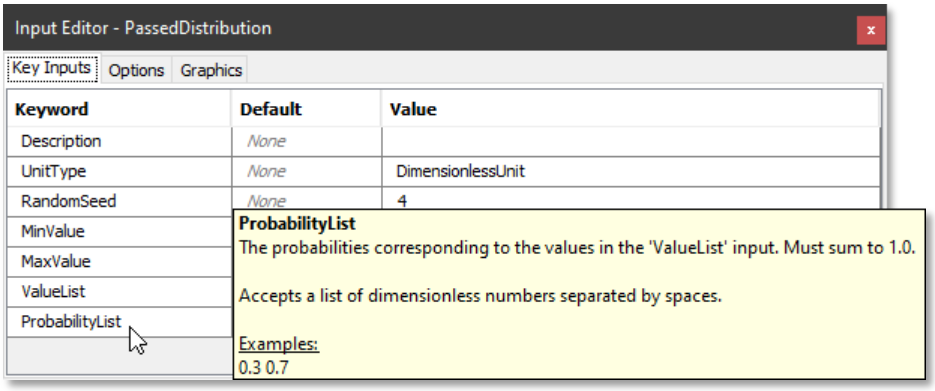
The default objects: XY-Grid, XYZ-Axis, Title, and Clock, can be deleted by the user. However, for the XY-Grid and XYX-Axis, it is better to hide them using the 'Show Grid' and 'Show Axes' buttons, which allows them to be re-instated at a later time.

4.7 Input Editor

The Input Editor allows the user to modify inputs for existing objects or assign inputs to new objects. When an object is selected, its parameters appear in the Input Editor window, grouped under a number of tabs. If a keyword has a default value assigned, it is shown in the Default column.

Hovering the cursor over a keyword will display a tooltip containing a brief description of the keyword and an example input.

Figure 4-7 Input Editor



The input for a keyword can be modified by clicking on the entry in the Value column and entering a new value with the appropriate units. Numbers must be entered without spaces or commas and Boolean keywords must take the value TRUE or FALSE (case sensitive). If an entry is made in the Value column, it will overwrite the default. If an input is not valid, an error message will be displayed showing the cause of the error.

Depending on the object, different keyword values will have different data types as shown in the following table.

Table 4-16 Input Data Types

Data Type	Description	Examples
Numbers without units	A number with or without a decimal point.	5 5.0
Numbers with units	A number followed by a unit separated by one or more spaces. The unit can be enclosed by square brackets, if preferred.	1000 mm 1.0 m 0.001 km 0.001 [km]
Times	Times can be entered normally as a number and time unit or it can be entered in date/time format. The following formats are supported: hh:mm hh:mm.ss hh:mm:ss.sss, 'YYYY-MM-DD hh:mm' 'YYYY-MM-DD hh:mm:ss' 'YYYY-MM-DD hh:mm:ss.sss' YYYY-MM-DDThh:mm YYYY-MM-DDThh:mm:ss YYYY-MM-DDThh:mm:ss.sss If the date/time format includes a space, the entire text must be enclosed by a pair of single quotes.	30.4 h 30:24:00.0 '0000-01-02 06:24:00.0' 0000-01-02T06:24:00.0
Vectors and Points	Values for the three components followed by a unit. One or more spaces separate the values and unit. If only two values are entered, the z-component is assumed to be zero. The unit can be enclosed by square brackets, if preferred.	2.0 1.0 0.0 m 2.0 1.0 m 2.0 1.0 0.0 [m]

Data Type	Description	Examples
Expressions	A formula containing object outputs and/or mathematical functions. Expressions are described in detail in Section 6.1.	`'1 + 2*[Queue1].QueueLength'` `'2[s] + [Queue1].QueueTimes(1)'`
Booleans	A value of either TRUE or FALSE (case-sensitive).	`TRUE` `FALSE`

Colours	A colour can be specified by a colour keyword or by RGB values. A list of named colours and their RGB equivalents are given in Section 0. Transparency can be specified by adding a fourth number after the RGB values.	`pink`
		`255 192 203`
		`255 192 203 125`

Strings	Text enclosed by a pair of single quotes. The single quotes can be omitted if the text does not include any spaces.	`'Quick red fox'` `Quick_red_fox`
Objects	Specified by the object's name.	`Server1` `Queue1`

Braces are used to delineate distinct entries in a list. For example, a list of two points would be entered as `{ 0.0 1.0 0.0 m } { 1.0 1.0 0.0 m }`. When an input has only one set of inner braces, it can be entered without the inner braces. For instance, `{ 1 2 3 }` can be entered as `1 2 3`.

A drop-down menu is available for most types of inputs.

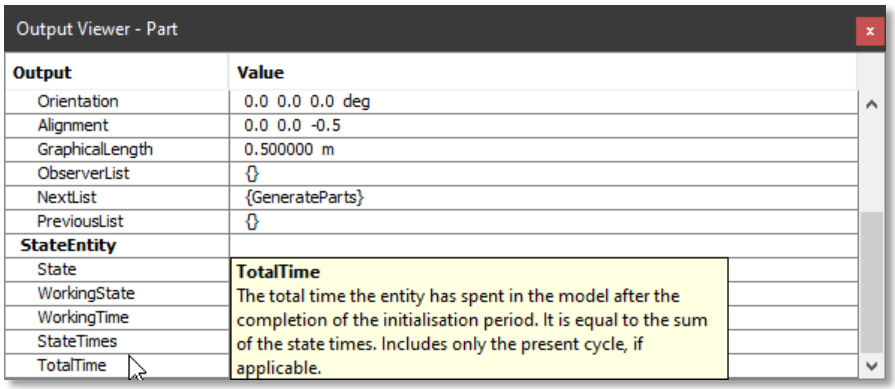
4.8 Output Viewer

The Output Viewer displays the available outputs for the selected object. The values in the Output Viewer are updated continuously as the simulation progresses.

Numerical outputs are normally shown in the appropriate SI unit, except for time which is shown in hours. When it is more convenient to view an output in another unit, the user can specify a set of preferred units using the Units entry in the Menu bar (see Section 4.3.1).

Hovering the cursor over an output name will display a tooltip containing a brief description of the output.

Figure 4-8 Output Viewer

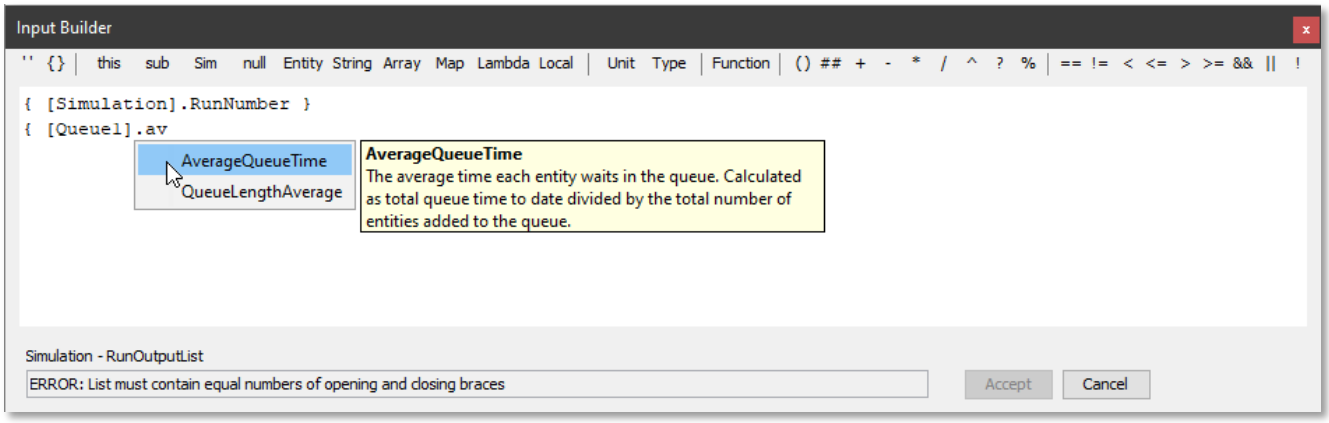


4.9 Input Builder

The Input Builder is a tool for constructing inputs that involve expressions. It is opened through the Input Editor by clicking on the 'Input Builder' entry in the dropdown menu for a keyword.

The following figure shows the Input Builder being used to construct the input to the RunOutputList keyword for the Simulation object.

Figure 4-9 Input Builder



The Input Builder has an auto-complete feature that provides a menu of valid names for both objects and outputs. The feature is activated by entering a left square bracket (indicating the start of an object name) or a period (indicating the start of an output name). The menu becomes shorter as the user enters any portion of the object's or output's name. Hovering the cursor over a menu item generates a pop-up description of the object or output. For an object, the description pop-up is generated from the input to its Description keyword.

The input for a keyword that uses inner curly braces such as AttributeDefinitionList or RunOutputList can be spread over several lines in the Input Builder. Line feeds are deleted automatically when the input is accepted. When an existing input is edited, the input for each pair of curly braces is placed automatically on a separate line.

The field at the bottom of the Input Builder shows whether or not the present input is valid. The present value for the input is shown if the input is valid. An error message is shown if it is not valid. Only a valid input can be accepted.

A button bar is provided that includes all the symbols and data types that are used by the expression system's syntax. The buttons provide both a convenient way to enter the correct symbols as well as pop-up descriptions that define the syntax and provide examples.

4.10 Event Viewer

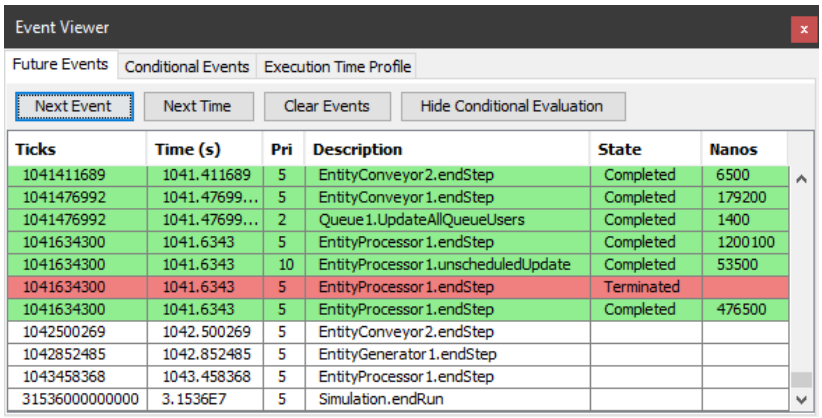
The Event Viewer is a tool for debugging and optimizing a simulation model. It is intended primarily for advanced users and programmers, but can also be used to demonstrate the underlying logic behind discrete event simulation.

4.10.1 Future Events tab

The 'Future Events' tab displays the future events that are scheduled for execution by the model at the present simulation time. The most recent events that have been executed, including conditional events, are also displayed along with the computer processing time in nanoseconds that was required to execute each event. A maximum of 1,000 completed events can be recorded, after which the oldest ones are discarded.

The following figure shows the Event Viewer for a simple model with just a few future events. A complex model can have many hundreds or thousands of future events.

Figure 4-10 Event Viewer



Events that have been completed are shown with a green background in the table. An event that was scheduled and then cancelled is shown with a red background. A conditional event that has been completed is shown with a yellow background. Future events are shown with a white background.

The following information is provided by the columns in the table:

- **Ticks** – the internal simulation time for the event expressed as an integer number of 'clock ticks'. The duration of a clock tick is defined by the TickLength input for the Simulation object. The default value is one microsecond.

- **Time** – the simulation time for the event. The time unit for this display is set using the Unit menu item on the Control Panel. The default unit is hours.

- **Pri** – the priority of the event that was assigned by the internal JaamSim logic. Events that are scheduled for the same simulation time are executed in order of increasing priority value, e.g. an event with priority 2 is executed before an event with priority 5.

- **Description** – the simulation object associated with the event and the JaamSim method to be executed by the event.

- **State** – the present state of the event: 'Completed', 'Terminated', or blank if the event has not been executed yet.

- **Nanos** – the computer processing time in nanoseconds that was required to execute the event. These times are accurate ONLY when the model is executed continuously with 'Real Time' mode turned off. Processing times are many times longer when events are executed

one by one using the 'Next Event' button or when 'Real Time' mode is used to slow the execution speed for the model. The large reduction in processing time is due to the 'just-in-time' compiling used by the Java virtual machine, not to the way the execution time data is collected.

The following buttons are provided:

- **'Next Event' button** – executes the next future event. If the event time is later than the present simulation time, the model is advanced to that time in accordance with the 'Real Time' mode setting, i.e. at the specified execution speed if it is 'on' or instantaneously if it is 'off'. After the event is executed, it is common for one or more new events to be scheduled for execution at future times.

- **'Next Time' button** – executes the next future event along with any other events that are scheduled for the same simulation time.

- **'Clear Events' button** – removes all the completed events from the Event Viewer.

- **'Hide Condition Events' button** – turns off the display of any conditional events that were executed. Conditional events are tested and, if appropriate, executed before time is advanced to the next event time. The processing time required to execute these events can be significant which makes it important to display them. However, if there are a large number of these events, they can create unnecessary clutter in the display. This button gives the user the ability to display or hide these results as required.

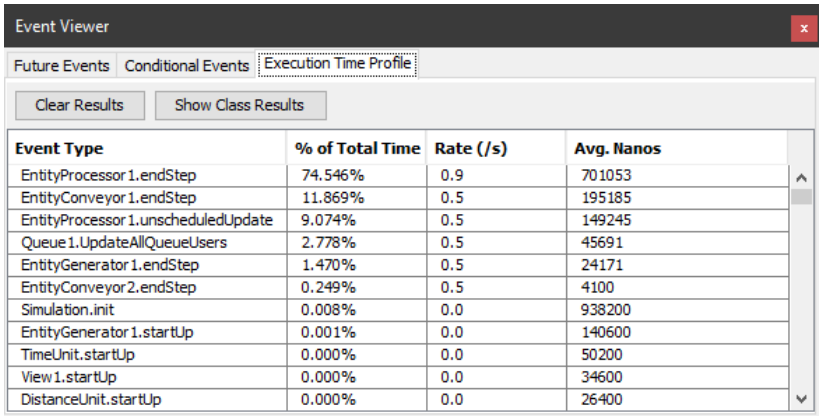
4.10.2 'Conditional Events' tab

The 'Conditional Events' tab provides a list of the conditional events to be tested and possibly executed at each event time. Most conditional events are associated with an ExpressionThreshold object. Note that often it is possible to eliminate a conditional event by providing an input to the WatchList keyword for the object in question.

4.10.3 'Execution Time Profile' tab

The 'Execution Time Profile' tab provides a list of event types sorted by the total processing time consumed by each event type. The following figure shows this display for a simple model with just a few types of events.

Figure 4-11 Event Viewer – Execution Time Profile



Event Type	% of Total Time	Rate (/s)	Avg. Nanos
EntityProcessor1.endStep	74.546%	0.9	701053
EntityConveyor1.endStep	11.869%	0.5	195185
EntityProcessor1.unscheduledUpdate	9.074%	0.5	149245
Queue1.UpdateAllQueueUsers	2.778%	0.5	45691
EntityGenerator1.endStep	1.470%	0.5	24171
EntityConveyor2.endStep	0.249%	0.5	4100
Simulation.init	0.008%	0.0	938200
EntityGenerator1.startUp	0.001%	0.0	140600
TimeUnit.startUp	0.000%	0.0	50200
View1.startUp	0.000%	0.0	34600
DistanceUnit.startUp	0.000%	0.0	26400

For accurate results, a model MUST be executed with 'Real Time' mode set to 'off' and allowed to run long enough for the 'Speed' display on the Control Panel to have reached its maximum value. The model should then be paused and the 'Clear Results' button clicked to reset the execute time data. The model can then be resumed to collect fresh data. The displayed profile data is updated continuously as the model runs.

The following data is displayed on the 'Execution Profile' tab:

- **'Event Type'** – the simulation object and JaamSim method name for the event.
- **'% of Total Time'** – the total processing time for the events of this type divided by the total processing time for all types of events, expressed as a percentage.
- **'Rate'** – the number of executions for this event type per unit simulation time. The time unit for this display is set using the Unit menu item on the Control Panel. The default unit is hours.
- **'Avg. Nanos'** – the average processing time in nanoseconds required to execute events of this type.

The 'Rate' and 'Avg. Nanos' displays can be used to determine whether the total processing time for an event type is the result of a long execution time or a large number of events being executed.

The 'Show Class Results' button changes the display to show results for each simulation object type instead of for each simulation object. For example, the normal display will show separate entries for EntityConveyor1 and EntityConveyor2, whereas with 'Show Class Results' selected, it will show a single entry for all EntityConveyors.

5 Units

5.1 Unit Types

JaamSim performs all internal calculations in SI units (meters, kilograms, seconds, etc.). However, as it can sometimes be more convenient to specify quantities using other unit systems, JaamSim natively supports the units shown in the following table.

Table 5-1 Supported Unit Types and Units

Unit Type	Supported Units
DimensionlessUnit	not applicable
TimeUnit	ns, us, ms, s, min, h, d, w, y
DistanceUnit	mm, cm, m, km, nmi, in, ft, mi
SpeedUnit	m/s, km/h, knots, mph
AccelerationUnit	m/s2, ft/s2
MassUnit	kg, t, kt, Mt
MassFlowUnit	(any mass unit)/(s, h, d, y)
VolumeUnit	m3, km3, bbl, mbbl, mmbbl
VolumeFlowUnit	(any volume unit)/(s, h, d, y)
AngleUnit	rad, deg
AngularSpeedUnit	rad/s, rad/h, deg/s, deg/h
EnergyUnit	J, kWh
EnergyDensityUnit	J/m3, kWh/m^3
SpecificEnergyUnit	J/kg, kWh/t
PowerUnit	W, kW, MW
CostUnit	$
CostRateUnit	$/s, $/h, $/d
LinearDensityUnit	kg/m, t/m, kt/m
LinearDensityVolumeUnit	m3/m
DensityUnit	kg/m3
PressureUnit	Pa, kPa, psi
ViscosityUnit	Pa-s, P, cP
AreaUnit	m2, cm2, mm2, in2
RateUnit	/ns, /us, /ms, /s, /min, /h, /d, /w, /y

Units are mandatory for most numerical inputs with the exception of pure numbers and ratios. Inputs that are pure numbers are indicated by the DimensionlessUnit type.

Each unit type has a selected unit that is used in the various displays. For example, the displayed units for time, distance, and speed are hours (h), metres (m), and metres per second (m/s). The displayed units can be changed using the 'Unit' menu item in the Control Panel.

5.2 Defining a New Unit

In addition to the predefined units in JaamSim, new units can be specified using a Unit object. A new Unit object can be created by entering the appropriate Define statement in the configuration file (see Section 8). For example if a new TimeUnit called 'Fortnight' is required, then the define statement would be:

```
Define  TimeUnit { Fortnight }
```

All Unit objects have the same input keyword:

Table 5-2 Unit Key Inputs

Keyword	Description
ConversionFactorToSI	Two numbers that specify the numerator and denominator, respectively, of the multiplicative factor to convert from the new Unit to SI base units.

For example, for the 'Fortnight' time unit (two weeks) defined above, the ConversionFactorToSI keyword should be set as follows:

```
Fortnight  ConversionFactorToSI ( 1209600  1 }
```

Once the new unit is defined in this way, it can be used with any input that requires that type of unit.

6 Expressions and User-Defined Variables

Most real-world models involve detailed rules and variables that are specific to that application. The variety and complexity of these rules make it impossible to accommodate every possibility in the keywords for a finite set of built-in objects. The expressions, attributes and custom outputs described in this section provide the model builder with the tools necessary to customize the built-in objects for the application.

6.1 Expressions

Expressions are mathematical statements that are evaluated by JaamSim. Many keywords that expect a number, string, or entity can also accept an expression that returns the appropriate type of object.

6.1.1 Syntax

Expressions can manipulate and return the following types of entries:

- numbers (with or without units)
- strings
- objects
- arrays
- maps
- lambda functions

Arrays and maps are described in Section 6.1.6. Lambda functions and higher-order functions are described in Section 6.1.7.

The rules for entering numbers, strings, and entities in an expression are given in the following table.

Table 6-1 Numbers, Strings, and Objects

Entry	Rule	Examples
Number	If the number includes a unit, the unit must be enclosed by square brackets.	`1.0` `1[m]`
String	Strings are enclosed by double quotes.	`"Quick red fox"`
Object	Object names are enclosed by square brackets.	`[Server1]`

Expressions are entered as a single line of text using the following syntax.

Table 6-2 Basic Syntax for Expressions

Rule	Example	Result

If an expression includes spaces, curly brackets, or double quotes, it must be enclosed by a pair of single quotes. Spaces in an expression are ignored.	`1[m]+2[m]`
	`'1[m] + 2[m]'`
	`'"abc"'`
		`3[m]`
		`3[m]`
		`abc`

Rule	Example	Result

Rules for mathematical order of operation are respected for the standard operators: +, -, *, /, and ^. Round brackets can be used to modify the order of operation.	`1+2*3`
	`(1+2)*3`
	`2*3^2`
	`(2*3)^2`
		`7`
		`9`
		`18`
		`36`

All calculations must respect units. Unit conversions are performed automatically.	`1[m]+2`
	`1[m]+2[m]`
	`1[m]/2[s]`
	`1[m]/2[m]`
	`1[m]*2[m]`
		`syntax error`
		`3[m]`
		`0.5[m/s]`
		`0.5`
		`2[m2]`

Outputs and attributes are referenced using a dot notation.	`[Queue1].QueueLength`	Number of entities in Queue1
The reserved string 'this' is used to refer to the object evaluating the expression.	`this.State`	State of the present entity.
The reserved string 'null' is used to refer to an undefined object.	`this.obj == null`	TRUE if the output obj is undefined.
Outputs can be chained using the dot notation.	`[Queue1].QueueList(1).State`	State of the entity at the front of the queue.
Comments can be added at any point in an expression by enclosing the comment between a pair of hash marks ('#').	`'1+2*3 #order of operation#'`	`7`

When developing a complex expression it is usually best to use the Input Builder tool which is available through the drop down menu for the relevant keyword in the Input Editor.

6.1.2 Local Variables

To improve readability and to avoid repeated calculations, it is possible to define one or more local variables within an expression. A local variable can take the value of any valid type, i.e. a number, string, object, array, map, or lambda function.

The following syntax is used to define a local variable:

```
'<variable1> = <expression1>; … <variableN> = <expressionN>; <final expression>'
```

For example, the expression `'x = 1; y = 2; x + y'` would return the value 3.

6.1.3 Functions

The following mathematical functions can be used in expressions:

Table 6-3 Basic Mathematical and Logical Functions

Function	Description	Example	Result
PI	Mathematical constant ‘pi’	`PI()`	`3.14159…`
E	Mathematical constant ‘e’	`E()`	`2.71828…`
min	Smallest of a list of values	`'min(1[s], 2[s])'`	`1[s]`
max	Largest of a list of values	`'max(1[s], 2[s])'`	`2[s]`

Function	Description	Example	Result
indexOfMin	Position of the minimum in a list of values	'indexOfMin(1[s], 2[s])'	1
indexOfMax	Position of the maximum in a list of values	'indexOfMax(1[s], 2[s])'	2
abs	Absolute value	'abs(-1[s])'	1[s]
ceil	Smallest (closest to negative infinity) integer that is greater than or equal to the argument	'ceil(5.2[s])'	6[s]
floor	Largest (closest to positive infinity) integer that is less than or equal to the argument	'abs(5.2[s])'	5[s]
round	Returns the closest integer to the argument, with ties rounding to positive infinity. Accepts a number with or without units. Returns an integer with the same units.	'round(2.5[s])' 'round(-2.5[s])'	3[s] -2[s]
signum	Zero if the argument is zero, 1.0 if the argument is greater than zero, and -1.0 if the argument is less than zero.	'signum(5.2[s])'	1
sqrt	Square root	'sqrt(4.0)'	2.0
cbrt	Cube root.	'cbrt(8.0)'	2.0
%	Modulus (remainder) operator. Used as an operator, not a function.	'11.5 % 4' '11.5[s] % 4[s]'	3.5 3.5[s]
choose	Selects from a list using an index	'choose(2, 1[s], 2[s])'	2[s]

Table 6-4 Exponential and Trignometric Functions

Function	Description	Example	Result
exp	Exponential function	'exp(1)'	2.71828…
ln	Natural logarithm	'ln(2.71828)'	0.999999
log	Base-10 logarithm	'log(100)'	2.0
sin	Sine of an angle or dimensionless number	'sin(30[deg])'	0.5
cos	Cosine of an angle or dimensionless number	'cos(60[deg])'	0.5
tan	Tangent of an angle or dimensionless number	'tan(45[deg])'	1
asin	Arcsine function	'asin(0.5)'	30[deg]
acos	Arccosine function	'acos(0.5)'	60[deg]
atan	Arctangent function	'atan(1.0)'	45[deg]
atan2	Two-argument arctangent function. For Cartesian coordinates x and y, atan2(x, y) returns the angle for the corresponding polar coordinates.	'atan2(1.0, -1.0)'	135[deg]

Table 6-5 Functions Involving Objects

Function	Description	Example	Result
notNull	Determines whether an entity exists. It can be used to test whether an output such as obj has been set. A better way to test for a non-null value is the expression `'this.obj != null'`.	`'notNull(this.obj)'` `'this.obj != null'`	`0 or 1` `0 or 1`

Table 6-6 Functions Involving Strings

Function	Description	Example	Result

+	Strings can be concatenated using the + operator.
	If necessary, the right hand side of the + operator is converted to a string.
		`'"abc" + "def"'`
		`'"abc" + 5[m]'`
		`'"abc" + [Entity1]'`
			`"abcdef"`
			`"abc5.0[m]"`
			`"abc[Entity1]"`

format	Constructs a string using a format string and one or more additional arguments. Each argument can be a dimensionless number, string, object, array, or map. The function mirrors the `String.format` method provided in Java.
	Each entry of `%` in the format string indicates a format code that inserts the next argument in the resulting output text. The full set of Java format codes is supported. The two most relevant are:
	`%s` - displays any argument as a string.
	`%.nf` - displays a number with n decimal places.
		`'format("x = %s",[Entity1])'`
		`'format("x = %s",5)'`
		`'format("x = %.3f",5)'`
		`'format("x = %s cm", 5[m]/1[cm])'`
		`'format("%s x %s","abc","def")'`
			`"x = [Entity1]"`
			`"x = 5.0"`
			`"x = 5.000"`
			`"x = 500.0 cm"`
			`"abc x def"`

indexOfStr	Returns the index within the specified string of the first occurrence of the specified substring, starting at the specified index. Zero is returned if the substring is not found. If the index is omitted, the entire string is searched.	`'indexOfStr("abcdefg", "cd")'` `'indexOfStr("ffaffbc", "ff", 3)'`	`3` `4`
length	Returns the length of the specified string. The length is equal to the number of characters in the string.	`'length("abc")'`	`3`

Function	Description	Example	Result
parseNumber	Converts a string to a floating point number.	`'parseNumber("1.5")'`	`1.5`

split	Divides the specified string into an array of substrings around matches to a specified 'regex' string. An optional third 'limit' argument specifies the maximum number of substrings to return. The function mirrors the split method provided in Java. See the Java documentation for 'regex' string (regular expression).	`'split("ab:cd:ef", ":")'`
		`'split("ab.cd.ef", "\\.")'`
		`'split("ab.cd.ef", "\\.\", 2)'`
			`{"ab","cd","ef"}`
			`{"ab","cd","ef"}`
			`{"ab","cd.ef"}`

substring	Returns a string that is a substring of the specified string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex-beginIndex. If the endIndex is omitted, the substring extends to the end of the string.	`'substring("abcdefg", 3)` `'substring("abcdefg", 3, 6)'`	`"cdefg"` `"cde"`
toUpperCase	Converts the specified string to upper case characters.	`'toUpperCase("abc")'`	`"ABC"`
toLowerCase	Converts the specified string to lower case characters.	`'toLowerCase("ABC")'`	`"abc"`
trim	Removes leading and trailing whitespace from the specified string.	`'trim("  abc  ")'`	`"abc"`

6.1.4 Logical Operations

Logical operations can also be performed in Expressions. All non-zero dimensionless values are interpreted as `TRUE`, while zero is interpreted as `FALSE`. The following are examples of syntactically valid logical Expressions:

```
'this.OutputA >= 1'
'(this.OutputA >= 1) && ([Entity1].OutputB == 0)'
```

The following logical operators are supported:

Table 6-7 Logical Operators

Operator	Description	Example	Result
==	Equal to	`'4.2 == 4.2'`	`1`
!=	Not equal to	`'3.5 != 4.2'`	`1`
<	Less than	`'3.5 < 4.2'`	`1`

Operator	Description	Example	Result
<=	Less than or equal to	'3.5 <= 3.5'	1
>	Greater than	'4.2 > 3.5'	1
>=	Greater than or equal to	'3.5 >= 3.5'	1
&&	Logical AND operation	'1 && 1'	1
||	Logical OR operation	'1 || 0'	1
!	Logical NOT operation	'! 0'	1

The && and || operators use short-circuited evaluation, that is, the right-hand side of the operator is only evaluated when necessary. For example, if the left-hand side of the && operator is FALSE, then the result is FALSE regardless of the value of the right-hand side. Similarly, if the left-hand side of the || operator is TRUE, then the result is TRUE regardless of the right-hand side.

Short-circuit evaluation allows an expression such as the following to be evaluated without returning an error when this.obj is null:

```
'this.obj != null && this.obj.attrib > 0'
```

6.1.5 Conditionals

Conditional operations are implemented with the ternary operator `'?'` using the following syntax:

```
<condition_expression> ? <true_expression> : <false_expression>
```

The `?` operator uses short-circuited evaluation, that is, the `<true_expression>` is evaluated only when the `<condition_expression>` is `TRUE` and the `<false_expression>` is evaluated only when it is `FALSE`.

Examples of the conditional operator are:

```
'2>1 ? 5 : 4'
'this.OutputA >= 2 ? [Entity1].OutputB + 1 : 0'
```

Complex logical expressions can be constructed by chaining a series of ternary operators to form an 'if, else-if' type structure, e.g.:

```
'this.OutputA <= 2 ? 2 : (this.OutputA <= 4 ? 4 : 6)'
```

Note that brackets were added to this expression to improve readability – they are not mandatory.

6.1.6 Arrays and Maps

An array is a sequence of entries that are indexed by an integer value starting with 1. A map is similar to an array except that its entries are indexed by a key (usually a string) instead of an integer. The entries in an array or map can be numbers, strings, entities, or other arrays or maps. Arrays and maps are sometimes referred to as 'Collections'.

Table 6-8 Arrays and Maps

Entry	Rule	Examples

Array	Arrays are enclosed by curly braces, with individual entries separated by commas.	`{5, 6, 7}`
		`{5[m], 6[m], 7[m]}`
		`{"a", "b", "c"}`
		`{[Queue1], [Queue2]}`
		`{{1, 2}, {3, 4}}`

Map	Maps are enclosed by curly braces with individual entries separated by commas. Each entry consists of a string (the key) followed an equal sign and the value for that key.	`{"a"=5, "b"=6, "c"=7}` `{"Q1"=[Queue1],"Q2"=[Queue2]}`

Arrays and maps can be referenced in an expression using the following syntax.

Table 6-9 Syntax for Arrays and Maps

Rule	Example	Result
Entries in an array are referenced by specifying an index. The index value can be either a constant or an expression that returns a dimensionless number. A non-integer value for the index will be truncated.	`'[Queue1].QueueList(2)'` `'{5,6,7}(2)'`	Second entity in the queue. `6`
Entities in a map are referenced by specifying a key (usually a string).	`'[Server1].StateTimes("Idle")'` `'{"a"=5, "b"=6, "c"=7}("b")'`	Total time Server1 has been in its Idle state. `6`
Entries in a nested array are referenced by providing multiple indices.	`'{{5,6}, {7,8}}(2)(1)'`	`7`
Arrays and maps can be concatenated and appended using the + operator.	`'{1,2,3} + {4,5,6}'` `'{1,2,3} + "abc"'`	`{1,2,3,4,5,6}` `{1,2,3,"abc"}`

A number of functions are provided that take an array or map as an argument.

Table 6-10 Functions of Arrays and Maps

Function	Description	Example	Result
indexOf	Index or key of the specified entry in an array or map. Zero is returned if the value is not included in the array or map.	`'indexOf( {5, -1, 2}, -1)'` `'indexOf( {5, -1, 2}. 3)'`	`2` `0`
indexOfMinCol	Index or key of the smallest entry in an array or map. If this value appears in several entries, the index of the first one is returned.	`'indexOfMinCol( {5, -1, 2} )'`	`2`
indexOfMaxCol	Index or key of the largest entry in an array or map.	`'indexOfMaxCol( {5, -1, 2} )'`	`1`
indexOfNearest	Index or key of the entry in an array or map that is closest to the specified value.	`'indexOfNearest({5,-1,2}, 1.5)'`	`3`
maxCol	Largest entry in an array or map.	`'maxCol( {5, -1, 2} )'`	`5`
minCol	Smallest entry in an array or map.	`'minCol( {5, -1, 2} )'`	`-1`

range	Generates an array of numerical values that can be used as an input to a higher-order function.	`'range(3)'`
		`'range(2, 4)'`
		`'range(2, 3, 0.5)'`
		`'range(2, 1)'`
			`{1,2,3}`
			`{2,3,4}`
			`{2,2.5,3}`
			`{}`

size	Number of entries in an array or map.	`'size( {5, -1, 2} )'`	`3`
sum	Returns the sum of the entries in an array or map that contains numbers with or without units.	`'sum( {5, -1, 2} )'` `'sum( {"a"=5, "b"=-1, "c"=2} )'`	`6` `6`

When several entries in an array have the same value, the functions `indexOfMinCol`, `indexOfMaxCol`, `indexOf` and `indexOfNearest` return the first index that satisfies the condition. When several entries in a map have the same value, the index returned by these functions is the first one that was entered in the map.

6.1.7 Lambda Functions and Higher-Order Functions

A lambda function is an expression that takes one or more input variables and returns a number, string, object, array, map, or another lambda function. Lambda functions can be used with higher-order functions to perform complex calculations that would otherwise require a loop structure.

Table 6-11 Lambda Functions

Entry	Rule	Examples
Lambda Function	Input variables are enclosed by bars and separated by commas. The expression that generates the returned value is enclosed by brackets. Input variables can be a number, string, array, map, or another lambda function. The object returned can be any of these same types of objects.	`|x,y|(x + y)`

Table 6-12 Syntax for Lambda Functions

Rule	Example	Result

A lambda function can be evaluated by providing input values enclosed by brackets.	`|x|(2*x)(5)`
	`|x,y|(x+2*y)(1,2)`
	`|x,y|(x+y)("abc","def")`
		`10`
		`5`
		`abcdef`

A lambda function with multiple inputs can be turned into one with fewer inputs by providing one or more of the inputs	`|x,y|(x+2*y)(1)`	`|y|(1+2*y)`

A higher-order function is one that takes a lambda function as an argument.

Table 6-13 Higher-Order Functions

Function	Description	Example	Result
map	Applies a one-input lambda function to each element of an array and returns an array with the resulting values. If a two-input lambda function is specified, the second input is the index of the element in the array.	`map(|x|(2*x),{1,2})` `map(|x,i|(2*i),{5,8}`	`{2,4)` `{2,4)`
filter	Applies a one-input lambda function to each element of an array and returns an array with only the ones that return TRUE (i.e. a non-zero number). If a two-input lambda function is specified, the second input is the index of the element in the array.	`filter(|x|(x>2),{1,2,3,4})` `filter(|x,i|(i>2),{5,6,7,8})`	`{3,4}` `{7,8}`

Function	Description	Example	Result

reduce	Applies the first input of a two-input lambda function to each element of an array. The second input to the reduce function is the initial value for an internal value maintained by the function during the calculation. The result of the calculation for each element is assigned to this internal value. After the last element is processed, the internal value is returned. The reduce function has three inputs: the function, the initial value to be assigned to the internal value, and the array to be processed.	`reduce(|x,accum|(x+accum), 0, {1,2,3})`
		
		`reduce(|x,accum|(max(x,accum)), 0, {1,2,3})`
		
		`reduce(|x,accum|(x||accum), 0, {0,1,0})`
			`6`
			
			`3`
			
			`1`

sort	Applies a two-input lambda function to the elements of an array and returns an array that has been re-ordered so that the lambda function returns TRUE (i.e. a non-zero number) for each adjacent pair of elements. The lambda function should return FALSE for entries that are equal.	`sort(|x,y|(x>y),{2,3,1})`	`{3,2,1}`

6.2 Attributes and Custom Outputs

Users can define two types of variables for individual objects in a model:

- **Attribute**. A variable whose value can be changed be changed by one or more Assign objects in a model (see Section 0). Attributes are a useful way to add application-specific information to generated entities or to the permanent objects in a model. For example, if there are two types of customers in a model that require different service times, an attribute named "type" can be added to the generated customer objects and randomly assigned the value 1 or 2. When the customer arrives at the server, its service time can then be calculated based on its type attribute.

- **Custom Output**. A variable whose value is calculated on demand by evaluating a specified expression during the simulation run. Custom outputs are a way for the user to supplement the JaamSim's built-in outputs or to avoid repeating a complicated calculation in more than one expression.

Attributes and custom outputs can have same types of values as expressions, i.e. numbers with or without units, strings, entities, arrays, maps, or lambda functions. They appear in the Output Viewer as outputs under the Entity heading.

The name for an attribute or custom output can be any alphanumeric text provided that no spaces, commas, or special symbols are included.

Attributes and custom outputs can be defined for any of the objects in a model using the AttributeDefinitionList and CustomOutputList keywords described in the following table.

Table 6-14 Keywords for Attributes and Custom Outputs

Keyword	Description

AttributeDefinitionList	Defines one or more attributes for the object. The following format is used for the input:
	
	`{ <Name1> <InitValue1> } ... { <NameN> <InitValueN> }`
	
	where `Name` is the name of the attribute and `InitValue` is an expression that returns the initial value for the attribute. The expression is evaluated when the simulation is first started or re-started. Normally, the expression is a simple constant, such as `5[km]`. The attribute type (number, string, etc.) is determined by its initial value. Do not reference any attributes, outputs, or custom outputs of any object in the InitValue expression.
	
	The following example defines a series of attributes of various types:
	
	`{ A 1 } { B 9[km] } { C '"abc"' } { D [Server1] } { E '{5,3,7}' }`

CustomOutputList	Defines one or more custom outputs for the object. The following format is used for the input:
	
	`{ <Name1> <Exp1> <Unit1> } ... { <NameN> <ExpN> <UnitN> }`
	
	where `Name` is the name of the custom output, `Exp` is the expression to be evaluated on demand, and `Unit` is the unit type for the output.
	
	For example, the following example defines a custom output called `TwiceSimTime`, whose value is equal to two times the present simulation time:
	
	`{ TwiceSimTime '2 * this.SimTime' TimeUnit }`

7 Simulation Runs and Experiments

7.1 Simulation Object

The Simulation object is used to store inputs that define basic parameters of the model, such as run duration. The Simulation object is created automatically when a new model is started, and thus does not appear in the Model Builder.

Table 7-1 Simulation Inputs

Keyword	Description
Key Inputs	
RunDuration	Duration of the simulation run in which statistics will be recorded.
InitializationDuration	The initialization interval for the simulation run. The model will run for the InitializationDuration interval and then clear the statistics and execute for the specified RunDuration interval. The total length of the simulation run will be the sum of the InitializationDuration and RunDuration inputs.
ExitAtStop	If TRUE, the program will be closed on completion of the last simulation run. Otherwise, the last run will be paused.
GlobalSubstreamSeed	Global seed that sets the sub-stream for each probability distribution. Must be an integer >= 0. GlobalSubstreamSeed works together with each probability distribution's RandomSeed keyword to determine its random sequence. It allows the user to change all the random sequences in a model with a single input. To run multiple replications, set the appropriate inputs under the Multiple Runs tab and then set the GlobalSubstreamSeed input to `[Simulation].RunNumber` or to `[Simulation].RunIndex(N)`, where N is a suitable integer.
PrintReport	If TRUE, a full output report is printed to the file `<configuration file name>.rep` at the end of the simulation run.
ReportDirectory	The directory where output files will be written. The default location is the directory of the input configuration file.
RunOutputList	One or more selected outputs to be printed at the end of each simulation run. Each output is specified by an expression. In script mode (-s tag), the selected outputs are printed to the command line (standard out). Otherwise, they are printed to the file `<configuration file name>.dat`. It is best to include only dimensionless quantities and non-numeric outputs in the RunOutputList. An output with dimensions can be made non-dimensional by dividing it by 1 in the desired unit, e.g. `'[Queue1].AverageQueueTime / 1[h]'` is the average queue time in hours. A dimensional number will be displayed along with its unit. The `format` function can be used if a fixed number of decimal places is required.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.

Keyword	Description
GeorgianCalendar	If TRUE, the simulation uses the standard Gregorian calendar that includes leap years. If FALSE, the simulation uses a simplified calendar that has a fixed 365 days per year.
StartDate	The calendar date and time that corresponds to zero simulation time.
PauseCondition	An optional expression that pauses the run when TRUE is returned.
ExitAtPauseCondition	If TRUE, the simulation run will be terminated when the PauseCondition expression returns TRUE. If multiple runs have been specified, then the next run will be started. If no more runs have been specified, the simulation will be paused or terminated depending on the input to the ExitAtStop keyword.
MaxEntitiesToDisplay	The maximum number of entities to display in the view windows. A model can contain more than this number of entities, but only this number will be displayed. This limit prevents JaamSim from becoming unresponsive when the number of entities in a model exceeds the graphics capabilities of the computer.
TickLength	The smallest time increment for JaamSim's internal integer-based time keeping. The default value of 1 microsecond will support simulation runs of more than 100 years.
Multiple Runs	
RunIndexDefinitionList	Defines the number of run indices and the maximum value N for each index. When making multiple runs, each index will be iterated from 1 to N starting with the last index. One run will be executed for every combination of the run index values. For example, if three run indices are defined with ranges of 3, 5, and 10, then at total of 3*5*10 = 150 runs will be executed.
StartingRunNumber	The first run number to be executed. The value can be entered as either an integer or as the equivalent combination of run indices. For example, if there are three run indices with ranges of 3, 5, and 10, then run number 22 can be expressed as 1-3-2 because 22 = (1-1)*5*10 + (3-1)*10 + 2.
EndingRunNumber	The last run number to be executed. The value can be entered as either an integer or as the equivalent combination of run indices. For example, if there are three run indices with ranges of 3, 5, and 10, then run number 78 can be expressed as 2-3-8 because 78 = (2-1)*5*10 + (3-1)*10 + 8.

7.2 Performing Multiple Simulation Runs

Multiple simulation runs can be executed automatically, one after another, using the keywords under the Multiple Runs tab for the Simulation object. The Simulation outputs RunNumber and RunIndex are used to change selected inputs between simulation runs. By default, the RunNumber output starts at 1 and is incremented by one with each simulation run that is performed. This output can be used to vary one or more inputs by referencing `[Simulation].RunNumber` in an expression. For example, setting the ServiceTime input for a Server to the following expression:

```
'1[s] + 0.1[s]*[Simulation].RunNumber'
```

assigns the service time to 1.1 s, 1.2 s, 1.3 s, etc., as the run number is incremented over multiple runs.

The RunIndex output is used when there are multiple inputs to test. This output contains an array of integers that are each incremented from 1 to N, where a separate value for N can be specified for each index. The number of run indices and the ranges over which they are incremented are determined by the RunIndexDefinitionList keyword.

For example, suppose there are two Servers and service times of 1.1 s, 1.2 s, and 1.3 s are to be tested for Server1 and service times of 2.1 s and 2.2 s are to be tested for Server2. Ten replications are to be made for each combination of service times. In this case, three run indices are needed – one for each variable that is to be changed. The run indices are defined by entering the values `3 2 10` to the RunIndexDefinitionList keyword. This input indicates that `RunIndex(1)` will be incremented over the range 1 to 3, `RunIndex(2)` will be incremented over 1 to 2, and that `RunIndex(3)` will be incremented over 1 to 10. The three run indices are used in the model inputs as follows:

- ServiceTime keyword for Server1: `'1[s] + 0.1[s]*[Simulation].RunIndex(1)'`
- ServiceTime keyword for Server2: `'2[s] + 0.1[s]*[Simulation].RunIndex(2)'`
- GlobalSubstreamSeed keyword for Simulation: `'[Simulation].RunIndex(3)'`

With these inputs, a total of sixty runs would be performed with the run indices incremented in the following sequence:

- `1-1-1, 1-1-2, … 1-1-10,`
- `1-2-1, 1-2-2, … 1-2-10,`
- `2-1-1, 2-1-2, … 2-1-10,`
- `2-2-1, 2-2-2, … 2-2-10,`
- `3-1-1, 3-1-2, … 3-1-10,`
- `3-2-1, 3-2-2, … 3-2-10,`

The notation `i-j-k` indicates run indices `RunIndex(1) = i`, `RunIndex(2) = j`, and `RunIndex(3) = k`.

It is not necessary to perform all the simulation runs defined by the run indices. The Simulation keywords StartingRunNumber and EndingRunNumber can be used to determine the runs that will be performed. To perform all sixty runs, the StartingRunNumber should be set to `1` (the default value) and the EndingRunNumber should be set to `60`.

Run indices are related to the run number by a mathematical equation that performs the necessary transformation. In the example given above, the RunNumber increases from 1 to 60, at the same time as the run indices increase from `1-1-1` to `3-2-10`. Run numbers and run indices in the `i-j-…-k` notation can be used interchangeably for the StartingRunNumber and EndingRunNumber inputs. For example, to perform all sixty runs, the StartingRunNumber could be set to `1-1-1` and EndingRunNumber to `3-2-10` instead of `1` and `60`.

7.3 Customized Output Report

When making multiple runs, the output reports for the runs are appended one after another in a single file. If more than just a few runs are to be made, it is best to create a customized output report using the RunOutputList keyword for Simulation. This keyword allows the user to specify the set of outputs needed to analyse the results from the runs. Each output is entered as a separate expression enclosed by curly braces. For example, if the desired outputs are the simulation run number, average utilisation of Server1, and average waiting time in hours for Queue1, then the input to the RunOutputList keyword would be as follows:

```
{ [Simulation].RunNumber } { [Server1].Utilisation } { [Queue1].AverageQueueTime/1[h] }
```

To make the custom output file easier to input to Excel, it is best to record only dimensionless values such as `[Queue1].AverageQueueTime/1[h]` in the above example. In this way, the value will be converted into the desired unit (hours in this case) and the unit will not appear in the entries for this column.

Prepare this input using the Input Builder, which can be opened from the dropdown menu in the Input Editor.

At the end of each run, the specified outputs are collected in a single output file with the name <configuration file name>.dat. The file has one row for each simulation run that was performed and is tab-delimited, which allows it to be imported directly into Excel for analysis.

The RunOutputList report can be generated with or without the main report, as specified by the PrintReport keyword.

8 Configuration File

The easiest way to create a simple model in JaamSim is to use the Graphical User Interface. However, once a model becomes more complex, it is often easier to edit the configuration file (CFG file) in a text editor. The configuration file is saved in plain text and has been designed to be human readable.

There are many advantages to a readable input file in plain text:

- Inputs can be easily reviewed and audited.
- Standard software for change control such as GIT can be used to track model inputs.
- Software for performing simulation experiments and optimisation can be developed by third-parties in other programming languages such as Python.

The recommended text editor is Notepad++, an open-source editor available for download at www.notepad-plus-plus.org.

8.1 Basic Structure

A JaamSim input configuration file consists of a series of lines, akin to a scripting language. Each line consists of a combination of object names, keywords, and values contained within braces. One or more spaces are used to separate these elements. Braces are also used to denote sets of arguments within the outer braces required for arguments in general. Blank lines are ignored by JaamSim.

Lines beginning with a hash mark ('#') can be used to create comments to document the input files. If a comment extends for several lines, each line must start with a hash mark.

8.2 Object Definitions

In JaamSim, an object is initialized by a Define statement. The statement contains Define followed by the object type, and the object name enclosed by braces. Multiple objects can be defined at the same time, provided that they are of the same type. For instance, the following two lines define respectively a single Arrow object and three Arrow objects.

```
Define Arrow  { SingleArrow }
Define Arrow  { Arrow1 Arrow2 Arrow3 }
```

Object instances can only be referenced after they have been defined.

8.3 Object Inputs

Once an object is defined, its keyword values can be set using a command of the following form:

```
<object name> <keyword> { <value1> <value2> … }
```

where value1, value2, … is the list of values for the keyword separated by one or more spaces. For instance, the following line sets the colour of the Arrow1 object to be black:

```
Arrow1 Colour { black }
```

Multiple parameters for an object can be set in one line containing the object name followed by keyword and value pairs.

```
Arrow1 Colour { black } Width { 2 }
```

Inner braces are used for keywords that accept multiple input values.

```
Arrow1 Points { { 0 0 0 m } { 1 1 1 m } }
```

8.4 Using Multiple Lines

The readability of a configuration file can often be improved by spreading lengthy inputs over multiple lines. This can be done for an input that accepts an expression and for inputs that use inner braces, such as the RunOutputList keyword for Simulation and the Value keyword for TimeSeries.

For inputs that accept an expression, only the contents of the expression within the single quote marks can be spread over multiple lines. For example,

```
Server1  ServiceTime  { 'this.obj.type == 1 ? 5[s] #some comment#
                                            : 2[s] #another comment#' }
```

For inputs that use inner braces, new lines can start only between on pair of inner braces and the next. For example:

```
Simulation RunOutputList { { [Simulation].RunNumber }
                           { [Server1].Utilisation }
                           { [Queue1].AverageQueueTime/1[h] } }
```

8.5 Include Statements

The user can store input data in multiple files and then refer to these files in an input configuration file using Include statements. These statements refer to other input configuration files by filename and path, surrounded by single quotes:

```
Include  '..\Base File\InputFile.cfg'
```

Include statements are particularly useful when only a few inputs are varied across many simulation runs. Include statements can be used to create incremental configuration files for additional runs that contain a base case configuration file:

```
Include  '..\Base File\Basecase.cfg'
Arrow1 Width { 2.0 }
```

This example includes the contents of Basecase.cfg and modifies the already-defined object Arrow1's keyword Width value to 2.0. Note that the changes from the base case configuration must appear after the Include statement. These simple configuration files are useful because it is easy to tell exactly how the configuration differs from the base case configuration.

8.6 Groups

Group objects bundle multiple objects together to simplify inputs. Instead of referring to a long list of objects, a single Group can be used instead. The Group may be used to set the value for a keyword

for all members instead of setting the value for each member of the Group. Certain keywords also accept Group objects as values.

Table 8-1 Group Inputs

Keyword	Description
List	A list of names of the objects included in this list, enclosed by braces.

The following example demonstrates the use of Groups:

```
Define     Arrow       { Arrow1 Arrow2 Arrow3 }
Define     Group       { ArrowList }
ArrowList  List        { Arrow1 Arrow2 Arrow3 }
ArrowList  Colour      { black }
```

In this example, a Group of three Arrow objects is created and each Arrow is set to the colour black.

By using the List keyword, a fourth Arrow can be added to the Group:

```
Define     Arrow       { Arrow4 }
ArrowList  List        { ArrowList Arrow4 }
ArrowList  Colour      { black }
```

8.7 RecordEdits Statement

The RecordEdits statement is used to preserve the organisation and formatting of a configuration file that has been prepared manually by the user.

It is usually best to construct a complex model manually using a text editor. These inputs are carefully formatted and organised, and include comments to document model design. However, once this material has been created, the easiest way to position the objects and to add graphics such as titles, labels, etc. is through the graphical user interface. If the model is then saved, all the formatting and comments would normally be lost.

JaamSim avoids this predicament with the RecordEdits statement. On saving, JaamSim copies all inputs before the RecordEdits statement line-by-line to the saved file, and then saves all the changes to the model using computer-written inputs. The following example illustrates this structure:

```
# Manually prepared inputs:
# - Everything before the RecordEdits statement is unchanged when JaamSim saves a
file.

RecordEdits

# Computer written inputs:
# - Everything that appears after the RecordEdits statement is written by the
computer.
```

The Save functionality in JaamSim is disabled when an input file is loaded without a RecordEdits statement. In this case, the Save As operation adds a RecordEdits statement to the end of the original configuration file and then writes out the new inputs.

8.8 Example Configuration File

The configuration file for the Basic Example described in Section 3 is given below.

```
RecordEdits

Define ColladaModel { Grid100x100  Axis }
Define DisplayEntity { XY-Grid  XYZ-Axis }
Define View { View1 }
Define TextModel { TitleTextModel  ClockTextModel }
Define OverlayText { Title }
Define OverlayClock { Clock }
Define SimEntity { Proto }
Define EntityGenerator { Gen }
Define EntityConveyor { GenToServ  ServToSink }
Define Server { Serv }
Define EntitySink { Sink }
Define Queue { ServQueue }
Define ExponentialDistribution { GenIATDist }

GenIATDist UnitType { TimeUnit }

Simulation Description { 'Simulation run control inputs' }
Simulation SnapToGrid { TRUE }
Simulation RealTime { TRUE }
Simulation RealTimeFactor { 2048 }
Simulation PauseTime {  }
Simulation ShowModelBuilder { TRUE }
Simulation ShowObjectSelector { TRUE }
Simulation ShowInputEditor { TRUE }
Simulation ShowOutputViewer { TRUE }
Simulation ShowPropertyViewer { FALSE }
Simulation ShowLogViewer { FALSE }

Grid100x100 ColladaFile { <res>/shapes/grid100x100.dae }

XY-Grid Description { 'Grid for the X-Y plane (100 m x 100 m)' }
XY-Grid Size { 100  100  m }
XY-Grid DisplayModel { Grid100x100 }
XY-Grid Movable { FALSE }

Axis ColladaFile { <res>/shapes/axis_text.dae }

XYZ-Axis Description { 'Unit vectors' }
XYZ-Axis Alignment { -0.4393409  -0.4410096  -0.4394292 }
XYZ-Axis Size { 1.125000  1.1568242  1.1266404  m }
XYZ-Axis DisplayModel { Axis }
XYZ-Axis Show { FALSE }
XYZ-Axis Movable { FALSE }

View1 Description { 'Default view window' }
View1 ViewCenter { -0.299610  -2.582932  2.866546  m }
View1 ViewPosition { -0.299610  -2.582933  11.526800  m }
View1 ShowWindow { TRUE }
View1 SkyboxImage { <res>/images/sky_map_2048x1024.jpg }

TitleTextModel Description { 'Text style for the Title' }
TitleTextModel FontColour { 150  23  46 }
TitleTextModel FontStyle { BOLD }

ClockTextModel Description { 'Text style for the Clock' }
ClockTextModel FontColour { 51  51  51 }
ClockTextModel FontStyle { ITALIC }

Title Description { 'Title for the simulation model' }
Title TextHeight { 18 }
Title Format { 'Getting Started with JaamSim' }
Title Position { 0.000000  0.000000  0.000000  m }
```

```
Title DisplayModel { TitleTextModel }
Title ScreenPosition { 15  15 }

Clock Description { 'Simulation date and time (no leap years or leap seconds)' }
Clock TextHeight { 10 }
Clock StartingYear { 2014 }
Clock DateFormat { 'yyyy-MMM-dd HH:mm:ss.SSS' }
Clock DisplayModel { ClockTextModel }
Clock ScreenPosition { 15  15 }
Clock AlignBottom { TRUE }

Proto Position { -4.400000  -0.600000  0.000000  m }
Proto Alignment { 0.0  0.0  -0.5 }

Gen NextComponent { GenToServ }
Gen InterArrivalTime { GenIATDist }
Gen PrototypeEntity { Proto }
Gen Position { -3.500000  -2.500000  0.000000  m }

GenToServ NextComponent { Serv }
GenToServ TravelTime { 1  s }
GenToServ Position { -2.500000  -2.500000  0.000000  m }
GenToServ Points { {  -2.500  -2.500  0.000  m  } {  -1.500  -2.500  0.000  m  } }

Serv NextComponent { ServToSink }
Serv WaitQueue { ServQueue }
Serv ServiceTime { 1  s }
Serv Position { -0.500000  -2.500000  0.000000  m }

ServToSink NextComponent { Sink }
ServToSink TravelTime { 1.5  s }
ServToSink Position { 0.600000  -2.500000  0.000000  m }
ServToSink Points { {  0.600  -2.500  0.000  m  } {  1.600  -2.500  0.000  m  } }

Sink Position { 2.500000  -2.500000  0.000000  m }

ServQueue Position { -0.500000  -4.500000  0.000000  m }

GenIATDist RandomSeed { 1 }
GenIATDist MaxValue { 10  s }
GenIATDist Mean { 2  s }
GenIATDist Position { -2.500000  -0.500000  0.000000  m }
```

9 Application Programming Interface

The Application Programming Interface (API) allows a JaamSim model to be constructed and/or launched from another program.

The API is implemented through the JaamSimModel class. There is one constructor for this class:

JaamSimModel() Constructs a new instance of the JaamSimModel class.

The following methods can be used.

Table 9-1 Method Summary for JaamSimModel

Modifier and Type	Method and Description
void	**autoLoad()** Pre-loads the simulation model with basic objects such as Simulation and Units.
void	**configure(File** file**)** Loads the specified configuration file to create the objects in the model. The autoLoad() method must be executed first.
void	**start()** Starts the simulation model on a new thread.
void	**pause()** Suspends model execution at the present simulation time.
void	**resume(**double simTime**)** Resumes a paused simulation model. The model will continue execution until the specified simulation time at which the model will be paused. Events scheduled at the next pause time will not be executed until the model is resumed.
void	**reset()** Sets the simulation time to zero and re-initialises the model. The start() method can be used to begin a new simulation run.
void	**clear()** Deletes all the objects in the present model and prepares the JaamSimModel to load a new input file using the autoLoad() and configure() methods.
double	**getSimTime()** Returns the present simulation time in seconds.
boolean	**isRunning()** Returns true if the simulation is executing.
void	**defineEntity(String** type**, String** name**)** Creates a new entity with the specified type and name. If the name already used, "_1", "_2", etc. will be appended to the name until an unused name is found.
void	**setInput(String** entName**, String** keyword**, String** arg**)** Sets the input for the specified entity and keyword to the specified string.
String	**getStringValue(String** expString**)** Evaluates the specified expression and returns its value as a string. Any type of result can be returned by the expression, including an entity or an array. If it returns a number, it must be dimensionless.

Modifier and Type	Method and Description
double	**getDoubleValue(String** expString**)** Evaluates the specified expression and returns its value. The expression must return a dimensionless number. All other types of expressions return NaN.
void	**save(File** file**)** Writes the inputs for the simulation model to the specified file.

Also provided are 'setValue' methods for the FileToVector, FileToMatrix, and FileToHashMap classes. The setValue method sets the data for these objects directly from a Java data structure, without the use of the DataFile input which can be left blank.

10 Maintenance, Breakdowns, and Thresholds

A significant feature of JaamSim is its ability to model planned maintenance, breakdowns, and process blockages, and to record the total time the process spends in each state. This feature allows a JaamSim model to combine a full Reliability, Availability, and Maintainability (RAM) analysis, normally done separately with specialised software, with a typical logistics model.

10.1 Thresholds

Process blockages are modelled by Threshold objects that open and close according to various types of rules:

- TimeSeriesThreshold. Opens and closes when the value provided by a TimeSeries object meets various conditions.
- ExpressionThreshold. Opens and closes according to a specified expression that returns TRUE or FALSE.
- SignalThreshold. Opens and closes when an entity is received by an EntitySignal object.

All types of Threshold objects share the following inputs and outputs.

Table 10-1 Threshold Inputs

Keyword	Description
OpenColour	The colour of the threshold graphic when the threshold is open.
ClosedColour	The colour of the threshold graphic when the threshold is closed.
ShowWhenOpen	A Boolean value. If TRUE, the threshold is displayed when it is open.
ShowWhenClosed	A Boolean value. If TRUE, the threshold is displayed when it is closed.

Table 10-2 Threshold Outputs

Output Name	Description
Open	If open, then return TRUE. Otherwise, return FALSE.
OpenFraction	The fraction of total simulation time that the threshold is open.
ClosedFraction	The fraction of total simulation time that the threshold is closed.
OpenCount	The number of times the threshold's state has changed from closed to open.
ClosedCount	The number of times the threshold's state has changed from open to closed.

The respond of an object to the closure of one of its Thresholds depends on which one of the following keywords was used.

Table 10-3 Keywords that accept Thresholds

Keyword	Description
ImmediateThresholdList	A list of thresholds that must be satisfied for the object to operate. Operation is stopped immediately when one of the thresholds closes. If a threshold closes part way though processing an entity, the work is considered to be partly done and the remainder is completed once the threshold re-opens.
ImmediateReleaseThresholdList	A list of thresholds that must be satisfied for the object to operate. Operation is stopped immediately when one of the thresholds closes. If a threshold closes part way though processing an entity, the work is interrupted and the entity is released.
OperatingThresholdList	A list of thresholds that must be satisfied for the object to operate. If a threshold closes part way though processing an entity, the remaining work is completed and the entity is released before the object is closed.
ReleaseThresholdList	A list of thresholds that must be satisfied for the object to operate. If a threshold closes part way though processing an entity, the remaining work is completed, but the entity is retained until the threshold re-opens.

A given Threshold can appear in only one of these keywords at a time. A single Threshold can be used by multiple objects.

10.2 Maintenance and Breakdowns

Planned maintenance and breakdowns are modelled using the DowntimeEntity (Section 0), which generates random or scheduled events based on either working time or calendar time. Normally, a maintenance activity is scheduled to occur at regular intervals based on calendar time. Breakdowns are normally modelled to occur randomly based on the working time for the object.

The object that will undergo the maintenance or breakdown has a series of keywords that determine how the object responds to a particular maintenance or breakdown event.

Table 10-4 Maintenance and Breakdowns Inputs

Keyword	Description
WorkingStateList	A list of states for which the entity is considered to be working.
ImmediateMaintenanceList	A list of DowntimeEntities representing planned maintenance that must be performed immediately, interrupting any work underway at present.
ForcedMaintenanceList	A list of DowntimeEntities representing planned maintenance that must begin as soon as task underway at present is finished.
OpportunisticMaintenanceList	A list of DowntimeEntities representing planned maintenance that can wait until task underway at present is finished and the queue of tasks is empty.
ImmediateBreakdownList	A list of DowntimeEntities representing unplanned maintenance that must be performed immediately, interrupting any work underway at present.
ForcedBreakdownList	A list of DowntimeEntities representing unplanned maintenance that must begin as soon as task underway at present is finished.
OpportunisticBreakdownList	A list of DowntimeEntities representing unplanned maintenance that can wait until task underway at present is finished and the queue of tasks is empty.

Only one breakdown or maintenance activity can occur at any given time. If multiple breakdown/maintenance activities are scheduled for the same time, they are performed sequentially. It is possible under some circumstances for a breakdown or maintenance to occur while a threshold is closed. In this case, the state is set to the appropriate breakdown or maintenance state.

The outputs for maintenance, breakdowns, and thresholds are grouped together as follows.

Table 10-5 Maintenance, Breakdowns, and Thresholds Outputs

Output Name	Description
Idle	Returns TRUE if able to work but there is no work to perform. For an EntitySystem, TRUE is returned if all the entities in the system are idle.
Working	Returns TRUE if work is being performed. For an EntitySystem, TRUE is returned if any of the entities in the system are working.
Setup	Returns TRUE if setup is being performed. For an EntitySystem, TRUE is returned if setup is being performed on any of the entities in the system.
Maintenance	Returns TRUE if maintenance is being performed. For an EntitySystem, TRUE is returned if maintenance is being performed on any of the entities in the system.
Breakdown	Returns TRUE if a breakdown is being repaired. For an EntitySystem, TRUE is returned if a breakdown is being repaired on any of the entities in the system.
Stopped	Returns TRUE if an operating limit prevents work from being performed. For an EntitySystem, TRUE is returned if an operating limit prevents work from being performed on any of the entities in the system.
Utilisation	The fraction of calendar time (excluding the initialisation period) that this object is in the Working state. Includes any completed cycles.
Commitment	The fraction of calendar time (excluding the initialisation period) that his object is in any state other than Idle. Includes any completed cycles.
Availability	The fraction of calendar time (excluding the initialisation period) that this object is in any state other than Maintenance or Breakdown. Includes any completed cycles.
Reliability	The ratio of Working time to the sum of Working time and Breakdown time. All times exclude the initialisation period and include any completed cycles.
Open	Returns TRUE if all the thresholds specified by the ThresholdList keywords are open.
NextMaintenanceTime	The estimated time at which the next maintenance activity will start.
NextBreakdownTime	The estimated time at which the next breakdown will occur.

10.3 States

Objects implementing states have the following keywords and outputs.

Table 10-6 States Inputs

Keyword	Description
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`

Table 10-7 States Outputs

Output Name	Description
State	The present state for this object, used for statistics collection. Typical states are: • **Working**. The object is performing its normal process. • **Idle**. The object is available for work, but has work to perform. • **Stopped**. One or more of the Thresholds are closed and there is no maintenance being performed or breakdown being repaired. • **Maintenance**. One of the DowntimeEntities entered to the MaintenanceList keywords is active. • **Breakdown**. One of the DowntimeEntities entered to the BreakdownList keywords is active. Additional states can be defined for some objects.
WorkingState	Set to true if the present state is one of the pre-defined working states.
WorkingTime	Total time spent in any one of the working states.
StateTimes	Total time spent in each of the states.
TotalTime	Total time the entity has spent in the model after the completion of the initialisation period. It is equal to the sum of the state times.

11 Graphics

11.1 DisplayEntity

The DisplayEntity is the basic 3D graphical object in JaamSim. All JaamSim objects that have graphics are subclasses of DisplayEntity.

All objects intended for visualization in a model display window in JaamSim have a set of basic graphics keywords used to define their appearance. These are found in the Graphics tab of the Input Editor when the object is selected. For polyline type objects the inputs for Position, Alignment, Size, and Orientation are replaced by those for Points and CurveType.

Table 11-1 DisplayEntity Inputs

Keyword	Description
Normal Objects	
Position	The location of the object in (x, y, z) coordinates.
Alignment	The point within the object that is located at the coordinates of its Position input. Expressed with respect to a unit box centered about (0 0 0).
Size	The size of the object in (x, y, z) coordinates. If only the x- and y-dimensions are given then the z-dimension is assumed to be zero.
Orientation	Euler angles defining the rotation of the object.
Polyline Type Objects	
Points	A list of points in (x, y, z) coordinates that define a polyline. When only two coordinates are given it is assumed that z = 0.
CurveType	The type of curve interpolation used for line type entities.
All Objects	
Region	If a Region is specified, the Position and Orientation inputs for the present object are relative to the Position and Orientation of the specified Region. If the specified Region is moved or rotated, the present object is moved to maintain its relative position and orientation.
RelativeEntity	If an object is specified, the Position input for the present object is relative to the Position for the specified object. If the specified object is moved, the present object is moved to maintain its relative position.
DisplayModel	The graphic representation of the object. If a list of DisplayModels is entered, each one is displayed provided that its DrawRange input is satisfied. This feature allows the object's appearance to change with its distance from the View window's camera.
Show	If TRUE, the object is shown in the View windows.
Movable	If TRUE, the object will respond to mouse clicks and can be positioned by dragging with the mouse.
VisibleViews	The view windows on which this entity will be visible.
DrawRange	The minimum and maximum distance from the camera for which this entity is displayed.

Table 11-2 DisplayEntity Outputs

Output Name	Description
Name	The unique input name for this entity.
ObjectType	The class of objects that this entity belongs to.
SimTime	The present simulation time.
Parent	The parent entity for this entity.
User-defined attributes and custom outputs	The present values for each of the attributes and custom outputs that were defined for the DisplayEntity.
Position	The present (x, y, z) coordinates of the DisplayEntity in its Region.
Size	The present (x, y, z) components of the DisplayEntity's size.
Orientation	The present (x, y, z) Euler angles of the DisplayEntity's rotation.
Alignment	The present (x, y, z) coordinates of a point on the DisplayEntity that aligns direction with the position output. Each component should be in the range [-0.5, 0.5].
GraphicalLength	Polyline type objects: the length of the polyline determined by its Points and CurveType inputs. Non-polyline type objects: the largest of the Size inputs.
ObserverList	The observers that are monitoring the state of this entity.
NextList	The entities that are immediately downstream from this entity.
PreviousList	The entities that are immediately upstream from this entity.

11.2 DisplayModel

The graphical appearance of a DisplayEntity and its subclasses is determined by its DisplayModel. The two objects work together to generate an object's display. In general, the DisplayEntity determines what is displayed, while its DisplayModel determines how it is displayed. A number of different subclasses of DisplayModel are available to match the various subclasses of DisplayEntity.

For example, the Text and TextModel objects work together to display text: the text to be displayed is determined by the Text object, while the style of the displayed text (font, bold, italics, colour, etc.) is determined by the TextModel. In this case, the TextModel plays the same role as a text style in word processing software. Users can ensure that the same text style is used for multiple Text objects by sharing the same TextModel between all these objects.

One DisplayModel can be shared between multiple DisplayEntities. This is an essential feature for the case of complex 3D content built from millions of triangles. In this case, a ColladaModel (a subclass of DisplayModel) stores 3D information that can be shared between multiple DisplayEntities. The 3D content is loaded and stored only once, even though it is displayed many times in various locations.

Although DisplayModels determine the appearance of a DisplayEntity, the DisplayModel has no graphics itself and therefore cannot be dragged and dropped in the normal manner. To create a new DisplayModel, simply duplicate an existing DisplayModel. JaamSim starts with a pre-defined example of each type of DisplayModel that can be used for this purpose. Duplicate an existing

DisplayModel by right-clicking its entry in the Object Selector and selecting Duplicate, then edit its keyword values and name as necessary.

A selection of DisplayModels can be found in the Display Models palette in the Object Selector. Each type of DisplayModel is intended for specific types of DisplayEntities.

Table 11-3 Types of DisplayModel

DisplayModel	Description	Usage
TextModel	A text style (font, colour, etc.).	Text, OverlayText, BillboardText, and OverlayClock objects.
PolylineModel	A series of line segments connected by nodes.	Selected sub-classes of DisplayEntity such as Polyline, Arrow, EntityConveyor, and EntityDelay.
ShapeModel	A flat geometric object such as a circle or rectangle.	DisplayEntity and all its sub-classes.
ImageModel	An imported picture.	DisplayEntity and all its sub-classes.
ColladaModel	An imported 3D object.	DisplayEntity and all its sub-classes.
GraphModel	Formatting inputs for graphs.	Graph object.

All of these DisplayModel objects have the same Graphics keywords that control optional rendering and scaling at different drawing ranges.

Table 11-4 DisplayModel Inputs

Keyword	Description
Graphics	
VisibleViews	A list of Views for which this DisplayModel is shown. If empty, the model appears on all Views.
DrawRange	A list of two values for the minimum and maximum distance from the camera this object is visible.
ModelScale	A list of three multiplicative factors by which to scale the model in the x, y and z dimensions respectively.

11.2.1 TextModel

TextModel objects specify the general appearance of Text objects and of overlay objects that display text (OverlayText, OverlayClock, and BillboardText).

The inputs to the TextModel keywords set the default values for the corresponding keywords for Text, OverlayText, etc. A TextModel can therefore be used as a style for these objects. For example, changing the FontName input for a TextModel will change the font shown by all the objects that use that TextModel as their DisplayModel input.

Table 11-5 TextModel Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
FontName	The font to be used for the text.
TextHeight	The height of the text as displayed in the view window.
TextHeightInPixels	The height of the text in pixels, used by billboard text and overlay text.
FontColour	The colour of the text.
FontStyle	The font styles to be applied to the text, e.g. Bold, Italic.
DropShadow	If TRUE, then a drop shadow appears for the text.
DropShadowColour	The colour for the drop shadow.
DropShadowOffset	The (x, y, z) coordinates of the drop shadow's offset, expressed as a decimal fraction of the text height.

11.2.2 PolylineModel

The PolylineModel is used to determine the graphics for polyline type objects that appear as a single-segment or multi-segment line, such as Polyline, Arrow, EntityConveyor, and EntityDelay. It cannot be used by other objects such as a DisplayEntity intended for 3D or 2D graphics.

The inputs to the PolylineModel keywords set the default values for the corresponding keywords for Polyline, Arrow, etc. A PolylineModel can therefore be used as a style for these objects. For example, changing the LineColour input for a PolylineModel will change the colour of the line shown by all the objects that use that PolylineModel as their DisplayModel input.

Table 11-6 PolylineModel Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Outlined	Determines whether or not the polyline is outlined. If TRUE, it is outlined with a uniform colour. If FALSE, it is drawn without an outline.
LineColour	The colour of the polyline.
LineWidth	The width of the polyline in pixels.
Filled	Determines whether the polyline is filled or hollow. If TRUE, then the polyline has a solid fill.
FillColour	The colour for the filled part of the polyline.
ShowArrowHead	If TRUE, an arrow head is displayed at the end of the polyline.
ArrowhHeadSize	A set of (x, y, z) numbers that define the size of the arrowhead.

11.2.3 ShapeModel

ShapeModel objects are used to display a 2D object whose geometry and colour can be adjusted.

The inputs to the ShapeModel keywords set the default values for the corresponding keywords for Shape, etc. A ShapeModel can therefore be used as a style for these objects. For example, changing the FillColour input for a ShapeModel will change the colour of the fill for all the objects that use that ShapeModel as their DisplayModel input.

Table 11-7 ShapeModel Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Shape	The graphical appearance of the object, chosen from a selection of predefined shapes.
FillColour	The colour of the filled part of the DisplayModel, defined by a colour name or RGB values.
OutlineColour	The colour of the outline of the DisplayModel, defined by a colour name or RGB values.
Filled	If TRUE, the DisplayModel will appear with a solid colour fill. Otherwise, the DisplayModel will appear hollow.
Bold	If TRUE, the DisplayModel outline will appear thicker than normal.

11.2.4 ImageModel

ImageModel objects are used to display custom 2D graphics in a simulation model, such as a picture or a map. Both DisplayEntity and OverlayImage can accept an ImageModel object as an input. Image file types currently supported by JaamSim include BMP, GIF, JPG, PCX and PNG files, or a ZIP file containing any one of these file types.

Table 11-8 ImageModel Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
ImageFile	A file path to the image file to be used for this DisplayModel. Must be enclosed in single quotes if the path contains spaces.
Transparent	If TRUE, transparency is enabled for supported image types (GIF and PNG).
CompressedTexture	If TRUE, image compression is applied to alleviate memory issues with large images.

11.2.5 ColladaModel

ColladaModel objects are used to display custom 3D graphics in a simulation model. The COLLADA file format (.DAE) is an interchange file format used for 3D graphics. Any 3D object such as a DisplayEntity and most of its sub-classes can accept a ColladaModel as its DisplayModel.

A number of other 3D formats can be used in addition to Collada. At the present time, JaamSim supports DAE, OBJ, and JSB formats as well as zipped versions of these files (ZIP). The JSB format is specific to JaamSim:

The JSB format is a binary format that allows complex 3D objects to be loaded much faster than is possible with the DAE and OBJ formats. A JSB file can be exported by right clicking on a ColladaModel in the Object Selector and selecting "Export 3D Binary File (*.jsb)". Note that the exported JSB will look for the same textures as the DAE file and that these should be located in the same relative position to the JSB file as they were to the DAE file.

Managing 3D assets can be complicated because of the large file sizes and the need to provide separate files for the textures. It is recommended that the user place each 3D asset and its texture files in its own ZIP file. This approach greatly reduces the number and size of the files being handled, and ensures that the files can be moved between computers without breaking the file paths. Note that it is often necessary to edit the DAE file with a text editor, such as Notepad++, to convert any absolute file paths for texture files to relative ones.

Table 11-9 ColladaModel Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
ColladaFile	A file path to the DAE, OBJ, or JSB file to be used for this DisplayModel. A ZIP file containing one of these files and its related texture files can also be used, and is the recommended option for this input. The file path must be enclosed in single quotes if it contains spaces.

11.2.6 GraphModel

The GraphModel is used to determine the presentation style and proportions for various types of Graph objects. To make the Graph scalable to any size, all length measurements are specified as a fraction of the Graph's total height.

Table 11-10 GraphModel Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
TitleTextHeight	Text height for the Graph title.
XAxisTitleTextHeight	Text height for the x-axis title.
YAxisTitleTextHeight	Text height for the y-axis title.

Keyword	Description
LabelTextHeight	The text height for both x-axis and y-axis labels.
TitleGap	The gap between the Graph title and the top of the Graph.
XAxisTitleGap	The gap between the x-axis labels and the x-axis title.
XAxisLabelGap	The gap between the x-axis and its labels.
YAxisTitleGap	The gap between the y-axis labels and the y-axis title.
YAxisLabelGap	The gap between the y-axis and its labels.
TopMargin	Size of the gaps between the respective edges of the outer pane and the Graph. Expressed as a fraction of the Graph's total height.
BottomMargin	
LeftMargin	
RightMargin	
TitleTextModel	The TextModel used to determine the font, colour, and style for the Graph title. The drop shadow settings are ignored. Black Verdana text is used if this input is left blank.
AxisTitleTextModel	The TextModel used to determine the font, colour, and style for the x-axis and y-axis titles. The drop shadow settings are ignored. Black Verdana text is used if this input is left blank.
LabelTextModel	The TextModel used to determine the font, colour, and style for the labels next to the x-axis and y-axis tick marks. The drop shadow settings are ignored. Black Verdana text is used if this input is left blank.
GraphColor	The colour of the Graph background.
BackgroundColor	The colour of the outer pane background.
BorderColor	The colour of the Graph border.

11.3 Importing a 3D Object or Image

The easiest way to create a new DisplayEntity for a 3D object or an image is to use the Import function provided in the Control Panel under File. This method creates a new DisplayEntity that is connected to a new DisplayModel with the imported 3D object or image.

11.4 View

The View object is used to hold the position and direction of the camera that creates the 3D image in a View window.

A new View can be created by clicking on the View item in the Menu bar. An existing View can be selected for editing by clicking on its entry in the Object Selector. Views can be copied, pasted, renamed, or deleted in the same way as other objects in the Object Selector. However, a renamed View must be closed and re-opened before its new name is displayed.

Table 11-11 View Inputs

Keyword	Description
Graphics	
Region	The region in which the view's coordinates are given.
ViewCenter	The position at which the view camera is pointed.
ViewPosition	The position of the view camera.
WindowSize	The size of the window in pixels (width, height).
WindowPosition	The position of the upper left corner of the window in pixels measured from the top left corner of the screen.
TitleBarText	Text to place in the title bar of the View window. The window must be closed and re-opened manually after changing the title.
ShowWindow	If `TRUE`, the View window is displayed on screen.
Movable	A Boolean indicating whether the view can be panned or rotated.
Lock2D	A Boolean indicating whether the view is locked to a downward view (the 2D default).
FollowEntity	The (optional) entity for this view to follow. Setting this input makes the view ignore ViewCenter and interprets ViewPosition as a relative offset to this entity.
ScriptedViewPosition	The (optional) scripted curve for the view position to follow.
ScriptedViewCenter	The (optional) scripted curve for the view center to follow.
SkyboxImage	The image file to use as the background for this view.

Table 11-12 View Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent	See Section 10.1.
View	
PointOfInterest	The point at which the view will zoom towards or rotate around.
DistanceToPOI	The distance from the camera position to the point of interest.

12 Graphics Objects Palette

Graphics Objects are used to create 3D objects, pictures, text, graphs, arrows, and other graphical components needed to visualise and monitor a simulation. The following objects are included in the Graphics Objects palette.

Table 12-1 Graphics Objects Palette

Object		Description
	Region	Local coordinate system.
	DisplayEntity	Graphical object displaying either a 3D shape or a 2D picture.
	Shape	Two-dimensional graphical object whose shape and colour can be selected.
	Polyline	Straight or curved line segments.
	Arrow	Line that terminates in an arrow head.
	Text	Text that appears in the 3D model universe.
	Graph	Chart that shows model outputs as they change in time.
	BarGauge	Gauge that displays the value of an expression.
	BooleanIndicator	Circular entity that changes colour to indicate TRUE and FALSE.
	EntityLabel	Text that labels another object. An EntityLabel can only be created by selecting the "Show Label" option in an object's context menu. Since they cannot be dragged and dropped, EntityLabel does not appear in the Model Builder.
	OverlayImage	Picture that appears in a fixed position in the display window.
	OverlayText	Text that appears in a fixed position in the display window.
	OverylayClock	Time and date display that appears in a fixed position in the display window.
	BillboardText	Text that follows a 3D position but is always upright on the screen.
	MimicEntity	Graphical object that copies the appearance of another DisplayEntity.
	View	Display window showing view of the 3D model universe. A View can only be created using the Views menu in the Menu Bar.
	VideoRecorder	Object that makes a video recording of the model.

12.1 Region

The Region object is used to define a local coordinate system. When a Region is specified for an object, its inputs for position and orientation are relative to the position and orientation of its Region. The global coordinate system is the default for objects that do not reference a specific Region.

The Position and Orientation keywords are used to define the origin for the coordinate system and the angles for its coordinate axes. Once these inputs have been set, it is advised to set the inputs for Show and Movable to FALSE so the Region object is no longer visible and cannot be moved accidentally.

The Scale keyword can be used to make the objects in a Region appear to be smaller and closer together compared to objects outside the Region.

Table 12-2 Region Inputs

Keyword	Description
Key Inputs	
Scale	The graphical scale for the Region's local coordinate system relative to the coordinate system in which it is embedded. For example, an input of 0.5 would make objects appear to be one-half smaller and closer together.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-3 Region Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.2 DisplayEntity

The DisplayEntity object is used to add an image or static 3D content in a model. Normally, a DisplayEntity is created automatically when a 3D object or an image is imported through the File > Import menu item.

A DisplayEntity can also be created by drag and drop from the Model Builder. The default appearance of a DisplayEntity is a grey cube. Its appearance can be changed by right-clicking the DisplayEntity and selecting "Change Graphics". The user can then select between the available DisplayModels or can create a new DisplayModel by importing a file in one of the supported 3D graphics formats.

Table 12-4 DisplayEntity Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-5 DisplayEntity Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.3 Shape

The Shape object is used to add simple 2D objects such as circles, squares, stars, etc. to a model. The DisplayModel input, which is restricted to ShapeModels, determines the type of 2D object that is displayed. It includes the keywords FillColour, LineColour, etc. that that override the default values for these inputs that are provided by its ShapeModel. The easiest way to enter these inputs is to use the formatting buttons on the Control Panel.

Table 12-6 Shape Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-7 Shape Outputs

Output Name	Description
Format	
FillColour	The colour with which the shape is filled.
LineColour	The colour with which the shape is outlined.
Filled	Determines whether or not the shape is filled. If TRUE, it is filled with a specified colour. If FALSE, it is hollow.
Outlined	Determines whether or not the shape is outlined. If TRUE, it is outlined with a specified colour. If FALSE, it is drawn without an outline.
LineWidth	Width of the outline in pixels.
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.4 Polyline

The Polyline object is a series of points that are connected by straight or curved line segments.

The CurveType keyword specifies whether the line segments are straight (`LINEAR`) or curved (`BEZIER`, `SPLINE`, `CIRCULAR_ARC`).

The Filled keyword determines whether or not the interior of the polyline is filled with a colour. The Outlined keyword determines whether or not a filled polyline has a border. The colour of the fill and the border are determined by the FillColour and LineColour keywords respectively. The border's width in pixel is determined by the LineWidth keyword.

Table 12-8 Polyline Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Format	
FillColour	The colour with which the polyline is filled.
LineColour	The colour with which the polyline is outlined.
Filled	Determines whether or not the polyline is filled. If TRUE, it is filled with a specified colour. If FALSE, it is hollow.
Outlined	Determines whether or not the polyline is outlined. If TRUE, it is outlined with a specified colour. If FALSE, it is drawn without an outline.
LineWidth	Width of the outline in pixels.
Graphics	
Points, CurveType, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-9 Polyline Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.5 Arrow

The Arrow object is similar to a Polyline except for the addition of an arrowhead at its end and the removal of the Filled, FillColour, and Outlined keywords.

Table 12-10 Arrow Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.

Keyword	Description
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Format	
LineColour	The colour of the arrow.
LineWidth	The width of the Arrow line segments in pixels.
ArrowSize	A set of (x, y, z) numbers that define the size of the arrowhead.
Graphics	
Points, CurveType, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-11 Arrow Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.6 Text

The Text object is used to show static or dynamic text in a model. Text objects are used for labelling various parts of the model and for monitoring the status of the model.

The output displayed by the Text object is determined primarily by its Format keyword. Dynamic text can be introduced by including a Java format code such as `%s` in the Format keyword and specifying the value to be displayed with the DataSource keyword. Some typical Java format codes are `%s`, which can display both text and numbers, and `%.6f`, which displays a numeric value with six decimal places.

If the variable text is a number with units, the value can be converted from SI to a specified unit through the Unit keyword.

The easiest way to modify the contents of a Text object is to edit it directly in the view window. Double-click on the object to place it in edit mode. In this mode, text can be entered, deleted, inserted, and highlighted using the same conventions as a typical text editor. Text can be copied from and pasted to the clipboard using Ctrl-C and Ctrl-V. Press the Return key or click on another object to save the changes and return the Text object to its normal mode.

The appearance and style of the text is determined by the keywords under the Font tab, including FontName, FontColour, etc. If no inputs are provided to these keywords, the appearance of the text is

determined by the TextModel entered for the DisplayModel keyword. The default TextModel is set to the black Verdana font.

TextModels can be used in the same way as the Text Styles in Microsoft Word. A new TextModel can be created through the following steps:

1. In the Object Selector, right-click on the TextModelDefault object and select Duplicate.
2. Rename the new TextModel object by clicking on its name in the Object Selector, entering the new name.
3. With the new TextModel selected in the Object Selector, use the formatting buttons in the button bar to specify its characteristics such as font, font style, colour, etc.

The new TextModel can be applied to the Text object by clicking the object and using the 'DisplayModel' button in the button bar.

Table 12-12 Text Inputs

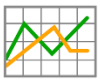
Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
TextHeight	The height of the font as displayed in the view window.
Format	The fixed and variable text to be displayed, as specified by a Java format string. If variable text is to be displayed using the DataSource keyword, include the appropriate Java format in the text. For example, `%s` will display a text output and `%.6f` will display a number with six decimal digits of accuracy. A new line can be started by entering `%n`. Note that a % character is generated by entering `%%`.
UnitType	The unit type for the numerical value to be displayed as variable text. Set to `DimensionlessUnit` if the variable text is non-numeric such as the state of a Server.
Unit	The unit in which to express an expression that returns a numerical value. For example, if the UnitType input has been set to `DistanceUnit`, then the output value could be displayed in kilometres, instead of meters, by entering km to this keyword.
DataSource	An expression that returns the variable text to be displayed. The expression can return a number that will be formatted as text or it can return text directly, such as the state of a Server.
FailText	The text to display if there is any failure while formatting the variable text or while evaluating the expression.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Font	
FontName	The font to be used for the text.
FontColour	The colour of the text specified by a colour keyword or RGB values.
FontStyle	The font styles to be applied to the text, e.g. Bold, Italic.
DropShadow	If TRUE, then a drop shadow appears for the text.

Keyword	Description
DropShadowColour	The colour for the drop shadow specified by a colour keyword or RGB values.
DropShadowOffset	The x, y, z coordinates of the drop shadow's offset, expressed as a decimal fraction of the text height.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-13 Text Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.7 Graph



The Graph object is a real-time visual representation of one or more quantities, showing their values over a specified time period.

The following figure illustrates the meanings of the keywords that are used to format a Graph.

Figure 12-1 Sample Graph with Formatting Keywords Defined Graphically

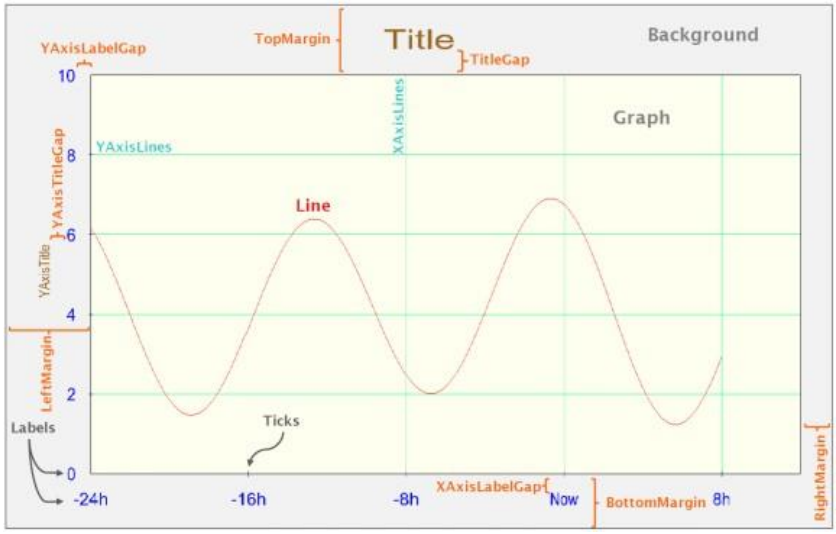


Table 12-14 Graph Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Title	Text for the graph title, enclosed in single quotes if it contains spaces.
NumberOfPoints	The number of data points to be displayed on the Graph. This determines the resolution of the graph.
UnitType	The type of units for each line specified by the DataSource input. Must be specified before the DataSource input and the Y-Axis inputs.
DataSource	One or more sources of data to be graphed against the primary y-axis. Specified as a series of Expressions, each enclosed by braces.
LineColours	A list of colours for the data series to be displayed. For multiple colours, each colour must be enclosed in braces as each colour can be itself defined as a list of RGB values.
LineWidths	A list of widths, in pixels, for the data series to be displayed.
SecondaryUnitType	The type of units for each line specified by the SecondaryDataSource input. Must be specified before the SecondaryDataSource input and the Secondary Y-Axis inputs.
SecondaryDataSource	One or more sources of data to be graphed against the secondary y-axis. Specified as a series of Expressions, each enclosed by braces.
SecondaryLineColours	A list of colours for the data series to be displayed. For multiple colours, each colour must be enclosed in braces as each colour can be itself defined as a list of RGB values.
SecondaryLineWidths	A list of widths, in pixels, for the data series to be displayed.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
X-Axis	
XAxisTitle	Title text for the x-axis.
XAxisUnit	The unit to be used for the x-axis.
XAxisStart	The minimum value for the x-axis.
XAxisEnd	The maximum value for the x-axis.
XAxisInterval	The interval between x-axis labels.
XAxisLabelFormat	The Java format to be used for the tick mark values on the x-axis. For example, the format `%.1f` would display the value 5 as 5.0.
XLines	A list of time values between XAxisStart and XAxisEnd where vertical gridlines are inserted.
XLinesColor	Colour of the vertical gridlines, or a list corresponding to the colour of each gridline defined in XLines, defined using a colour name or RGB values.
Y-Axis	
YAxisTitle	Title text for the primary y-axis.

Keyword	Description
YAxisUnit	The unit to be used for the primary y-axis.
YAxisStart	The minimum value for the primary y-axis, in units of the DataSource.
YAxisEnd	The maximum value for the primary y-axis.
YAxisInterval	The interval between primary y-axis labels.
YAxisLabelFormat	The Java format to be used for the tick mark values on the primary y-axis. For example, the format `%.1f` would display the value 5 as 5.0.
YLines	A list of values at which to insert horizontal gridlines.
YLinesColor	Colour of the horizontal gridlines, defined using a colour name or RGB values.
Secondary Y-Axis	
SecondaryYAxisTitle	Title text for the secondary y-axis.
SecondaryYAxisUnit	The unit to be used for the secondary y-axis.
SecondaryYAxisStart	The minimum value for the secondary y-axis.
SecondaryYAxisEnd	The maximum value for the secondary y-axis.
SecondaryYAxisInterval	The interval between secondary y-axis labels.
SecondaryYAxisLabelFormat	The Java format to be used for the tick mark values on the secondary y-axis.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-15 Graph Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.8 BarGauge

The BarGauge object is used to provide a graphical display of the numerical value returned by an expression.

Table 12-16 BarGauge Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
DataSource	Height of the bar expressed as a value between 0 and 1. Values outside this range will be truncated.
Colour	Colour of the bar.
BackgroundColour	Colour of the gauge's body.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-17 BarGauge Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
BarGauge	
Value	Value displayed by the gauge.

12.9 BooleanIndicator

The BooleanIndicator allows the state of a specified Expression returning TRUE or FALSE to be displayed.

Table 12-18 BooleanIndicator Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
DataSource	An expression returning a boolean value: zero = `FALSE`, non-zero = `TRUE`.
TrueColour	The colour of the indicator when the DataSource expression is `TRUE`.

Keyword	Description
FalseColour	The colour of the indicator when the DataSource expression is FALSE.
TrueText	The string returned by the Text output when the DataSource expression is TRUE.
FalseText	The string returned by the Text output when the DataSource expression is FALSE.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-19 BooleanIndicator Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
BooleanIndicator	
Text	If the DataSource expression is TRUE, then return TrueText. If it is FALSE, then return FalseText.

12.10 EntityLabel

The EntityLabel object is a special version of a Text object that is used to display the name of a selected object. The only way to create an EntityLabel is to right-click on the object to be labelled and select “Show Label”. It cannot be dragged and dropped from the Model Builder.

An EntityLabel moves automatically with the object and is destroyed when the object is destroyed. An object can have only one EntityLabel at a time.

An object name can be changed by editing its EntityLabel in the view window. Double-click on the EntityLabel to enter edit mode and revise the name as desired. Press the Return key or click on another object to accept the new name.

The appearance of all the EntityLabels in a model can be changed by editing the inputs for the EntityLabelModel (under Display Models > TextModel in the Object Selector). The standard colour, font and style (bold and/or italics) for all EntityLabels can be selected in this way.

Table 12-20 EntityLabel Inputs

Keyword	Description
Key Inputs	
TextHeight	Keywords for Text (see Table 12-12).
Font	
FontName, FontColour, FontStyle, DropShadow, DropShadowColour, DropShadowOffset	Keywords for Text (see Table 12-12).
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-21 EntityLabel Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.11 Overlay Objects (OverlayImage, OverlayText, OverlayClock)

Overlay objects are special versions of other objects that are used for graphical display in a simulation model. Unlike other display objects, the position of overlay objects is referenced to the corner of a view window, and so the object does not move when the View is panned or zoomed. These objects are useful for labelling View windows or displaying the model name and company logos. Examples of overlay objects are the default Title and Clock that are provided automatically when a new model is opened.

There are three types of overlay objects, each corresponding to a different graphical object type. The relationship between each overlay object, its parent object type, and its usage is summarized in the following table.

Table 12-22 Overlay Object and Usage Summary

Overlay Object	Parent Object	Usage
OverlayImage	DisplayEntity	Static image (Logos, other graphics)
OverlayText	Text	Static or dynamic text (Model name, states, rates)
OverlayClock	Text	Current time in the simulation model.

Apart from graphics inputs, each type of overlay object has the same keywords as its parent object. The graphics inputs for overlay objects are shown in the following table.

Table 12-23 Overlay Objects Graphics Inputs

Keyword	Description
ScreenPosition	A list of two numbers specifying the spacing, in pixels, between the left and top corner of the View window and the object.
AlignRight	If TRUE, the horizontal alignment is referenced to the right side of the View window instead of the left.
AlignBottom	If TRUE, the vertical alignment is referenced to the bottom side of the View window instead of the top.

12.12 BillboardText

A BillboardText object is similar to a Text object, except that the text is always oriented towards the View window and its height is given in pixels instead of metres. BillboardText retains its coordinates in space and in this way differs from the OverlayText object. Both static and dynamic text can be displayed by a BillboardText object using the same keywords as Text objects.

BillboardText is commonly used to label 3D objects. The advantage of using BillboardText is that the label is visible and readable from all view angles.

The keywords for BillboardText are identical to those for Text objects and have the same meaning, except for the TextHeight keyword, which now gives the height of the text in pixels instead of metres. At the present time, this input must include the units of metres (m) even though it is interpreted as pixels. This will be corrected in a future version of the software.

12.13 MimicEntity

The MimicEntity is used to copy the display of another entity, usually as part of a control panel for a model. The entity whose display is to be copied is specified by the input to the SourceEntity keyword. This input can be the name of a specific entity or an expression that returns an entity.

Table 12-24 MimicEntity Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
SourceEntity	The entity whose graphics are to be copied.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-25 MimicEntity Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

12.14 VideoRecorder

The VideoRecorder object is used to create a short video of a model in operation.

The AVI file created by the VideoRecorder is encoded using the VP8 codec, which is NOT supported by Windows Media Player. Furthermore, the present encoding algorithm is quite inefficient making the file size much larger than necessary. Both problems can be solved by re-encoding the video using free open-source software such as HandBrake (www.handbrake.fr).

The original AVI file in VP8 codec can be viewed with the VLC video player (www.videolan.org).

Table 12-26 VideoRecorder Key Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
CaptureStartTime	Simulation time at which to capture the first frame.
CaptureInterval	Simulation time between captured frames.
CaptureFrames	The total number of frames to capture for the video. The recorded video assumes 30 frames per second. Therefore, if a 2 minute video is required, the number of frames should be set to 120 x 30 = 3600.
CaptureArea	The size of the video/image, expressed as the number of horizontal and vertical pixels. The top left-hand corner of the captured frames will be the same as the top left-hand corner of the image on the monitor. If the specified image size is larger than the monitor resolution, then the image will be extended beyond the bottom and/or right sides of the monitor.
CaptureViews	The list of View windows to be captured.
VideoBackgroundColor	The background colour for the captured frames. Only the 3D view portion of the specified windows will be captured. The remainder of the frame, such as the Control Panel or any gaps between the view windows, will be replaced by the background colour.
VideoName	A label to append to the run name when the AVI file is saved. The saved file will be named <run name>_<VideoName>.avi.
SaveImages	If `TRUE`, an individual PNG file will be saved for each frame.

SaveVideo	If `TRUE`, an AVI file containing the video will be saved.
Options	
CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 12-27 VideoRecorder Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

13 Probability Distributions Palette

The Probability Distributions palette provides a selection of standard theoretical probability distributions as well as user-defined probability distributions.

Table 13-1 Probability Distributions

Distribution Name		Description
	UniformDistribution	Generates samples with a constant probability between minimum and maximum values.
	TriangularDistribution	Generates samples from a triangular probability distribution between minimum and maximum values. The distribution peaks at its mode.
	NormalDistribution	Generates samples from a normal probability distribution.
	ExponentialDistribution	Generates samples from a negative exponential probability distribution.
	NonStatExponentialDist	Generates samples from a time-varying negative exponential probability distribution. The correct 'non-stationary Poisson process' algorithm is used.
	ErlangDistribution	Generates samples from an Erlang probability distribution.
	GammaDistribution	Generates samples from a Gamma probability distribution.
	BetaDistribution	Generates samples from a Beta probability distribution.
	WeibullDistribution	Generates samples from a Weibull probability distribution.
	LogNormalDistribution	Generates samples from a Log-Normal probability distribution.
	LogLogisticDistribution	Generates samples from a Log-Logistic probability distribution.
	DiscreteDistribution	Generates samples from a discrete set of values.
	ContinuousDistribution	Generates samples over a continuous range of values.
	BooleanSelector	Randomly selects true/false with a user-selectable probability of true.

Most probability distributions use the following inputs and outputs.

Table 13-2 Distribution Inputs

Keyword	Description
UnitType	The unit type for the value returned by the distribution, e.g. TimeUnit. To keep the units consistent for other inputs, this input must be set first.

Keyword	Description
RandomSeed	Seed for the random number generator. Must be an integer greater than or equal to zero. The RandomSeed keyword works together with the GlobalSubstreamSeed under the Simulation object to determine the random sequence. The GlobalSubstreamSeed keyword allows the user to change all the random sequences in a model with a single input. When an object with this input is copied and pasted, the RandomSeed input is reset to an unused value for each copy that is pasted.
MinValue	The minimum value that can be returned by the distribution. A value less than the minimum is rejected and the distribution is re-sampled.
MaxValue	The maximum value that can be returned by the distribution. A value greater than the maximum is rejected and the distribution is re-sampled.

Table 13-3 Distribution Outputs

Output Name	Description
Value	The last value sampled from the distribution. When used in an expression, this output returns a new sample every time the expression is evaluated.
CalculatedMean	The mean value for the distribution calculated directly from the inputs. Ignores the values entered for the MinValue and MaxValue keywords.
CalculatedStandardDeviation	The standard deviation for the distribution calculated directly from the inputs. Ignores the values entered for the MinValue and MaxValue keywords.
NumberOfSamples	The total number of samples returned by the Probability Distribution.
SampleMean	The average of the samples returned by the Probability Distribution.
SampleStandardDeviation	The standard deviation of the samples returned by the Probability Distribution.
SampleMin	The minimum of the samples returned by the Probability Distribution.
SampleMax	The maximum of the samples returned by the Probability Distribution.

The Probability Distributions were coded using algorithms adapted from "Simulation Modeling & Analysis", 4th Edition, by Averill M. Law. Random numbers for these distributions are generated by the Multiple Recursive Generator developed by L'Ecuyer ("Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators", Operations Res., 47: 159-164 (1999a)).

13.1 Changing the Random Seed

Each Probability Distribution has the RandomSeed keyword, which generates a different pseudo-random sequence of values for each Probability Distribution. Often, a simulation model is executed multiple times, called ‘replications’, using different random seeds to determine the statistical accuracy of the output parameters.

One way to achieve multiple replications is to change the RandomSeed for each Probability Distribution. However, this can be impractical when large numbers of distributions are used in a model. The GlobalSubstreamSeed keyword for the Simulation object simplifies this process. Changing the input for this keyword causes all Probability Distributions in the model to generate a different

random sequence. In effect, the seed for each Probability Distribution is the combination of the inputs to the RandomSeed and GlobalSubstreamSeed keywords.

When multiple objects behave according to a Probability Distribution with the same characteristics, it is advisable to have a unique instance of the Probability Distribution for each object. This avoids any cross-correlation effects arising from the order in which a single distribution may be sampled.

13.2 UniformDistribution

The UniformDistribution object generates random samples with a constant probability between specified minimum and maximum values.

Table 13-4 UniformDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-5 UniformDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.3 TriangularDistribution

The TriangularDistribution object generates random samples from a triangular probability distribution between specified minimum and maximum values. The distribution peaks at its mode.

Table 13-6 TriangularDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Mode	The mode of the triangular distribution, i.e. the value with the highest probability.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-7 TriangularDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.4 NormalDistribution

The NormalDistribution object generates random samples from a normal probability distribution.

Table 13-8 NormalDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Mean	The mean of the normal distribution (ignoring the MinValue and MaxValue keywords).
StandardDeviation	The standard deviation of the normal distribution (ignoring the MinValue and MaxValue keywords).
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-9 NormalDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.5 ExponentialDistribution

The ExponentialDistribution object generates random samples from a negative exponential probability distribution.

Table 13-10 ExponentialDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.

Keyword	Description
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Mean	The mean of the exponential distribution.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-11 ExponentialDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.6 NonStatExponentialDist

The NonStatExponentialDist object generates random samples from a time-varying negative exponential probability distribution. The correct "non-stationary Poisson process" algorithm is used.

This distribution is used to generate the inter-arrival times for a customers, orders, etc. that arrive randomly with an average arrival rate that varies over time. The calculation is based on the expected ("average") cumulative number of arrivals over the time period. This data is entered as a TimeSeries using the ExpectedArrivals keyword.

The ScaleFactor input allows the average arrival rate from the distribution to be scaled up or down from the values in the ExpectedArrivals input. Note that it is incorrect to scale the inter-arrival times returned the distribution, hence the need for this keyword.

Table 13-12 NonStatExponentialDist Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
RandomSeed, MinValue, MaxValue	See Section 13.
ExpectedArrivals	A TimeSeries containing the expected cumulative number of arrivals as a function of time.
ScaleFactor	A factor that is applied to the cumulative number of arrivals returned by the ExpectedArrivals input.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-13 NonStatExponentialDist Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.7 ErlangDistribution

The ErlangDistribution object generates random samples from an Erlang probability distribution.

Table 13-14 ErlangDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.

Keyword	Description
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Mean	The scale parameter for the Erlang distribution.
Shape	The shape parameter for the Erlang distribution. An integer value >= 1. Shape = 1 gives the Exponential distribution. For Shape > 10 it is better to use the Gamma distribution.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-15 ErlangDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.8 GammaDistribution

The GammaDistribution object generates random samples from a Gamma probability distribution.

Table 13-16 GammaDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Mean	The mean of the Gamma distribution.
Shape	The shape parameter for the Gamma distribution. A decimal value > 0.0.

Keyword	Description
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-17 GammaDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.9 BetaDistribution

The BetaDistribution object generates random samples from a Beta probability distribution.

Table 13-18 BetaDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
AlphaParam	The alpha tuning parameter.
BetaParam	The beta tuning parameter.
Scale	The scale parameter for the distribution. This scales the value of the distribution so it returns values between 0 and scale.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.

Keyword	Description
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-19 BetaDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.10 WeibullDistribution

The WeibullDistribution object generates random samples from a Weibull probability distribution.

Table 13-20 WeibullDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Scale	The scale parameter for the Weibull distribution.
Location	The location parameter for the Weibull distribution.
Shape	The shape parameter for the Weibull distribution. A decimal value > 0.0.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-21 WeibullDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.11 LogNormalDistribution

The LogNormalDistribution generates random samples from a Log-Normal probability distribution.

Table 13-22 LogNormalDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Scale	The scale parameter for the Log-Normal distribution.
NormalMean	The mean of the dimensionless normal distribution (not the mean of the lognormal).
NormalStandardDeviation	The standard deviation of the dimensionless normal distribution (not the standard deviation of the lognormal).
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-23 LogNormalDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.12 LogLogisticDistribution

The LogLogisticDistribution object generates random samples from a Log-Logistic probability distribution.

Table 13-24 LogLogisticDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
Scale	The scale parameter for the Log-Logistic distribution.
Shape	The shape parameter for the Log-Logistic distribution. A decimal value > 0.0.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-25 LogLogisticDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.13 DiscreteDistribution

The DiscreteDistribution object generates random samples from a discrete set of values.

Table 13-26 DiscreteDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
ValueList	The list of discrete values that can be returned by the distribution. The values can be any positive or negative and can be listed in any order. No interpolation is performed between these values.
ProbabilityList	The list of probabilities corresponding to the discrete values in the ValueList. Must sum to 1.0.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-27 DiscreteDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.14 ContinuousDistribution

The ContinuousDistribution object generates random samples over a continuous range of values.

Table 13-28 ContinuousDistribution Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType, RandomSeed, MinValue, MaxValue	See Section 13.
ValueList	The list of values for the user-defined cumulative probability distribution.
CumulativeProbabilityList	The list of cumulative probabilities corresponding to the values in the ValueList. The cumulative probabilities must be given in increasing order. The first value must be exactly 0.0. The last value must be exactly 1.0.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-29 ContinuousDistribution Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Distribution	
Value, CalculatedMean, CalculatedStandardDeviation, NumberOfSamples, SampleMean, SampleStandardDeviation, SampleMin, SampleMax	See Section 13.

13.15 BooleanSelector

The BooleanSelector returns a randomly-selected value of `TRUE` or `FALSE` based on a user-specified probability.

Table 13-30 BooleanSelector Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
RandomSeed	See Section 13.
TrueProbability	The probability of the Selector returning true.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 13-31 BooleanSelector Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

Output Name	Description
BooleanSelector	
Value	The last value sampled from the distribution. When used in an expression, this output returns a new sample every time the expression is evaluated.

14 Basic Objects Palette

The Basic Objects palette contains a number of objects that can be used in many types of simulation models. The following objects are provided in the Basic Objects palette.

Table 14-1 Basic Objects Palette

Object		Description
	Controller	Signals the updating of each component in the specified sequence.
	InputValue	Provides a way to enter a numerical value directly into the simulation model screen.
	TimeSeries	Provides a floating point number that changes in simulated time following a series of input values.
	TimeSeriesThreshold	Specifies a range of values from a TimeSeries for which an activity is permitted.
	ExpressionThreshold	Specifies a logical condition for which an activity is permitted.
	ExpressionLogger	Records the values for one or more expressions to a log file at regular intervals.
	EntitlementSelector	Selects an index on the basis of entitlement from a given set of proportions.
	ExpressionEntity	Calculates the value for a specified expression.
	DowntimeEntity	Provides Breakdown and Maintenance controls.
	ValueSequence	Generates a repeating sequence of numerical values.
	EventSchedule	Generates a sequence of inter-arrival times from a list of event times.
	FileToVector	Populates a one-dimensional array with data from a specified file.
	FileToMatrix	Populates a two-dimensional array with data from a specified file.
	FileToHashMap	Populates a HashMap with data from a specified file.
	ExternalProgram	Executes an external program when an entity is received.
	EntitySystem	Represents a group of entities whose state is determined by the states of those entities.
	ScriptEntity	Changes model inputs during a simulation run.

14.1 Controller

The Controller object generates the update signals for the individual objects in a continuous type model that are managed by this Controller.

Table 14-2 Controller Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
FirstTime	Simulation time for the first update signal.
Interval	Time interval between update signals.
MaxUpdates	Maximum number of updates to perform.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-3 Controller Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
EntityList	Objects that receive update signals from this Controller, listed in the sequence in which the updates are performed.
Count	Total number of updates that have been performed.

14.2 InputValue

1.0

The InputValue object is a special version of a Text object that is used to allow a user to enter a numerical input directly in the view window, without using the Input Editor. The input value can be modified by double-click on the displayed value to enter edit mode. After editing is complete, press the Return key or click on another object to accept the new value.

The present value for an InputValue object can be accessed by the inputs to the keywords for other objects either through an expression, e.g. [InputValue1].Value, or by simply entering the name of the InputValue.

Table 14-4 InputValue Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
TextHeight, UnitType	Keywords for Text (see Table 12-12).
Value	The numerical value for the input.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Font	
FontName, FontColour, FontStyle, DropShadow, DropShadowColour, DropShadowOffset	Keywords for Text (see Table 12-12).
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-5 InputValue Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
InputValue	
Value	The present value for this input.

14.3 TimeSeries

The TimeSeries object simulates a numerical value that varies with time.

Many JaamSim keywords are structured to accept a constant value, a Probability Distribution, a TimeSeries, or an Expression, which makes TimeSeries a powerful and flexible component for building simulation models.

The data for the object consists of a series of time stamps and values specified by the Value keyword. The time stamps can be irregularly spaced and it is possible to repeat the time series over and over again during the simulation using the CycleTime keyword.

The value returned by a TimeSeries object has a specific unit type indicated by the UnitType keyword. To maintain consistency, the UnitType must be specified prior to any other input.

The OffsetToFirst keyword determines how a time stamp entry in the time series is converted to simulation time. When set to TRUE, the simulation time corresponding to a time stamp is calculated as the elapsed time between the time stamp and the first time stamp in the series. This option ensures that the first entry in the TimeSeries corresponds to zero simulation time. If set to FALSE, the simulation time for the time stamp is the elapsed time between the time stamp and the StartDate input for the Simulation. If the first time stamp is earlier than the StartDate input, it will be assigned a negative simulation time. This possibility has been allowed for in the internal logic for TimeSeries.

Table 14-6 TimeSeries Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
OffsetToFirst	If TRUE, the simulation times corresponding to the time stamps entered to the Value keyword are calculated relative to the first time stamp. This offset sets the simulation time for the first time stamp to zero seconds.
UnitType	The unit type for the time series. The UnitType input must be specified before the Value input.
Value	A list of time series records with format { time value }, where: 'time' is the time stamp for the record and 'value' is the time series value. Records are entered in order of increasing simulation time. The appropriate units should be included with both the time and value inputs.
CycleTime	The time at which the time series will repeat from the start.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-7 TimeSeries Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
TimeSeries	
PresentValue	The time series value for the present time.

Output Name	Description
NextTime	Time at which the time series value is updated next.
NextValue	Value for the time series when it is updated next.

Timestamps can be entered in the normal fashion, such as 0 s, 5 h, etc., or in various year/month/day formats (see Table 4-16).

The first timestamp must always be zero seconds (i.e. 0 s) or January 1, 00:00 of an arbitrary year. If a non-zero year is used, for example 2010-01-01, then the TimeSeries considers this date to be time zero of the simulation and all other timestamps are offset accordingly.

The value for the TimeSeries is constant between one timestamp and the next. If the simulation time is greater than the CycleTime, the TimeSeries will repeat from the beginning of the Value list.

For time series with more than a few entries, it is best to put the Value inputs in a separate file and use the Include feature described in Section 8.5. This will require manual editing of the configuration file (see Section 8). In the following example, the include file, TimeSeries1.inc, contains the following entries:

```
TimeSeries1  Value {
{   0 d  0.00  km/h  }
{   2 d  0.76  km/h  }
{  10 d  0.24  km/h  }
}
```

Assuming that the include-file is located in the same directory as the configuration file, the inputs to the configuration file would be:

```
Define  TimeSeries  { TimeSeries1 }
TimeSeries1  UnitType  { SpeedUnit }
Include '.\TimeSeries1.inc'
TimeSeries1  CycleTime  { 14 d }

RecordEdits
```

Note that all the inputs associated with TimeSeries1 appear at the beginning of the configuration file before the RecordEdits statement (see Section 8.7). This position will allow editing of the model in the Input Editor and saving the results without changing the Include file arrangement.

In the example, TimeSeries1 takes on a new value (0.76 km/h) on when simulation time reaches 24 hours and another new value (0.24 km/h) at 240 hours. After 14 days, the TimeSeries completes a cycle and takes on the original value (0 km/h). This 14-day cycle would repeat until the end of the simulation run.

The following inputs in year/month/day format are equivalent to those for the example given above.

```
TimeSeries1  Value {
{  2014-01-01T00:00:00  0.00  km/h  }
{  2014-01-03T00:00:00  0.76  km/h  }
{  2014-01-11T00:00:00  0.24  km/h  }
}
```

14.4 TimeSeriesThreshold

The TimeSeriesThreshold object varies its state between open and closed depending on the present value for a TimeSeries object.

The TimeSeries to be monitored is specified by the TimeSeries keyword. Trigger levels are determined by the MinOpenLimit and MaxOpenLimit keywords. For the TimeSeriesThreshold to be open, the present value for the TimeSeries must be within the range specified by these two keywords.

The LookAhead and Offset keywords provide additional flexibility:

- If a non-zero value is specified for the LookAhead keyword, then the present value for the TimeSeries must be within the range specified by the MinOpenLimit and MaxOpenLimit keywords over the entire LookAhead duration, starting from the present time.
- If a non-zero value is specified for the Offset keyword, then the above rule is modified so that the LookAhead duration begins at the present time plus the offset.

Table 14-8 TimeSeriesThreshold Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType	The unit type for the threshold (e.g. `DistanceUnit`, `TimeUnit`, `MassUnit`).
TimeSeries	The TimeSeries object whose values are to be tested.
MaxOpenLimit	The largest TimeSeries value for which the threshold is open. The threshold is closed for TimeSeries values greater than MaxOpenLimit.
MinOpenLimit	The smallest TimeSeries value for which the threshold is open. The threshold is closed for TimeSeries values smaller than MinOpenLimit.
LookAhead	The length of time over which the TimeSeries values must be >= MinOpenLimit and <= MaxOpenLimit. The threshold is open if the TimeSeries values x(t) satisfy MinOpenLimit `<=` `x(t)` `<=` MaxOpenLimit for simulation times t from `(simTime +` Offset`)` to `(simTime +` Offset `+` LookAhead`)`.
Offset	The amount of time that the threshold adds on to every time series lookup.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Format	
OpenColour, ClosedColour, ShowWhenOpen, ShowWhenClosed	See Section 10.1.

Keyword	Description
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-9 TimeSeriesThreshold Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
Threshold	
Open, OpenFraction, ClosedFraction, OpenCount, ClosedCount	Outputs inherited from Threshold (see Table 10-2).
TimeSeriesThreshold	
TimeSeriesValue	The value of the TimeSeries object at the present time plus the offset.
NextOpenTime	The next time at which the threshold will be open.
NextCloseTime	The next time at which the threshold will be closed.

14.5 ExpressionThreshold

The ExpressionThreshold object varies its state between open and closed depending on the value returned by an expression.

The expression to be evaluated is specified by the OpenCondition keyword. Normally, an ExpressionThreshold is open when the OpenCondition is `TRUE` and is closed when it is `FALSE`. However, it is possible to specify a separate expression to close the ExpressionThreshold using the CloseCondition keyword. When this keyword is specified, the threshold remains open until the CloseCondition expression is `TRUE`.

The CloseCondition keyword introduces some special cases to be addressed:

- If the two conditions conflict by both returning `TRUE`, then the threshold is considered to be open.
- If both conditions are `FALSE`, then the threshold's previous open or closed state is retained.

- If both conditions are FALSE at the start of the simulation run, then InitialOpenValue keyword determines whether the threshold is open or closed.

The OpenCondition and CloseCondition expressions are evaluated whenever simulation time is about to be advanced (conditional event) and when the ExpressionThreshold's open/closed state is tested by another object or by the input expression to another object. This procedure can become very inefficient for a model with a large number of ExpressionThresholds. The solution is to use the WatchList keyword. When an input is provided to this keyword, the expressions are evaluated only when one of the objects on the WatchList changes state.

Expression thresholds CANNOT be used in some circumstances. The expressions entered to the OpenCondition and CloseCondition keywords are tested only when an event occurs in the model such as the arrival of an entity or the completion of processing a Server. Normally, this restriction has no effect on a discrete event simulation since every change of the model's state is associated with an event. However, it is possible to enter an expression that changes between TRUE and FALSE without an event occurring. This can occur when simulation time is used explicitly in an expression, e.g. `'this.SimTime > 5[s]'`. An ExpressionThreshold will detect this condition becoming TRUE at the first event that occurs after 5 seconds. One way to model this type of condition is to use a TimeSeries and a TimeSeriesThreshold. Another solution is to use a Controller object to generate events at regular intervals and to add the object to the WatchList.

The ShowPendingStates keyword is used to detect situations where an ExpressionThreshold is being used inappropriately. A "pending state" is the situation where the present values for the OpenCondition and CloseCondition expressions are inconsistent with the present state of the ExpressionTheshold. This situation can be detected when the ExpressionTheshold is visible in the View window because the OpenCondition and CloseCondition expressions are re-evaluated every time the computer screen is refreshed.

Table 14-10 ExpressionThreshold Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
OpenCondition	The logical condition for the ExpressionThreshold to open.
CloseCondition	The logical condition for the ExpressionThreshold to close. If not specified, the CloseCondition defaults to the opposite of the OpenCondition. If the OpenCondition and CloseCondition are both TRUE, then the ExpressionThreshold is set to open.
InitialOpenValue	The initial state for the ExpressionThreshold: TRUE = Open, FALSE = Closed. This input is only relevant when the CloseCondition input is used and both the OpenCondition and CloseCondition are FALSE at the start of the simulation run. Otherwise, the initial state is determined explicitly by the OpenCondition and CloseCondition.

Keyword	Description
WatchList	An optional list of objects to monitor. If the WatchList input is provided, the ExpressionThreshold evaluates its OpenCondition and CloseCondition expression inputs and set its open/closed state ONLY when triggered by an object in its WatchList. This is much more efficient than the default behaviour which evaluates these expressions at every event time and whenever its state is queried by another object. Care must be taken to ensure that the WatchList input includes every object that can trigger the OpenCondition or CloseCondition expressions. Normally, the WatchList should include every object that is referenced directly or indirectly by these expressions. The VerfiyWatchList input can be used to ensure that the WatchList includes all the necessary objects.
VerifyWatchList	Allows the user to verify that the input to the WatchList keyword includes all the objects that affect the ExpressionThreshold's state. When set to TRUE, the ExpressionThreshold uses both the normal logic and the WatchList logic to set its state. An error message is generated if the threshold changes state without being triggered by a WatchList object.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Format	
OpenColour, ClosedColour, ShowWhenOpen, ShowWhenClosed	See Section 10.1.
PendingOpenColour	The colour of the ExpressionThreshold graphic when the threshold condition is open, but the threshold is still closed.
PendingClosedColour	The colour of the ExpressionThreshold graphic when the threshold condition is closed, but the threshold is still open.
ShowPendingStates	A Boolean value. If TRUE, the ExpressionThreshold displays the pending open and pending closed states.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-11 ExpressionThreshold Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.

Output Name	Description
Threshold	
Open, OpenFraction, ClosedFraction, OpenCount, ClosedCount	Outputs inherited from Threshold (see Table 10-2).
ExpressionThreshold	
Open	If open, then return TRUE. Otherwise, return FALSE.

14.6 ExpressionLogger

The ExpressionLogger object provides the ability to record the value for one or more expressions whenever the object is triggered during the simulation run. An ExpressionLogger can be triggered by one or more types of events:

- At regular time intervals,
- Whenever an object changes state, and
- Whenever the value of an expression changes.

Logging at regular intervals is handled by the keywords in the Key Inputs tab. State and value tracing is handled by the keywords in the Tracing tab.

Table 14-12 ExpressionLogger Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Interval	A fixed interval at which entries will be written to the log file. This input is optional if state tracing or value tracing is specified.
DataSource	One or more sources of data to be logged. Each source is specified by an Expression. It is best to include only dimensionless quantities and non-numeric outputs in the DataSource input. An output with dimensions can be made non-dimensional by dividing it by 1 in the desired unit, e.g. `'[Queue1].AverageQueueTime / 1[h]'` is the average queue time in hours. A dimensional number will be displayed along with its unit. The `'format'` function can be used if a fixed number of decimal places is required.
IncludeInitialization	If `TRUE`, entries are logged during the initialization period.
StartTime	The time at which the log starts recording entries.
EndTime	The time at which the log stops recording entries.
Options	
Active	If `TRUE`, the object is used in the simulation run.

Keyword	Description
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Tracing	
StateTraceList	A list of entities whose states will be traced. An entry in the log file is made every time one of the entities changes state. Each entity's state is written automatically to the log file - it is not necessary to add an expression to the DataSource keyword's input.
ValueTraceList	One or more sources of data whose values will be traced. An entry in the log file is made every time one of the data sources changes value. Each data source's value is written automatically to the log file - it is not necessary to add an expression to the DataSource keyword's input. It is best to include only dimensionless quantities and non-numeric outputs in the ValueTraceList input. An output with dimensions can be made non-dimensional by dividing it by 1 in the desired unit, e.g. `'[Queue1].AverageQueueTime / 1[h]'` is the average queue time in hours. A dimensional number will be displayed along with its unit. The `'format'` function can be used if a fixed number of decimal places is required.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-13 ExpressionLogger Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Logger	
LogTime	The simulation time at which the last log entry was made.

14.7 EntitlementSelector

The EntitlementSelector object is similar to the DiscreteDistribution object in that it returns an index in some range, except that instead of returning a random selection based on probabilities it makes its choice using an entitlement algorithm based on proportions. The entitlement algorithm chooses the index that minimizes the difference between the actual number returned for each index and the expected number based on the proportions.

The ProportionList keyword is used to specify the relative proportions for the N choices.

Table 14-14 EntitlementSelector Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
ProportionList	A list of N numbers equal to the relative proportion for each of the N indices. Must sum to 1.0.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-15 EntitlementSelector Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
EntitlementSelector	
Value	The last index that was selected.
NumberOfSamples	The total number of times that an index has been selected.
SampleCount	The number of times each of the N indices has been selected.
SampleDifference	The difference between the number of times each index has been selected and the expected number calculated from the proportions.

14.8 ExpressionEntity

The ExpressionEntity is used to evaluate an expression. It is useful when a complicated expression is used in several different places. Instead of entering the expression several times, it is better to use an ExpressionEntity to evaluate the expression and then use its output Value to pass the result to other expressions.

Table 14-16 ExpressionEntity Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.

Keyword	Description
UnitType	The unit type for the value returned by the expression.
Expression	The expression to be evaluated.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-17 ExpressionEntity Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
ExpressionEntity	
Value	The present value for the expression.

14.9 DowntimeEntity

The DowntimeEntity object is used to generate planned and unplanned maintenance events for various types of objects. The DowntimeEntity generates the downtime events and their durations, but the objects that use one or more DowntimeEntities must provide their own logic for halting operations.

Table 14-18 DowntimeEntity Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
FirstDowntime	The calendar or working time for the first planned or unplanned maintenance event. If an input is not provided, the first maintenance event is determined by the input for the Interval keyword. A number, an object that returns a number, or an expression can be entered.
IntervalWorkingEntity	The object whose working time determines the occurrence of the planned or unplanned maintenance events. Calendar time is used if the input is left blank.

Keyword	Description
DurationWorkingEntity	The object whose working time determines the completion of the planned or unplanned maintenance activity. Calendar time is used if the input is left blank.
Interval	The calendar or working time between the start of the last planned or unplanned maintenance activity and the start of the next maintenance activity. A number, an expression, or an object that returns a number can be entered.
Duration	The calendar or working time required to complete the planned or unplanned maintenance activity. A number, an expression, or an object that returns a number can be entered.
MaxDowntimePending	The maximum number of downtimes pending for the downtime event.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-19 DowntimeEntity Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
DownTimeEntity	
StartTime	The time that the most recent downtime event started.
EndTime	The time that the most recent downtime event finished or will finish.
NextStartTime	The time at which the next downtime event will begin. If downtime is based on the working time for an entity, then the next start time is estimated assuming that it will work continuously until the downtime event occurs.
CalculatedDowntimeRatio	The value calculated directly from model inputs for: (avg. downtime duration)/(avg. downtime interval)
Availability	The fraction of calendar time (excluding the initialisation period) during which this type of downtime did not occur.

14.10 ValueSequence

The ValueSequence object is used to generate a specified sequence of numbers. It can be used instead of a Probability Distribution when a model is to be validated against recorded data. It can also be used to specify an interval or duration for a planned maintenance activity.

The next value in the sequence is returned every time the ValueSequence is referenced. The values are recycled after the last value in the list is returned.

Table 14-20 ValueSequence Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType	The unit type for the generated values.
ValueList	The sequence of numbers to be generated. Note that the appropriate unit for the numbers must be entered in the last position.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-21 ValueSequence Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
ValueSequence	
Index	The position of the last value returned in the list.
Value	The last value returned by the ValueSequence. When used in an expression, this output returns a new value every time the expression is evaluated.

14.11 EventSchedule

The EventSchedule object is similar to the ValueSequence object in that a specified sequence of values is returned. The difference is that an EventSchedule returns the inter-arrival times for a specified sequence of event times. When applied to the InterArrivalTime keyword for an EntityGenerator, it generates a specified sequence of entities at the simulation times provided to the EventSchedule.

Table 14-22 EventSchedule Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
TimeList	A sequence of monotonically-increasing simulation times at which to generate events. If entered in date format, an input of '0000-01-01 00:00:00' corresponds to zero simulation time.
CycleTime	Defines when the event times will repeat from the start.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-23 EventSchedule Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
ValueSequence	
Index	The position of the event time in the list for the last inter-arrival time that was returned.
Value	The last inter-arrival time returned from the sequence. When used in an expression, this output returns a new value every time the expression is evaluated.

14.12 FileToVector, FileToMatrix, and FileToHashMap

The FileToVector, FileToMatrix, and FileToHashMap objects read data contained in a file.

The DataFile keyword specifies the file to be read. The data must be delimited by either spaces or tabs (but NOT commas), and can consist of any mixture of numbers (with or without units), strings, objects, arrays, maps, and lambda functions. An entry that contains spaces must be enclosed by single quotes.

An entry can be given in expression format or it can take advantage of rules that allow a somewhat simpler format. The following table gives some examples of valid data entries:

Table 14-24 Data File Entries

Entry	Rule	Examples
Comment	Records that begin with a # symbol are ignored.	`# This is a comment`

Number	Numbers are entered in expression format.	`1.0`
		`1.0[m]`
		`'1.0 [m]'`

Time Stamp	Time stamps can be entered as either:
	`'YYYY-MM-DD HH:MM:SS.SSS' or`
	`YYYY-MM-DDTHH:MM:SS.SSS`
	See Table 4-16 for more details.
		`'0000-01-02 06:24:00.0'`
		`0000-01-02T06:24:00.0`

Object	Object names can be entered with or without square brackets	`[Server1]` `Server1`
Array	Arrays are entered in expression format. They must be enclosed by single quotes.	`'{1,2,3}'` `'{1, 2, 3}'`

String	Any entry that cannot be interpreted as a valid number, object, or array is treated as a string.	`'"Quick red fox"'`
		`'Quick red fox'`
		`Quick_red_fox`

When JaamSim is executed from the API, the DataFile input can be replaced by a call to the setValue method for this object, which populates the data directly.

The data read from the file is made available to the model through an output named Value. The two objects differ in the type of data returned by this output:

- FileToVector: the Value output returns a single array combining all the records in the file.
- FileToMatrix: the Value output returns an array of arrays, with one internal array for each record that was read.

- FileToHashMap: the Value output returns a map with the first entry in each line of the file treated as the key and with the corresponding value consisting of the remaining entries collected as an array.

The data file is first read when the simulation is started. It is re-read and the Value output updated whenever the object receives an entity.

Table 14-25 FileToVector, FileToMatrix, and FileToHashMap Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
DataFile	A file containing entries that are delimited by spaces and/or tabs.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-26 FileToVector, FileToMatrix, and FileToHashMap Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
FileToVector, FileToMatrix, and FileToHashMap	
Value	An array or map containing the data from the DataFile input.

14.13 ExternalProgram

The ExternalProgram object executes an external program when it receives an entity. Model execution is suspended until the external program terminates. The external program can be written in Python or any other language.

Table 14-27 ExternalProgram Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
ProgramFile	The executable file for the external program. If the external program is written in Python, the program file is the executable for Python interpreter (python.exe).
InputFile	Optional input file for the external program. If the external program is written in Python, the input file is the python code file (*.py)
DataSource	A list of expressions that provide the parameters to the external program. The inputs must be provided in the order in which they are to be entered in the external program's command line.
InitialValue	The Value output prior to receiving the first entity.
TimeOut	Maximum time in milliseconds for the external program to finish executing.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-28 ExternalProgram Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	Not used.

Output Name	Description
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
ExternalProgram	
Value	An array of values returned by the external program after parsing. Returned strings are converted automatically to numbers, times, or entities, if appropriate. For example, if the external program returns a single number, its value is `'this.Value(1)'`.

14.14 EntitySystem

The EntitySystem object allows the state for a group of objects to be assigned.

An EntitySystem's state is set to 'Working' if any of the objects in the system are working, and to 'Idle' if all the objects are idle. An optional keyword StateExpression allows user-defined states to be assigned to the EntitySystem depending on the states of the individual object in the system.

Table 14-29 EntitySystem Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
StateExpression	An expression returning a string that sets this object's present state. If left blank, the state will be set to Working if any of the entities specified by the WatchList input are working. It will be set to Idle if all of the entities in the WatchList are idle.
WatchList	A list of objects to monitor. The system's state will be re-calculated whenever one of the WatchList objects changes state.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Maintenance	
WorkingStateList	See Section 10.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-30 EntitySystem Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability	See Section 0.
EntitySystem	
EntityList	Entities included in this system. Consists of the entities in the WatchList for which a state can be obtained.

14.15 ScriptEntity

Originally developed for video capture, a ScriptEntity can be used to change window Views, to create automatic zooming and panning, and to toggle video capture during a run. Furthermore, keywords defined in the script file can be used to modify simulation and object parameters initially defined in the input configuration file.

At present, the ScriptEnity cannot be dragged and dropped into a model. It can only be created by editing the configuration file (see Section 8).

The ScriptEntity object takes only one keyword, which is the path to a script (.scr) file.

Table 14-31 ScriptEntity Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Script	Path to the file containing the scripting instructions to be loaded
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 14-32 ScriptEntity Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

The script file contains sets of model inputs preceded by the Time keyword under the ScriptEntity object.

Table 14-33 Inputs for .scr file

Keyword	Description
Time	The simulated time at which the subsequent inputs are executed.

For example, the following inputs can be entered into a .scr file and then referenced by a ScriptEntity object to slow down the model at a given point in the simulation run:

```
ScriptEntity1 Time            { 24.0 h }
Simulation    RealTime        { TRUE }
Simulation    RealTimeFactor  { 1200 }

ScriptEntity1 Time            { 30.0 h }
Simulation    RealTime        { FALSE }
```

15 Resource Objects Palette

Resources are objects that can be seized and released by other objects in a model, such as Seize, Release, and EntityProcessor. Once a resource is seized by an object, it cannot be seized by another object until it is released again. Resource objects do not carry out any tasks of their own.

Object		Description
	Resource	Set of identical resource units that can be seized and released by various processes.
	ResourcePool	Set of unique resource units that can be seized and released by various processes.
	ResourceUnit	A single resource that can be seized and released individually from a ResourcePool.

15.1 Resource

The Resource object is used to represent a pool of identical equipment or processor units.

The number of resource units is specified using the Capacity keyword, which can accept a constant value or an expression. If the number of units is allowed to vary, it is possible for the number of units in use to be greater than the present value for Capacity. If this situation occurs, the model continues normally and takes resource units out of service one-by-one as they are released. If the Capacity increases, the Resource attempts to make use of the additional capacity immediately.

Table 15-1 Resource Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
StrictOrder	If TRUE, the next entity to seize the resource will be chosen strictly on the basis of priority and waiting time. If this entity is unable to seize the resource because of other restrictions such as an OperatingThreshold input or the unavailability of other resources it needs to seize at the same time, then other entities with lower priority or shorter waiting time will NOT be allowed to seize the resource. If FALSE, the entities will be tested in the same order of priority and waiting time, but the first entity that is able to seize the resource will be allowed to do so.
Capacity	The number of equivalent resource units that are available. Only an integer number of resource units can be specified. A decimal value will be truncated to an integer. If the capacity changes during the simulation run, the Resource will attempt to use an increase in capacity as soon as it occurs. However, a decrease in capacity will have no effect on entities that have already seized Resource capacity.

Keyword	Description
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 15-2 Resource Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
AbstractResourceProvider	
UserList	The objects that can seize units from this resource.
Capacity	The total number of resource units that can be used.
UnitsInUse	The number of units that are in use.
AvailableUnits	The number of resource units that are not in use.
UnitsSeized	The number of units that have been seized.
UnitsReleased	The number of units that have been released.
UnitsInUseAverage	The average number of units that have been in use.
UnitsInUseStandardDeviation	The standard deviation of the number of units that have been in use.
UnitsInUseMinimum	The minimum number of units that have been in use.
UnitsInUseMaximum	The maximum number of units that have been in use.
UnitsInUseTimes	The total time that the number of resource units in use was 0, 1, 2, etc.

15.2 ResourcePool

A ResourcePool is a collection of separate ResourceUnits that have individual characteristics.

The ResourceUnits in a ResourcePool is determined by the ResourcePool input for the ResourceUnits.

Table 15-3 ResourcePool Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
StrictOrder	If TRUE, the next entity to seize the resource will be chosen strictly on the basis of priority and waiting time. If this entity is unable to seize the resource because of other restrictions such as an OperatingThreshold input or the unavailability of other resources it needs to seize at the same time, then other entities with lower priority or shorter waiting time will NOT be allowed to seize the resource. If FALSE, the entities will be tested in the same order of priority and waiting time, but the first entity that is able to seize the resource will be allowed to do so.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 15-4 ResourcePool Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
AbstractResourceProvider	
UserList	The objects that can seize units from this resource.
Capacity	The total number of resource units that can be used.
UnitsInUse	The number of units that are in use.
AvailableUnits	The number of resource units that are not in use.
UnitsSeized	The number of units that have been seized.
UnitsReleased	The number of units that have been released.
UnitsInUseAverage	The average number of units that have been in use.
UnitsInUseStandardDeviation	The standard deviation of the number of units that have been in use.
UnitsInUseMinimum	The minimum number of units that have been in use.
UnitsInUseMaximum	The maximum number of units that have been in use.
UnitsInUseTimes	The total time that the number of resource units in use was 0, 1, 2, etc.

15.3 ResourceUnit

A ResourceUnit represents a single resource that can be seized individually or from a ResourcePool.

An entity can select the most appropriate ResouceUnit to seize from a ResourcePool using the inputs to the AssignmentCondition and AssignmentPriority keywords for each ResourceUnit. The AssignmentCondition input determines whether the entity is eligible to seize the unit. The AssignmentPriority input determines which of the eligible units is most appropriate for the entity.

Inputs for the various ThresholdList, MaintenanceList, and BreakdownList keywords determine whether the unit is available to be seized.

A ResourceUnit will attach itself to the entity that seized it if its FollowAssignment input is set to TRUE.

The appearance of a ResourceUnit changes to indicate its present state. By default, it changes colour, but the StateGraphics keyword allows it to change to another shape or image.

Table 15-5 ResourceUnit Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
ResourcePool	The name of the ResourcePool from which this ResourceUnit can be selected. If no pool is specified, the ResourceUnit itself is considered to be a ResourcePool with one unit.
AssignmentCondition	An optional expression that tests whether an entity is eligible to seize this unit. The entry `'this.Assignment'` represents the entity that is being tested in the expression. The expression should return `1` (`TRUE`) if the entity is eligible. For example, the following expression requires that the entity's attribute `'type'` has the value `1`: `'this.Assignment.type == 1'`
AssignmentPriority	An optional expression that returns the priority for this unit to be used by the ResourcePool when choosing the next unit to be seized. The calculated priority should be a positive integer, with a lower value indicating a higher priority. The entry `'this.Assignment'` can be used in the expression to represent the entity that would seize the unit. For example, the following expression assigns the priority `1` to the ResourceUnit if the entity's attribute `'type'` has the value `1`, otherwise a priority of `2` is assigned: `'this.Assignment.type == 1 ? 1 : 2'`
Options	
Active	If TRUE, the object is used in the simulation run.
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.

Keyword	Description
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
FollowAssignment	If `TRUE`, the ResourceUnit will move next to the entity that has seized it, and will follow that entity until it is released.
AssignmentOffset	The position of the ResourceUnit relative to the entity that has seized it.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 15-6 ResourceUnit Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
ResourceUnit	
UserList	The objects that can seize this resource unit.
Assignment	The entity to which this unit is assigned.

16 Process Flow Palette

The Process Flow palette contains all the objects needed to create process flow type models. These models are characterized by a passive entity that is passed from one object to another following a process flow diagram. These types of models are often used to simulate a manufacturing process where the entities represent parts that are moved between processing stations. The following objects are provided in the Process Flow palette.

Table 16-1 Process Flow Palette

Object		Description
	SimEntity	The basic entity for use in a process flow type model.
	EntityLauncher	Creates a copy of a prototype entity when clicked by the user.
	EntityGenerator	Creates copies of a prototype entity at specified intervals.
	EntitySink	Destroys any entity it receives.
	Server	Processes a received entity over a specified duration.
	Queue	Stores received entities until they are needed.
	EntityConveyor	Transports a received entity along a specified path at a fixed speed.
	EntityDelay	Delays a received entity by a specified duration.
	Seize	Seizes one or more units of a Resource.
	Release	Releases one or more units of a Resource.
	EntityProcessor	Seizes one or more resources, delays the received entity by a specified duration, and releases the resources.
	Assign	Assigns new values to attributes.
	Branch	Directs a received entity to a selected destination.
	Duplicate	Sends copies of the received entity to a set of destinations.
	Combine	Matches entities from multiple queues. The entity from the first queue is passed on while the other entities are destroyed.
	SetGraphics	Changes the appearance of the received entity.
	EntityGate	Blocks received entities from progressing further until the EntityGate is opened by one or more Thresholds.
	EntitySignal	Opens or closes a specified SignalThreshold when an entity is received.
	SignalThreshold	Threshold that is opened and closed directly by an EntitySignal object.
	Assemble	Combines sub-components to create an assembled part.
	EntityContainer	An entity that can hold one or more entities.

Object		Description
	Pack	Inserts entities in a new EntityContainer.
	Unpack	Removes all the entities from an EntityContainer which is subsequently destroyed.
	AddTo	Add entities to an existing EntityContainer.
	RemoveFrom	Removes some or all of the entities from an EntityContainer.
	EntityLogger	Records the outputs and state data for a generated entity in an output log.
	Statistics	Collects statistics from the received entities.

Many of the objects in the Process Flow palette provide the following outputs.

Table 16-2 LinkedComponent Outputs

Output Name	Description
obj	The entity that was received most recently.
NumberAdded	The number of entities received from upstream after the initialization period.
NumberProcessed	The number of entities processed by this component after the initialization period.
NumberInProgress	The number of entities that have been received but whose processing has not been completed yet.
ProcessingRate	The number of entities processed per unit time by this component after the initialization period.
ReleaseTime	The time at which the last entity was released.

16.1 SimEntity

The SimEntity object is the basic entity that is passed through a process flow type model.

The main feature of SimEntity is that its state can be assigned at various stages of the process flow. Process Flow objects that can receive a SimEntity, such as Server and Queue, can assign a state to the received SimEntity using their StateAssignment keyword. For example, if the StateAssignment input for a Queue is set to "Waiting", then the state for each SimEntity received by the Queue will be set to "Waiting" when it arrives at the Queue. The SimEntity will remain in this state until it is set to a new state by a subsequent object.

The time spent a SimEntity in each state can be accessed using its StateTimes output. For example, the total time spent by the SimEntity in the state "Waiting" is given by `StateTimes("Waiting")`.

Unlike other objects, the StateTimes data for a SimEntity is not cleared at the end of the initialization period (warm-up). This is done to avoid having a Statistics object record artificially small times for some states that would be become the minimum times for those states.

Table 16-3 SimEntity Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
DefaultStateList	A list of states that will always appear in the output report, even if no time is recorded for this state.
InitialState	The state of the SimEntity at the start of the simulation run or when it is first created during a simulation run.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-4 SimEntity Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 11.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.

16.2 EntityLauncher

The EntityLauncher object creates an entity when it is clicked by the user or when a specified keyboard key is pressed.

The simulation model must be executing in Real Time mode for entities to be generated. For the keyboard action to be detected, the View window containing the EntityLauncher must be selected.

The PrototypeEntity keyword identifies the entity to be copied. This entity can be any type of object, no matter how complex. Either a specific object or an expression that returns an object can be entered. The generated copies retain both the graphics of the prototype as well as the values of all its inputs.

The ActionKey input is used to specify a keyboard shortcut for creating an entity.

Table 16-5 EntityLauncher Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
ActionKey	An optional keyboard key that will cause the EntityLauncher to generate an entity.
PrototypeEntity	The entity to be copied by the EntityLauncher.
NextComponent	The next object to which a generated entity is passed.
BaseName	The base for the names assigned to the generated entities. The generated entities will be named Name1, Name2, etc.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable	See Section 11.1.

Table 16-6 EntityLauncher Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 11.1.

16.3 EntityGenerator

The EntityGenerator object creates a series of entities that are passed to the next object in a process.

The PrototypeEntity keyword identifies the entity to be copied. This entity can be any type of object, no matter how complex. Either a specific object or an expression that returns an object can be entered. Copies retain both the graphics of the prototype as well as the values of all its inputs.

The rate at which entities are generated is determined by the InterArrivalTime and FirstArrivalTime keywords. These inputs have units of time and can be a constant value, an object that returns a number with units of time (e.g. TimeSeries or Probability Distribution), or an expression that returns such a number.

Table 16-7 EntityGenerator Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which a generated entity is passed.
FirstArrivalTime	The time at which the first entity is to be generated.
InterArrivalTime	The elapsed time between one generated entity and the next.
EntitiesPerArrival	The number of entities to be generated for each arrival time.
PrototypeEntity	The entity to be copied by the EntityGenerator.
BaseName	The base for the names assigned to the generated entities. The generated entities will be named Name1, Name2, etc.
MaxNumber	The maximum number of entities to be generated.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-8 EntityGenerator Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	Inherited output that is not relevant for EntityGenerator.
EntityGenerator	
NumberGenerated	The total number of entities generated, including the initialization period.
PresentIAT	The total working time required before the next entity is created.
ElapsedTime	The working time that has been completed towards the creation of the next entity.
FractionCompleted	The portion of the total working time towards the creation of the next entity that has been completed.

16.4 EntitySink

The EntitySink object destroys incoming entities.

Table 16-9 EntitySink Key Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.

Keyword	Description
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-10 EntitySink Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

16.5 Server

The Server object processes an incoming entity and then passes it to the next object.

Entities that are waiting to be processed are held by a Queue object identified by the keyword WaitQueue. All entities received by the Server first pass through this Queue object, even if the Server is Idle. Entities can be sent to the Server or directly to the Queue. The choice has no effect on the model logic.

Whenever an entity is added to a Queue, all the Servers and other type of objects that specified this Queue as its WaitQueue will be notified. The first one that is available will then remove the entity for processing.

The rate at which entities are processed is determined by the ServiceTime keyword. This input has the units of time and can be a constant value, an object that returns a number with units of time (e.g. TimeSeries or Probability Distribution), or an expression that returns such a number.

The Server can be stopped under various circumstances using the OperatingThresholdList keyword, which specifies a list of threshold objects such as SignalThreshold, TimeSeriesThreshold, or

ExpressionThreshold. All of the specified threshold objects must be open for the Server to operate. However, if a threshold closes while the Server is processing an entity, it will complete its work on that entity before ceasing further operation.

Although a Server is alerted automatically when an entity arrives to its WaitQueue, it is NOT alerted automatically when its Match input changes or when an expression entered WaitQueue keyword changes the Queue it returns. The WatchList keyword provides a way to alert the Server to these types of changes.

Table 16-11 Server Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.
WatchList	An optional list of objects to monitor. The queue will be inspected for an entity to process whenever one of the WatchList objects changes state.
ServiceTime	The service time required to process an entity.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.

Keyword	Description
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the processor.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-12 Server Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.
Server	
ServiceDuration	The total working time required for the present service activity.
ServicePerformed	The working time that has been completed for the present service activity.
FractionCompleted	The portion of the total service time for the present service activity that has been completed.

16.6 Queue

A Queue object defines a location for simulation entities to wait for processing by other entities.

Unlike many other objects in this palette, an entity received by a Queue is not passed automatically to the next object. It must wait in the Queue until it is removed by some other object. Queues are used in this way by the Server, Seize, EntityGate, Assemble, Combine, Pack, Unpack, AddTo, and RemoveFrom objects. Whenever an entity is added to a Queue, all the objects that use this Queue are notified that a new entity is available. The first available object will remove the entity from the Queue and start processing it.

Queued entities can be sequenced by an optional priority value specified by the Priority keyword. In most cases, the Priority is specified by an Expression that is evaluated when the entity first arrives at the Queue. Priority is integer valued and decimal values will be truncated, which means, for example, that priority values of 3.2 and 3.6 are identical (i.e. truncated to 3).

Entities with the same priority values can be sequenced in either the default first-in-first-out (FIFO) order, or in last-in-first-out order (LIFO).

Lastly, a queued entity can be provided with an optional identification using the Match keyword. Objects that use Queues, such as a Server, can request its Queue to provide the first entity with a specified value for the Match keyword. As with the Priority keyword, in most cases the Match value is specified by an Expression that is evaluated when the entity first arrives at the Queue. The Match variable is stored as a string, but will also accept a dimensionless integer or an entity which will be converted to a string. If a decimal value is provided, it will be truncated.

Table 16-13 Queue Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Priority	The priority for positioning a received entity in the Queue. Priority is integer valued and a lower numerical value indicates a higher priority. Priority is normally specified by an Expression, however, a dimensionless number or an object that returns a dimensionless number such as a TimeSeries or a Probability Distribution can also be used.
Match	An expression returning a string value that categorizes the queued entities. The expression is evaluated and the value saved when the entity first arrives at the queue. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer.
FIFO	The order in which entities are placed in the queue. TRUE indicates first-in-first-out order (FIFO). FALSE indicates last-in-first-out order (LIFO).

Keyword	Description
RenegeTime	The time an entity will wait in the queue before deciding whether or not to renege. Evaluated when the entity first enters the queue. A constant value, a distribution to be sampled, a time series, or an expression can be entered.
RenegeCondition	A logical condition that determines whether an entity will renege after waiting for its RenegeTime value. Note that TRUE and FALSE are entered as 1 and 0, respectively. A constant value, a distribution to be sampled, a time series, or an expression can be entered.
RenegeDestination	The object to which an entity will be sent if it reneges.
MaxValidLength	Maximum number of objects that can be placed in the queue. An error message is generated if this limit is exceeded. This input is intended to trap a model error that causes the queue length to grow without bound. It has no effect on model logic.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Format	
Spacing	The amount of graphical space between objects in the Queue.
MaxPerLine	Maximum number of objects in each row of the Queue.
MaxRows	The number of rows in each level of the Queue.
ShowEntities	If TRUE, the objects in the Queue are displayed.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-14 Queue Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

Output Name	Description
Queue	
QueueLength	The present number of objects in the Queue.
QueueList	The entities in the queue.
QueueTimes	The waiting time for each entity in the queue.
PriorityValues	The Priority expression value for each entity in the queue.
MatchValues	The Match expression value for each entity in the queue.
QueueLengthAverage	The average number of objects in the Queue.
QueueLengthStandardDeviation	The standard deviation of the number of objects in the Queue.
QueueLengthMinimum	The fewest number of objects in the Queue.
QueueLengthMaximum	The largest number of objects in the Queue.
QueueLengthTimes	The total time that the queue has length 0, 1, 2, etc.
AverageQueueTime	The average time each entity waited in the Queue. Calculated as total accumulated queue time divided by the number of entities added to the Queue.
MatchValueCount	The present number of unique Match values in the queue.
UniqueMatchValues	The list of unique Match values for the entities in the queue.
MatchValueCountMap	The number of entities in the queue for each Match expression value. For example, `'[Queue1].MatchValueCountMap("SKU1")'` returns the number of entities whose Match value is `"SKU1"`.
MatchValueMap	Provides a list of entities in the queue for each Match expression value. For example, `'[Queue1].MatchValueMap("SKU1")'` returns a list of entities whose Match value is `"SKU1"`.
NumberReneged	The number of entities that reneged from the queue.
QueuePosition	The position in the queue for an entity undergoing the RenegeCondition test. First in queue = 1, second in queue = 2, etc.

16.7 EntityConveyor

The EntityConveyor object moves an incoming entity along a path at a fixed speed, and then passes it to the next object.

The travel time for the EntityConveyor is determined by the TravelTime keyword. If a variable travel time is specified through an expression, the conveyor's speed is updated whenever an entity is added to the conveyor or an entity reaches the end of the conveyor.

Table 16-15 EntityConveyor Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
TravelTime	The time required to convey an entity from the start to the end.
Width	The width of the line representing the EntityConveyor in pixels.
Color	The colour of the line representing the EntityConveyor.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
RotateEntities	If TRUE, the entities are rotated to match the direction of the path.
LineWidth	The width of the conveyor in pixels.
LineColour	The colour of the conveyor.
Graphics	
Points, CurveType, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-16 EntityConveyor Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Attributes, Custom Outputs, Position, Size, Orientation, Alignment	See Section 10.1.

Output Name	Description
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	Not used.
EntityConveyor	
EntityList	The entities being processed at present.

16.8 EntityDelay

The EntityDelay object delays an incoming entity by a variable duration before passing it to the next object.

The delay is represented as motion along a line that is similar in appearance to the EntityConveyor object. It differs, however, in that the entities moving along the line can pass one another due to their different delay times.

The duration of each entity's delay is determined by the Duration keyword. This input has units of time and accepts a constant value, a TimeSeries, a Probability Distribution, or an Expression.

Table 16-17 EntityDelay Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
Duration	The delay time for the path.
AllowOvertaking	If TRUE, an entity can pass a second entity that started ahead of it. If FALSE, the entity's duration is increased sufficiently for it to arrive no earlier than the previous entity.
MinSeparation	The minimum time between the previous entity leaving the path and the present entity leaving the path. Applicable only when AllowOvertaking is FALSE.

Keyword	Description
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
Animation	If TRUE, an entity is moved along the specified path to indicate its progression through the delay activity.
RotateEntities	If TRUE, the entities are rotated to match the direction of the path.
LineWidth	The width in pixels of the line representing the EntityDelay.
LineColour	The colour of the line representing the EntityDelay.
Graphics	
Points, CurveType, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-18 EntityDelay Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Attributes, Custom Outputs, Position, Size, Orientation, Alignment	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
EntityDelay	
EntityList	The entities being processed at present.

16.9 Seize

The Seize object allocates one or more units of a specified set of Resources on receiving an incoming entity.

If any of the Resources have insufficient units available, the received entity is directed to a Queue object identified by the WaitQueue keyword. All entities received by the Seize object first pass through this Queue object, even if sufficient units of the Resources are available.

Entities can be sent directly to the Queue specified by the WaitQueue keyword. Whenever an entity is added to the Queue, all the Seize objects that specified this Queue as their WaitQueue will be notified. The first Seize block that has sufficient resource units available will then remove the entity for processing.

Entities waiting for the same Resources at more than one Seize object are processed in order of the priority assigned to them by the input to their Queue's Priority keyword. If the entities in two or more Queues have the same priority value, then the one that has waited the longest is chosen.

All of the specified Resources for the selected entity must be available before any are seized. Once all of them are available, they are seized simultaneously.

Table 16-19 Seize Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.
ResourceList	The Resources from which units are to be seized. All the resource units must be available to be seized before any one unit is seized. When more than one object attempts to seize the same resource, the resource is assigned based on the priorities and arrival times of entities waiting in the objects' Queues. An entity's priority is determined by the Priority input for its Queue, and is assigned to the entity when it first arrives to the Queue. This priority determines both the position of the entity in the queue and its priority for seizing a resource. If several entities have the same priority, the resource is assigned to entity that arrived first to its Queue.

Keyword	Description
NumberOfUnits	The number of units to seize from the Resources specified by the ResourceList keyword. The last value in the list is used if the number of resources is greater than the number of values. Only an integer number of resource units can be seized. A decimal value will be truncated to an integer.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-20 Seize Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.

Output Name	Description
Seize	
SeizedUnits	The number of resource units seized by the last entity.

16.10 Release

The Release object de-allocates one or more units of a specified set of Resources on receiving an incoming entity. All of the Resources are released simultaneously.

On release of the Resources, the entities waiting for these Resources are processed in order of the priority assigned to them by the input to their Queue's Priority keyword. If the entities in two or more Queues have the same priority value, then the one that has waited the longest is chosen.

Table 16-21 Release Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
ResourceList	The Resources from which units are to be released.
NumberOfUnits	The number of units to release from the Resources specified by the ResourceList keyword. The last value in the list is used if the number of resources is greater than the number of values. Only an integer number of resource units can be released. A decimal value will be truncated to an integer.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-22 Release Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

16.11 EntityProcessor

The EntityProcessor simulates multiple parallel servers that share the same queue. The servers are assumed to have exactly the same characteristics, which makes them interchangeable.

The Capacity input specifies the number of parallel servers.

The WaitQueue input identifies the Queue from which the entities are drawn for processing.

The optional ResourceList and NumberOfUnits keywords determine whether additional resources need to be seized before processing can begin. All the resource units must be available before the next entity can be removed from the WaitQueue. The resource units are seized simultaneously.

When the processing of an entity is completed, its resources are released, the entity is sent to the object specified by the NextComponent input.

Table 16-23 EntityProcessor Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.

Keyword	Description
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.
WatchList	An optional list of objects to monitor. The queue will be inspected for an entity to process whenever one of the WatchList objects changes state.
ResourceList	The Resources from which units are to be seized. All the resource units must be available to be seized before any one unit is seized. When more than one object attempts to seize the same resource, the resource is assigned based on the priorities and arrival times of entities waiting in the objects' Queues. An entity's priority is determined by the Priority input for its Queue, and is assigned to the entity when it first arrives to the Queue. This priority determines both the position of the entity in the queue and its priority for seizing a resource. If several entities have the same priority, the resource is assigned to entity that arrived first to its Queue.
NumberOfUnits	The number of units to seize from the Resources specified by the ResourceList keyword. The last value in the list is used if the number of resources is greater than the number of values. Only an integer number of resource units can be seized. A decimal value will be truncated to an integer.
Capacity	The maximum number of entities that can be processed simultaneously. If the capacity changes during the simulation run, the EntityProcessor will attempt to use an increase in capacity as soon as it occurs. However, a decrease in capacity will have no effect on entities that have already started processing.
ServiceTime	The service time required to process an entity.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.

Keyword	Description
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the processor.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-24 EntityProcessor Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.
Server	
Capacity	The present number of entities that can be processed simultaneously.
UnitsInUse	The present number of capacity units that are being used.
EntityList	The entities being processed at present.
RemainingTime	The remaining processing time for the entities being processed at present.

16.12 Assign

The Assign object performs one or more Attribute assignments whenever it receives an incoming entity. Once the assignments have been performed, the received entity is passed to the next object without delay.

The Assign object is the only place where an Attribute's value can be modified. The Assign object can modify the value of any attribute in the model: its own attributes, the received entity's attributes, or any other object's attributes.

The attribute assignments to be performed are specified by the input to the AttributeAssignmentList keyword. Each assignment has the following form:

```
{ <left-hand side expression> = <right-hand side expression> }
```

The right side of each assignment equation is an expression to be evaluated. The left side is an expression that identifies the attribute whose value is to be modified. For example, if object Assign1 has a dimensionless numerical attribute `A`, then the following input to its AttributeAssignmentList keyword causes the attribute to be increased by 1:

```
{ 'this.A = this.A + 1' }
```

The Assign object attribute `obj` can be used to access the attributes of the received entity. For example, if the received entity has an attribute `B` whose value is a string, then the following input to the AttributeAssignmentList keyword causes this attribute to take the value `'New String'`:

```
{ 'this.obj.B = "New String"' }
```

If another object, say Server1, has an attribute `C` whose value is a number with units of distance, then the following input to the AttributeAssignmentList keyword causes this attribute to be increased by 1 kilometre:

```
{ '[Server1].C = [Server1].C + 1[km]' }
```

The entries in an array or map can also be modified. For example, the following input assigns the value `1` to the second entry in an array-valued attribute `ArrayAttrib` and the value `2` to the key `"abc"` in the map-valued attribute `MapAttrib`:

```
{  'this.ArrayAttrib(2) = 1' }  { 'this.MapAttrib("abc") = 2' }
```

Furthermore, the index for an array assignment or the key for a map can be a calculated quantity. For example, the following input assigns the value `1` to the index of `ArrayAttrib` that is given by the attribute `N` of the entity that is being processed:

```
{  'this.ArrayAttrib(this.obj.N) = 1' }
```

An entry in a two-index array (an array of arrays) can be set by specifying two separate indices. For example, if the attribute `MatrixAttrib` is the two-index array `{{0,0},{0,0}}`, then the following input assigns the value `1` to the second entry in the first nested array:

```
{  'this.MatrixAttrib(1)(2) = 1' }
```

An unlimited number of assignments can be performed by one Assign object. The assignments are processed in the order in which they appear in the AttributeAssignmentList input.

Table 16-25 Assign Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
AttributeAssignmentList	A list of attribute assignments that are triggered when an entity is received.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-26 Assign Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

16.13 Branch

The Branch object directs an incoming entity to a destination that is chosen from a list of alternatives.

The value for the Choice keyword determines the destination that is chosen: 1 = first branch, 2 = second branch, etc. The input to Choice can be a constant value, a DiscreteDistribution, a TimeSeries, or an Expression. The use of an Expression allows the choice to be made based on the Attributes of the incoming entity.

Table 16-27 Branch Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponentList	A list of possible objects to which the processed DisplayEntity can be passed.
Choice	A number that determines the choice of next component: 1 = first branch, 2 = second branch, etc.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-28 Branch Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

16.14 Duplicate

The Duplicate object sends copies of the received entity to one or more objects.

Table 16-29 Duplicate Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
TargetComponentList	The list of components that will receive the duplicated entities. One duplicated entity will be sent to each entry in the list.
BaseName	The base for the names assigned to the duplicated entities. The duplicated entities will be named Name1, Name2, etc.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-30 Duplicate Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

16.15 Combine

The Combine object takes one entity each from multiple queues and passes on a single entity to the next object.

If the MatchRequired input it TRUE, then each of the entities must have the same match value calculated by the input to Match keyword for its Queue.

Entities are combined when the number of entities specified by the NumberRequired input is available in each queue. When a match is found, the entity in the first Queue is passed to the object specified by the NextComponent keyword, while the entities from the other Queues are destroyed.

Table 16-31 Combine Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
ServiceTime	The time required to process the incoming entity. Can be a constant value, a TimeSeries, a Probability Distribution, or an Expression.
WaitQueueList	A list of Queue objects that hold the entities waiting to be combined.
NumberRequired	The number of entities required from each queue for the assembly process to begin. The last value in the list is used if the number of queues is greater than the number of values.
MatchRequired	If TRUE, the all entities to be processed must have the same Match value. The Match value for an entity is determined by the Match keyword for its queue. The value is calculated when the entity first arrives at its queue.
FirstQueue	Determines which Match value to use when several values have the required number of entities. If FALSE, the entity with the earliest arrival time in any of the queues determines the Match value. If TRUE, the entity with the earliest arrival time in the first queue determines the Match value.
RetainAll	If TRUE, all the matching entities are passed to the next component. If FALSE, only the entity in the first queue is passed on.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.

Keyword	Description
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
ProcessPosition	The position of the entity being processed relative to the position of the Combine object.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-32 Combine Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.

16.16 SetGraphics

The SetGraphic object is used to change the graphical appearance of a specified entity.

Table 16-33 SetGraphics Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
TargetEntity	The entity whose graphics are to be changed. Defaults to the entity that was received.
GraphicsList	List of entities whose graphics can be chosen for assignment to the target entity.
Choice	A number that determines the choice of entities from the GraphicsList: 1 = first entity's graphics, 2 = second entity's graphics, etc.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-34 SetGraphics Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

16.17 EntityGate

The EntityGate object allows incoming entities to be blocked temporarily.

When the EntityGate is open and the Queue is empty, incoming entities pass through without delay. When it is closed, incoming entities are directed to a Queue where they are held until the EntityGate becomes Open. When the EntityGate opens, the queued entities are released one at time, separated by a delay determined by the ReleaseDelay keyword. This delay does not apply to entities that arrive when the EntityGate is open. However, if queued entities are still being released, an incoming entity is placed at the end of the Queue even though the EntityGate is Open.

The EntityGate's state, either Open or Closed, is determined by the input to the OperatingThresholdList keyword, which specifies a list of threshold objects such as SignalThreshold, TimeSeriesThreshold, or ExpressionThreshold. All of the specified threshold objects must be open for the EntityGate to become Open.

Table 16-35 EntityGate Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.
WatchList	An optional list of objects to monitor. The queue will be inspected for an entity to process whenever one of the WatchList objects changes state.
ReleaseDelay	The time delay before each queued entity is released. Entities arriving at an open gate are not delayed.
NumberToRelease	Maximum number of entities to release each time the gate is opened.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.

Keyword	Description
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the processor.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-36 EntityGate Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.

16.18 EntitySignal

The EntitySignal object sets the state of a SignalThreshold object when it receives an incoming entity.

The SignalThreshold and its new state are specified by the TargetSignalThreshold and NewState keywords, respectively.

Table 16-37 EntitySignal Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
TargetSignalThreshold	The SignalThreshold object whose state will be changed by this EntitySignal.
NewState	The state to be set for the SignalThreshold: TRUE if it is open, FALSE if it is closed.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-38 EntitySignal Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

16.19 SignalThreshold

The SignalThreshold object varies its state between open and closed when instructed to do so by one or more EntitySignal objects.

SignalThreshold has no internal logic of its own for changing state.

Table 16-39 SignalThreshold Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
InitialState	The initial state for the SignalThreshold at the start of the run. `TRUE` = open, `FALSE` = closed.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.
OpenColour, ClosedColour, ShowWhenOpen, ShowWhenClosed	See Section 10.1.

Table 16-40 SignalThreshold Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
Threshold	
Open, OpenFraction, ClosedFraction, OpenCount, ClosedCount	See Section 10.1.

16.20 Assemble

The Assemble object combines a number of sub-components into an assembled part.

Sub-components waiting for processing are collected into a series of Queue objects, identified by the WaitQueueList keyword, one for each type of sub-component.

Incoming sub-components must be sent directly to the appropriate queue, not to the Assemble object itself. If necessary, a Branch object can be used to direct incoming entities to the various queues.

The assembly process begins when there is at least one sub-component entity in each of the Queues. When this occurs, the first sub-component in each Queue is removed and destroyed, and a new assembled part is created by copying the object specified by the PrototypeEntity keyword. The time required to complete the assembly process is given by the ServiceTime keyword.

Table 16-41 Assemble Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
ServiceTime	The service time required to perform the assembly process.
WaitQueueList	A list of Queue objects in which to place the arriving sub-component entities.
NumberRequired	The number of entities required from each queue for the assembly process to begin. The last value in the list is used if the number of queues is greater than the number of values.
MatchRequired	If TRUE, the all entities to be processed must have the same Match value. The Match value for an entity is determined by the Match keyword for its queue. The value is calculated when the entity first arrives at its queue.
FirstQueue	Determines which Match value to use when several values have the required number of entities. If FALSE, the entity with the earliest arrival time in any of the queues determines the Match value. If TRUE, the entity with the earliest arrival time in the first queue determines the Match value.
PrototypeEntity	The prototype for entities representing the assembled part.
BaseName	The base for the names assigned to the entities representing the assembled parts. The entities will be named Name1, Name2, etc.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.

Keyword	Description
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the position of the Assemble object.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-42 Assemble Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.

16.21 EntityContainer

The EntityContainer object is used to store a collection of entities for subsequent processing as a unit.

An EntityContainer can be passed to any object that accepts a SimEntity. The Pack and AddTo objects are used to place entities in an EntityContainer. The Unpack and RemoveFrom objects are used to remove entities from an EntityContainer.

Table 16-43 EntityContainer Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
DefaultStateList	A list of states that will always appear in the output report, even if no time is recorded for this state.
InitialState	The state assigned to the entity when it is first created.
Priority	The priority for positioning the received entity in the EntityContainer. Priority is integer valued and a lower numerical value indicates a higher priority. For example, priority 3 is higher than 4, and priorities 3, 3.2, and 3.8 are equivalent.
Match	An expression returning a string value that categorizes the entities in the EntityContainer. The expression is evaluated and the value saved when the entity is first loaded into the EntityContainer. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer.
FIFO	Determines the order in which entities are placed in the EntityContainer (FIFO or LIFO): TRUE = first in first out (FIFO) order (the default setting), FALSE = last in first out (LIFO) order.
SetEntityState	Determines whether the states for the entities held by the EntityContainer are updated to match each change to the EntityContainer's state.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
PositionOffset	The position of the first entity in the container relative to the EntityContainer.
Spacing	The amount of graphical space shown between entities in the EntityContainer.
MaxPerLine	The number of entities in each row inside the EntityContainer.

Keyword	Description
MaxRows	The number of rows in each level of entities inside the EntityContainer.
ShowEntities	If TRUE, the entities in the EntityContainer are displayed.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-44 EntityContainer Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
EntityContainer	
obj	The entity that was loaded most recently.
NumberAdded	The number of entities loaded after the initialization period.
NumberRemoved	The number of entities unloaded after the initialization period.
Count	The present number of entities in the EntityContainer.
EntityList	The entities contained by the EntityContainer.
PriorityValues	The Priority expression value for each entity in the EntityContainer.
MatchValues	The Match expression value for each entity in the EntityContainer.
StorageTimes	The elapsed time since each entity was placed in the EntityContainer.
MatchValueCount	The present number of unique Match values in the EntityContainer.
UniqueMatchValues	The list of unique Match values for the entities in the EntityContainer.
MatchValueCountMap	The number of entities in the EntityContainer for each Match expression value. For example, `'[EntityContainer1].MatchValueCountMap("SKU1")'` returns the number of entities whose Match value is `"SKU1"`.
MatchValueMap	Provides a list of entities in the EntityContainer for each Match expression value. For example, `'[EntityContainer1].MatchValueMap("SKU1")'` returns a list of entities whose Match value is `"SKU1"`.

16.22 Pack

The Pack object is used to generate EntityContainers and fill them with incoming entities.

EntityContainers are created automatically on demand by the Pack object. The number of entities to be packed in each EntityContainer is determined by the NumberOfEntities keyword. Once the specified number of entities is available in the Queue, the entities are packed one by one in the EntityContainer in the same order as the Queue. The time to pack each entity is specified by the ServiceTime keyword.

Table 16-45 Pack Key Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.
WatchList	An optional list of objects to monitor. The queue will be inspected for an entity to process whenever one of the WatchList objects changes state.
NumberOfEntities	The number of entities to pack into the container.
ServiceTime	The service time required to pack each entity in the container.
PrototypeEntityContainer	The prototype for the EntityContainers to be generated. The generated EntityContainers will be copies of this entity.
BaseName	The base for the names assigned to the generated EntityContainers. The generated containers will be named Name1, Name2, etc.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.

Keyword	Description
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
ContainerStateAssignment	The state to be assigned to container on arrival at this object. No state is assigned if the entry is blank.
NumberToStart	The minimum number of entities required to start packing.
WaitForEntities	If `TRUE`, the EntityContainer will be held in its queue until sufficient entities are available to start packing.
Thresholds	See Section 10.1.
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the processor.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-46 Pack Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.

Output Name	Description
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.
Pack	
Container	The EntityContainer that is being filled.

16.23 Unpack

The Unpack object is used to remove the entities from incoming EntityContainers.

Entities are unpacked one by one from the EntityContainer in the same order as they were packed. The time to unpack each entity is specified by the ServiceTime keyword. The received EntityContainer is destroyed once it has been unpacked.

Table 16-47 Unpack Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.

Keyword	Description
WatchList	An optional list of objects to monitor. The queue will be inspected for an entity to process whenever one of the WatchList objects changes state.
MatchForEntities	An expression returning a string value that determines which of the entities in the container are eligible to be removed. If used, the only entities eligible for selection are the ones whose inputs for the container's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer.
ServiceTime	The service time required to unpack each entity.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
ContainerStateAssignment	The state to be assigned to container on arrival at this object. No state is assigned if the entry is blank.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the processor.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-48 Unpack Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.
AbstractUnpack	
Container	The EntityContainer that is being unpacked.

16.24 AddTo

The AddTo object is used to add a given number of entities to a received EntityContainer.

The number of entities to be added to each EntityContainer is determined by the NumberOfEntities keyword. Once the specified number of entities is available in the WaitQueue and there is an EntityContainer waiting in the ContainerQueue, the entities are added one by one to the EntityContainer in the same order as they appeared in the WaitQueue. The time to add each entity is specified by the ServiceTime keyword. EntityContainers are filled in the order in which they appear in the ContainerQueue.

When an input to the Match keyword is provided, only the entities in the WaitQueue that have the same Match value are added to the EntityContainer. The Match keyword for the ContainerQueue is not used by this logic.

Table 16-49 AddTo Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.
WatchList	An optional list of objects to monitor. The queue will be inspected for an entity to process whenever one of the WatchList objects changes state.
NumberOfEntities	The number of entities to pack into the container.
ServiceTime	The service time required to pack each entity in the container.
ContainerQueue	The Queue in which the arriving EntityContainers are stored while they wait for processing.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
ContainerStateAssignment	The state to be assigned to container on arrival at this object. No state is assigned if the entry is blank.
NumberToStart	The minimum number of entities required to start packing.
WaitForEntities	If TRUE, the EntityContainer will be held in its queue until sufficient entities are available to start packing.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.

Keyword	Description
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the processor.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-50 AddTo Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.
Pack	
Container	The EntityContainer that is being filled.

16.25 RemoveFrom

The RemoveFrom object is used to remove a given number of entities from an incoming EntityContainer.

Entities are removed one by one from the EntityContainer in the same order as they were packed. The time to unpack each entity is specified by the ServiceTime keyword. EntityContainers are passed to another object once the specified number of entities has been unpacked.

Table 16-51 RemoveFrom Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
WaitQueue	Queue from which the next entity for processing will be selected. If an expression is entered that can return various Queues, it is necessary for each Queue to be included in the entry to the WatchList keyword. If a Queue is not included, then the arrival of an entity to that Queue will not wake up this processor.
Match	An expression returning a string value that determines which of the queued entities are eligible to be selected. If used, the only entities eligible for selection are the ones whose inputs for the Queue's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer. Note that a change in the Match value does not trigger the processor automatically to re-check the Queue. The processor can be triggered by adding one or more objects to the WatchList input.
WatchList	An optional list of objects to monitor. The queue will be inspected for an entity to process whenever one of the WatchList objects changes state.
MatchForEntities	An expression returning a string value that determines which of the entities in the container are eligible to be removed. If used, the only entities eligible for selection are the ones whose inputs for the container's Match keyword are equal to value returned by the expression entered for this Match keyword. Expressions that return a dimensionless integer or an object are also valid. The returned number or object is converted to a string automatically. A floating point number is truncated to an integer.
ServiceTime	The service time required to unpack each entity.
NumberOfEntities	The maximum number of entities to remove from the container.
NextForContainers	The next object to which the processed EntityContainer is passed.

Keyword	Description
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
StateAssignment	The state to be assigned to each entity on arrival at this object. No state is assigned if the entry is blank.
ContainerStateAssignment	The state to be assigned to container on arrival at this object. No state is assigned if the entry is blank.
Thresholds	
ImmediateThresholdList, ImmediateReleaseThresholdList, OperatingThresholdList	See Section 10.1.
Maintenance	
WorkingStateList, ImmediateMaintenanceList, ForcedMaintenanceList, OpportunisticMaintenanceList, ImmediateBreakdownList, ForcedBreakdownList, OpportunisticBreakdownList	See Section 10.2.
Format	
StateGraphics	A list of state/DisplayEntity pairs. For each state, the graphics will be changed to those for the corresponding DisplayEntity, e.g. `{ Idle DisplayEntity1 } { Working DisplayEntity2 }`
ProcessPosition	The position of the entity being processed relative to the processor.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-52 RemoveFrom Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
StateUserEntity	
Idle, Working, Setup, Maintenance, Breakdown, Stopped, Utilisation, Commitment, Availability, Reliability, Open, NextMaintenanceTime, NextBreakdownTime	See Section 0.

Output Name	Description
LinkedDevice	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
LinkedService	
MatchValue	The present value to be matched to an entity in the queue.
AbstractUnpack	
Container	The EntityContainer that is being unpacked.

16.26 EntityLogger

The EntityLogger object records the outputs, attributes, and state data for each entity that it receives.

The output log file is created automatically when the simulation run begins. The output file is named <configuration file name>-<EntityLogger name>.log to ensure that it is unique for the simulation run. For example, if the configuration file is named "run1.cfg" and the EntityLogger's name is EntityLogger1, then the name of the log file will be "run1-EntityLogger1.log". A pre-existing file with this name will be overwritten once the simulation run is started.

Table 16-53 EntityLogger Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
DataSource	One or more sources of data to be logged. Each source is specified by an Expression. It is best to include only dimensionless quantities and non-numeric outputs in the DataSource input. An output with dimensions can be made non-dimensional by dividing it by 1 in the desired unit, e.g. `'[Queue1].AverageQueueTime / 1[h]'` is the average queue time in hours. A dimensional number will be displayed along with its unit. The `'format'` function can be used if a fixed number of decimal places is required.
IncludeInitialization	If TRUE, entries are logged during the initialization period.
StartTime	The time at which the log starts recording entries.
EndTime	The time at which the log stops recording entries.
Options	
Active	If TRUE, the object is used in the simulation run.
AttributeDefinitionList, CustomOutputList	See Section 6.2.

Keyword	Description
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-54 EntityLogger Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
Logger	
LogTime	The simulation time at which the last log entry was made.
EntityLogger	
obj	The entity that was received most recently.

16.27 Statistics

The Statistics object collects statistical information on the entities it receives.

The quantity to be tracked is specified using the SampleValue keyword, which accepts an Expression. Some example inputs are:

- `'this.obj.A'` - the sample value is the Attribute 'A' carried by the received entity
- `'this.SimTime - this.obj.t'` - the sample value is the simulation time that has elapsed since the Attribute 't' for the received entity was set to the simulation time at some earlier point in the model

The Statistics object can also aggregate the state statistics for the arriving entities using the RecordEntityStateTimes keyword.

Table 16-55 Statistics Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
UnitType	The unit type for the variable whose statistics will be collected.

Keyword	Description
SampleValue	The variable for which statistics will be collected.
HistogramBinWidth	Width of the histogram bins into which the recorded values are placed. Histogram data will not be generated if the input is left blank.
RecordEntityStateTimes	If `TRUE`, the state times for received entities are recorded for statistics generation.
ResetEntityStateTimes	If `TRUE`, the state times for received entities are set to zero on departure.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 16-56 Statistics Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.
Statistics	
SampleMinimum	The smallest value that was recorded.
SampleMaximum	The largest value that was recorded.
SampleAverage	The average of the values that were recorded.
SampleStandardDeviation	The standard deviation of the values that were recorded.
StandardDeviationOfTheMean	The estimated standard deviation of the sample mean.
TimeAverage	The average of the values recorded, weighted by the duration of each value.
TimeStandardDeviation	The standard deviation of the values recorded, weighted by the duration of each value.

17 Calculation Objects Palette

The Calculation Objects Palette contains objects for building continuous type models and mixed discrete-event/continuous models. As the name suggests, a continuous model changes state continuously as time advances. JaamSim is able to model this type of behaviour very efficiently using outputs, Attributes, and Expressions. The following objects are provided in the Calculation Objects palette.

Table 17-1 Calculation Objects Palette

	Object	Description
	WeightedSum	Calculates the weighted sum of the input values.
	Polynomial	Evaluates a polynomial function of the input value.
	Integrator	Integrates the input value over time.
	Differentiator	Differentiates the input value over time.
	PIDController	Proportional-Integral-Differential controller.
	Lag	Calculates the LAG operation for the input value.
	MovingAverage	Calculates a moving average of the input value over a specified range of time.
	SineWave	Generates a sinusoidal wave.
	SquareWave	Generates a square wave.
	UnitDelay	Delays the input value by one Controller time step.

Many calculation objects use the following inputs and outputs.

Table 17-2 Calculation Object Inputs

Keyword	Description
Controller	The Controller object that signals the updating of the calculation.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before one with a higher value.
UnitType	The unit type for the input value(s) to the calculation.
InputValue	The input value for the present calculation.

Table 17-3 Calculation Object Outputs

Output Name	Description
Value	The result of the calculation at the present time.

17.1 WeightedSum

The WeightedSum object adds two or more input values. A set of dimensionless constants can be provided to multiply each input value prior to addition.

$$y = C_1*x_1 + C_2*x_2 + \ldots + C_N*x_N, \text{ if the coefficients } C_i \text{ are specified}$$
$$= x_1 + x_2 + \ldots + x_N \quad , \text{ if the coefficients } C_i \text{ are not specified}$$

where:

y = present output value for the weighted sum
x_i = present value for the i^{th} input to the weighted sum
C_i = i^{th} entry in the **CoefficientList** input
N = number of inputs to the weighted sum

The value returned by the WeightedSum is calculated on demand.

Table 17-4 WeightedSum Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType	The unit type for the inputs to the weighted sum and for the value returned.
InputValueList	The list of inputs to the weighted sum. All inputs must have the same unit type.
CoefficientList	The list of dimensionless coefficients to be applied to the input values. If left blank, the input values are simply added without applying any coefficients.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-5 WeightedSum Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

Output Name	Description
WeightedSum	
Value	The calculated value for the weighted sum.

17.2 Polynomial

The Polynomial object evaluates a specified polynomial for the present input value:

$$y = C_0 + C_1*x + C_2*x^2 + \ldots + C_N*x^N$$

where:

y = present output value for the polynomial
x = present input to the polynomial
C_i = i^{th} entry in the **CoefficientList** input

The value returned by the Polynomial is calculated on demand.

Table 17-6 Polynomial Key Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
InputValue	The dimensionless input value to the polynomial
CoefficientList	The list of dimensionless coefficients for the polynomial function. The number of coefficients provided determines the number of terms in the polynomial. For example, inputs c0, c1, c2 specifies the second order polynomial P(x) = c0 + c1*x + c2*x^2.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-7 Polynomial Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

Output Name	Description
WeightedSum	
Value	The calculated value for the polynomial.

17.3 Integrator

The Integrator object integrates the input value with respect to time using the trapezoidal rule:

```
y = Y + (t - T)*0.5*(x + X)
```

where:

```
y = present output value for the integrator
x = present input to the integrator
t = present simulation time
Y = output value for the integrator at the last update time
X = input to the integrator at the last update time
T = simulation time at the last update
```

The value returned by the Integrator is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, and T used in the calculation.

Table 17-8 Integrator Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller	The Controller object that signals the updating of the calculation.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before one with a higher value.
UnitType	The unit type for the input value(s) to the calculation.
InputValue	The input value for the present calculation.
InitialValue	The initial value for the integral at time = 0.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-9 Integrator Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
DoubleCalculation	
Value	The result of the calculation at the present time.

17.4 Differentiator

The Differentiator object calculates the time derivative of the input value:

```
y = (x - X)/(t - T), for t > T
  = Y              , for t = T
```

where:

```
y = present output value for the differentiator
x = present input to the differentiator
t = present simulation time
Y = output value for the differentiator at the last update time
X = input to the differentiator at the last update time
T = simulation time at the last update
```

The value returned by the Differentiator is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, and T used in the calculation.

Table 17-10 Differentiator Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller	The Controller object that signals the updating of the calculation.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before one with a higher value.
UnitType	The unit type for the input value(s) to the calculation.
InputValue	The input value for the present calculation.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.

Keyword	Description
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-11 Differentiator Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
DoubleCalculation	
Value	The result of the calculation at the present time.

17.5 PIDController

The PIDController object simulates the Proportional-Integral-Differential (PID) type controller that is widely used in electronic control systems.

```
error = SetPoint - x
integral = INTEGRAL + (error - ERROR)*(t - T)
derivative = (error - ERROR)/(t - T)

y' = ProportionalGain/SetPointScale * (error + integral/IntegralTime +
                derivative*DerivativeTime)

y = min( max( y', OutputLow ), OutputHigh )
```

where:

```
y = present output value for the PID controller
x = present ProcessVariable input to the PID controller
t = present simulation time
Y = output value for the PID controller at the last update time
X = ProcessVariable input to the PID controller at the last update time
T = simulation time at the last update
ERROR = value for error at the last update time
INTEGRAL = value for integral at the last update time
```

The value returned by the PIDController is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, T, ERROR, and INTEGRAL used in the calculation.

Table 17-12 PIDController Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller	The Controller object that signals the updating of the calculation.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before one with a higher value.
UnitType	The unit type for the input value(s) to the calculation.
SetPoint	The set point for the PIDController. The unit type for the set point is given by the UnitType keyword.
ProcessVariable	The process variable feedback to the PIDController. The unit type for the process variable is given by the UnitType keyword..
ProcessVariableScale	A constant with the same unit type as the process variable and the set point. The difference between the process variable and the set point is divided by this quantity to make a dimensionless variable.
OutputUnitType	The unit type for the output from the PID controller.
ProportionalGain	The coefficient applied to the proportional feedback loop. The unit type for the proportional gain is given by the OutputUnitType keyword.
IntegralTime	The time scale applied to the integral feedback loop.
DerivativeTime	The time scale applied to the differential feedback loop.
OutputLow	The lower limit for the output signal.
OutputHigh	The upper limit for the output signal.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-13 PIDController Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
DoubleCalculation	
Value	The result of the calculation at the present time.

Output Name	Description
PIDController	
Error	The difference between the set point and the process variable values divided by the process variable scale.
Integral	The integral of the dimensionless error value.
Derivative	The derivative of the dimensionless error value.
ProportionalValue	The proportional component of the output value.
IntegralValue	The integral component of the output value.
DerivativeValue	The derivative component of the output value.

17.6 Lag

The Lag object calculates the lag operation used in electronic control systems.

```
y = Y + (t - T)*(x - Y)/LagTime
```

where:

```
y = present output value for the Lag object
x = present input to the Lag object
t = present simulation time
Y = output value for the Lag object at the last update time
X = input to the Lag object at the last update time
T = simulation time at the last update
```

The value returned by the Lag object is calculated on demand. The update signal received from the Controller is used only to record the values Y, X, and T used in the calculation.

Table 17-14 Lag Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller	The Controller object that signals the updating of the calculation.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before one with a higher value.
UnitType	The unit type for the input value(s) to the calculation.
InputValue	The input value for the present calculation.
LagTime	A value with units of time used in the lag calculations.

Keyword	Description
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-15 Lag Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
DoubleCalculation	
Value	The result of the calculation at the present time.
Lag	
Error	The value for (InputValue - Value).

17.7 MovingAverage

The MovingAverage object calculates the average value of the input over a given time interval.

$$y = \{ x + \sum_{(i = M \text{ to } M-N-2)} X(i) \} / N$$

where:

```
y = present output value for the MovingAverage object
x = present input to the MovingAverage object
N = NumberOfSamples
M = number of updates that have been performed previously
X(i) = input to the MovingAverage object at the ith update time
```

The value returned by the MovingAverage object is calculated on demand. The update signal received from the Controller is used only to record the values `X(i)` used in the calculation.

Table 17-16 MovingAverage Key Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller	The Controller object that signals the updating of the calculation.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before one with a higher value.
UnitType	The unit type for the input value(s) to the calculation.
InputValue	The input value for the present calculation.
NumberOfSamples	The number of input values over which to average.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-17 MovingAverage Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
DoubleCalculation	
Value	The result of the calculation at the present time.

17.8 SineWave

The SineWave object generates a sine wave output:

$$y = Offset + Amplitude * \sin(2\pi\ t/Period + PhaseAngle)$$

where:

```
y = present output value for the SineWave object
t = present simulation time
```

The value returned by the SineWave is calculated on demand.

Table 17-18 SineWave Key Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType	The unit type for the value returned by the wave.
Amplitude	Amplitude of the generated wave.
Period	Period of the generated wave.
PhaseAngle	Initial phase angle of the generated wave.
Offset	Offset added to the output of the generated wave.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-19 SineWave Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
WaveGenerator	
Value	The present value for the wave.

17.9 SquareWave

The SquareWave object generates a square wave output:

```
y = Offset + Amplitude, if sin(2π t/Period + PhaseAngle) >= 0
  = Offset – Amplitude, if sin(2π t/Period + PhaseAngle) < 0
```

where:

```
y = present output value for the SquareWave object
t = present simulation time
```

The value returned by the SquareWave is calculated on demand.

Table 17-20 SquareWave Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
UnitType	The unit type for the value returned by the wave.
Amplitude	Amplitude of the generated wave.
Period	Period of the generated wave.
PhaseAngle	Initial phase angle of the generated wave.
Offset	Offset added to the output of the generated wave.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-21 SquareWave Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
WaveGenerator	
Value	The present value for the wave.

17.10 UnitDelay

The UnitDelay object holds the value from the last update time. It is used to prevent an infinite loop from occurring when a modelled calculation includes a feedback loop.

```
y = X
```

where:

```
y = present output value for the UnitDelay object
X = input to the UnitDelay at the last update time
```

Table 17-22 UnitDelay Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller	The Controller object that signals the updating of the calculation.
SequenceNumber	The sequence number used by the Controller to determine the order in which calculations are performed. A calculation with a lower value is executed before one with a higher value.
UnitType	The unit type for the input value(s) to the calculation.
InputValue	The input value for the present calculation.
InitialValue	The value for the UnitDelay function at simulation time equal to zero.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 17-23 UnitDelay Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
UnitDelay	
Value	The result of the calculation at the present time.

18 Fluid Objects Palette

The Fluid Objects Palette contains objects for building dynamic models of hydraulic flow involving tanks, pipes, pumps, etc. The flow rate in the system is calculated by solving the unsteady Bernoulli equation.

The following objects are provided in the Fluid Objects palette.

Table 18-1 Fluid Objects Palette

Object		Description
	Fluid	Defines a specific fluid and its properties.
	FluidFlow	Computed flow of a specified fluid between a source and a destination.
	FluidFixedFlow	Constant flow to/from a specified tank.
	FluidTank	Cylindrical vessel for storing fluid.
	FluidPipe	Cylindrical conduit for transporting fluid.
	FluidCentrifugalPump	Type of pump with a rotating impeller.

Many fluid objects use the following inputs and outputs.

Table 18-2 Fluid Object Inputs

Keyword	Description
Previous	The upstream component that feeds this component.
Diameter	The hydraulic diameter of the component. Equal to the inside diameter of a pipe with a circular cross-section.

Table 18-3 Fluid Object Outputs

Output Name	Description
FlowArea	The cross-sectional area of the component.
Velocity	The velocity of the fluid within the component.
ReynoldsNumber	The Reynolds Number for the fluid within the component. Equal to (velocity)(diameter)/(kinematic viscosity).
DynamicPressure	The dynamic pressure of the fluid flow. Equal to (0.5)(density)(velocity^2).
InletPressure	The static pressure at the component's inlet.
OutletPressure	The static pressure at the component's outlet.

18.1 Fluid

The Fluid object defines the basic properties of the fluid.

Table 18-4 Fluid Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Density	The density of the fluid (default = water).
Viscosity	The dynamic viscosity of the fluid (default = water).
Colour	The colour used to represent the fluid.
Gravity	The acceleration of gravity to be used in the fluid flow calculations.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 18-5 Fluid Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

18.2 FluidFlow

The FluidFlow object calculates the flow rate between a source and a destination by solving the unsteady Bernoulli equation.

Table 18-6 FluidFlow Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller, SequenceNumber	Keywords for Calculation Objects (see Table 17-2).
Fluid	The Fluid being moved by the flow.
Source	The source object for the flow.
Destination	The destination object for the flow.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 18-7 FluidFlow Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
FluidFlowCalculation	
FlowRate	The volumetric flow rate for the system.
FluidFlow	
FlowAcceleration	The time derivative of the volumetric flow rate.
FlowInertia	The sum of (density)(length)/(flow area) for the hydraulic components in the route.

18.3 FluidFixedFlow

The FluidFixedFlow object moves Fluid between a source and a destination at a fixed volumetric flow rate.

Table 18-8 FluidFixedFlow Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Controller, SequenceNumber	Keywords for Calculation Objects (see Table 17-2).
Fluid	The Fluid being moved by the flow.
Source	The source object for the flow.
Destination	The destination object for the flow.
FlowRate	The constant volumetric flow rate from the source to the destination.
Width	The width of the pipe segments in pixels.
Colour	The colour of the pipe.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Points, CurveType, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 18-9 FluidFixedFlow Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Attributes, Custom Outputs, Position, Size, Orientation, Alignment	See Section 10.1.
FluidFlowCalculation	
FlowRate	The volumetric flow rate for the system.

18.4 FluidTank

The FluidTank object represents a cylindrical storage tank containing Fluid. FluidTanks are modelled as large diameter FluidPipes so that the velocity and inertia of the Fluid are preserved in calculations.

Table 18-10 FluidTank Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Previous, Diameter	Keywords for Fluid Objects (see Table 18-2).

Keyword	Description
Capacity	The total volume of fluid that can be stored in the tank.
InitialVolume	The volume of fluid in the tank at the start of the simulation.
AmbientPressure	The atmospheric pressure acting on the surface of the fluid in the tank.
InletHeight	The height of the flow feeding the tank. Measured relative to the bottom of the tank.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 18-11 FluidTank Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
FluidComponent	
FlowArea, Velocity, ReynoldsNumber, DynamicPressure, InletPressure, OutletPressure	Outputs inherited from Fluid Objects (see Table 18-5).
FluidTank	
FluidVolume	The volume of the fluid stored in the tank.
FluidLevel	The height of the fluid from the bottom of the tank.

18.5 FluidPipe

The FluidPipe object represents a cylindrical pipe for transporting Fluid.

Table 18-12 FluidPipe Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Previous, Diameter	Keywords for Fluid Objects (see Table 18-2).

Keyword	Description
Length	The length of the pipe.
HeightChange	The height change over the length of the pipe. Equal to (outlet height - inlet height).
Roughness	The roughness height of the inside pipe surface. Used to calculate the Darcy friction factor for the pipe.
PressureLossCoefficient	The pressure loss coefficient or 'K-factor' for the pipe. The factor multiplies the dynamic pressure and is applied as a loss at the pipe outlet.
Width	The width of the pipe segments in pixels.
Colour	The colour of the pipe.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Points, CurveType, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 18-13 FluidPipe Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Attributes, Custom Outputs, Position, Size, Orientation, Alignment	See Section 10.1.
FluidComponent	
FlowArea, Velocity, ReynoldsNumber, DynamicPressure, InletPressure, OutletPressure	Outputs inherited from Fluid Objects (see Table 18-5).
FluidPipe	
DarcyFrictionFactor	The Darcy Friction Factor for the pipe.

18.6 FluidCentrifugalPump

The FluidCentrifugalPump object models the performance of a centrifugal pump.

Table 18-14 FluidCentrifugalPump Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
Previous, Diameter	Keywords for Fluid Objects (see Table 18-2).

Keyword	Description
MaxFlowRate	Maximum volumetric flow rate that the pump can generate.
MaxPressure	Maximum static pressure that the pump can generate (at zero flow rate).
MaxPressureLoss	Maximum static pressure loss for the pump (at maximum flow rate).
SpeedController	The CalculationEntity whose output sets the rotational speed of the pump. The output value is ratio of present speed to maximum speed (0.0 - 1.0).
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 18-15 FluidCentrifugalPump Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
FluidComponent	
FlowArea, Velocity, ReynoldsNumber, DynamicPressure, InletPressure, OutletPressure	Outputs inherited from Fluid Objects (see Table 18-5).

19 SubModel Objects Palette

Sub-models provide a way to group the objects in a model into functional units and to create a hierarchical structure.

Table 19-1 SubModel Objects Palette

	Object	Description
	SubModel	A collection of objects that can be used in multiple places within a model.
	SubModelStart	The first component of a sub-model.
	SubModelEnd	The last component of a sub-model.

19.1 SubModel

The SubModel object is the basis for the implementation of sub-models in JaamSim. The components of SubModel are displayed in a Region that is created as part of the SubModel object. A component is added to the SubModel by dragging and dropping it on top of its Region or by copying and pasting an existing object on top of the Region.

A component has both a local name with the context of its SubModel as well as a global name. For example, a component with the local name `Server1` within `SubModelA` has a global name of `SubModelA.Server1`. The SubModel that contains a component is called the parent SubModel of the component.

A SubModel can contain another SubModel as a component. For example, if `Server1` is component of `SubModelA` which in turn is a component of `SubModelB`, then the global name for Server1 is `SubModelB.SubModelA.Server1`.

A SubModel's Region can be displayed or hidden temporarily using the 'Show SubModels' button  , or it can be displayed or hidden permanently using the 'Show Components' item in its context menu (right click).

A SubModel can receive entities as part of a process flow type model or it can operate independently. A SubModel uses its SubModelStart and SubModelEnd components specify the entry and exit points. An entity arriving to a SubModel is directed automatically to its SubModelStart component. An entity arriving to a SubModelEnd component is directed automatically back to the parent SubModel and from there to the object specified by its NextComponent input. An entity can be sent directly to any other object by providing an input to the NextComponent input to the SubModelEnd object. A SubModel that receives entities must have a single SubModelStart object as a component, but can have multiple SubModelEnd components.

A SubModel object can be copied in the same way as other objects. The SubModel's components and their inputs are copied along with the SubModel. Inputs to SubModel components that reference the parent SubModel or another component of the SubModel are changed automatically to refer to

the copied SubModel or SubModel component. RandomSeed inputs to probability distributions are changed to the first unused integer value.

The keyword `'sub'` in an expression refers to the parent SubModel. It is equivalent to entering `'this.parent'`.

Table 19-2 SubModel Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object to which the processed entity is passed.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 19-3 SubModel Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.
StateEntity	
State, WorkingState, WorkingTime, StateTimes, TotalTime	See Section 0.
LinkedComponent	
obj, NumberAdded, NumberProcessed, NumberInProgress, ProcessingRate, ReleaseTime	See Section 16.

19.2 SubModelStart

The SubModelStart object is used as the entry point for the components of a SubModel. An entity arriving to a SubModel is directed automatically to its SubModelStart component.

Table 19-4 SubModelStart Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next component in the sub-model.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 19-5 SubModelStart Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

19.3 SubModelEnd

The SubModelEnd object is used as the exit point for the components of a SubModel. An entity arriving to a SubModelEnd component is directed automatically back to the parent SubModel and from there to the object specified by its NextComponent input. An entity can be sent directly to any other object by providing an input to the NextComponent input to the SubModelEnd object.

Table 19-6 SubModelEnd Inputs

Keyword	Description
Key Inputs	
Description	A free-form string describing the object.
NextComponent	The next object, external to the sub-model, to which the processed entity is passed. If left blank, the entity is returned to the parent sub-model which directs it to the object specified by its NextComponent input.
Options	
AttributeDefinitionList, CustomOutputList	See Section 6.2.

Keyword	Description
Graphics	
Position, Alignment, Size, Orientation, Region, RelativeEntity, DisplayModel, Show, Movable, VisibleViews, DrawRange	See Section 11.1.

Table 19-7 SubModelEnd Outputs

Output Name	Description
Entity and DisplayEntity	
Name, ObjectType, SimTime, Parent, Attributes, Custom Outputs, Position, Size, Orientation, Alignment, GraphicalLength, ObserverList, NextList, PreviousList	See Section 10.1.

20 Pre-Built SubModels

A pre-built SubModel is one that has already been populated with components and can be dragged and dropped directly into a model.

20.1 ServerAndQueue

The ServerAndQueue object provides an example of how a sub-model can be programmed in JaamSim. It consists of a Server, its Queue, and an ExpressionThreshold that closes when a maximum queue length is reached.

ServerAndQueue includes the following components:

- SubModelStart
- Server
- Queue
- Threshold
- SubModelEnd

Inputs to the ServerAndQueue can be made to both the ServerAndQueue object itself or to it components. However, pre-programmed inputs, such the NextComponent keyword for Server, cannot be modified by the user. These input values are shown in grey in the Input Editor.