extendsim

DISCRETE EVENT

QUICK START GUIDE

License, Software Copyright, Trademark, and Other Information

The software described in this manual is furnished under a separate license and warranty agreement. The software may be used or copied only in accordance with the terms of that agreement. Please note the following:

> ExtendSim blocks and components (including but not limited to icons, dialogs, and block code) are copyright © by Imagine That Inc. and/or its Licensors. ExtendSim blocks and components contain proprietary and/or trademark information. If you build blocks, and you use all or any portion of the blocks from the ExtendSim libraries in your blocks, or you include those ExtendSim blocks (or any of the code from those blocks) in your libraries, your right to sell, give away, or otherwise distribute your blocks and libraries is limited. In that case, you may only sell, give, or distribute such a block or library if the recipient has a valid license for the ExtendSim product from which you have derived your block(s) or block code. For more information, contact Imagine That Inc.

Acknowledgments

Extend was created in 1987 by Bob Diamond; it was re-branded as ExtendSim in 2007.

Chief architects for ExtendSim 10: Steve Lamperti and Bob Diamond

Simulation Engineers: Anthony Nastasi, Cecile Pieper, Peter Tag, and Dave Krahl

Graphics, documentation, and editing: Kathi Hansen, Carla Sackett, and Pat Diamond

Imagine That Inc • 6830 Via Del Oro, Suite 230 • San Jose, CA 95119 USA
408.365.0305 • fax 408.629.1251 • info@extendsim.com
www.ExtendSim.com

Table of Contents

Discrete Event Quick Start

Introduction

Welcome!

Thank you for using ExtendSim, the power tool for simulation modeling! We hope you enjoy using ExtendSim and that you find this Quick Start Guide helpful.

About the Quick Start Guides

The purpose of a Quick Start Guide is to get new users quickly familiar with a specific ExtendSim module and aware of the ExtendSim features and capabilities. There are three Quick Start Guides—Continuous Process Modeling, Discrete Event Simulation, and Discrete Rate Modeling. Depending on the product purchased, one or more of these will be installed as eBooks in the Documents/ExtendSim/Documentation folder.

☞ Each of the Quick Start Guides has similar structure and information, so it is not necessary to read all of them.

For more in depth information, the guides will refer you to the User Reference, the Technical Reference, or the Tutorial & Reference for the ExtendSim Database or Advanced Resource Management, all of which are installed in the Documents/ExtendSim/Documentation folder.

Who should read this Quick Start Guide

This guide is for model developers who build *discrete event models*. These models simulate systems in which it is important to individually identify and monitor system entities, and where state variables, which describe the system at any point in time, change only when an event occurs.

Chapters in this Quick Start Guide

1) Introduction to Discrete Event Simulation (DES):
    - What DES is
    - Where and when to use DES
    - How it is being used
    - How DES compares to continuous process and discrete rate simulation
    - Where to get more information
2) Model basics and how to run a model
3) Tutorial 1: building a simple discrete event model
4) Tutorial II: expanding the model
5) Concepts and terminology specific to discrete event modeling

What is Discrete Event simulation?

Discrete event simulation (DES) is the methodology used to simulate the behavior of a system or process in which events, rather than the mere passing of time, cause changes to system components and hence to the system. In a DES model, nothing happens between points in time until an event occurs. At

each event, the status of system components (such as entities and resources) is recalculated and the state of the system changes.

DES models involve queues, activities, routing, and the monitoring of the movement, position, and properties of uniquely identifiable system components called items. These models have several things in common:

- They involve a combination of system entities such as customers, procedures, materials, equipment, information, space, and energy (called *items* in ExtendSim) together with system *resources* such as equipment, tools, and personnel.
- Each process is a series of logically related *activities* undertaken to achieve a specified outcome, typically either a product or a service. Activities have a duration and usually involve the use of process elements and resources.
- Processes are organized around *events*, such as the receipt of parts, a request for service, or the ringing of a telephone. Events occur at random but somewhat predictable intervals and can be economic or noneconomic.

Why simulate using DES?

Discrete event simulation is indispensable for understanding, analyzing, and predicting the behavior of complex and large-scale systems. It is used to insure that the functioning of existing systems is well understood and that the design of new systems is efficient and beneficial.

DES models allow companies and governmental agencies to look at their fundamental processes from a cross-functional perspective and ask “Why?” and “What if?”

What is modeled and where is DES used?

DES is an integral part of Lean, Six Sigma, business reengineering, risk analysis, capacity planning, throughput analysis, and reliability engineering projects. Discrete event models are also useful for examining the effects of variations such as labor shortages, equipment additions, and transmission breakdowns.

The following table gives some of the most common areas where discrete event modeling is used.

Discipline	Fields	Applications
Manufacturing	Aerospace, Biotech, Agriculture, Semiconductor, Food and Beverage, Automotive, Pharmaceutical, Consumer products	Inventory and resource management, Six Sigma/Lean initiatives, scheduling, capacity planning, evaluation of procedures.
Service Industries	Retail, Banking, Health Care, Finance, Restaurants, Hotels, Insurance, Utilities	Service levels, scheduling, throughput analysis, personnel management, evaluation of procedures, Six Sigma/Lean initiatives, workflow.
Communications/Networks	Call Centers, Satellite Systems, Airborne and Ground Communication Systems	Capacity planning, performance evaluation, throughput analysis, determination of reliability and fault tolerance.

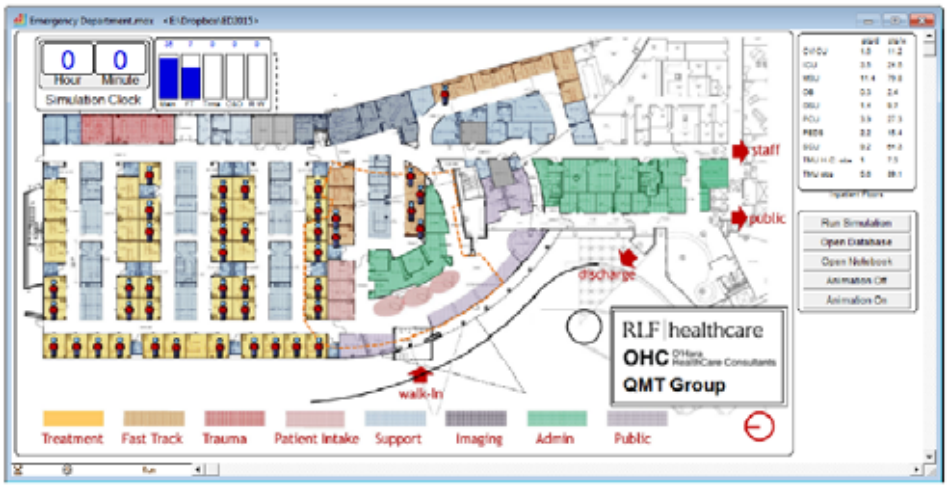
Discipline	Fields	Applications
Transportation/Material Handling	Airlines, Railroads, Freight and Mail, Moving and Cargo, Warehousing, Logistics, Defense	Emergency planning, scheduling, service level, product mix, Six Sigma/Lean initiatives.

How ExtendSim is being used for DES

ExtendSim pioneered the combination of a graphical user interface integrated with a simulation language, which allowed discrete event simulation to become accessible and wide-spread. Some examples where ExtendSim is being used for DES include:

- The Public Lands Planning and Management group at RSG models visitor use at national parks and forests, providing a basis for standards that frame acceptable conditions. The goal is to optimize park operations and maximize visitor experiences while minimizing search-and-rescue incidents in wilderness areas such as the Half Dome Trail.
- The Surgeon General of the US Army sponsored research by the RAND Corporation to develop and evaluate alternative strategies for equipping the Army's combat support hospitals (CSH's) to meet needs in all phases of the deployment and redeployment cycle. The focus was on developing an equipping/maintenance strategy that would result in fewer, but regularly modernized, full hospital sets system-wide, reducing maintenance and upgrade costs.
- In conjunction with RLF Architects and the QMT Group, O'Hara Healthcare Consultants used ExtendSim DES to redesign a community hospital's emergency department—architecturally, operationally, and clinically. In addition to other factors, the model included daily patient volume, arrival profiles, emergency severity index (ESI) probabilities, and diagnostic probabilities by ESI, as well as service times (by ESI, treatment area, and disposition). The

insight gained helped staff guide the design of the new department and the model provides analytics for evaluating future impacts.



- Grayrock & Associates LLC helped executives at an injection modeling plant determine if more equipment and an expanded space would be needed to meet increased production demands.The model substantiated that throughput could be increased by 25% without plant expansion or capital expenditures.

•The Department of Homeland Security (DHS) commissioned the MITRE Corporation to create an agent-based DES model of airport defense and security systems. Attacker and defense behavior are modeled using intelligent agents with decision making and learning capabilities. The model simulates the performance of the airport defense against threat vectors (path-weapon combinations) so security can continuously and rapidly adapt to shifting threats.

- Johnson & Johnson commissioned the OpStat Group to create a simulation model which would examine current operations and develop solutions for the future for one of its large pharmaceutical plants. The model allowed the company to pre-test process changes and assist in the selection of techniques as part of a Lean Six Sigma program. The scope evolved from improvement and capacity projects to becoming integral to the ongoing planning and management process. Ultimately significant benefits ensued—better utilization of personnel, accelerated delivery of value-added projects, and improved accuracy and timeliness of planning information—allowing plant and supply chain management to evaluate options for sourcing and capacity within the plant and across the supply chain.

- Dow Chemical Company performs reliability modeling to identify and understand the impact of different failures on overall production capabilities in chemical plants. The model is used for understanding the key equipment components that contribute towards maximum production loss and for analyzing the impact of change policies, such as the installation of new equipment or an increased stock level for failure-prone components. A Failure Summary Report provides information for further phases of the analysis.

Record #	ArrivalTime	FailID	TTR	TBF	Reorder Time
1	12/12/2007 1:21:23	4	34.07	29.55	200.29
2	4/13/2008 7:57:22	29	28.98	118.33	205.82
3	8/12/2010 23:15:08	6	19.31	737.12	192.85
4	12/18/2011 13:29:02	30	32.96	1094.37	196.47
5	3/20/2012 7:36:14	3	57.58	1160.08	203.45
6	4/8/2012 9:14:08	21	22.70	1173.26	197.78
7	2/16/2013 1:37:23	21	19.76	227.53	207.67
8	7/17/2014 6:13:39	8	20.32	1775.59	199.21
9	7/11/2015 23:51:22	29	29.20	1918.00	209.94
10	3/27/2016 20:58:43	19	89.80	2222.62	198.07
11	6/16/2016 18:45:39	10	26.06	2280.67	201.06
12	4/3/2017 2:59:44	22	39.39	2490.63	208.78

Figure 3: Failure summary report

•Northrop Grumman Mission Systems developed tools for the simulation and visualization of National Polar-Orbiting Observation Environmental Satellite System (NPOESS) weather satellite data collection and processing systems. The model simulates orbiting weather satellites, data retrieval through global receptors, and weather product generation and distribution. The models are used to size the architecture and evaluate cost/performance trade offs.

- A consortium of the DOE, National Labs, and other institutions created a discrete event biomass supply chain model that includes the logistical features of the supply (number of farms involved, average yield, harvest schedule, moisture content of the crop, and so forth). The model takes input for daily weather data (temperature, relative humidity, wind speed, etc.) and user-defined parameters for safe moisture content for bailing. The model also calculates costs per ton of biomass, energy input, and CO2 emissions from farm equipment.
- Hormel Foods Corporation had issues with its product flow due to over-utilization of inventory locations. The company modeled the different systems that were supplying and demanding material from locations throughout the country, verifying adequate inventory turns, minimizing equipment and material handing costs, and meeting capacity requirements. optimizing the product mix and the number of supply items at various storage locations.

How DES compares to other modeling methodologies

The most important aspects of the three main modeling methodologies—continuous, discrete event, and discrete rate—are shown below.

	System State Changes When	Models	Attributes for Tracking	Queued Order Method
Continuous Process	Time changes	Generic stuff	No	N/A
Discrete event	Event occurs	Identifiable things	Yes	FIFO, LIFO, Priority, or custom order based on Item Attributes
Discrete rate	Event occurs	Generic or aggregated stuff	Yes	FIFO, LIFO, or custom order based on Flow Level Attributes

- State change. Models are either event-based or time-based. Discrete event models only change state when an event occurs; the same is true for discrete rate models. Continuous process models have a fixed time step where time advances in equal increments and the model changes state at each time step.
- What is modeled. Discrete event models monitor the status of individually identifiable "things" or "entities"—called items in ExtendSim—while discrete rate and continuous models both model homogeneous or heterogeneous "stuff".
- Attributes. The entities in discrete event models can be assigned attributes—properties that allow the entity to be individually identified and manipulated; this is also true for the quantities of flow in discrete rate models. However, values in continuous models do not have attributes.
- Order method. The order in which the flow in discrete rate models or the things in discrete event models leave a queue which is holding more than one quantity or thing. The stuff (values) in continuous models cannot be put into any order.

☞ For more information about how discrete event modeling differs from continuous and discrete rate modeling, see the User Reference chapter on Modeling Methodologies.

Where to get more information

The ExtendSim documentation, example models, and the video files and documents on the ExtendSim.com website provide comprehensive help.

Tutorial & Reference documents

In addition to the Quick Start Guides, there are three Tutorial & Reference documents that are included as eBooks in the Documents/ExtendSim/Documentation folder:

- *ExtendSim Database*. This internal relational database provides model developers with a systematic way to manage information for the model and makes models scalable.
- *Advanced Resource Management*. ARM is a sophisticated architecture for systematically dealing with multiple types of resources and a model's complex resource requirements.

- *Reliability*. Graphically capture and validate complex availability behavior. Determine when scheduled and unscheduled downs occur for individual resources and what impact that has on the availability of the system as a whole.

User Reference

The ExtendSim User Reference has a lot of information you will find helpful when building, using, and presenting models.

How To chapters cover general modeling and simulation topics:

- Using libraries and blocks
- Performing analysis
- Enhancing presentations
- Creating a user interface
- Using equation-based blocks
- And much more

Discrete Event chapters in the User Reference discuss specific topics:

- Items, properties, and values
- Queueing
- Routing items
- Processing, batching, and unbatching
- Resources and shifts

Appendices list Menu Commands and the ExtendSim libraries and blocks

Every menu command is explained; the main libraries are described block by block.

Technical Reference

You probably won't build your own discrete event block, but it's very common to use an Equation block (Value library) in a model. The Technical Reference lists over 1,000 functions and has information about using include files and other programming tools.

☞ The eBooks ship with the appropriate ExtendSim product. To access these documents, see the Documents/ExtendSim/Documentation/folder or launch the books from the Getting Started model that opens when ExtendSim launches. The User Reference and Technical Reference are also available if you select the Help menu when using ExtendSim.

Example models and videos show you how

ExtendSim includes numerous tutorial models as well as videos and example models that explain concepts discussed in the User Reference. For example models, see the Documents/ ExtendSim/Examples/ Discrete Event folder.

Discrete Event Quick Start

Model Basics

Overview

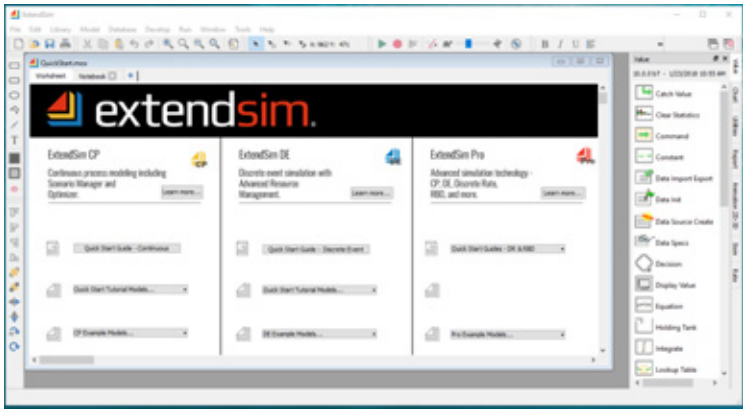
This chapter shows how to open and run a model of a simple discrete event system and discusses the components of a model.

If you already know how to open, run, and save an ExtendSim model and are familiar with blocks, connectors, and model layout, skip this chapter and go to the next one.

Opening and exploring a model

▸ Launch ExtendSim

As seen on the right, when ExtendSim launches it opens an application window with:



- A model (on the left)
- Library windows (stacked on the right) for every open library
- Toolbars on the top and sides

By default, the model that opens will be the Getting Started model. If that option has been removed from the Edit > Options > Model tab, a new blank model will open instead.

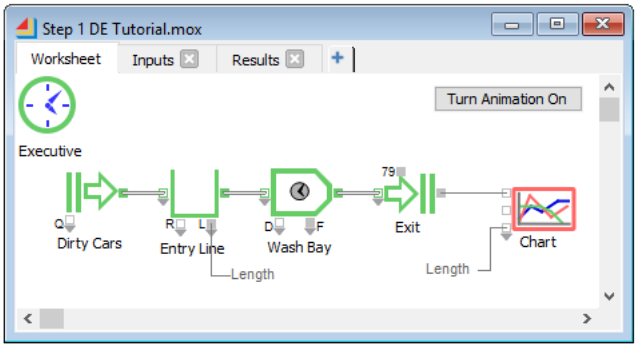
- The Getting Started model has information about ExtendSim products as well as links to the manuals, video tutorials, and more.
- If instead a new model opens, in most situations it will open with an Executive block in the upper left corner. This is discussed in “Launch ExtendSim” on page 21.

This tutorial will focus on the model window. Other aspects of the application window are discussed later in this Guide or in the User Reference.

Open the example model

- Select File > Open
- Browse to Documents/ExtendSim/Examples/**Tutorials**/Discrete Event
- Select the model named *Step 1 DE Tutorial*
- Click Open

This opens the *model window* as shown on the right.



About the model

This is a simple example of a car wash. The model uses six blocks (labeled Executive, Dirty Cars, Entry Line, Wash Bay, Exit, and Chart) to simulate a simple system where cars enter a line to go through a wash system.

The model window

Notice that there are tabs on the top of the model window:

- The *Worksheet* is where the model resides.
- The other tabs are notebooks that provide s a central place for storing inputs, outputs, and so forth; notebooks are discussed on page 11.

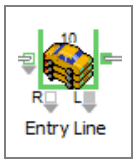
Run the example model

- Select Run > Run Simulation or click the Run Simulation button in the tool bar.

As the simulation runs, progress information will be displayed in the status bar at the bottom left of the application window. (With a small model like this, and with animation turned off, the messages may go by too quickly to be read.) Since it has been set to do it, at the end of the simulation run the Line Chart block opens and displays the graph.

Run with animation

ExtendSim provides built-in and customizable 2D animation on block icons and on the worksheet. To see 2D animation in this model:



- Select Show 2D Animation in the Run menu, select the Show 2D Animation tool, or click the Turn Animation On button on the model worksheet.
- Run the simulation again

With animation on, the Queue and Activity blocks (labeled Entry Line and Wash Bay) respectively show the buildup of cars waiting to be washed and the status (busy or idle) of the wash bay.

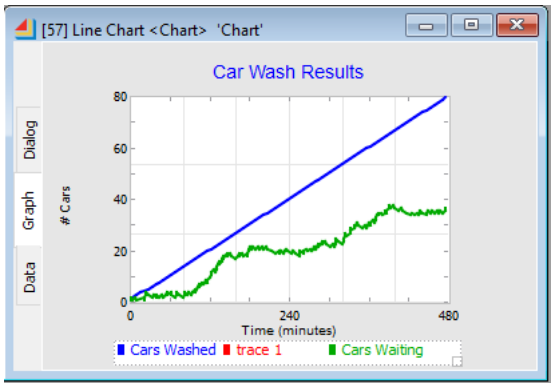
See results

In addition to what animation shows, there are many ways to see information during and at the end of the simulation run. For this model, information is reported on graphs and data tables, in a notebook, in block dialogs, and (as seen above) through animation.

On graphs and in data tables

ExtendSim comes with a number of flexible graphs and charts to use in models; this model uses a Line Chart block.

The example model runs for a simulated 480 minutes. At the end of the simulation, ExtendSim opens the Line Chart block and displays the history of the values over time on the block's Graph tab.



The Line Chart block in this model has information entering two of its input connectors, so the Graph tab displays two lines.

In this case, the blue line shows the number of cars that have been processed and the green line shows the status of the waiting line, or queue.

Time	Cars Washed
6	1
12	2
18	3
24	4

The Line Chart's Data tab shows the data points which produce the line, as seen on the left.

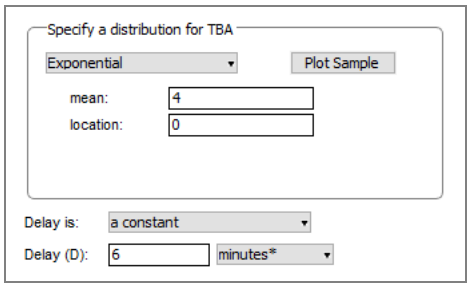
The Line Chart's Dialog tab has options for when to show the graph, when to autoscale, and so forth.

☞ To access windows for customizing how the graph is displayed and how the lines are drawn on it, right click on the Graph tab. To tear off one of the Line Chart's tabs so it is in a separate window, click and drag the tab by its name. Click a torn-off tab's close box to return it to the Line Chart.

In Notebooks

Notebooks are customizable tear-off windows that can be used as a "dashboard" for the model - to control model parameters, report simulation results, and document the model.

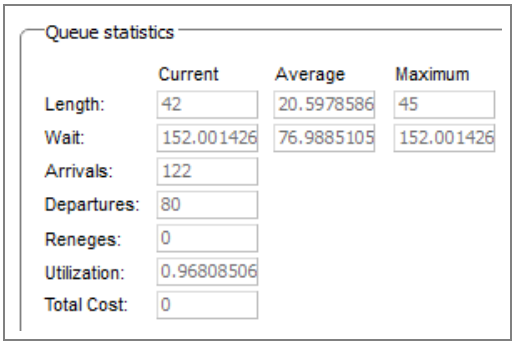
In this model, some important parameters and tables from the dialogs of blocks in the yogurt model have been *cloned* (duplicated) onto two notebooks, one for inputs and one for outputs.



☞ Cloning dialog objects is discussed in the How To: Custom Interfaces section of the User Reference.

From the model's blocks

Information can often be found on block icons, in block dialogs, and at block connectors. For example, when the icon of the Activity block (labeled Wash Bay) is double-clicked, its dialog opens and the Results tab gives a lot of information about wait times, utilization, blocking, and so forth for the cars that have been washed. And if 2D animation is on when the model is run, the icon of the Queue block (labeled Entry Line) indicates that approximately 56 cars are waiting to be washed.



Using other blocks

This model is so simple that it doesn't need further analysis. However, for larger and more complex models, the blocks in the Report library gather information as the model runs and display and calculate statistics on the results.

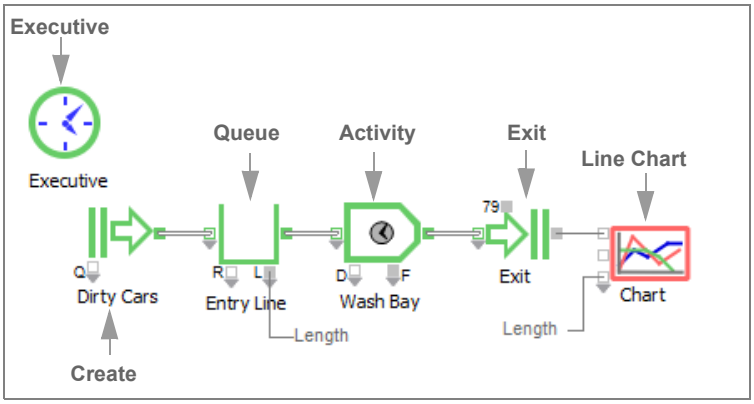
Model basics

As summarized in "What is Discrete Event simulation?" on page 1 and explained more in "Items and informational values" on page 38, discrete event models simulate the movement, position, and properties of uniquely identifiable system components, called *items*, which change state during the simulation run.

In the simplest terms, ExtendSim models are composed of *blocks* and *connections*. This model, for example, has six blocks, as seen on the model worksheet below. As the model runs, items go into a block, are processed and/or modified, and are then sent on to the next block, usually via a connection. Except for some hierarchical blocks, blocks are stored in repositories called *libraries*.

Blocks

Each block in ExtendSim represents a portion of the process or system that is being modeled.



- Blocks have *names*, such as Queue or Activity, that signify the function they perform. An Activity block, for example, will have the same functional behavior in every model you build.
- You can also add a *label* to a block to indicate what it represents in your specific model, such as an Activity block labeled Wash Bay.
- Blocks come from libraries, which are discussed in "Libraries" on page 16.

Icons

A block's icon is usually a pictorial representation of its function. For instance, in this model the block labeled Entry Line is a Queue block. Its icon symbolizes a holding tank that can have quantities added or removed from it.

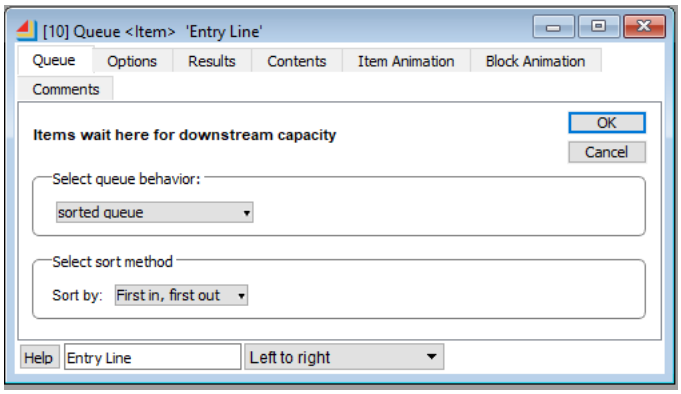
The small circles and squares attached to icons are *connectors*, which are discussed in more detail in "Connectors and connections" on page 14.

☞ Place your cursor over a block's icon on a model worksheet to see a tooltip with the block's object ID, name, and library. If *Include additional block information* is checked in the Edit > Options > Model tab, the tooltip will also include a short description of the block's function.

Dialogs

Most blocks have a dialog associated with them. Dialogs are used to enter values and settings before running simulations and to see results during and after the run.

To access a block's dialog, double-click its icon or right-click the icon and select Open Dialog. For example, if you double-click the Queue icon, the dialog at right opens.



At the top is the block's global object ID, its name, the name of the library it resides in (inside the angle brackets), and its label (if present). Global object IDs are unique identifiers assigned sequentially to blocks and other objects as they are placed in a model.

☞ When there are multiple models and several open dialogs, it can be difficult to remember which block comes from which model. To see which model a block is in, hover over the block's name in the dialog's title bar.

Dialog tabs

Block dialogs may have tabs on top and/or on the left side.

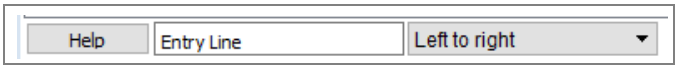
- Most dialogs have tabs across the top for entering values and settings, getting results, controlling animation, and so forth. As shown above for the Queue block, the tabs are Queue, Options, Results, Contents, Item Animation, Block Animation, and Comments.
- If present, tabs on the left side are for switching between windows. For example, the Line Chart's window has three tabs on the left (Dialog, Graph, and Data).

Interactive dialog items

Some dialogs calculate and display values that are generated as the model runs. If you leave a dialog open during the simulation, it will slow the run but you can watch the impact on different variables. This *interactive simulation* capability means you can even change some of the settings in a dialog during a simulation run, such as choosing different buttons or entering new values.

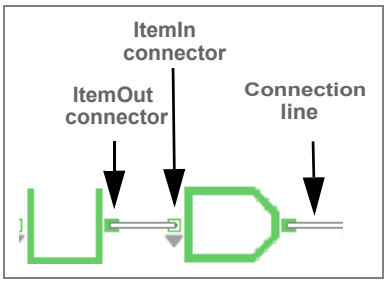
☞ When you click a button while the simulation is running, the block gets that changed value on the next event. However, if you type text or enter numbers into a parameter field or dialog table, the model pauses while you are typing in order to get your entire input.

Help, label, and view



As is true for most blocks, at the bottom of the Queue block's dialog is a Help button. It provides information about the block, such as its purpose and use, connector usage, descriptions of each dialog object, and so on. To the right of the Help button is a text box for entering a label for that specific instance of the block in the model. The View popup is for changing the icon's orientation or appearance when you build models—for example, the Activity block offers a choice of *Right to left* and *Left to right*.

Connectors and connections



- *Connectors* are used to get data into and out of a block during a simulation run
- *Connections* are the mechanisms used to transfer data and information between blocks

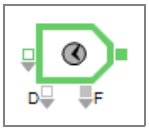
Connectors

Most blocks in ExtendSim have input and output connectors (the small circles or squares attached to the block's icon). As you might expect, information flows into a block at input connectors and out of the block at output connectors.

In ExtendSim, the behavior of most connectors is predefined for each specific block. For example, the ItemIn connector on an Activity block automatically takes in the items passed to it from the Queue. This makes model building easy since you can connect blocks and run simulations without having to write equations or logic to define what each block should do with its inputs or outputs.

A block can have many input and/or output connectors; some blocks have none. In addition, a block can have single connectors or arrays of connectors depending on how the block is constructed.

Connectors in the Item blocks



The blocks in the Item library have primary connectors for getting and passing items (itemIn and itemOut) as well as secondary connectors for getting and passing values (input and output).

For instance, the Activity block (labeled Wash Bay) has two item connectors and two value connectors:

- The green squares on the left and right sides of the Activity's icon are for getting and passing items.
    - The green outlined *ItemIn* connector on the left is a variable connector for items to enter. Variable connectors act like an array of single connectors that can be expanded or contracted to enable a required number of connections. As shown here, a variable connector is usually designated by a dark gray arrow.

- The solid green *ItemOut* connector on the right is a single connector for items to exit the block.

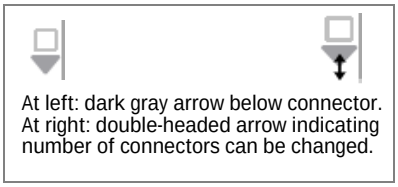
- V*alue* connectors are represented by gray squares and are used for getting and reporting values. As with item connectors, they can be a variable connector or a single connector. There are two variable value connectors on the bottom of the Activity.

  - The gray outlined variable input connector on the bottom left can be used to control specific block behavior, such as the block's delay time.
  - The solid gray variable output connector on the bottom right is for providing specific results.

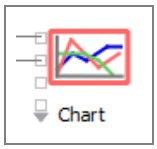
Expanding a variable connector

To expand the connector array for a variable connector:

- Hover over the dark gray arrow that is below the bottom input connector on the left side of the Line Chart block. (Don't hover over or click on the input connector, just hover over the gray arrow.) If you are in the correct spot, the cursor will turn into a dragging cursor—a double-headed up/down arrow—indicating that the array of connectors can be resized.


At left: dark gray arrow below connector.
At right: double-headed arrow indicating number of connectors can be changed.

- Once the cursor turns into the double-headed arrow, click and drag down until the desired number of connectors are exposed, as shown here. As you drag, more connectors appear and the number of connectors is displayed.



☞ See the User Reference for more information about variable connectors, including how to collapse them.

Connector function

To get information about a connector's function, click the Help button in the bottom left-hand corner of the block's dialog or see the User Reference.

☞ Place your cursor over a connector to see its name and current value; that value will change as the model runs. You may also see additional information depending on how the block is programmed.

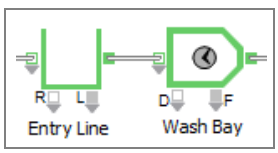
Connections

The connections between itemOut and itemIn connectors represent the path of items from block to block. Connections between value connectors represent the flow of information. The simulation itself is a series of calculations and actions that occur at each event.

Making connections

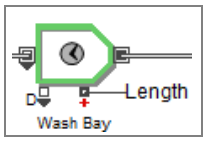
There are three ways to make connections between blocks:

1) *Line connections* join the output of one block to the input of another using connection lines.
   - ExtendSim uses the Point and Click method to physically draw connection lines.



- When an ExtendSim block's connector is right-clicked, a special *Smart Block technique* can be used to add a block to the model and at the same time automatically connect it to the original block with a connection line.
- For the Item and Rate libraries only, their primary connectors support *Bump Connect*. To use this feature, click on a block in the library window then bump its itemIn or inflow connector into the itemOut or outflow connector of a block already on the model worksheet. This places the new block on the worksheet and connects the blocks together.

2) *Named connections* use text labels as outputs and inputs, causing data to jump from the output to the input without using connection lines. The model has a named connection titled "Length" that reports the Queue's waiting line length on the Line Chart.



3) The *Throw/Catch* feature allows remote connections in a model, even between hierarchical levels; this is discussed in the User Reference.

Libraries

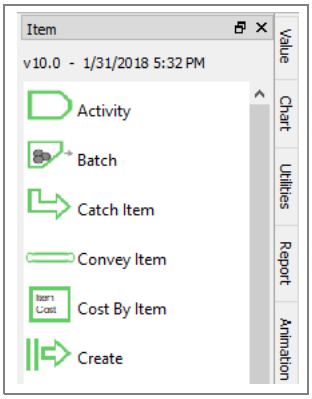
Libraries are repositories for the blocks used to build models. When you open an existing model, ExtendSim automatically opens the libraries that contain the blocks used in the model. Libraries are also automatically opened if listed in the table in the Edit > Options > Libraries tab.

Libraries can be manually opened, closed, and created in the Library menu. Open libraries are listed in alphabetical order at the bottom of the Library menu.

Note that any data entered in the block's dialog is stored within the model, not within the library.

Library windows

Each library has an associated *library window*, showing its blocks and their names. By default when you open ExtendSim, a library window for each open library opens and docks on the right side of the application window; to change that, see the Edit > Options > Libraries tab. The name of each library window is shown on the top of the docked window and in a tab along the right side.



Library windows behave like other docking windows. For example, to detach a library window from its docked position such that it becomes a floating window, double-click its title bar or click and drag to detach it. To re-dock a library window, click its close box, double-click its title bar, or click on its title bar and drag it back to the docking area.

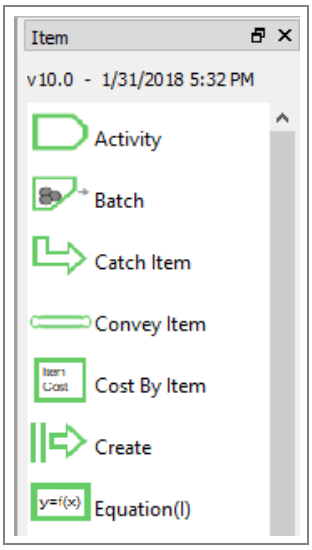
Use the close box to close a docked library window. To easily close all the library windows, use the Library > Close All Library Windows command or the corresponding button in the toolbar at the top of the windows. The Open All Library Windows command and button re-opens the library window for every open library.

☞ Closing a library window does not close the library; use the Library menu to close a library.

Blocks for building discrete event models

As discussed more below, to create discrete event models you will primarily use blocks from the Item library. However, it is common to also use blocks from other libraries that ship with ExtendSim, especially the Chart, Report, and Value libraries.

Item library



Blocks in the Item library correspond to typical activities, operations, and resources in many environments. These blocks are connected in an activity or data flow diagram that represents a system. The complexities of generating and posting events are handled within the blocks, alleviating the need to do any programming in the ModL language.

Item library blocks are optimized for modeling service, manufacturing, material handling, and other discrete event systems. The blocks represent queues, machines, labor, conveyors, and so forth and incorporate high-level modeling concepts such as variable batching, conditional routing, and preemptive operations. Built-in performance calculations and statistical reports allow you to predict the value, effectiveness, and cost of implementing changes before committing resources.

These blocks have been specifically designed to meet most discrete event modeling needs, allowing you to quickly and easily perform complex modeling tasks. For instance, you can use a popup menu in a Queue block to specify that stored items are sorted in first-in-first-out order, last-in-first-out order, or in a custom order based on their assigned priorities or attributes.

Custom discrete event blocks

Because of the Item library's extensive capability, it is unlikely that you would need to create your own discrete event blocks. If you do want a custom discrete event block, they are created the same way the blocks in the ExtendSim libraries were created—using the included ExtendSim IDE and its programming tools—then saved in libraries for use in all models. Two important notes:

- If you use an ExtendSim block as a basis for your custom block, be sure to save your block under a different name. Otherwise, your code will be replaced whenever ExtendSim updates.
- Store your custom blocks in a new library, not in an existing ExtendSim library. Otherwise, you will loose your custom blocks whenever ExtendSim is updated.

☞ See the Appendix in the ExtendSim User Reference for a listing and brief description of the blocks in the Item library.

Value library

Most models use one or more continuous blocks from the Value library. They are pre-programmed to simplify data management and model-specific tasks and allow you to perform complex modeling steps often with just the click of a button. For example:

- Use the Random Number block's popup menu to select one of 35 distributions.

- The Equation block is for creating compiled equations, from a simple operation to a full programming segment. It uses formulas and/or ModL code entered in its dialog to calculate values for, and perform operations in, models.
- The Optimizer block facilitates optimization.
- The Scenario Manager block allows you to strategically explore the outcome of different model configurations and analyze alternatives

See the Continuous Process Quick Start Guide or the User Reference for more information about blocks in the Value library.

☞ Using Value library blocks with Item library blocks does not change the fundamental architecture of discrete event models—they will still be event-based rather than use the time-based architecture of continuous models.

Chart library

The blocks in the Chart library are useful for any type of model. The most commonly used block is the Line Chart, which traces the history of values over time during the simulation.

Report library

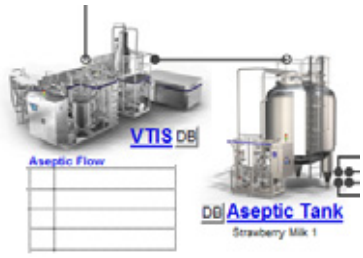
The blocks in the Report library are also useful for any type of model. They gather information as the model runs and display and calculate statistics on the results.

Utilities library

The Utilities library has several blocks that are useful for creating a custom user interface for models and for performing other tasks and activities. See the “How To” section of the User Reference for some examples.

Hierarchical blocks

If your model becomes too cluttered with blocks or you’d like it better organized, encapsulate portions of the model into a hierarchical block. To see the submodel, double-click the hierarchical block. Hierarchical blocks are created using simple menu commands; they can be copied from model to model and stored in libraries for reuse in other models.



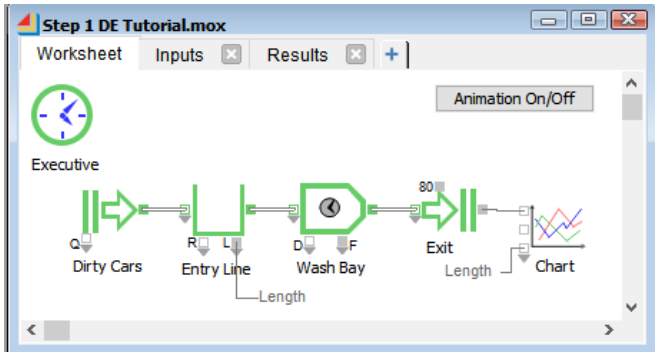
Two hierarchical blocks

Discrete Event Quick Start

Tutorial Part I

Overview

This tutorial shows how to build the simple discrete event model shown below.



⚠ It is important that you have read the previous chapter, or understand ExtendSim basics, before proceeding.

A simple discrete event process

When building any simulation model, it is best to start with a simplified subset of the process. Then continue adding detail, verifying and validating at each stage, until you arrive at a completed representation of the system. This will make the model building process more manageable.

About the model

The most common discrete event model involves the handling of one or more waiting lines or queues, such as those found in supermarkets or factories. This example model represents a business operation where cars get washed; it is an example of a single-server waiting line.

☞ This model will be similar to the model you opened and ran in the previous chapter.

Assumptions

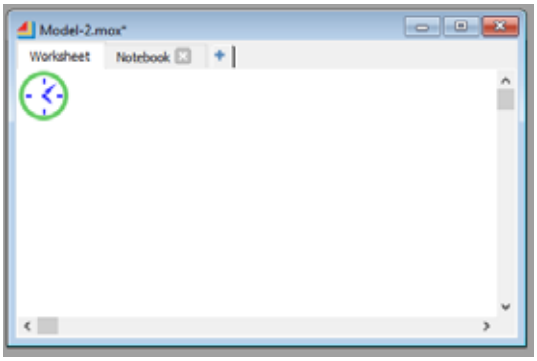
- There is only one route into the car wash
- Cars arrive approximately every 4 minutes
- It takes 6 minutes to wash a car
- The model runs for a simulated time of 480 minutes (8 hours)

Launch ExtendSim

By default, when you launch ExtendSim the Getting Started model opens as well as the major ExtendSim libraries and their library windows.

Open a new model

- Use the toolbar button or the File menu to open a new model.



By default, two things happen:

1) A new model window, named Model-X, opens
2) An Executive block is placed on the left-most side of the worksheet

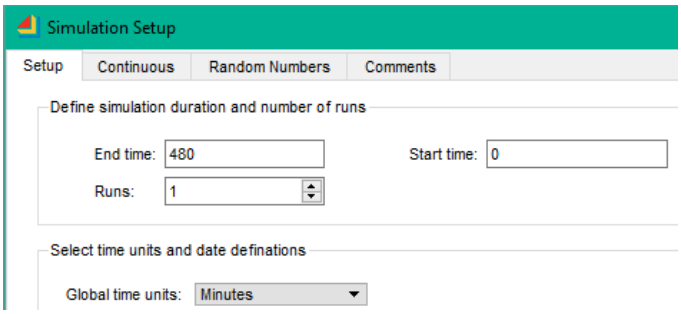
⚠ The Executive block will only be automatically placed in new models if the Item library is open.

If the major libraries have not been set to open in the Edit > Options > Libraries tab, you will need to manually open the Chart, Item, and Value libraries using the Library menu. Then place an Executive block (Item library) on the worksheet.

Set the simulation parameters

- Select the command Run > Simulation Setup (Ctrl/control J)

This command opens a window for entering global settings for the model, such as how long and how many times the simulation will run.



The most common simulation settings you will need to enter in the Simulation Setup window (and often the only ones) are located on the Setup tab: the *End time*, the number of *Runs*, and *Global time units*. For most purposes, you want the simulation to start at the beginning, so you would use the default start time of 0.

Use the model assumptions to set up the simulation run:

- In the Setup tab, enter the simulation parameters:
  - **End time: 480**
  - **Start time: 0** (default)
  - **Runs: 1** (default)
  - **Global time units: Minutes**
- Leave the other Simulation Setup settings at their defaults
- Click **OK** to close the window

With these settings, the model will run for a simulated time of 480 minutes.

Save the model

▸ Choose File > Save Model As and name the file *My Car Wash*.

☞ Models with changes that haven't been saved have an asterisk (*) at the end of their name in the title bar. Each time you save a model, ExtendSim creates a backup, *ModelName.bak*, of your previously saved model. To recover using a backup file, rename the backup file as *ModelName.mox*.

Blocks used for the simple Car Wash model

The following table lists the six blocks used in the model.

Name (Label)	Library	Block Function	Purpose in Model
Executive 	Item	Does event scheduling and provides for simulation control, item allocation, attribute management, and more.	Must be present and on the left-most side in every event-based model. *Note: If the Item library is open, this block will automatically be added to new models as they open.*
Create 	Item	Generates items or values, either randomly or on schedule. If used to generate items, it pushes them into the simulation and should be followed by a queue-type block.	Generates cars that arrive randomly, approximately every 4 minutes.
Queue 	Item	Acts as a sorted queue or as a resource pool queue. As a sorted queue, holds items in FIFO or LIFO order, or sorts items based on their attribute or priority.	Holds the cars and, when the wash bay is available, releases cars one by one in first-in, first-out (FIFO) order.
Activity 	Item	Processes one or more items simultaneously. Processing time is either a constant or is based on a distribution or an item's attribute.	Washes the car for a simulated 6 minutes.
Exit 	Item	Removes items from the simulation and counts them as they leave.	Removes cars from the model.
Line Chart 	Chart	Displays information about the model.	Reports the length of the waiting line and how many cars have been washed.

There is nothing fundamentally different about the structure of these different blocks. Any block may create, modify, or present information, and many blocks perform more than one of these functions. You can, of course, have multiple instances of the same block (such as the Activity) within a model.

Placing and connecting blocks on the model worksheet

Methods for placing blocks in a model

There are two main methods for placing and connecting blocks in a model:

1) The Point and Click method is useful for adding any block anywhere on the model worksheet:

   - Click a block in the Library menu or library window to select it
   - Click somewhere on the worksheet to place it
   - As needed, connect it to another block using one of the connection methods discussed on page 15

2) The Smart Block method is a quick way to add a new block to a model and connect it to an existing block at the same time:

   - On the model worksheet, right-click on one block's connector and, from the list, select the new block to add. This will place the new block next to the original and connect them.

You will use both methods for this tutorial. However, to use the right-click technique, a non-Executive block must already be on the worksheet to start the connection chain. So you will place the first block on the worksheet using Point and Click.

☞ A third method, Bump Connect, only works with the primary connectors on Item and Rate library blocks. See the User Reference for more information about Bump Connect.

Use Point and Click to place an Item library block on the worksheet

To use Point and Click to place a **Create** block (Item library) on the model worksheet:

- Bring the **Item** library window forward by clicking once on its name in the list of library windows on the far right of the application window
- In the Item library window, click once on the **Create** block's icon

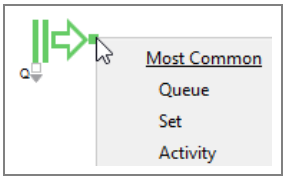
- Then click on the model worksheet somewhere below and to the right of the Executive

☞ If you select the wrong block, click the Esc (escape) key to unselect it. (You can also place multiple instances of the same block on a worksheet by holding down the Alt or Option key while repeatedly clicking on the worksheet.)

Use the Smart Blocks technique to place the remaining Item blocks

Now, because it's an easy and fun way to add blocks to a model, use the Smart Block technique to place additional Item library blocks on the worksheet.

Queue block



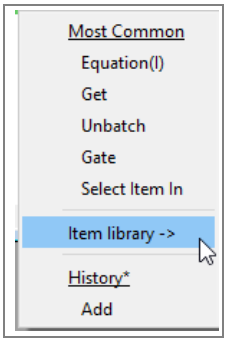
- Go to the model worksheet.
- Right-click on the Create block's **ItemOut connector** (the solid green square on the right of the icon). A list of the blocks most commonly added to the Create appears.
- Select the **Queue** from the list of common blocks. Note that the Queue gets automatically placed on the worksheet and connected to the Create.

Activity block

- Next, right-click on the *Queue* block's ItemOut connector and choose an **Activity**. This places an Activity block on the worksheet and connects it to the Queue.

Exit block

Finally, since the Exit block isn't one of the five most common blocks that follow an Activity:



- Right-click on the Activity's ItemOut connector.
- In the popup that appears, click the **Item library** as shown at right. This opens a new popup listing all the blocks in the Item library.
- Select an **Exit** block in the new popup. This connects the output of the Activity to an Exit block's input connector.

☞ While not needed here, the Smart Block technique can also be used to connect a second block to an output (hold down the Ctrl/Cmd key while right-clicking on the output) or to insert a new block between two connected blocks (separate the two blocks to provide room for the new block, then right-click on the output or input of one of the existing blocks to insert the new block).

⚠ Since Smart Block involves right-clicking on a connector, you must already have at least one block, other than the Executive, on the model worksheet for it to work. This technique works with the major ExtendSim libraries and, if they have been programmed to use it, with custom libraries.

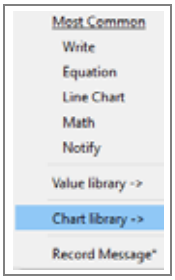
Place the Line Chart block on the worksheet

As noted in the table on page 21, there are two pieces of information you want displayed as the simulation runs:

- The total number of cars that have been washed
- The length of the waiting line

The Line Chart block (Chart library) is useful for tracing the history of values during the simulation.

Report the number exited

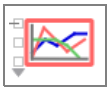


To report the number of cars processed, you will use the Smart Block technique to add the Line Chart block to the model.

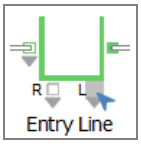
The Exit block has two value output connectors. The output on the top reports the total number of cars that exit the block; the output on the right is used if there are multiple streams of items entering the block. Eventually there will be multiple streams, so use the output on the right side of the Exit's icon.

- Right-click on the output connector that is on the right side of the **Exit block**.
- In the popup that appears, click the Chart library as shown at right.
- Select the **Line Chart** block. This connects the Exit block's output connector to the top input of the Line Chart block.

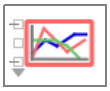
Report the queue length



You could use the Smart Block technique to add a second Line Chart block to the model. However, you will instead draw a connection line to the existing Line Chart block so that it displays both the number exited and the length of the queue.



- Hover the cursor over the Queue's **L** (length) output connector, located on the bottom right side of the icon, until that connector enlarges.
- Click once on that output connector.
- Then, since you're going to use the second input on the Chart in the next tutorial, move the cursor and hover it over the **third** input from the top of the Line Chart block. That input connector enlarges.
- When the connection line turns thick and blue, click once on the Line Chart's input connector to make the connection.

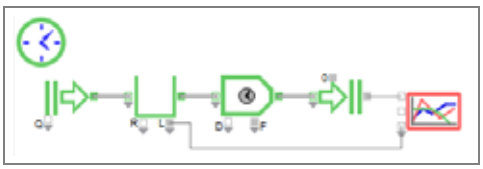


This connection line will cause the Chart to report the number of cars exiting during the simulation.

☞ Connection lines will turn solid dark gray when properly connected; they appear as a dashed red line if the connection has not been made correctly.

Adjust the connections and blocks

By default, connection lines are drawn using a Smart Connection technique that provides intelligent connection line control when moving blocks around. If Smart Connections don't result in a good appearance for the finished model, you can change the connection line style and adjust the line.



- To change the connection line from the Queue's L connector so the line doesn't cross over anything:
  - Right-click on the connection line
  - Choose the Right Angle or Free Form Connection
  - Move the line segments as desired by selecting and moving the anchor points.
- Reposition the blocks as desired.

When finished, the model should look similar to the one shown above.

☞ To change connection line behavior and appearance in a model, go to the Model > Connection Line Style menu or use the right-click menu. To change the default connection line behavior for all models, use the Edit > Options menu.

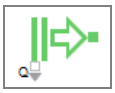
Save the model

- Choose File > Save Model to save your changes.

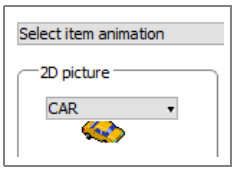
Enter dialog parameters and settings

There are only a few values to enter to reflect the basic car wash assumptions.

Create block

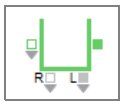


- To open the dialog of the Create block, double-click its icon or right-click on the icon and choose Open Dialog
- In the *Create* tab of the Create block:
  - The default setting is that items are created randomly using an exponential distribution. Since this is exactly what you want, just enter **Mean: 4**. With this setting, one car will arrive approximately every 4 minutes.
  - In the field at the bottom of the dialog, label the block **Dirty Cars**
- By default, ExtendSim animates items as green circles. This can be changed in the block's *Item Animation* tab:



  - Notice that the popup is set to *Select item animation*
  - From the popup list for the 2D picture, select **Car**.
  - Click OK to save changes and close the block's dialog.

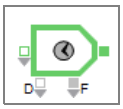
Queue block



- Open the dialog for the Queue block.
- By default, the Queue block is specified as a sorted queue, with items stored and released in first-in, first-out order. Since this is what the model specifies, do not make any changes to the settings.
- Label the block **Entry Line**.
- Click OK to save changes and close the block's dialog.

Activity block

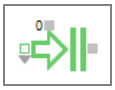
The assumptions indicate that cars are washed one at a time and that it takes the same amount of time to wash each car.



- In the dialog of the Activity block, the default settings are that the capacity is 1 and the delay is a constant amount of time. Since those settings are what you want, enter **Delay (D): 6,** indicating that it takes 6 minutes to wash each car.
- Label the block **Wash Bay**.
- Close the block's dialog.

Exit block

The Exit block automatically counts and passes items out of the simulation. There are no parameters to enter for this block.



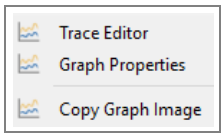
- In its dialog, label the block **Exit**.

Configure the Line Chart block

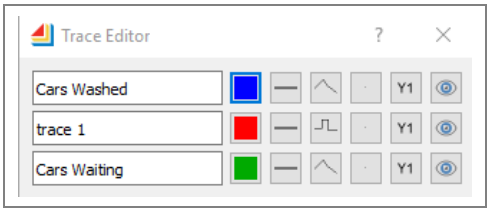
As this model is constructed, the Line Chart will get the number of cars that have been washed through its top input connector and the number of cars waiting to be washed on the third input.

Trace Editor

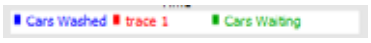
- Open the Line Chart block by double-clicking its icon or by right-clicking and choosing Open Dialog.



- Right-click on the block's Graph window. The two options at the top of the popup are Trace Editor and Graph Properties.
- Choose the Trace Editor. This is where you specify how the lines are drawn on the graph.

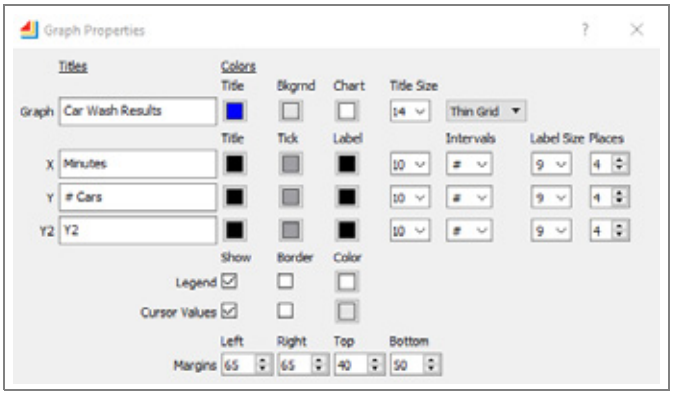


  - Name the first trace **Cars Washed**
  - Leave the second trace blank
  - Name the third trace **Cars Waiting**
  - For both the first and third traces, click on the line style (the third button) until it changes to interpolated, as shown above
  - Close the Trace Editor
  - Notice that the new labels appear in the key. You can resize the key and place it anywhere you want on the graph.



Graph Properties

- Right-click the graph window again and this time choose Graph Properties. This is where you customize the titles, colors, and other properties of the axes, as well as characteristics of the numbers that are displayed.



- For the titles:
  - Name the graph **Car Wash Results**
  - Name the X axis **Minutes**
  - Name the Y axis **# Cars**.
- Close the Graph Properties window.

☞ You can also enter titles directly on the graph—click on the text and change it, then press the Enter key.

Dialog tab

- Click the Dialog tab on the left side of the graph. The tabs in the Dialog are for labeling the block and for setting what happens to the graph during or after the simulation run. In the Dialog:
  - In the block's Display tab, choose if you want the graph to not open, or to open at the beginning or the end of the simulation run. Also choose when you want the graph to autoscale.
  - At the bottom of the dialog, label the block **Chart**.
  - Close the block's dialog.

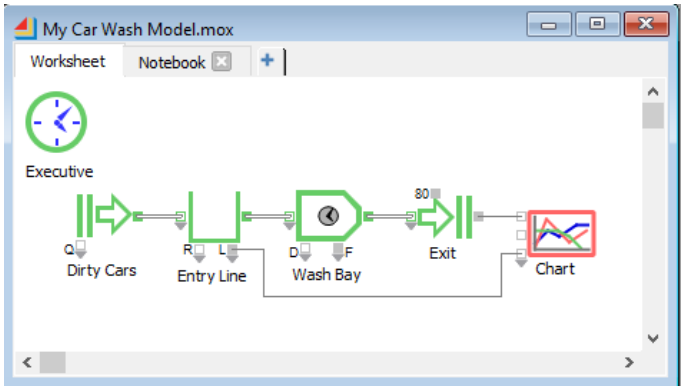
☞ See the Line Chart's Help button or the User Reference for more information about the tools and dialogs that control this block.

Save the model

- Save the model to save your changes

Your tutorial model

If you have followed all the steps, your model should look similar to the model shown below.



Running a simulation

Before running the model, decide if you are going to run a single model one or more times or run different models all at the same time.

Run modes

There are two modes toggled using the Run Mode toolbar button:

1) To run a single model as quickly as possible, even if you are running that model multiple times, use the fastest run mode as shown here. This is the default and what you will use for this tutorial.
2) To run different models all at the same time, use the multi-threaded run mode . This takes advantage of the ExtendSim multi-threading capability—the user interface is pro-

cessed in one thread and the processing for each model occurs in separate threads. In this mode, the Run Simulation button changes to .

Run the simulation

- Click the Show 2D Animation button in the toolbar or choose the Run > Show 2D menu command to enable 2D animation.
- Click the Run Simulation button in the tool bar or choose the Run > Run Simulation command or right-click to run the simulation. (Since you are only running one model at a time; leave the Run Mode at the default fastest mode.)
- If it isn't open, double-click the Chart block to see its graph. The block shows the results as a graph, seen below, and as data in its Data tab.

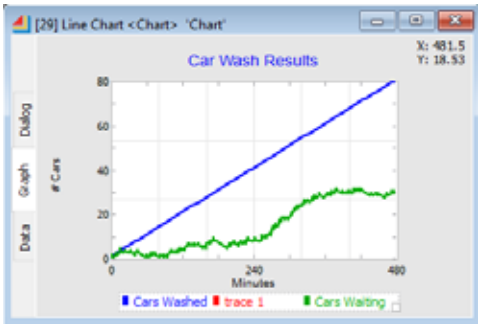
As seen in the previous chapter, if you run with 2D Animation on, the blocks will automatically display information and status on their icons. With animation on, it is easy to see that cars are arriving faster than they can be washed.

☞ While animation is very useful for debugging models or for making presentations, it can considerably increase the time it takes a discrete event simulation to run.

Verify and validate results

This is a good opportunity to verify and validate the results. Given how the model is constructed and the settings in its dialogs, are the numbers in the graph what you expected to see? Does the model accurately represent the system being modeled?

The Line Chart shows the results as a graph, seen at right, and as data in its Data tab.



Your numbers may differ because there is randomness in this model, but at the end of the simulation the graph might show that 80 cars have been washed and that there are around 40 cars waiting to be washed. This would correspond to the information in the Create block, which shows that about 120 cars were generated. These numbers make sense considering that 1 car is generated approximately every 4 minutes and the simulation runs for 480 minutes.

☞ See the User Reference's Process of Simulation chapter for information about verifying and validating a model.

Save the model

Since you'll use this model in the next chapter:

- Select the command File > Save.

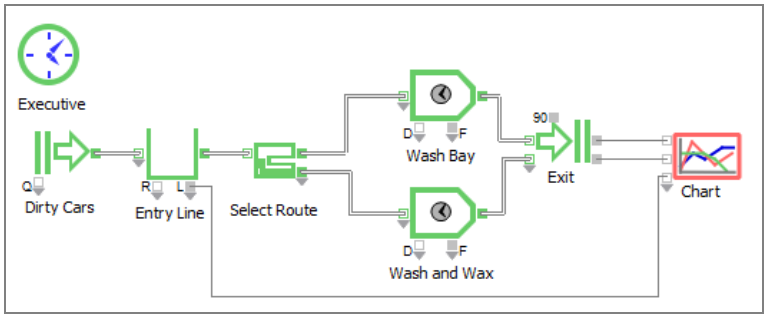
To compare your model to ours, see the example model named *Step 1 DE Tutorial* in the folder Documents/ExtendSim/Examples/Tutorials/Discrete Event.

Discrete Event Quick Start

Tutorial Part II

The goal

The previous chapter showed how to build a simple discrete event model. This chapter adds some details to that model and discusses some ExtendSim features that can be used to enhance the model experience.



Model assumptions

The Car Wash model represents a business operation where cars can be washed and waxed. The assumptions for this model are:

- There is only one route into the car wash
- Cars arrive approximately every 4 minutes
- It takes 6 minutes to wash a car
- There are two bays—one for washing only, and one for washing and waxing
- Approximately 25% of the cars want to be waxed and it takes 8 minutes to wash and wax a car
- The model runs for a simulated time of 8 hours (480 minutes)
- You want to enhance the model's appearance

☞ The final Car Wash model and the models for the steps in between are located in the folder Documents/ExtendSim/Examples/Tutorials/Discrete Event. However, to get the maximum benefit of this tutorial it is recommended that you build the models yourself then compare your model to the example models.

Steps you will take in this chapter

1) Start with an existing model
2) Add a second wash bay
3) Route the cars sequentially so that both bays get used approximately the same amount of time
4) Configure the model such that 25% of the cars need to be waxed and can only go through a specific wash/wax bay
5) Optional: add user interface and other helpful features to your model

Start with the model from the previous chapter

- Open the *My Car Wash* model you built in the previous chapter
- Or, open the example model from the Tutorial folder:
  - Go to Documents/ExtendSim/Examples/Tutorial/Discrete Event
  - Select the example named *Step 1 DE Tutorial*, then click Open

Create a second wash bay

The model in the previous chapter had more cars waiting to be washed than could be accommodated. Since long lines deter customers, it would be better to keep the entry line short.

To increase the number of cars that the wash bay can process at any one time, you could simply change the capacity setting in the Activity's dialog. For instance, allowing a maximum of 2 items in the Activity would simulate a wash bay that could wash two cars at a time.

However, since the assumptions state that the final model has two bays, you will add a second bay. One way to do this would be to duplicate the existing Activity block and connect it. Or you could use Smart Blocks to add and connect a second Activity block in parallel with the first, as done below.

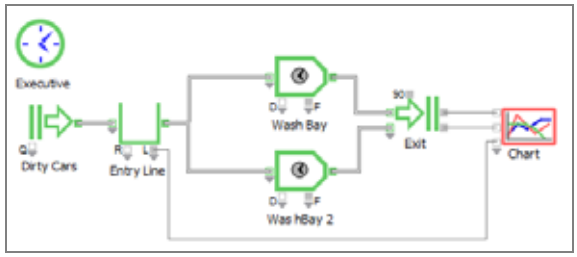
Append a second Activity block

- Hold down the **Ctrl/control key** while you right-click on the **itemOut** connector of the **Entry Line** (Queue) block.
- From the popup, select the Activity block. This adds a second Activity to the model, in parallel with the first one, and connects it to the Entry Line queue.
- In the Activity's dialog:
  - Set the **Delay (D): 6**
  - Label the new block **Wash Bay 2**.

☞ When you instead want to insert a block between two connected blocks, move the two blocks apart to provide room for the new block. Then right-click on the output connector of the first block and choose the block to insert.

Connect the new block to the Exit

- Expand the variable input connector on the **Exit** block so that it reveals a second input for items. (The instructions for expanding variable connectors were given on page 15.) Notice that expanding the number of input connectors on the Exit block also expands the number of its outputs.
- Connect Wash Bay 2's itemOut connector to the Exit block's second itemIn connector.



Connect to the Line Chart

- Connect from the Exit block's second **output** connector to the second (from the top) input connector on the Line Chart block.
- Save and run the simulation.

With the second wash bay, the entry line's queue length stays near 0 most of the time, as shown in the graph and the Queue block's Results tab.

Your model should now be similar to the *Step 2 DE Tutorial* model located at Documents/ExtendSim/Examples/Tutorial/Discrete Event.

Route the cars sequentially

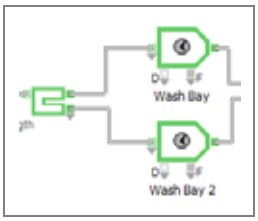
Since you have not specified any rules concerning how the cars are routed to a wash bay, a car will go to the first available bay. However, if both bays are free, the car will go to the bay that was first connected in the model. This implicit routing is not obvious and is rarely what you want.

⚠ Unless it is completely unimportant in the model, you should always explicitly state the routing. The most common method to do this is by using the Select Item In and Select Item Out blocks from the Item library. Otherwise, the order in which connections were made could dictate the routing, as discussed in the User Reference's Discrete Event Routing chapter.

Explicitly stating the routing method

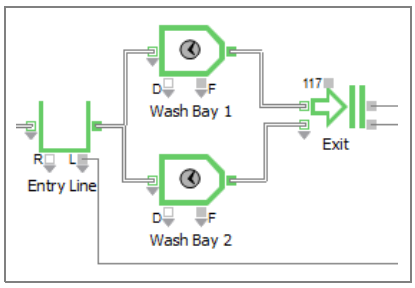
☞ Right-clicking is a great way to insert one block between two other blocks. However, if there are more than two blocks as is the case in this model, the Smart Block technique won't know where to make the insertion. So the following first deletes the connections to the two wash bays.

- Since you're going to insert a block after the Entry Line, frame-select to move everything that is on the right of the Entry Line further right.
- Delete the connections from the Entry Line to the two Wash Bays
- Right-click on the Entry Line's itemOut connector:
  - In the popup that appears, click the **Item library**.
  - Select an **Select Item Out** block in the new popup, connecting that block to the Entry Line.
- Connect the itemIn and itemOut connectors of the blocks as shown on the right.



- In the dialog of the Select Item Out block:
  - Label the block **Select Route**
  - Choose the option **Select output based on: sequential**
  - So that cars won't be blocked at one bay when the other is free, select **If output is blocked: item will try unblocked outputs**

This causes the cars to be sequentially routed between the two bays. If the selected bay is blocked, the car won't wait for it to be free but will instead be routed to the other bay if available. When you run the model, a similar number of cars will have been washed as in the previous model, but as seen by the dialog of the Exit block, each wash bay will have been used equally.



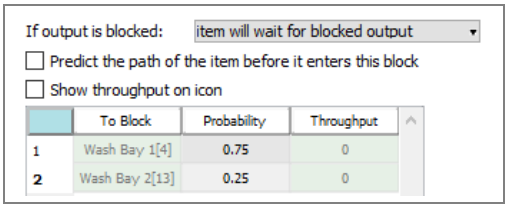
Your model should now be similar to the *Step 3 DE Tutorial* model located at Documents/ExtendSim/Examples/Tutorial/Discrete Event.

Designate some cars to require waxing

Most car washes allow cars to have wax applied after the wash and the model assumptions state there is a wax option. Assume that Wash Bay 2 is where cars go to get washed and waxed.

- In the dialog of the Select Route block:
  - Change the selection condition to *Select output based on:* **random**. Notice that this changes other options in the dialog.

  - In the first row of the table (Wash Bay), set the Probability to **0.75**
  - In the second row (Wash Bay 2), set the Probability to **0.25**
  - So that cars will respect the probability designation rather than looking around for a free bay if their assigned bay is blocked, select **If output is blocked: item will wait for blocked outputs**



- In the dialog of the Activity block labeled Wash Bay 2:
  - Change the **Delay (D)** to **8** to reflect the model assumption that waxing takes an additional 2 minutes after washing
  - Change the label for the block to be **Wash and Wax**

Save the model

- Save the model to save your changes

Your model should now be similar to the *Step 4 DE Tutorial* model located at Documents/ExtendSim/Examples/Tutorial/Discrete Event.

☞ Using attributes to specify which cars required waxing would be a more powerful and elegant solution than setting a percentage. Attributes give items unique properties and characteristics, allowing them to be tracked, examined, and routed individually throughout a model. See the User Reference, Discrete Event section, for how to set and use item attributes.

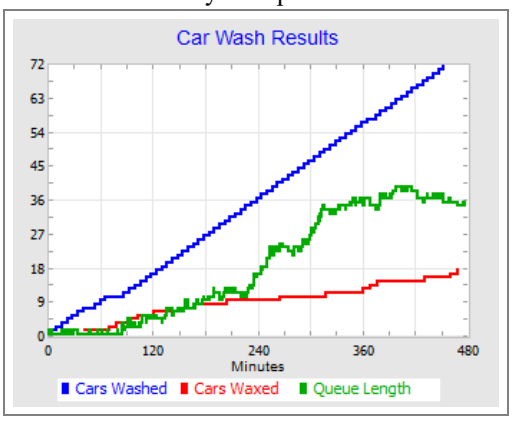
Model verification and validation

As mentioned in the previous chapter, it is important to verify that the model runs as expected and accurately represents the system. When the model is run, the Line Chart indicates that having some cars require waxing at a special bay:

- Increases the line of cars waiting to be processed
- Reduces the total number of cars that exit the simulation

This makes sense because there is one entry line, waxing takes longer than washing, and cars are limited to their specific bay. So even if the wash bay is available, a car waiting to be washed could be blocked by a car that is waiting to be waxed, further impacting processing.

To see how many cars have exited from each bay, open the dialog of the Exit block or see the Chart's Data tab. Are the numbers what you expected?



A final Car Wash model

So that you can see some additional features and capabilities, the Documents/ExtendSim/Examples/Tutorial/Discrete Event folder contains a *Car Wash Final* model that:

- Assigns attributes to the cars as they enter the simulation:
  - A "carType" attribute has the cars determine whether they only want to be washed or also want wax.
  - A "delayTime" attribute determines how long it takes to process a car based on its "car type".
  - Attributes give items unique properties and characteristics, allowing them to be tracked, examined, and routed individually throughout a model.
- Has dynamic rather than static processing times. There are many ways to do this: select a random distribution for the processing time in the Activity block, use a Lookup Table block (Value library) to schedule processing to be dependent on the time of day, cause an Equation block (Value library) to calculate a wash time based on model conditions, or (as done in this model using the Equation block), assign attributes to the cars to represent a processing time depending on the type of vehicle.
- Create the model such that attendants are needed to drive the cars through the wash. Note that, as discussed in the Resources and Shifts section of the User Reference, and in the separate document *Advanced Resource Management Tutorial and Reference*, there are many ways to model resources.
- Have arriving cars look at the waiting line and not enter the car wash if the line is too long (balking) or leave the line after arrival if the wait time reaches a certain point (reneging). These concepts are discussed more in the User Reference, Discrete Event Queueing chapter.
- Consider using a hierarchical block from the Templates library to replace some of the functionality in this model. For example, the "Create - Arrivals Vary by Time" template would be helpful if you had an historical record of car arrivals.

 See the User Reference's section on Discrete Event Modeling for how to accomplish the above.

Your model could have other enhancements such as:

- If the model were more complex or required a lot of data, the ExtendSim internal database could be used to track inputs and outputs.
- The Scenario Manager (Value library) could evaluate and compare different model configurations.
- For model verification and validation, you could use the Reports Manager block (Report library) to get detailed information about every block and event in the model.

Enhancing the user interface

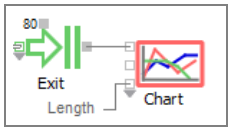
The Car Wash model you built in these tutorials adheres to the assumptions and runs as expected. But it could use some enhancements to make it easier to use and more presentable..

⚠ This section gives a brief overview of a few ExtendSim user interface features. See the How To: Big Picture chapter of the User Reference for a more extensive summary.

Named connections

The connection lines in this model are manageable, but this model has an unsightly line stretching from the Queue (Entry Line) to the Line Chart. Furthermore, larger models could have connection lines going all over the place.

Named connections use text labels as outputs and inputs, causing data to jump from the output to the input without using connection lines. For example, the Step 1DE Tutorial model uses a named connection to report the length of the queue.



☞ If you don't like how a connection line gets drawn, select the line and right-click to choose a different line style. To select a different default line style, use the Edit > Options > Model tab.

Clones and notebooks

Clones are exact copies of the parameter fields, text, tables, or graphs from a block's dialog. Similar to a shortcut or alias, clones behave identically to the original. Clones can be placed on a model worksheet or on notebooks.

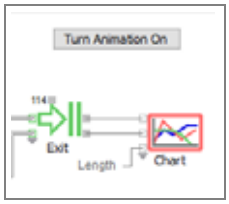
Notebooks are customizable windows for organizing and managing a model's information. You can clone dialog objects, insert text and graphics, and paste pictures onto a notebook, then use it to control model parameters, report simulation results, and document the model. And notebooks can be torn off from the model for better accessibility.

As seen on page 11, the Car Wash models have parameter fields, tables, and other objects cloned to two notebooks labeled Inputs and Results.

☞ You can clone one dialog object, several, or an entire frame. Delete unwanted clones using the Clone cursor.

Interactive dashboard

ExtendSim has many tools and features for creating an interactive dashboard or control panel for models. For example, the Buttons block (Utilities library) has several pre-built buttons that can be cloned onto the worksheet, such as the Turn Animation On button shown here.

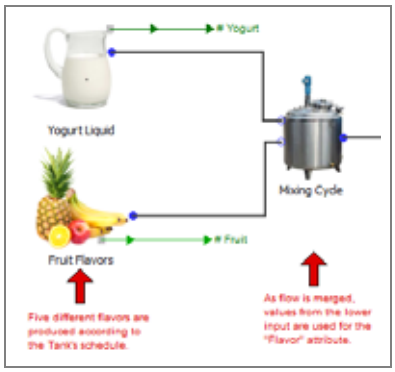


☞ For custom buttons, enter an equation in the Buttons block dialog and give your button a name.

Hierarchy

Hierarchy is a method for representing multiple blocks or an entire submodel as one block within the model. With hierarchy you can:

- Encapsulate sections of a model into one or more blocks for scalability
- Create generic model structures and store in libraries for reusability
- Customize and animate the icons that represent sections of your model



☞ Hierarchical blocks can be created from the top down or from the bottom up, can be copied within a model and from one model to another, and can be saved in libraries for reuse.

Custom animation

Item library blocks have an Item Animation tab for specifying how an item gets represented when 2D animation is on. You saw this on page 25, where you chose a picture of a car to represent vehicles entering the car wash.

By default each block's Item Animation tab will use whichever picture was selected in the preceding block. You can also cause a block to use a different picture to represent items that pass through a block, or have the animation picture change depending on the item's properties—such as its attribute or priority.

ExtendSim comes with many common pictures for animation; you can easily add more. For more information, see the How To: Presentation/Animation section of the User Reference.

Next steps

When you've finished this chapter:

1) Please read the next chapter in this Quick Start Guide. It explains the discrete event terms and concepts and provides a top level picture of the how the Item library works.

2) As you build models, refer to the "Discrete Event"' section of the User Reference and to the numerous example models located at Documents/ExtendSim/Examples/Discrete Event. Supported by the example models, the Discrete Event section of the User Reference is devoted to various aspects of building a discrete event model so that you can learn how to:

   - Generate items with properties and values
   - Specify different types of queues

- Route items throughout the model
- Process items in series or in parallel, interrupt processing, and multi-task
- Batch and unbatch items
- Model resources and shifts; schedule resource availability
- And much more

3) Also refer to the “How To” section of the User Reference, which points out ExtendSim features that will help you to:

- Represent the dynamics of the system you’re modeling
- Create a user interface for the model
- Get data into the model
- Manage data for use in the model
- Perform analysis and assessments
- Report model results and export data
- Communicate with other applications and devices

Discrete Event Quick Start

Concepts & Terminology

Before building a discrete event model, it is helpful to understand the terminology that will be used and to have an overview of ExtendSim discrete event architecture.

Overview of a discrete event model

Discrete event models pass entities (called *items*) from block to block as events occur during the simulation run. The items in the simulation are usually generated as a random distribution within specific parameters, or as a scheduled list of when events will occur. These items can have properties, such as attributes and priorities, which help them correspond more closely to parts, customers, jobs, and so forth in real life. Items are processed by activities, and the time and extent of processing is often dependent on the availability of resources.

The main source of discrete event blocks is the Item library. Most of the blocks in the Item library have item connectors and value connectors. An item connector passes an item and all the information associated with it to the next item connector. Value connectors and dialog parameters provide specific information about the item and its properties (attributes, timing, and so on) as well as information about the effects that the item has in the model (such as queue length and wait).

☞ It is this value information which is plotted and displayed in a discrete event model, not the items themselves.

Often the purpose of a simulation is to determine where there are bottlenecks in a process and to see which parts of the process might be improved. Each branch of the flow diagram should either feed into another block or end in an Exit block.

A model can combine continuous blocks, typically those in the Value library, with discrete event blocks from the Item library. If you use any discrete event blocks in a model, the model will become discrete event and will require the Executive block (Item library).

Layout of a discrete event model

You can place the blocks in a model anywhere you want, remembering that ExtendSim evaluates discrete event blocks along the path of the connections. The only exception to this generality is that the Executive block must be to the left of all other blocks.

Executive block

The Executive block, which is required for all discrete event simulations, controls and performs event scheduling. An Executive block must be present and located on the left of all other blocks in a discrete event model. Its use in a model changes the timing so that simulation time advances from one event to the next, rather than at uniform intervals.

For more information, see the User Reference, Discrete Event Tips chapter.

☞ Most of the Executive's options are for advanced users. Unless you use string attributes, it is rare that you would need to make any changes in the Executive's dialog.

Events, activities, and resources

DES models involve queues, activities, routing, and the monitoring of the movement, position, and properties of the uniquely identifiable system components called items.

☞ See below for more information about items.

Events

ExtendSim moves items in a discrete event model only when an event happens. Events are occurrences such as receipt of an order, a telephone call, or a customer arriving. They are managed by the Executive block (discussed in the User Reference, Discrete Event Tips chapter) and only occur when particular blocks specify that they should.

Blocks that depend on time cause events to happen at the appropriate time. For instance, an Activity block holding an item until a particular time will cause an event to be posted to the ExtendSim internal event calendar. When the time is reached, the event occurs and the model recalculates its data.

Blocks that do not generate events allow the blocks after them to pull items during a single event. Thus a single event can cause an item to pass through many blocks if those blocks do not stop them. For instance, a Set block could set the item's attribute and pass the item to the next block in the same event.

For more information, see event scheduling in the User Reference, Discrete Event tips chapter.

Activities

Activities are undertaken to achieve a specified outcome, typically either a product or a service. They have a duration and usually involve the use of process elements and resources. An activity could involve processing, moving, transporting, or otherwise manipulating an item. For more information, see the User Reference, Discrete Event Processing chapter.

Resources

Resources are the means by which process activities and operations are performed. Typical resources include equipment, personnel, space, energy, time, and money. Resources can be available in unlimited quantities but are most often limited or constrained. They can be consumed in the process or kept for reuse.

In ExtendSim, a resource that is required for a process or activity to take place can be modeled explicitly either as an item or as a unit in a pool:

- Item resources get batched with the items that require them (Resource Item method).
- A resource pool contains a count of the resources that are available to the model. The count can be a number in a Resource Pool block (Resource Pool method) or the number of records in an internal ExtendSim database (Advanced Resource Management method).

See the User Reference, Discrete Event Resources chapter for complete information.

Items and their properties

Items and informational values

The basic units that are passed between discrete event blocks are *items*. An item is an individual entity that represents an element of the system being modeled; it can only be in one place at a time. Items have a life cycle in which they are created, transformed, and eventually destroyed. They change state (physically move, are delayed, or have their properties altered)

when events occur, such as a part being assembled, a customer arriving, and so on. In manufacturing models, items may be parts on an assembly line; in network models, an item would be a packet of information; in business models, items may be invoices or people. Items are passed from block to block through item connectors.

The Create block is the main method for generating items in a model. It can create items using a random distribution, at a constant rate of arrival, at a fixed schedule, or on demand. The Resource Item block can also provide a finite pool of items to the model.

Items can have *properties* — different pieces of information attached to an item that make the item unique. Item properties include attributes, priorities, and quantities, as discussed below.

Values provide information about items and about model conditions. Values tell you the number of customers in queue, how many parts have been shipped, and how frequently telephone calls occur. Values also report processing time, utilization, and cycle time. These informational values are passed through value connectors. When you use a chart in a discrete event model you are graphing information about items, not the items themselves. For example, when the top output of an Exit block (total exited) is connected to a chart, it displays the time that each item left the model and the number of items that have exited.

Types of item handling blocks

Each Item library block is identified in its dialog as being a residence, passing, or decision type of block:

- *Residence blocks* are able to store an item for some amount of time. Examples of residence-type blocks are the Queue and Activity.
- *Passing blocks* must pass the item along before any simulation time elapses. Example blocks include the Set, which sets item properties, and the Equation (I) which performs a calculation as an item passes through.
- *Decision blocks* conditionally allow an item to pass through. Examples include the Gate and Select Item In blocks, which hold or pass items based on dialog settings.

Knowing these categories of blocks and how they relate to the processing of items will help you to build better models. For complete information, see block types in the User Reference, Discrete Event Tips chapter.

Item properties

A property is a characteristic of an item that stays with the item as it moves through the simulation. Item properties include attributes, priorities, and quantities.

Attributes

Attributes are an important part of a discrete event simulation because they provide information about items. Each attribute consists of a name that characterizes the item and a value that indicates some dimension of the named characteristic. For example, an item's attribute name might be "color" and its value could be "1" (for "red"). Or the attribute name might be "ProcessTime" and its value "4.76". Attributes are often used for routing instructions, operation times, or part quality in statistical process control; they are discussed fully in the User Reference, Discrete Event Items chapter.

Priorities

Priorities allow you to specify the importance of an item. For instance, there might be a step in a manufacturing process where a worker looks at all the pending job orders and chooses the

one that is most urgent. Each item can only have one priority. The top priority has the lowest value, including negative values (that is, an item with a priority of "-1" has a higher priority than an item with a priority of "2"). Priorities are discussed fully in the User Reference, Discrete Event Items chapter.

Quantities

Each item can be a single entity or a group of duplicates. If the quantity of an item is 1, it represents one item; if it is greater than 1, it represents a group. By default, items have a quantity of 1. The quantity can be changed by a block such as the Set block. For more information, see the User Reference, Discrete Event Items chapter.

Connectors

Blocks in the Item library contain item and value connectors. Most of the discrete event blocks pass an *item index* through item connectors at each event. Each passed index contains a set of information about the item – its attributes, priority, quantity, and so on. This is different from value connectors which only pass informational values.

Making connections

When combining discrete event blocks with blocks from other libraries, you will only be able to connect compatible connectors. Item connectors can only connect to item or universal connectors; they cannot connect with value input or output connectors. Likewise, value connectors can only connect with value or universal connectors; they cannot connect with the item input or output connectors. For more information about connector types, see the section on connector types in the User Reference, How To: Libraries and Blocks chapter.

Closed and open systems

Blocks that provide a finite number of resources can be part of closed or open systems. How blocks are connected in the model determines whether the system is considered open or closed.

Closed systems

In a *closed system*, resources are routed from a resource block and used in the model. Once they are no longer being used, the resources are recycled back to the resource block and become available for further use. For example, assume a technician (the resource) is required to assemble parts of a television. While the technician is assembling the parts, he/she will be busy and will not be available to perform work elsewhere. In a closed system, the technician will return to the technician pool after assembling the parts and will become available for other assignments.

In a *partially closed system*, only a portion of the resources are returned to the resource block for re-use. For example, consider a case were there are different shifts of laborers (the resource). Suppose three laborers are assigned to a task. Upon completion, the shift for one of the laborers is finished and he does not return to the labor pool to be assigned to a new task.

Open systems

Resource blocks may also be part of *open systems* when the block's resources are not recycled. In an open system, resources at the end of the line are not passed back to the Resource block. The most common example of an open system is stock. Normally, stock passes out of the model at the end of the line. Another example of an open system is a consumable resource such as a disposable fixture that makes only one pass through the manufacturing process.

The User Reference and example models

The discrete event portion of the ExtendSim User Reference has chapters that discuss specific discrete event modeling concepts and techniques, such as queuing, resources and shifts, and costing. ExtendSim ships with numerous example models such as the models used in the Quick Start tutorial and examples that correspond to the concepts discussed in the User Reference.

INDEX

A-C

D-F

G-I

J-L

M-N

O-P

Q-S

T-V

W-Z