



extendSIM®

# USER GUIDE

*Imagine  
That!*®

Imagine That Inc. • 6830 Via Del Oro, Suite 230 • San Jose, CA 95119 USA  
408.365.0305 • fax 408.629.1251 • [info@extendsim.com](mailto:info@extendsim.com)  
[www.extendsim.com](http://www.extendsim.com)

Copyright © 2007 by Imagine That Inc. or its Licensors.

All rights reserved. Printed in the United States of America.

You may not copy, transmit, or translate this manual or any part of this manual in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use without the express written permission of Imagine That Inc.

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 (including icons, dialogs, and block code) are copyright © by Imagine That Inc. and its Licensors. ExtendSim blocks may contain proprietary and/or trademark information. If you build blocks, and you use all or any portion of the blocks from the BPR, Discrete Event, Flow, Item, Items (db), Mfg, Rate, SDI Tools, or Quick Blocks library 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 legal license for the ExtendSim product from which you have derived your block(s) or block code. For more information, contact Imagine That Inc.

Imagine That!, the Imagine That logo, ExtendSim, Extend, and ModL are either registered trademarks or trademarks of Imagine That Incorporated in the United States and/or other countries. Mac OS is a registered trademark of Apple Computer, Inc. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation. GarageGames, Inc. is the copyright owner of the Torque Game Engine (TGE), Simulation Dynamics, Inc. is the copyright owner of SDI Industry, Wolverine Software Corporation owns the copyright for Proof Animation, and the copyright for Stat::Fit® is owned by Geer Mountain Software. TGE, SDI Industry, Proof Animation, and Stat::Fit are licensed to Imagine That, Inc. for distribution with ExtendSim. All other product names used in this manual are the trademarks of their respective owners. All other ExtendSim products and portions of products are copyright by Imagine That Inc. All right, title and interest, including, without limitation, all copyrights in the Software shall at all times remain the property of Imagine That Inc. or its Licensors.

Extend was originally created by Bob Diamond

Chief architects for ExtendSim 7:

Bob Diamond, Steve Lamperti, Dave Krahl, Anthony Nastasi, and Cecile Damiron

Graphics, documentation, and production for ExtendSim 7:

Carla Sackett, Pat Diamond, and Kathi Hansen

ExtendSim 7 is dedicated to the memory of Peg Feasby – she would have been so proud.  
Special thanks to Lynn Scheurman, whose insight and dedication inspired us all.

# **Table of Contents**

## Table of Contents

..... 1

### About ExtendSim

**Preface**..... 1

**Introduction**..... 3

Why simulation is important .....4

Simulation with ExtendSim .....4

What ExtendSim can do .....4

    Modeling capabilities .....4

    Simulation architecture .....5

Levels of use .....5

About this User Guide.....6

Additional resources .....7

    Electronic documentation .....7

    ExtendSim Help.....8

    User forums .....8

    Support.....8

Model illustrations .....9

### Tutorial

**Running a Model** ..... 13

Opening the Reservoir model..... 14

Model basics..... 15

    Blocks .....15

    Connections.....16

Blocks used in the Reservoir model .....17

Running the Reservoir model.....18

Displaying the results on the Plotter .....18

Notebooks.....19

Making changes to the model.....19

    Adding and removing blocks.....20

    Changing dialog parameters .....20

Other modifications .....21

**Building a Model** ..... 23

Steps to create the Reservoir model .....24

Opening a new model worksheet .....25

Setting the simulation parameters.....25

Building the model.....25

    Basic steps .....25

    About libraries.....26

    Adding blocks to the model.....26

    Connecting blocks .....27

Working with block dialogs.....29



Rainfall source.....	29
Stream source.....	30
Combining the sources.....	31
Water in the reservoir.....	31
Displaying the results.....	32
Running the simulation.....	32
Additional ways of connecting blocks.....	32
Straight line connection.....	32
Multi-segment line connection.....	33
Named connection.....	33
Plotting against multiple axes.....	34
The final Reservoir model.....	36
Additional enhancements.....	36
Introduction to hierarchy.....	36
The ExtendSim Navigator.....	37
Cloning.....	38
Other modifications.....	39
Next steps.....	40
<b>Simulation Concepts.....</b>	<b>41</b>
Systems, models, and simulation.....	42
Systems.....	42
Models.....	42
Simulation.....	43
Modeling methodologies.....	43
Comparison of main modeling methodologies.....	44
Comparison table.....	45
Table of continuous, discrete event, and discrete rate differences.....	45
Other modeling approaches.....	47
Monte Carlo modeling.....	47
State/Action models.....	49
Agent-based models.....	51
The modeling process.....	54
Goals of modeling.....	54
The simulation process.....	55
Before you build a model.....	55
Refining models.....	56
Model verification.....	56
Model validation.....	57
Additional modeling terminology.....	57
Model parameters, variables, inputs, and outputs.....	57
Constant values and random variables.....	57
<b>Continuous Modeling</b>	
<b>Introduction.....</b>	<b>59</b>
How the Continuous module is organized.....	60
Blocks for building continuous models.....	60
Using the ExtendSim blocks.....	60

Building custom continuous blocks.....	60
Third-party libraries.....	61
Application areas.....	61
Next steps.....	62
<b>Tutorial .....</b>	<b>63</b>
Removing overflow from the Holding Tank.....	64
Setting the maximum capacity.....	64
Determining if there is too much water.....	65
Comparing contents to overflow limit.....	66
Calculating how much water to remove.....	66
Removing the overflow.....	67
Simplifying the model.....	68
Adding an Equation block.....	68
Specifying input variables.....	69
Specifying output variables.....	69
Entering the equation.....	69
Improving the accuracy of the model.....	69
Next steps.....	70
<b>Areas of Application .....</b>	<b>71</b>
Scientific.....	72
Predator/Prey.....	72
Drug Ingestion.....	73
Engineering.....	75
Noisy FM system.....	75
Business.....	76
Inventory Management.....	76
Social sciences.....	77
City Planning.....	77
Custom blocks.....	79
Planet Dance.....	79
Fish Pond.....	80
<b>Concepts, Tips, and Techniques.....</b>	<b>81</b>
Simulation timing.....	82
Delta time.....	82
Delta times other than 1.....	83
Determining which dt to use.....	83
Specifying dt or the number of steps.....	83
Feedback and delays.....	84
Feedback.....	85
Delays in feedback loops.....	85
Integration.....	85
Simulation order.....	86
Flow order.....	86
Left to right order.....	86
Custom order.....	86
Mixing block types.....	87

Connections to multiple inputs .....	87
Using plotters as inputs.....	87
Using a plot line as reference or standard .....	88
Uncluttering models .....	88

## Discrete Event Modeling

### **Introduction.....89**

About the Discrete Event module .....	90
How the Discrete Event module is organized .....	90
What the Introduction to the Discrete Event module covers .....	91
Discrete event systems and processes.....	91
Blocks for building discrete event models.....	92
Item library .....	92
Third-party libraries.....	92
Creating custom discrete event blocks .....	92
Terminology and architecture .....	93
Overview of a discrete event model .....	93
Layout of a discrete event model .....	93
Executive block .....	93
Items and informational values.....	93
Item properties.....	94
Events .....	94
Activities .....	95
Resources .....	95
Connectors.....	95
Closed and open systems.....	95
Types of item handling blocks .....	96
Application areas.....	96
Next steps .....	97

### **Tutorial.....99**

A basic discrete event model.....	100
About the model .....	100
Starting a model and setting simulation parameters.....	100
Start small .....	101
Adding complexity.....	103
Creating a second wash bay.....	103
Explicit routing .....	104
Requiring resources .....	105
Item attributes.....	106
Further exploration.....	108

### **Items, Properties, and Values.....109**

Blocks of interest.....	110
Item generating and removing.....	110
Item properties.....	110
Property-aware blocks .....	111
Item generation.....	111

Generating items at random intervals.....	111
Random intervals with dynamic parameters .....	112
Generating items according to a schedule.....	114
Item properties.....	115
Attributes.....	115
Priority.....	122
Quantities.....	124
Other item properties.....	126
<b>Queueing.....</b>	<b>127</b>
Blocks of interest.....	128
Queueing disciplines.....	128
Queue/server systems.....	129
M/M/1 queues.....	130
Priority queues.....	130
Queueing considerations.....	131
Blocking.....	131
Balking.....	131
Reneging.....	131
Jockeying.....	132
Sorting items using the Queue Equation block.....	133
Variables and rules.....	134
Least dynamic slack.....	135
Minimizing setup.....	136
Maximizing service levels.....	136
Combined rules.....	137
Matching items using the Queue Matching block.....	138
Queue Matching model.....	138
Other models that use the Queue Matching block.....	138
Advanced queue topics.....	139
Viewing and manipulating queue contents.....	139
Initializing a queue.....	139
Animating queue contents.....	140
<b>Routing.....</b>	<b>143</b>
Commonly used blocks.....	144
Blocks that route items.....	144
Blocks that affect the flow of items.....	144
Items from several sources.....	145
Select Item In dialog.....	146
Merging several item flows into one stream.....	146
Balancing multiple input lines.....	147
Throw Item and Catch Item blocks for merging item streams.....	148
Items going to several paths.....	149
Select Item Out dialog.....	149
Implicit routing.....	151
Simple routing.....	152
Scrap generation.....	153
Sequential ordering.....	154

Explicit ordering .....	155
Routing decisions based on properties .....	155
Conditional routing .....	157
Machines that can only process certain types of items .....	161
<b>Processing.....</b>	<b>163</b>
Commonly used blocks.....	164
Systems and processes.....	164
Processing in series.....	165
Processing in parallel.....	166
Parallel processing using one block.....	166
Simple parallel connections .....	166
Setting the processing time .....	167
Processing time for an Activity .....	167
Processing time for other activity blocks.....	168
Fixed processing time .....	168
Scheduled processing time.....	168
Random processing time .....	169
Custom processing time .....	170
Implied processing time .....	171
Cumulative processing time: time sharing .....	171
Adding setup time.....	172
Bringing an activity on-line.....	173
Scheduling activities.....	173
Controlling the flow of items to an activity.....	175
Fixed number of items .....	175
Fixed period of time .....	177
Interrupting processing.....	177
Preemption .....	178
Shutting down .....	179
Multitasking .....	183
Simulate Multitasking Activity model .....	184
Kanban system.....	185
Transportation and material handling.....	185
Travel time .....	185
Transport blocks.....	186
Convey Item blocks.....	188
How the length is calculated .....	189
Transportation models .....	190
<b>Batching and Unbatching.....</b>	<b>193</b>
Blocks of interest.....	194
Batching .....	194
Batch dialog .....	195
Simple batching .....	196
Batching by matching items .....	197
Batching a variable number of items.....	198
Properties when items are batched.....	199
Delaying kits .....	201

Unbatching .....	201
Simple unbatching .....	202
Variable batching and unbatching .....	203
Properties when items are unbatched.....	204
Preserving the items used to create a batch .....	204
Both blocks choose to preserve uniqueness.....	204
Either block chooses to preserve uniqueness.....	205
Additional models .....	205
<b>Resources and Shifts.....</b>	<b>207</b>
Blocks of interest .....	208
Resource pool blocks.....	208
Other resource blocks.....	209
Modeling resources.....	209
How to model resources .....	209
Resource Pool method.....	209
Resource Item method .....	213
Other methods for modeling resources.....	215
Closed and open systems .....	216
Scheduling resources .....	216
Scheduling resource pools and resource items.....	216
Scheduling resource items .....	217
The Shift block .....	218
Shift types and what they control .....	218
Status connectors .....	219
Shift models .....	219
<b>Activity-Based Costing.....</b>	<b>223</b>
Blocks of interest .....	224
Modeling with activity-based costing.....	225
Item types .....	225
Defining costs and cost rates .....	226
Combining resources with cost accumulators .....	229
Combining cost accumulators.....	231
Working with cost data .....	231
How ExtendSim tracks costs .....	234
Setting the _cost and _rate attributes.....	234
Combining resources with cost accumulators .....	234
Calculating costs .....	235
Combining multiple cost accumulators.....	236
<b>Statistics and Model Metrics .....</b>	<b>237</b>
Commonly used blocks .....	238
Gathering statistics .....	238
Clearing statistics .....	239
Clearing Statistics model .....	239
Using the History block to get item information .....	239
History model.....	240
Verifying Information model.....	240

Accumulating data .....	241
Non-Processing model .....	241
Processing model.....	241
Time weighted versus observed statistics .....	242
Time Weighted Statistics model .....	242
Timing the flow of items in a portion of the model.....	243
<b>Tips and Techniques .....</b>	<b>245</b>
Moving items through the simulation .....	246
How items move through the simulation .....	246
Connections to multiple item input connectors .....	248
An item's travel time.....	248
Using scaling for large numbers of items .....	249
Preprocessing.....	249
Restricting items in a system .....	249
Connecting to the select connector .....	250
Continuous blocks in discrete event models.....	250
Setting time-based parameters using connectors .....	251
Varying a distribution's arguments.....	251
Using the Holding Tank block to accumulate values .....	252
Cycle timing .....	254
Using the Timing attribute feature .....	254
Using a Set or Equation(I) and Information blocks .....	255
Item library blocks .....	255
Executive block .....	255
Block types.....	256
Common connectors on discrete event blocks.....	257
Event scheduling.....	258
Event calendars .....	259
Zero time events.....	259
Event Scheduling model.....	259
Messaging in discrete event models .....	260
Block messages .....	260

## Discrete Rate Modeling

<b>Introduction .....</b>	<b>265</b>
What this chapter covers .....	266
Discrete rate application areas .....	266
Simulating discrete rate systems .....	267
Comparison to discrete event and continuous modeling .....	267
Discrete rate models .....	268
Blocks for building discrete rate models .....	269
Rate library .....	269
Creating custom discrete rate blocks.....	269
Terminology and architecture .....	269
LP technology .....	269
Layout of a discrete rate model.....	270
Executive block .....	270

Connectors and connections .....	270
Units and unit groups .....	271
Rates .....	271
How the Discrete Rate module is organized .....	272
<b>Tutorial for Discrete Rate Systems .....</b>	<b>273</b>
A basic discrete rate model .....	274
About the model .....	274
Starting a model and setting simulation parameters.....	275
Start small .....	275
Add a dynamic constraint.....	278
Add a fruit processing line.....	279
Add maintenance .....	280
Change the flow unit to containers for the filling process .....	281
Cool the mixture.....	282
Package the containers.....	283
Add a palletizing area .....	284
Add a second palletizing area.....	286
Further exploration .....	287
<b>Sources, Storage, and Units .....</b>	<b>289</b>
Blocks of interest .....	290
Residence blocks for holding flow .....	290
Changing the flow unit group .....	291
Capacity.....	291
Full and not-full.....	291
Tank block's capacity .....	291
Interchange block's capacity .....	292
Convey Flow block's capacity .....	293
Setting an initial contents.....	293
Empty and not-empty .....	294
Tank initialization .....	294
Interchange initialization.....	294
Convey Flow initialization .....	295
Indicators.....	295
Setting indicators .....	296
Getting information about levels.....	296
Units and unit groups .....	297
Definitions.....	297
Declaring and selecting flow units.....	298
Defining block units .....	299
Time units .....	300
Changing the unit group.....	300
Change Units block.....	300
<b>Rates, Constraints, and Movement.....</b>	<b>301</b>
Blocks of interest.....	302
Rates, rate sections, and the LP area .....	303
Types of rates .....	303



Rate sections .....	305
Rate precision.....	306
LP area .....	306
Flow rules .....	306
Critical and relational constraints .....	307
Defining a critical constraint.....	308
Valve .....	309
Tank and Interchange.....	310
Convey Flow .....	311
Merge and Diverge.....	311
Meeting the critical constraint requirement.....	312
Valve or Convey Flow .....	312
Tank or Interchange .....	312
Merge or Diverge blocks .....	313
Comprehensive example .....	315
Rate sections .....	315
Critical constraints .....	315
Relational constraint .....	315
Simulation's impact on the effective rates .....	315
<b>Merging, Diverging, and Routing Flow .....</b>	<b>317</b>
Blocks of interest.....	318
Merging and diverging flow .....	318
Mode table.....	319
Select mode.....	319
Batch/Unbatch mode.....	321
Proportional mode .....	321
Priority mode .....	322
Distributional mode.....	324
Sensing mode.....	325
Neutral mode.....	326
Features of the Merge and Diverge blocks .....	327
Bias Order – resolving competing requests for flow .....	327
Internal throw and catch .....	328
Changing decision rules dynamically.....	328
Throwing flow and catching flow remotely.....	329
Creating a throw/catch connection.....	330
Filter options to facilitate throw/catch connections .....	330
Examples of throw and catch connections .....	331
<b>Delaying Flow .....</b>	<b>333</b>
Blocks of interest.....	334
Controlling a Valve's maximum rate.....	334
Using the Flow Control tab.....	335
Observing the maximum rate for a goal .....	335
Setting a Valve's quantity goal.....	335
Setting a Valve's duration goal .....	338
Setting hysteresis in a Valve .....	341
Delaying flow with the Shift block.....	342

Adding a Shift to a model.....	342
Convey Flow block.....	342
Dialog settings .....	343
Constraining rates .....	344
Convey Flow information .....	344
When to avoid using the Convey Flow block .....	346
<b>Mixing Flow and Items .....</b>	<b>347</b>
Controlling flow with items and items with flow.....	348
Items controlling flow .....	348
Flow controlling items.....	349
Flow controlling items and items controlling flow.....	350
Step The Flow Process model .....	350
Using the Interchange block to mix items with flow .....	352
Behavioral rules .....	352
The flow connector configuration .....	353
Item release conditions .....	353
Interchange modes .....	354
<b>Miscellaneous.....</b>	<b>359</b>
Precision.....	360
Biasing flow.....	360
Bias order .....	361
Bias block.....	361
Merge and Diverge blocks .....	362
Global and advanced options in the Executive.....	364
Global options .....	364
Advanced options.....	366
Common connectors on discrete rate blocks.....	368
Animation.....	370
Tank.....	370
Interchange .....	371
Valve .....	371
Sensor .....	373
Convey Flow.....	373
<b>Advanced Topics.....</b>	<b>375</b>
What this chapter covers .....	376
LP technology .....	376
Overview.....	376
The LP area.....	377
The sequence of events.....	377
Types of information provided to the Executive .....	379
The LP calculation .....	382
Upstream supply and downstream demand .....	382
Definition .....	383
Requirements for the supply/demand calculation .....	383
Cautions when using potential rates .....	383
Messaging in discrete rate models.....	386

Block messages .....	387
<b>3D Animation</b>	
<b>Introduction to E3D.....</b>	<b>389</b>
What this chapter covers .....	390
Blocks and objects for 3D animation .....	390
Item library blocks .....	390
Animation library.....	391
Custom 3D objects and blocks.....	391
Overview .....	391
Features.....	391
Controlling the E3D environment .....	393
Prerequisites.....	393
Software and hardware .....	393
Preparation.....	393
How the E3D module is organized .....	394
<b>Tutorial I .....</b>	<b>395</b>
The E3D environment .....	396
Opening the E3D window.....	396
Exploring the E3D window.....	396
Changing the associated model .....	398
Navigating within the E3D window.....	398
Manipulating the E3D window.....	399
3D animation modes .....	400
Mode descriptions.....	400
QuickView versus Concurrent or Buffered.....	400
Running a model with 3D animation .....	400
Opening the model .....	401
Running the model with 3D animation .....	402
Next step .....	403
<b>Tutorial II.....</b>	<b>405</b>
Adding 3D behavior to an existing model .....	406
The goal.....	406
Open the starter model .....	407
Cause objects to move simultaneously.....	409
Create objects to represent items .....	409
Create objects to represent blocks.....	410
Enhancing the model.....	411
Add scenery.....	411
Add a 3D Controller block.....	411
Launch with the E3D window .....	412
Some things to notice .....	412
Internal animation .....	412
Rotation of 3D objects .....	412
Mounting objects.....	412
Moving blocks linked to objects .....	413

Conveyor .....	413
Item length .....	413
Conveyor capacity.....	414
<b>Tutorial III .....</b>	<b>415</b>
Animating a bank line .....	416
The goal.....	416
Open the starter model .....	416
Animate the model in 3D .....	418
Unmount the Activity blocks .....	419
Add Transport blocks .....	419
Animating the travel time.....	419
What the model needs.....	419
Walking and waiting in a line.....	420
Leaving the bank.....	420
Minimizing the icons of the existing Transport blocks.....	421
The model so far .....	421
Block positions to determine a path's length .....	421
Setting the speed and determining the distance .....	422
Mounting objects .....	423
Steps for mounting an object.....	423
Create the object .....	423
Create an attribute .....	424
Mount the object on the item .....	424
Create a hierarchical block .....	424
Unlinking objects from blocks.....	425
Unlinking positions.....	425
Creating custom pathways.....	426
Use the correct Transport behavior .....	426
Creating paths.....	427
Create a new environment file.....	427
Create a path object .....	427
Create path markers .....	428
Select the path.....	428
Repeat the process for another path.....	429
Enhancing the model .....	429
<b>Environment Files &amp; E3D Editors .....</b>	<b>431</b>
Environment files.....	432
Modifying the environment .....	432
The E3D Editor .....	433
Learning about the E3D Editor.....	433
E3D Editor modes .....	435
Mode categories .....	435
World modes.....	436
Terrain modes .....	438
Editor menus and commands.....	440
<b>3D Objects.....</b>	<b>441</b>

3D objects .....	442
Types of objects .....	442
Object properties .....	443
Actions .....	443
Creating objects .....	444
Create an object that represents a block .....	445
Create an object that represents an item or other moveable entity .....	445
Create a 3D object as scenery .....	446
Create an environmental effect .....	448
Deleting objects .....	449
Changing object properties .....	449
Changing skins .....	450
Move an object .....	451
Show or hide objects .....	453
Rotate an object .....	455
Scale an object .....	457
Saving changes .....	459
Saving an environment file .....	459
WayPoints .....	459
Creating a waypoint .....	459
Choosing a waypoint as a destination .....	460
Mounting objects .....	460
Item object on block object .....	461
Object on item object .....	461
Scenery object on scenery object .....	461
Other object information .....	462
Collision .....	462
Gravity, friction, and momentum .....	462
Sound .....	462
Object ID .....	463
BlockNumber .....	463
GroupTag and UserTag .....	463
<b>Movement, Paths, and Terrains.....</b>	<b>465</b>
Traveling time .....	466
Setting travel time in a Transport or Convey Flow block .....	466
Creating paths .....	467
Terrains .....	470
Modifying the terrain .....	471
<b>Tips and Reference .....</b>	<b>473</b>
Tips .....	474
Using an Equation block to call E3D functions .....	474
Hierarchical blocks and 3D animation .....	474
Items stack on top of each other .....	474
Performance Considerations .....	475
E3D commands, options, and settings .....	475
Opening the E3D window .....	475
3D tab in Options dialog .....	476

3D Animation tab of Simulation Setup dialog .....	477
Dialog tabs for animation.....	478
Item Animation tab.....	478
Block Animation tab .....	480
Transport Animation tab.....	481
Animation 2D-3D blocks.....	482
3D Controller block.....	482
3D Scenery block.....	483
3D Text block .....	483
Animate 3D block.....	483
E3D Editor menu commands .....	484
File.....	484
Edit.....	485
Camera .....	485
Window.....	485
Lighting Tools.....	485
World.....	485
Action .....	485
Brush .....	485

## How To

<b>Libraries and Blocks.....</b>	<b>487</b>
The ExtendSim libraries .....	488
Animation 2D-3D library .....	488
Electronics library .....	488
Item library (not available in ExtendSim CP) .....	488
Plotter library.....	489
Rate library (not available in ExtendSim CP or ExtendSim OR) .....	489
Utilities library .....	489
Value library.....	489
Example Libraries folder.....	489
Legacy folder.....	489
Using libraries .....	490
Opening a library .....	490
Closing a library.....	491
Searching for libraries and blocks .....	491
Library windows .....	493
Creating and maintaining libraries .....	493
Creating a new library.....	494
Saving and compiling libraries.....	494
Substituting one library for another.....	495
Arranging blocks in libraries.....	495
Protecting the code of library blocks.....	495
Converting libraries to RunTime format .....	496
Working with blocks .....	496
Customizing block icons .....	496
Icon views .....	496
Connectors.....	497

Connecting to different connector types.....	500
Dialogs.....	501
Animating blocks .....	502
Hierarchical blocks .....	502
Managing blocks.....	502
Copying blocks .....	502
Changing a block's name.....	502
Removing blocks.....	502
Corrupted blocks .....	502
<b>Creating a Custom User Interface.....</b>	<b>503</b>
Cloning .....	504
How to clone a dialog item .....	504
Using cloned items.....	505
Unlinked clones .....	506
Centralizing data in a database.....	506
Hierarchy.....	506
Creating a dashboard interface .....	506
Buttons .....	507
Popup menus .....	508
On/Off Switch.....	508
Additional blocks to control model execution .....	508
Notebooks .....	508
Controls .....	509
Slider.....	509
Switch.....	510
Meter.....	510
Interacting with the model user .....	510
Notify block.....	511
Equation blocks .....	512
Additional interactive features if you program .....	513
External applications as an interface.....	514
Documenting models .....	514
Text and graphics .....	514
Help block .....	514
<b>Model Execution.....</b>	<b>515</b>
Simulation setup.....	516
Setup tab.....	517
Continuous tab .....	518
Random Numbers tab.....	519
3D Animation tab.....	521
Comments tab .....	521
Running a model .....	522
Menu commands and toolbar buttons.....	522
Running a model multiple times .....	522
Stepping through a model .....	522
Other points when running models.....	523
Status bar .....	524

Blocks that control or monitor simulation runs .....	525
Saving intermediate results .....	525
Timing .....	526
Continuous simulation timing .....	526
Discrete event simulation timing .....	526
Simulation order ( <i>continuous models</i> ) .....	526
Time units .....	526
Global time unit .....	526
Local time unit .....	527
Calendar dates .....	528
Time unit conversions (non-Calendar dates) .....	529
Other Units .....	529
Flow units .....	529
Length .....	529
Length and number of runs .....	530
Terminating systems .....	530
Non-terminating systems .....	530
Determining the length and number of runs .....	531
Speeding up a simulation .....	531
Displaying data or movement .....	532
Inefficient settings or block code .....	532
Other factors that affect simulation speed .....	533
Slowing down simulations .....	533
Working with multiple models .....	533
How ExtendSim passes messages in models .....	533
Application messages .....	534
Block messages .....	535
<b>Presentation .....</b>	<b>537</b>
Working with text .....	538
Entering text .....	538
Moving and copying text .....	538
Drag and drop text .....	539
Formatting text .....	539
Navigator .....	539
Hierarchy .....	540
Uses for hierarchy .....	540
Hierarchical blocks .....	541
Making a selection into a hierarchical block .....	542
Building a new hierarchical block .....	543
Saving hierarchical blocks .....	547
Modifying hierarchical blocks .....	548
Animation .....	551
Blocks with built-in animation .....	551
Blocks for customized animation .....	553
Animation functions .....	556
Animation pictures .....	556
Displaying messages on a block's icon .....	557
ExtendSim databases .....	557



Connections .....	557
Connection lines .....	557
Named connections .....	560
Model appearance .....	560
Showing and hiding connections and connectors .....	560
Changing model styles .....	561
Graphic shapes, tools, and commands.....	561
Drawing objects in the Shapes menu.....	561
Shuffling graphics .....	561
Modifying objects .....	562
Patterns and colors.....	562
Working with pictures .....	562
<b>Analysis .....</b>	<b>563</b>
Blocks that calculate statistics.....	564
Statistics .....	564
Clear Statistics.....	566
Mean & Variance .....	566
Information.....	567
Cost Stats.....	567
Confidence intervals .....	567
Sensitivity analysis .....	568
Overview.....	568
Steps for using sensitivity analysis.....	568
Specifying the sensitivity method .....	570
Turning sensitivity on and off.....	570
Reporting the results .....	571
Multi-dimensional scenarios.....	571
Optimization .....	572
How optimization works.....	573
Steps for using optimization .....	573
Optimization tutorial.....	573
Adding constraints .....	580
Using the Optimizer block .....	583
Stat::Fit (Windows only).....	586
Tutorial .....	587
Plotters .....	588
Plot and data panes .....	588
Plotter tools.....	589
Plotter dialogs .....	592
Types of plotters.....	593
Copying plotted information .....	596
Clearing plotted information.....	596
Reports .....	596
Types of reports.....	596
Generating reports .....	597
Steps for reporting.....	597
Reporting example .....	597

<b>Math and Statistical Distributions .....</b>	<b>599</b>
Blocks that represent functions .....	600
Other options .....	601
Equation-based blocks .....	601
Overview .....	602
Equation components .....	602
Random numbers .....	604
Random number generators .....	605
Random seeds .....	605
Resetting random numbers for consecutive runs .....	606
Probability distributions .....	606
Characteristics of distributions .....	606
Choosing a distribution .....	606
Distribution fitting .....	607
ExtendSim distributions .....	607
Integration vs. summation in the Holding Tank block .....	610
<b>Debugging Tools .....</b>	<b>613</b>
Debugging hints .....	614
Verifying results as you build a model .....	615
Connector information .....	615
Cloning dialog items .....	615
Blocks for debugging .....	615
Measuring performance to debug models .....	616
Find command .....	617
The Source Code Debugger .....	618
Dotted lines for unconnected connections .....	618
Animation features for debugging .....	618
Animating the model .....	618
Animating item properties (discrete event models only) .....	618
Notebook .....	618
Stepping through the simulation .....	619
Show Simulation Order command .....	619
Slow simulation speed .....	620
Model reporting .....	620
Model tracing .....	620
Generating traces .....	620
Tracing example .....	621
<b>Data Management and Exchange .....</b>	<b>623</b>
User interfaces for data exchange .....	624
Copy/Paste .....	625
Importing and exporting data .....	626
Read and Write .....	628
Dynamic linking to internal data structures .....	629
DDE links (Windows only) .....	636
Internal data storage and management methods .....	638
ExtendSim databases for internal data storage .....	638

How this section is organized .....	639
Advantages of using internal databases .....	639
Creating and interacting with internal databases .....	640
How to create an ExtendSim database .....	640
Establishing Parent/Child relationships .....	643
Linking a database to data .....	645
Database management .....	646
Database dialogs and popup menus.....	648
Excel Add-In for ExtendSim databases .....	650
Monte Carlo model.....	651
Other internal data storage and management methods.....	651
Global arrays .....	652
Dynamic arrays .....	654
Embedding an object (Windows only) .....	655
Linked lists.....	657
Exchanging data with external applications.....	657
Spreadsheets .....	658
External databases .....	658
Blocks for data management and exchange .....	659
Read and Write blocks .....	659
Data access blocks .....	660
Other blocks for modelers .....	661
Blocks for developers.....	661
Data source indexing and organization .....	661
Transferring data between a data table and a spreadsheet .....	662
Transferring data between a spreadsheet and a database.....	662
Communicating with external devices.....	662
Technologies for communication .....	663
Text files.....	663
ActiveX/COM/OLE (Windows only) .....	665
DDE (Windows only).....	666
ODBC/SQL .....	667
FTP .....	667
DLLs and Shared Libraries.....	668
Mailslots (Windows only) .....	668
<b>Miscellaneous .....</b>	<b>669</b>
Navigator.....	670
Opening the Navigator .....	670
Model Navigator mode .....	671
Database List mode.....	671
Library Window mode.....	671
Printing .....	672
Selecting what to print .....	672
The Print command.....	672
Printing and Print Setup hints.....	674
Copy/Paste and Duplicate commands.....	674
Copying within ExtendSim.....	674
Copying from ExtendSim to other applications.....	675

Copying from other applications to ExtendSim.....	675
Tool tips.....	676
Changing parameters dynamically.....	676
Methods.....	676
Sharing model files.....	677
Locking the model.....	677
The ExtendSim LT-RunTime version.....	678

## Reference

<b>Menu Commands and Toolbars .....</b>	<b>679</b>
ExtendSim menu (Mac OS only) .....	680
File menu.....	680
New Model.....	680
New Text File.....	680
Open.....	680
Close.....	680
Revert Model/Revert Text File.....	681
Save Model and Save Model As.....	681
Save Text File and Save Text File As.....	681
Update Launch Control (Windows only).....	681
Import Data Table .....	681
Export Data Table.....	681
Import DXF File (Windows only).....	682
Show Page Breaks.....	682
Print Setup (Windows) and Page Setup (Mac OS) .....	682
Print.....	682
Network License (Windows only; network license only).....	683
Properties .....	683
Five most recent models or text files.....	684
Exit/Quit .....	684
Edit menu.....	684
Undo.....	684
Cut .....	684
Copy.....	684
Paste.....	684
Clear .....	684
Delete Selected Records.....	685
Select All.....	685
Duplicate .....	685
Find .....	685
Find Again .....	686
Replace.....	686
Replace, Find Again .....	686
Replace All.....	686
Enter Selection.....	686
Create/Edit Dynamic Link.....	686
Open Dynamic Linked Blocks.....	687
Sensitize Parameter.....	687
Open Sensitized Blocks .....	687

Paste DDE Link (Windows only).....	687
Delete DDE Link (Windows only) .....	687
Show DDE Links (Windows only).....	688
Refresh DDE Links (Windows only).....	688
Insert Object (Windows only).....	688
Design Mode (Windows only) .....	688
Object (Windows only).....	688
Show Clipboard .....	688
Options.....	688
Text menu .....	695
Library menu .....	695
Open Library .....	695
Close Library .....	695
New Library.....	695
Tools .....	695
List of libraries .....	697
Model menu .....	698
Make Selection Hierarchical.....	698
New Hierarchical Block .....	698
Open Hierarchical Block Structure.....	698
Connection Lines.....	698
Show Named Connections.....	698
Hide Connections.....	698
Hide Connectors.....	698
Controls.....	698
Align .....	698
Rotate Shape .....	699
Flip Horizontally/Flip Vertically.....	699
Border Thickness .....	699
Shape Fill/Border .....	699
Change Model Style.....	699
Lock Model.....	699
Use Grid .....	699
Show Block Labels .....	699
Show Block Numbers.....	699
Show Simulation Order .....	700
Set Simulation Order.....	700
Database menu .....	700
New Database .....	700
Import New Database .....	700
Export Database.....	701
Rename Database.....	701
New Table.....	701
Import Tables.....	701
Export Selected Tables.....	701
Rename Table .....	701
New Tab .....	702
Rename or Delete Tab.....	702
Clone Selected Tables to Tab.....	702

Append New Field .....	703
Insert New Field.....	703
Append New Records.....	703
Insert New Records .....	704
Develop menu.....	704
New Block .....	704
Open Block Structure .....	704
Rename Block.....	704
Set Block Category.....	704
Compile Block.....	705
New Dialog Item.....	706
New Tab .....	706
Rename or Delete Tab.....	706
Move Selected Items to Tab.....	706
New Include File.....	706
Open Include File .....	706
Delete Include File .....	707
Shift Selected Code Left.....	707
Shift Selected Code Right.....	707
Go To Line .....	707
Go To Function/Message Handler .....	707
Match Braces.....	707
Match IFDEF/ENDIF.....	707
Set Breakpoints .....	707
Open Breakpoints Window.....	707
Open Debugger Window.....	707
Continue.....	708
Step Over.....	708
Step Into .....	708
Step Out .....	708
Run menu .....	708
Run Simulation.....	708
Continue Simulation.....	708
Run Optimization.....	708
Simulation Setup.....	708
Prioritize Front Model.....	709
Use Sensitivity Analysis .....	709
Show 2D Animation .....	709
Show 3D Animation .....	709
Show Movies (Mac OS only).....	709
Launch Proof (Windows only) .....	709
Launch StatFit (Windows only) .....	710
Generate Report.....	710
Report Type .....	710
Add Selected To Report.....	710
Add All To Report.....	710
Remove Selected From Report .....	710
Remove All From Report .....	710
Show Reporting Blocks .....	710

Stop .....	710
Pause .....	710
Step .....	710
Resume .....	711
Debugging .....	711
Window menu .....	712
Notebook .....	712
Navigator .....	713
Database List .....	713
Calendar .....	713
E3D Window .....	713
Help menu .....	713
ExtendSim Help .....	713
Support Resource Center .....	713
Downloads and Updates .....	713
User Forum .....	713
What's New .....	713
ExtendSim Product Line .....	713
Imagine That Inc. Online .....	714
About ExtendSim (Windows only) .....	714
Toolbar buttons .....	714
ExtendSim database tool bars .....	714
<b>Value Library Blocks .....</b>	<b>715</b>
Submenus .....	716
Data Access .....	716
Holding .....	717
Inputs .....	718
Math .....	718
Optimization .....	719
Outputs .....	720
Routing .....	720
Statistics .....	721
<b>Item Library Blocks .....</b>	<b>723</b>
Submenus .....	724
Activity .....	724
Batching .....	725
Data access .....	725
Information .....	726
Properties .....	726
Queues .....	727
Resources .....	727
Routing .....	728
Executive .....	729
<b>Rate Library Blocks .....</b>	<b>731</b>
Block descriptions .....	732

<b>Utilities Library Blocks .....</b>	<b>735</b>
Submenus .....	736
Developer Tools .....	736
Discrete Event Tools.....	736
Information.....	737
Math.....	737
Model Control.....	737
Time .....	738
<b>Upper Limits.....</b>	<b>739</b>
<b>Cross-Platform Considerations .....</b>	<b>741</b>
Libraries .....	742
Models .....	742
Menu and keyboard equivalents .....	742
Transferring files between operating systems.....	743
File name adjustments.....	743
Physically transferring files .....	743
File conversion .....	743
<b>Index</b>	



# About ExtendSim

## Preface

ExtendSim's architect  
talks about simulation

*“What we experience of nature is in models,  
and all of nature's models are so beautiful.”*  
— R. Buckminster Fuller

Dedicated to the pleasure of finding things out

Simulation is defined as the act of imitation. Even a word processor simulates pen and paper, but how do you get the computer to behave like the stock market, or an electronic circuit, or even a car, and how can you communicate this power to the user? My search for the answer began in the early days of the space race.

I was attending the Polytechnic Institute of Brooklyn when the head of the Electronics Engineering department told us that a new department was being formed... a combination of mathematics, computers, physics and engineering. Being into math, and curious about the large IBM mainframe lurking down the hall, I immediately joined and made a constant pest of myself at the computer center.

The bug bit hard, I guess, and I began to realize that I could use computers to duplicate the laboratory experiments in class so well, that I never really did them, I just simulated them on the computer. NASA then asked if I could develop a simulation of their new liquid fuel booster for something called Project Apollo. I came up with the Rocket-Drop simulation, a monstrously large program that only had one function: follow the path of a single droplet of fuel, from the shower heads (as the fuel sprayers at the top of the engine were called) to the rocket engine exhaust, via subsonic, supersonic, and hypersonic flow.

It hit me then that simulation was inaccessible, except to the select few who had the resources to put together an entire system dedicated to one function. A generalized simulation application would be a great and useful thing, if one could find the computer that was both powerful enough and widespread enough to support it. This was 1965, and Seymour Cray was still building his superfast (at the time!) computers by hand and graphic user interfaces were still decades in the future.

When I saw the graphical user interfaces (GUIs) on the Mac OS and Windows machines, I realized that I could use these tools to fulfill that long awaited dream. ExtendSim is built upon those roots. Imagine That Inc. was founded in 1987 to develop and market Extend and its successor ExtendSim, the first simulation applications allowing users of any discipline to use simulation and to develop their own libraries of customized simulation tools.

Imagine That! is dedicated to bringing the art, science, and fun of simulation to the desktop, in a form digestible and accessible by everyone. ExtendSim is the first user-extendible simulation package that meets those expectations.

Bob Diamond  
President

*“You see? That’s why scientists persist in their investigations, why we struggle so desperately for every bit of knowledge, stay up nights seeking the answer to a problem, climb the steepest obstacles to the next fragment of understanding, to finally reach that joyous moment of the kick in the discovery, which is part of the pleasure of finding things out.”* attributed to Richard P. Feynman

# About ExtendSim

## Introduction

Learn about ExtendSim's capabilities  
and how to get started using them

*“Begin at the beginning,’ the King said, gravely,  
‘and go ‘til you come to the end; then stop.’”  
— Lewis Carroll*

ExtendSim is a powerful, leading edge simulation tool. Using ExtendSim, you can develop dynamic models of real-life processes in a wide variety of fields. Use ExtendSim to create models from building blocks, explore the processes involved, and see how they relate. Then change assumptions to arrive at an optimum solution. ExtendSim and your imagination are all you need to create professional models that meet your business, industrial, and academic needs.

## Why simulation is important

Simulation involves designing a model of a system and carrying out experiments on it as it progresses through time. Models enable you to see how a real-world activity will perform under different conditions and test various hypotheses at a fraction of the cost of performing the actual activity.

One of the principal benefits of a model is that you can begin with a simple approximation of a process and gradually refine the model as your understanding of the process improves. This “step-wise refinement” enables you to achieve good approximations of very complex problems surprisingly quickly. As you add refinements, the model more closely imitates the real-life process.

## Simulation with ExtendSim

ExtendSim is an easy-to-use, yet extremely powerful, tool for simulating processes. It helps you understand complex systems and produce better results faster. With ExtendSim you can:

- Predict the course and results of certain actions
- Gain insight and stimulate creative thinking
- Visualize your processes logically or in a virtual environment
- Identify problem areas before implementation
- Explore the potential effects of modifications
- Confirm that all variables are known
- Optimize your operations
- Evaluate ideas and identify inefficiencies
- Understand why observed events occur
- Communicate the integrity and feasibility of your plans

## What ExtendSim can do

ExtendSim allows you to simulate any system or process by creating a logical representation in an easy-to-use format.

### Modeling capabilities

With ExtendSim, you get powerful modeling constructs, including:

- A full set of building blocks that allow you to build models rapidly
- A customizable graphical interface that depicts the relationships in the modeled system
- Unlimited hierarchical decomposition making enterprise-wide models easy to build and understand
- Dialogs, Notebooks, and an integrated database for changing model values, so you can quickly try out assumptions and interface with your model dynamically
- 2D and realistic 3D animation of the model for enhanced presentation

- A full-featured authoring environment for building user-friendly front ends that simplify model interaction and enhance communication
- The ability to adjust settings dynamically, while the simulation is running
- An equation editor for creating custom-compiled equations
- The ability to create new blocks with custom dialogs and icons
- Complete scalability since model size is limited only by the limits of your system
- Evolutionary optimization, Monte Carlo, batch-mode, and sensitivity analysis
- Customizable reports and plotters for presentation and in-depth analysis
- Activity-based costing capabilities for analyzing cost contributors
- Full connectivity and interactivity with other programs and platforms

### Simulation architecture

A robust architecture adds advanced features to make it the most scalable simulation system available:

- **Multi-purpose simulation.** ExtendSim is a multi-domain environment so you can dynamically model continuous, discrete event, discrete rate, agent-based, linear, non-linear, and mixed-mode systems.
- **Library based.** The blocks you build can be saved in libraries and easily reused in other models.
- **Integrated compiled programming language and dialog editor, optimized for simulation.** Modify ExtendSim's blocks or build your own for specialized applications.
- **Scripting support.** Build and run models remotely, either from an ExtendSim block or from another application.
- **Integrated support for other programming languages.** Use ExtendSim's built-in APIs to access code created in Delphi, C++ Builder, Visual Basic, Visual C++, etc.
- **Over 1000 functions.** Directly access functions for integration, statistics, queueing, animation, IEEE math, matrix, sounds, arrays, FFT, debugging, DLLs, string and bit manipulation, I/O, and so on; you can also define your own functions.
- **Message sending.** Blocks can send messages to other blocks interactively for subprocesses.
- **Sophisticated data-passing capabilities.** Pass values, arrays, or structures composed of arrays.
- **Full support for a wide range of data types and structures.** Arrays, linked-lists, and integers, real, and string data types are built in.
- **Integrated data linking.** Connect block dialog data to internal databases.

To see the new features added in this release, go to the ExtendSim web site or choose the menu command Help > What's New.


### Levels of use

You can use ExtendSim on many levels:

- Run pre-assembled models and explore alternatives by changing the data. If you work in a group environment, one or more authors can create models for others to run for experimentation. The author can also build a custom front end to facilitate user interaction with the model. The LT-RunTime version of ExtendSim allows non-modelers to run pre-assembled models, change data,

and obtain results. For more information, see “The ExtendSim LT-RunTime version” on page 678.

- Assemble your own models from the blocks that come with ExtendSim. ExtendSim is shipped with libraries of blocks to handle most modeling needs. To assemble a model, pull blocks from libraries and link connectors on the blocks. You can also assemble your own hierarchical blocks of subsystems and save them in libraries. This saves starting from scratch when you’re building a model of a process that has elements in common with a previous model.
- Use the integrated development environment to create new blocks that conform to the ExtendSim modeling architecture. The development environment is optimized for simulation and allows you to create blocks with custom code, dialogs, and icons and use them in your models just as you would other ExtendSim blocks. You can also modify the blocks that come with ExtendSim to work with your specific needs.
- Develop your own modeling architecture, conventions, and features. With the ExtendSim development environment, you can create a custom set of blocks with unique interfaces, communication protocols, and behaviors. This new architecture can be continuous, discrete event, discrete rate, agent-based, or an entirely new type of simulation.
- Automate your model building using the scripting functions to build wizards, or by using ActiveX/COM. You can use ActiveX/COM or block-based wizards to cause models to be automatically created or modified. Models can also be programmatically created from a user input form or data file. This allows the modeling environment to be utilized indirectly by end-users who have little or no simulation experience.

 An ExtendSim ASP license is required to distribute the functionality of ExtendSim to other users or to provide internet or intranet access to ExtendSim or to its functionality. Contact Imagine That, Inc. for more information.

As you read this manual, you will see how ExtendSim caters to the needs of users at all levels.

## About this User Guide

This manual is a general-purpose tutorial and reference for using ExtendSim. Every effort has been made to present sample models that you can easily understand, whatever field you are in, so you can quickly learn how to use this powerful tool. While you may find that your subject area is not represented in the manual, or that the sample models reflect some disciplines which are unfamiliar to you, remember that their purpose is to teach you how to use ExtendSim. What you model, and how you model it, are determined mostly by your knowledge and expertise in your subject area.

This manual is divided into several modules:


- **About ExtendSim**

This module includes the preface and introduction to ExtendSim.

- **Tutorial**

The Tutorial module starts on page 14. The first two chapters use a simple model to explain the most important concepts of modeling with ExtendSim. It is highly recommended that you build and run the models as you read the tutorials.

The third chapter in this module provides a more in-depth discussion of simulation technologies, general modeling concepts, and terminology.

 The first two Tutorial chapters use a very simple model to make it easier to learn basic modeling techniques. In real life, ExtendSim is used to simulate complex processes, as illustrated by the models shown at the end of this chapter.

- **Continuous Modeling**

An introduction to continuous modeling concepts, a tutorial, example areas of application, and additional modeling tips not covered elsewhere in the manual. The Continuous module starts on page 60.

- **Discrete Event Modeling**

An introduction to event-based modeling of discrete items, a tutorial, and several chapters with tips and concepts specific to discrete event modeling. The Discrete Event module starts on page 90.

- **Discrete Rate Modeling**

An introduction to rate-based modeling, a tutorial, several modeling tips and concepts chapters, and an advanced topics chapter. This module starts on page 266.

- **E3D Animation**

Shows how to use 3D animation to enhance the simulation experience. This module starts on page 389.

- **How To**

The chapters in this module provide additional concepts and techniques for developing and running models, such as creating a custom user interface, statistical analysis, data management and transfer, and so forth. The How To module starts on page 488.

- **Appendix**

The Appendix module, which starts on page 680, has several reference chapters, including a description of each of the menu commands and toolbars, lists and descriptions of blocks in the main libraries, and upper-limit values for a variety of parameters.

 A separate **Developer Reference** is also available for advanced users who want to create libraries of blocks for specific purposes.

## Additional resources

In addition to the printed documentation, there are several resources to support your simulation experience.

### Electronic documentation

If you install ExtendSim from a CD, PDF files of the User Guide and Developer Reference are installed in the application's Documentation folder. You can also download the documentation files from [www.ExtendSimManuals.com](http://www.ExtendSimManuals.com).

## ExtendSim Help

### *Context-sensitive help*

Context-sensitive help is available any time you are using ExtendSim. Just select the menu command Help > ExtendSim Help or press F1 on your keyboard.

### *Block help*

Blocks provide a complete definition of how they work, including descriptions of their dialog items and connectors. You can access this information through a button labeled Help in each block's dialog. The window that opens also has a Blocks button at the bottom left that will let you access the Help information for blocks from any library that is currently open.

### *Tool tips*

Additional information, such as the name of a block and its purpose, or the name and sometimes the value of an output connector, can be obtained by mousing over the block or connector, respectively.

### **User forums**

The ExtendSim E-Xchange is a user forum for sharing ideas, insights, and modeling techniques with other ExtendSim users. Use this forum to post issues and solutions, share blocks and models, and to talk directly to other people developing simulations. You will also find useful information about upcoming training sessions and seminars. You must register to join, but access is free and available to all ExtendSim users at [www.ExtendSimUsers.com](http://www.ExtendSimUsers.com), or select the command Help > User Forum.

The ExtendSim Academic E-Xchange is a user forum for educators using ExtendSim to teach modeling concepts and simulation to their students. Access is free but there is an approval process to obtain membership in this forum. To apply, go to [www.ExtendSimAcademic.com](http://www.ExtendSimAcademic.com).

### **Support**

Following are some suggestions if you need help while using ExtendSim.

#### ***How to get technical and modeling support***

To find answers to your modeling or technical questions as quickly as possible, we recommend that you refer to the various resources in the order shown below:

- 1) ExtendSim context-sensitive Help, as discussed on page 8.
- 2) User Guide or Developer Reference.
- 3) Frequently Asked Questions at [www.ExtendSimFAQ.com](http://www.ExtendSimFAQ.com).
- 4) For modeling questions, the ExtendSim E-Xchange or the ExtendSim Academic E-Xchange, as discussed at “User forums” on page 8. (Note: User forums are for modeling questions and are not appropriate for technical support questions.)
- 5) For technical support questions, choose the command Help > Support Resource Center or go to [www.ExtendSimSupport.com](http://www.ExtendSimSupport.com) and submit your question online.

#### ***Contacting Imagine That Inc. Technical Support***



You must be a registered customer to receive technical support from our support staff. Register online when you install ExtendSim (Windows only), register online after installation using the Register.exe file in the ExtendSim7\Online Registration folder (Windows only), or mail or fax the registration card that was included in your ExtendSim package.



When you contact our support representatives, please provide the following information:

1) **ExtendSim serial number, product name, and release number:**

*Windows:* Located in the Help > About ExtendSim menu command, on the title page of the User Guide, or on the tear-off remainder of your registration card.

*Mac OS:* Located in the ExtendSim > About ExtendSim menu command, on the title page of the User Guide, or the tear-off remainder of your registration card.

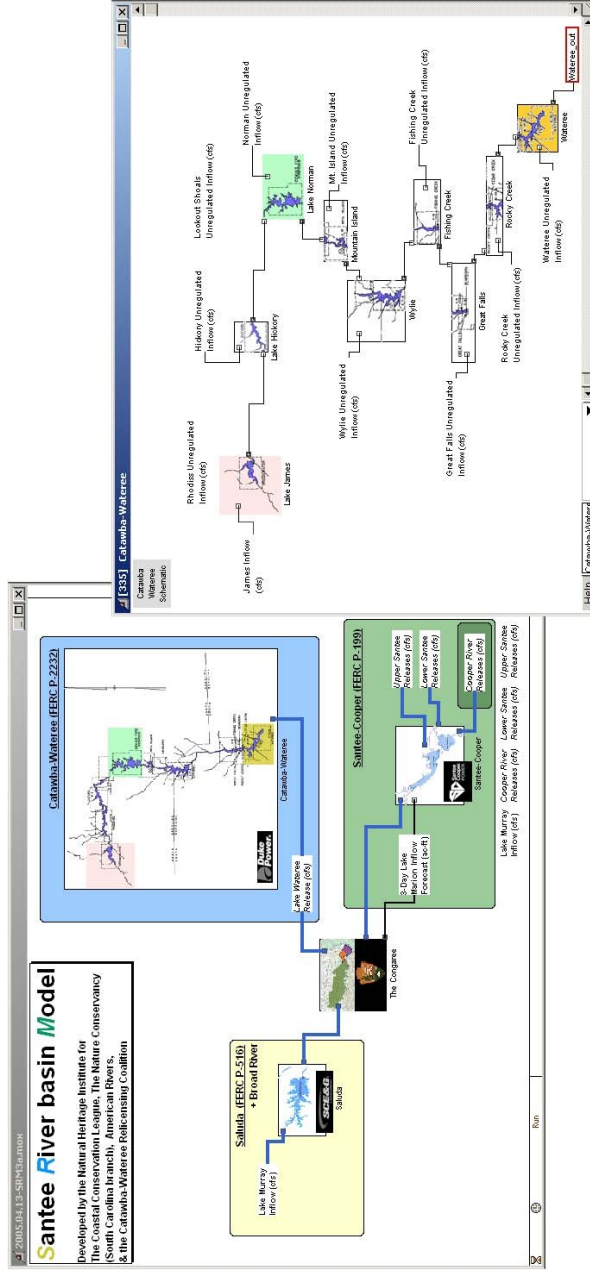
2) **Name and contact information** (telephone, email, and/or fax), so we can reply.

3) **Type of computer.**

4) **Operating system and version.**

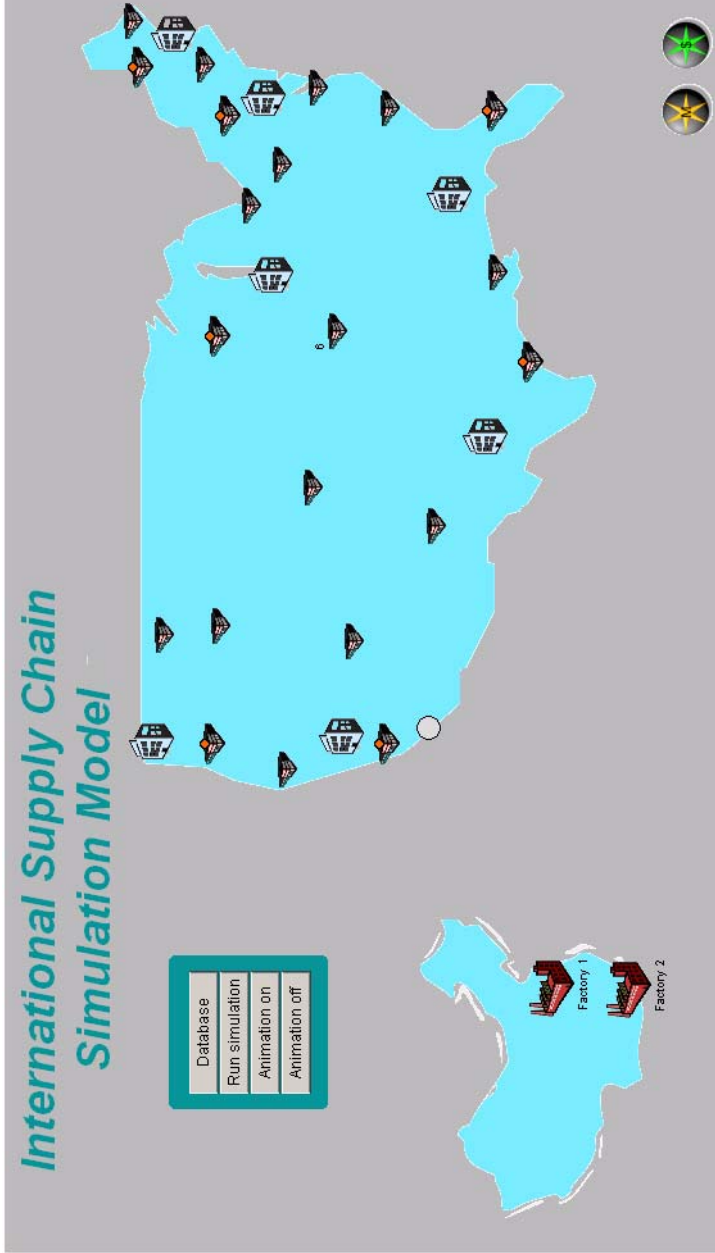
### Model illustrations

The tutorials used throughout this guide are specifically simplified so that you can easily learn how to use ExtendSim. In contrast, simulation is more typically used to model complex processes, like the ones shown on the following pages.



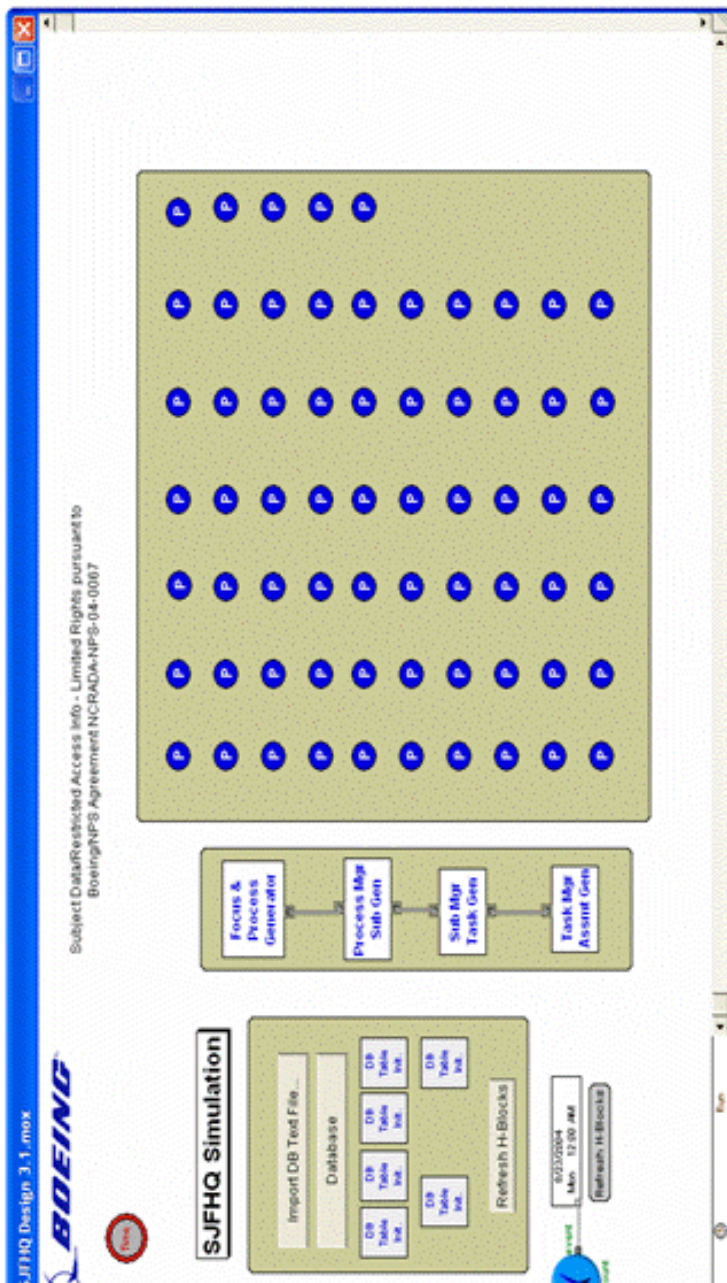
**Santee River Basin Model (SRM), developed by the National Heritage Institute, et al**

This continuous hydrologic model predicts how operational alternatives will affect hydropower generation, lake levels, in-stream flow, and water uses. It analyzes how dam operations can be altered to improve stream flow in over 300 river miles and helps assess the effects on lake levels and power generation. A series of hierarchical blocks with nested submodels keep details in layers below the model's realistic top level. For instance, the Catawba-Watersee hierarchical block contains 9 additional hierarchical blocks, some of which contain even more layers.



**International Supply Chain model, by James Dailey and Associates**

This discrete event model captures two key supply chain realities: the variances of supply chain dynamics and the non-linearity of business environments. The model uses an internal database to store the extensive amount of data, including SKUs, SKU Groups, Stock Points, and Assembly Lines.



**Standing Joint Force Headquarters (SJFHQ) model by the Naval Postgraduate School and The Boeing Company**

This discrete event model simulates the planning processes performed by SJFHQ members, analyzes time-critical information systems, and performs trade studies to obtain system measures of performance. It captures all processes, sub-processes, information flows, and personnel task assignments. The simulation takes approximately three seconds to run through one full cycle of processes, simulating 260 hours of operational work time and 6000 individual task assignments. The database allows the modeler to use a set of rules to drive the model architecture and simulation parameters.

Reference: "Modeling and Simulation Support for the Standing Joint Force Headquarters Concept" by Susan G. Hutchins et al. Published in the 10th International Command and Control Research and Technology Symposium.

# Tutorial

## Running a Model

Learn how to run an ExtendSim model  
and investigate its components

*“For the things we have to learn before  
we can do them, we learn by doing.”  
— Aristotle*


## 14 | Running a Model

### Opening the Reservoir model

The first two chapters of this User Guide provide a tutorial that will help you learn the basics of working with ExtendSim models. This chapter covers:

- Opening a model
- Blocks, including their icons, connectors, and dialogs
- Connections between blocks
- Running a model
- Displaying simulation results on a Plotter
- Using the Notebook to display model inputs and outputs
- Modifying models

The following chapter will show how to build the model seen in this chapter. To get the most out of the tutorial, we recommend that you follow along by performing the actions described.

 The ExtendSim Tutorial uses a continuous model to illustrate how to run and build a model. Even if you will be building non-continuous (discrete event or discrete rate) models, it is important to complete this Tutorial because:

- The tutorials in the non-continuous modules assume you have completed this Tutorial.
- It is common to use continuous blocks when building non-continuous models.

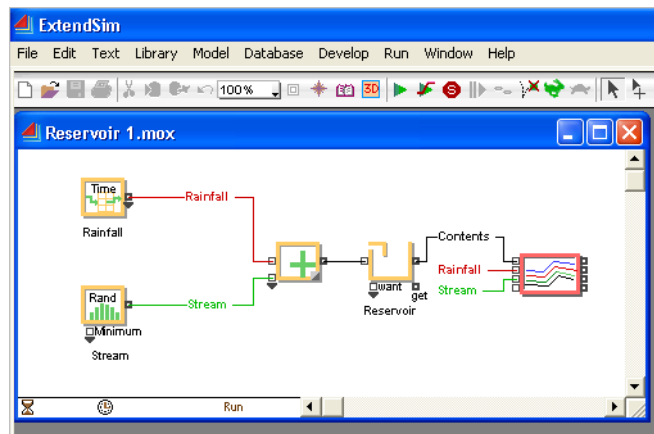
### Opening the Reservoir model

The tutorial uses the sample Reservoir model. This simple continuous model is useful for illustrating concepts because it can be accessed and understood by all ExtendSim modelers.

To open the model,

- ▶ Select File > Open.
- ▶ Browse to  
\\ExtendSim7\\Exam-  
ples\\Tutorials\\
- ▶ Select **Reservoir 1** and  
click Open.

This model simulates a reservoir being filled by two water sources—rainfall and an incoming stream. The purpose of the model is to see how much water accumulates in the reservoir over time.



Interface with Reservoir 1 model open

## Model basics

In the simplest terms, ExtendSim models are made up of *blocks* and *connections*. The Reservoir model, for example, has five blocks, as you can see in the model window. As the model runs, information goes into a block, is processed and/or modified, and is then sent on to the next block via a connection.

### Blocks

Each block in ExtendSim represents a portion of the process or system that is being modeled. Blocks have *names*, such as Math or Queue, that signify the function they perform. A Queue block, for example, will have the same functional behavior in every model you build. You can also add your own *label* to a block to indicate what it represents in your specific model, such as a Queue block labeled Waiting Line.

- Blocks are stored in *Libraries*. You will learn more about libraries and how to access blocks from them at “About libraries” on page 26.

Most blocks are composed of an *icon*, *connectors*, and a *dialog*.

### Icons

A block’s icon is usually a pictorial representation of its function. For instance in the Reservoir model, the block labeled Reservoir is a Holding Tank block. Its icon symbolizes an actual tank that can have quantities added or removed from it. The small squares attached to the sides of the icon are connectors, which are discussed in more detail in the following section.

- Place your cursor over a block’s icon to see a Tool Tip with its number (a unique identifier based on when the block was placed in the model), block name, and the library it comes from. To also display a description of the block, go to Edit > Options and check ***Include additional block information*** in the Model tab.

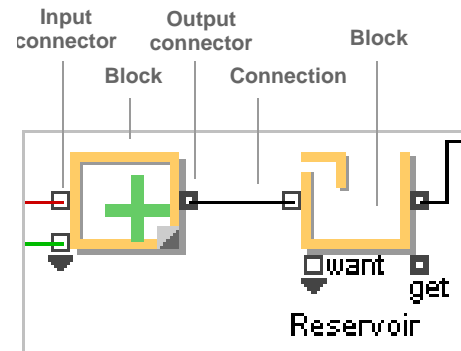
### Connectors

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

A block can have many input and/or output connectors; some blocks have none. For instance, the Holding Tank block labeled Reservoir has an input connector on the left for values to enter. The output connector on the right reports the results of the block’s computations; in the tank it reports the contents at each time step. Additional inputs on the bottom are for controlling specific tank behavior.

The function of a connector is specific to the block; you can get information about a connector’s function by clicking the Help button in the bottom left-hand corner of the dialog, as discussed in “Dialogs”, below. Since connectors are more important when you build a model (as compared to when you run it), they are discussed in more detail in “Connecting blocks” on page 27.

- Place your cursor over a connector to see its name and current value. You may also see additional information depending on how the block is programmed.



Parts of a model

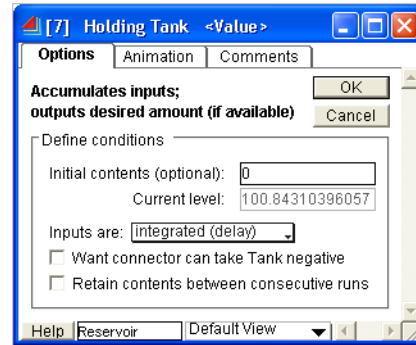
### Dialogs

Most blocks have a dialog associated with them. Dialogs are used to enter values and settings before running simulations and to see results as the simulation runs.

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

At the top of the dialog is the block's global block number, its name, and, in braces, the library it resides in. Global block numbers are unique identifiers assigned sequentially to blocks as they are placed in a model.

At the bottom of every dialog is a Help button. The block's Help provides information about the block, such as its purpose and use, connector usage, descriptions of each dialog item, and so on. Beside the Help button is a text box where you can enter a label for the block, up to 31 characters. The View popup is for changing the icon's orientation or appearance when you build models—for example, the Holding Tank offers a choice of *Default View* and *Default View Reverse*.



Holding Tank dialog

Some dialogs also calculate and display values that are generated as the model runs, so if you leave a dialog open during the simulation, 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 typing new values.

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

### Connections

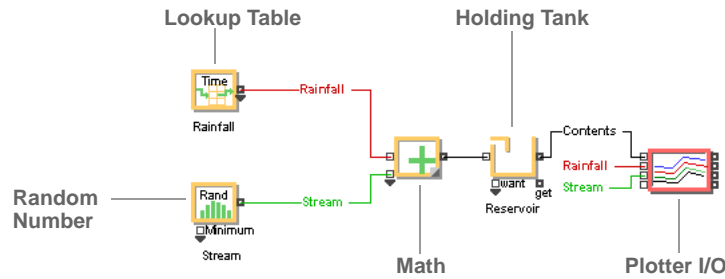
Connections are the lines that are used to join blocks together. They represent the flow of information from block to block through the model. The simulation itself is a series of calculations and actions which proceed along the path of the connections repetitively. Each repetition is called a *step* for continuous models or *event* for discrete event and discrete rate models.

In the Reservoir model, the blocks calculate in an order determined by the connections, starting at the left and going to the right.







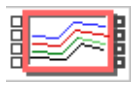
## Blocks used in the Reservoir model

There are five blocks in the Reservoir model.



Blocks in the Reservoir model

The following chart lists the blocks and their functions.

Name (Label)	Block Function	Purpose in Reservoir Model
Lookup Table (Rainfall) 	Acts as a lookup table. You can choose to set it to output data based on the current time or based on the value it receives at its input.	Represents rainfall entering the reservoir each month. The amount of rainfall is based on historical averages and varies with the month.
Random Number (Stream) 	Generates random integers or real numbers based on the selected distribution.	Generates random values that correspond to the changing flow of the stream. In this model, the stream increases the reservoir level between 0 and 1 inch of water per month.
Math 	Performs mathematical functions. The same Math block can be used for a wide variety of purposes by selecting the desired function from a popup box in the dialog.	Adds the amounts from the two different water sources and transfers them to the Reservoir.
Holding Tank (Reservoir) 	Accumulates the total of the input values. It also allows you to request an amount to be removed and outputs that requested amount, if available.	Accumulates water from its two sources. In this model, the tank has an infinite capacity and nothing is removed.
Plotter I/O 	Displays plots and tables of data for up to four value inputs for continuous models. Can be used to input its results to another section of the model or to another model.	Shows the amount of input from the two sources and the level of water in the reservoir, as it is affected by the amount of water entering it.

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 within a model.

### Running the Reservoir model

Now that you have seen the basic parts of a model, you are ready to run the Reservoir simulation and see how models operate. After running the model, you will see how easy it is to modify models as you change your assumptions. For now, do not change anything in the blocks' dialogs.

- ▶ Select Run > Run Simulation or click the Run Simulation button  in the toolbar.

As the simulation runs, progress information will be displayed in the status bar at the bottom left of the model window. (For a simple model like Reservoir that is completed very quickly, the messages may go by too quickly to be read.)

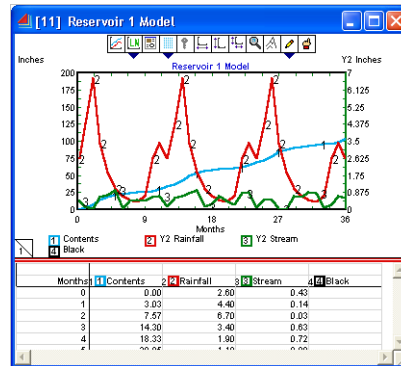
You can learn more about commands for running, stopping, and pausing models in “Running a model” on page 522.

### Displaying the results on the Plotter

Plotters show both a graphical representation of the numbers fed to them as well as a table of the numerical values. As described on page 588, ExtendSim comes with a number of flexible plotters to use in your models.

The Reservoir model runs for a simulated 36 months. While the model runs, ExtendSim displays the results on the Plotter, which by default remains on the screen when the simulation is finished. This is your primary method for determining what happened during the simulation.


The Plotter block in this model has information entering three of its four input connectors, so the graph displays three lines. The Plotter also has two value axes, each using a different scale. The legend below the graph indicates which axis is used for each line (Y2 indicates the values are plotted on the right axis) and what each line represents.



Plotter results for the model

In this case, the blue line, labeled **1**, is matched with the left axis to show the total contents of the reservoir over time. Using the right axis, the red line (**2**) shows the amount of water entering the reservoir from the rainfall alone and the green line (**3**) shows the input from the stream. (Because the model uses random numbers for the stream, specific values may be different after each run.)


The bottom of a plotter window shows the data points which produce the line. Scroll down this list to see the numerical values for each line. You can also observe the values for any given point by moving the cursor anywhere in the graph. The corresponding values are displayed above the data table's column headings.

- 🔍 ExtendSim plotters remember the pictures (but not the data) of the last four plots. You can see the previous plots by clicking on the small turned-up page symbol  at the bottom left of the graph part of the plot window.

## Notebooks

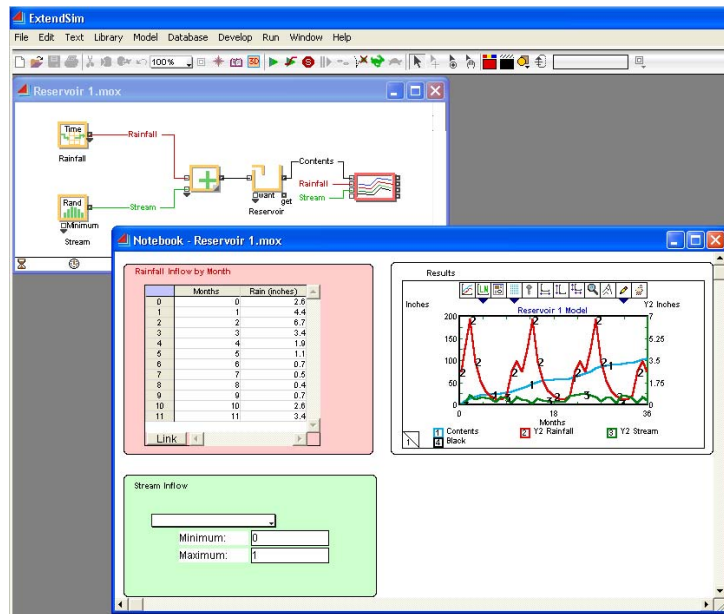
ExtendSim features like the Notebook give you capabilities that go beyond the basics of building and running models. A Notebook is a window you can customize to help organize and manage the data in a model.

You can use a Notebook as a “front-end” to the model - to control model parameters, report simulation results, and document your model. Each model has its own Notebook which can contain plots, text, pictures, drawing items, and cloned duplicates of dialog and plotter items.

- ▶ Select Window > Notebook or click the Open Notebook button  in the toolbar.

The Notebook for the Reservoir model opens. As you can see, the important parameters and tables you saw in the dialogs of the Reservoir model have been placed in its Notebook.

- ▶ Run the simulation again. Note that the results shown on the Plotter and the results shown in the Notebook are the same (the results will change slightly from one simulation run to the next because the Stream source uses a random distribution).



Reservoir 1 model with Notebook containing clones of dialog items from the Lookup Table, Random Number, and Plotter I/O blocks.

Notebooks are easy to create and are especially useful for documenting models and to view the impact of inputs on results. For more information, see “Notebooks” on page 508.

## Making changes to the model

So far, you have run the simulation and viewed the Notebook without changing any of the assumptions that were supplied when the model was created. One of ExtendSim’s strongest features is the ability to change assumptions on the fly and see the results instantly. Since the Plotter remembers the previous four plots, you can easily compare the results after you change assumptions.

You can change a model by adding or removing blocks or by changing parameter values in a block's dialog.

### Adding and removing blocks

If you have some processes running in parallel (such as the two water sources in the Reservoir model), you can easily test the results of adding additional parallel processes or removing existing ones.

You will learn how to add blocks in “Building a Model” on page 23.

To remove a block, click it to select it, then choose Edit > Clear Blocks or use the Delete or Backspace key. Note that deleting a block removes its associated connections as well.

### Changing dialog parameters

You can change a dialog value by clicking in a parameter field while the model is running. When you do this, ExtendSim pauses the simulation. To continue the simulation, select Run > Resume or click the Pause/Resume tool in the toolbar. Note that when you make changes to a dialog and then save the model, the changes are saved with it.

On page 509 you will see how to vary a dialog value manually using Controls, such as Sliders, and on page 568 how to use sensitivity analysis to automatically explore various scenarios. For simplicity, parameters in the following blocks are entered in the dialog as static values that do not change based on model conditions. To see how easy it is to change dialog parameters and see the impact on results, try some of the following suggested changes.

#### Lookup Table block (Rainfall)

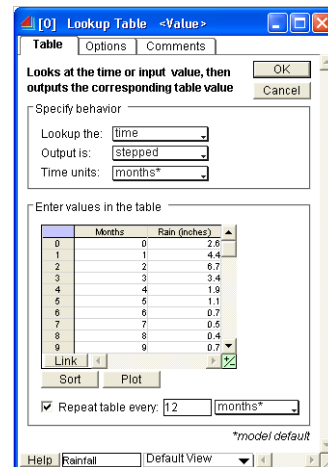


The Lookup Table block provides a time-varying value (in this case, the amount of rainfall each month) for the simulation.

This block looks at the current simulation time, compares it to the time values in the column labeled **Months**, and outputs the corresponding **Rain (inches)** value.

Right now, the Lookup Table block shows that 2.6 inches of water are fed into the reservoir at time 0, the beginning of the run, 4.4 units at the beginning of the next month, and so on.

- Change the first value in the **Rain** column from 2.6 to a high number, such as 90, then run the simulation again to see the impact on the results.



Lookup Table block dialog

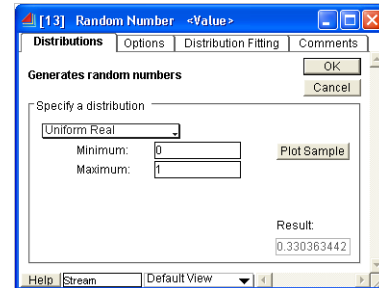
### Random Number block (Stream)



The Random Number block generates values based on a random distribution. In this model, it outputs a real number from 0 to 1.

► Change the distribution's **maximum** output from 1 to a different number, such as 30, and run the simulation again.

☞ ExtendSim lets you choose from dozens of probability distributions or create your own. Learn more about how to use random numbers in your models in “Math and Statistical Distributions”.



Random Number block dialog

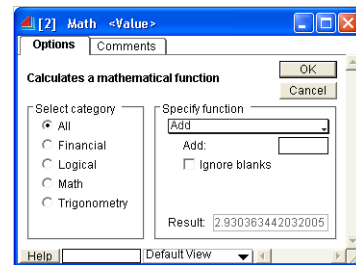
### Math block



The Math block can be used to perform any mathematical function. In this model, it is used to add the amounts of water coming in from the rainfall and the stream.

► To add a constant value to the input number, enter an amount in the **Add** field.

The Math block is very versatile. By clicking radio buttons you can quickly change the type of mathematical function that the block performs.



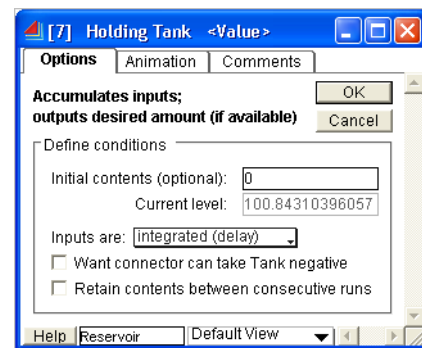
Math block dialog

### Holding Tank block (Reservoir)



In addition to accumulating the amounts entering, the Holding Tank allows you to enter an initial amount and to remove some or all of the contents. You can also specify which mathematical method is performed on the inputs to calculate the contents.

► Enter an amount in the **Initial contents** field and run the simulation again to see how the results change.



Holding Tank block dialog

### Other modifications

As you have seen, ExtendSim offers many options to change the way a model runs so that you can explore different scenarios. The Reservoir model can be expanded upon in other ways that will be explored later since they involve more advanced techniques. For example, you can:

- Add more real-life contingencies by adding blocks to the model. For example, add more water sources, then use a Notify block (Utilities library) to monitor the simulation and sound an alarm if the water level goes above a certain value. For more information, see “Notify block” on page 511.
- Add blocks and connect them to the Holding Tank to remove some of the water from the Reservoir. The blocks can cause the water to be removed randomly or based on a table of

expected outputs. The Reservoir models in “Tutorial” on page 63 demonstrate some ways of doing this.

- View the status of blocks by connecting their outputs to the Plotter, by leaving their dialogs open while running the simulation, or by taking some of their dialog items and putting them in the model window, which is referred to as *cloning*. You’ll learn more about cloning in “Creating a dashboard interface” on page 506.
- Perform analysis such as Sensitivity Analysis or Optimization and have ExtendSim find the best set of parameters. See the chapter “Analysis” on page 563.
- Build a user interface for the model by putting sections of the model in layers (hierarchy), building custom reports (Notebook and reporting features), or creating a dashboard front-end (using buttons and cloned dialog parameters to run the model). These features are discussed in “Creating a Custom User Interface” on page 503.

Now that you understand how easy it is to run an ExtendSim model, the next chapter will show you how to build a model from scratch. When you are finished with that chapter, you will see how easy it was to create the Reservoir model and the ease with which you can create your own models.

# Tutorial

## Building a Model

How to build an ExtendSim model  
and use the model window features

*“He builded better than he knew;  
The conscious stone to beauty grew.”  
— Ralph Waldo Emerson*

This chapter continues the Tutorial by describing the steps required to create the Reservoir model that you ran in Chapter 1. As stated earlier, this model uses continuous simulation, but the concepts described in this chapter also apply to other types of modeling.

- ☞ Other modules in this guide describe specific discrete event and discrete rate concepts and show you how to build those types of models, but they assume that you have already completed this Tutorial.

The topics covered in this chapter include:

- Opening a new model window
- Simulation setup and run options
- Opening a library of blocks
- Adding blocks to the model
- Connecting blocks using different methods
- Using dialogs to set parameters
- Making adjustments to the Plotter graph
- Hierarchy - top down
- Navigating the model
- Cloning duplicates of dialog and plotter items

### Steps to create the Reservoir model

The basic concept behind the Reservoir model is to simulate what happens to a reservoir's water level as water enters it over a 36 month period. There is no water in the reservoir at the start of the simulation and no water is removed from it. As water sources add their contributions each month, the water level rises.

The steps to create the Reservoir model are:

- 1) Open a new model worksheet
- 2) Set simulation run parameters
- 3) Build the model using blocks from libraries
- 4) Select block settings and enter dialog parameters

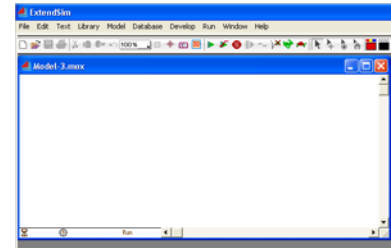
- ☞ For your reference, the final version of the Reservoir 1 model is located in the `ExtendSim7\Examples\Tutorials` folder and is shown on page 36. However, you will learn more about ExtendSim and modeling if you build the model yourself.



## Opening a new model worksheet

To start a new model:

- ▶ Choose File > New Model.
- ExtendSim opens a blank model worksheet titled Model-1.



New model worksheet (Windows)

## Setting the simulation parameters

You enter simulation parameters, such as the model's time units and duration, in the Simulation Setup command.

- ▶ Choose Run > Simulation Setup

The Simulation Setup command opens a dialog for setting a variety of simulation and 3D animation parameters, such as how long and how many times the simulation will run, when the random number seed gets reset, the mode of interaction between the simulation and the 3D window, and so forth. The dialog has tabs for Setup, Continuous, Random Numbers, 3D Animation, and Comments.

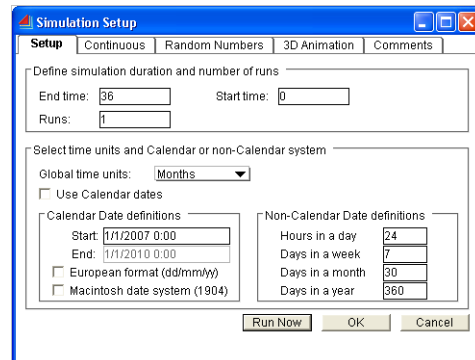
The most common simulation settings you will need to enter in the Simulation Setup window (and often the only ones) are the **End time** and **Global time units** parameters located on the **Setup tab**. For most purposes, you want the simulation to start at the beginning, so you would use the default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

Customize the Setup tab by entering the following parameters:

- ▶ **End time: 36**
- ▶ **Start time: 0** (Default)
- ▶ **Runs: 1** (Default)
- ▶ **Global time units: Months**
- ▶ Click OK

This model will run for 36 months, performing calculations once each month.

Each time you run a simulation, ExtendSim uses the same values entered in the Simulation Setup window. Thus, you will usually only configure the settings once per model. The Simulation Setup command is discussed fully in "Simulation setup" on page 516.



Setup tab of Simulation Setup window

## Building the model

The Reservoir model requires five blocks. There are two sources of water: rainfall and a stream; the data for rainfall comes from a table while the stream contributes a random amount. You need a block to add the two water sources, another block that can hold values (representing the reservoir), and a plotter to display the results of the simulation.

### Basic steps

The basic steps for building a model are:

- 1) Open the relevant libraries, if necessary.

- 2) Add the blocks to the model.
- 3) Move them to the desired positions.
- 4) Add connections between blocks.

### About libraries

Blocks used in a model are stored in repositories called *libraries*. The entire definition for a block (its program, icon, dialog, and so on) is stored in the library. When you include a block in a model, the block itself is not copied to the model. Instead, a reference to the block is included in and stored with the model. Any data you enter in the block's dialog is also stored within the model.

There are many advantages to this method of using references to libraries instead of actual blocks in models. If you change the definition of a block in a library, all models that use that block are automatically updated. Also, block definitions are quite large, so storing just a reference to the library saves memory and reduces processing time.

- ☞ When you save a model, ExtendSim saves the names of the blocks as well as the locations of the libraries that store the blocks. The next time you open the model, ExtendSim automatically opens the libraries the model uses. You can also set a preference in the Options window to have up to seven libraries load automatically whenever ExtendSim is launched (see “Options” on page 688).

### Opening the relevant libraries

To add a block to a model, the library in which that block resides must be open. For the Reservoir model, you need to open the Value and Plotter libraries.

To open the Value library:

- ▶ Choose Library > Open Library. ExtendSim takes you to the Libraries folder.
- ▶ Select the *Value* library.
- ▶ Click Open.
- ▶ Repeat the above steps to open the *Plotter* library.

Open libraries are listed in alphabetical order at the bottom of the Library menu.

### Adding blocks to the model

There are two methods for adding a new block to a model:

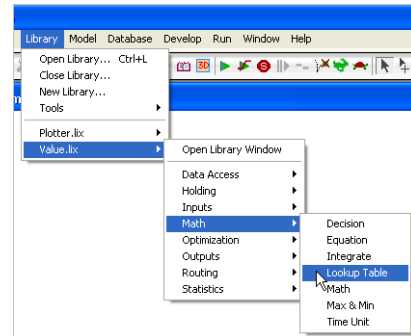
- Select the block from its library within the Library menu.
- Drag the block from a Library window, as discussed on “Library Window mode” on page 671.

For this Tutorial, you will use the first method and select blocks from the Library menu.

The first block needed for the Reservoir model is a Lookup Table block, which will be used to enter data about the amount of rainfall entering the reservoir.


To add the Lookup Table block to the model worksheet:

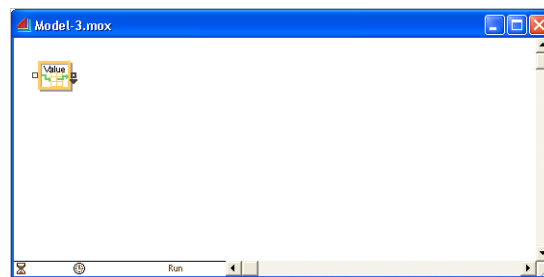
- ▶ From the Library menu, scroll to the Value library, which is listed at the bottom of the menu. When the Value library is highlighted, a secondary menu opens that lists several categories, each of which further expands to show the blocks contained in that category.
- ▶ Under the Math category, click Lookup Table.



Library menu with Value library categories

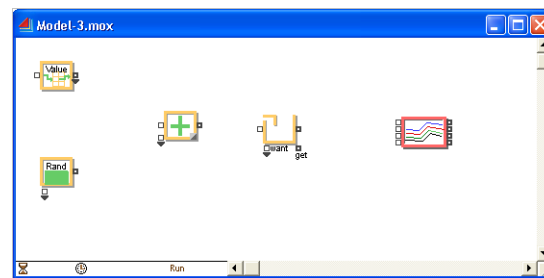
An icon for the Lookup Table block appears in the top-left corner of the model window. By default, the icon is selected. To deselect a block, click anywhere in the window. To move a block, select it and then drag it to the desired position in the model window or use your keyboard arrow keys to move it one pixel at a time.

 If you click at a location on the model window before you pick a block from the menu, the block will appear at the point where you clicked.



Lookup Table block added to model

- ▶ From the Library menu choose the library, category, and block (as indicated below), to add the four remaining blocks to the model:
  - ▶ Value library > Inputs category > Random Number block
  - ▶ Value library > Math category > Math block
  - ▶ Value library > Holding category > Holding Tank block
  - ▶ Plotter library > Plotter I/O block



All blocks added to model

When you have finished, the model should look similar to the screenshot above.

This is a good time to save the model so far.

- ▶ Choose File > Save Model As and name the file *My Reservoir*.

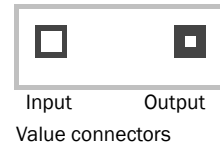
### Connecting blocks

As mentioned in Chapter 1, connections pass information from one block to another. Blocks are usually connected together by drawing connections from one block's output connector to another block's input connector.

### Connectors

In ExtendSim, the behavior of most connectors is predefined for each specific block. For example, when you set a Math block to use a function (add, subtract, divide, etc.) it knows what to do with the values that are input into the block. This makes model building easy since you can connect blocks and run simulations without having to write equations to define what each block should do with the inputs or outputs.

There are several types of connectors in ExtendSim. Continuous models, such as Reservoir, only use *value* input and output connectors to pass information from one block to another. Other types of connectors will be discussed in later chapters.



Each input connector can only have one source of information. Therefore, blocks that need to have many sources of input require a separate input connector for each piece of information.

### Types of connections

There are two types of connections in ExtendSim: *line connections* and *named connections*. Line connections join the output of one block to the input of another using connection lines; named connections use text labels as outputs and inputs, causing data to jump from the output to the input without using connection lines.

Connection lines can be drawn using three different styles: *right-angle*, *straight*, and *multi-segment*. The default style is right-angle, which you will use in the following example.

The other styles of line connections, as well as named connections, will be discussed at “Additional ways of connecting blocks” on page 32.

### Connecting the Lookup Table block to the Math block’s variable connector


In the Reservoir model, the two water sources need to be connected to the Math block so that the amounts of water entering the reservoir from rainfall and from the stream can be added together. The first step is to connect the Lookup Table block to the Math block.

As mentioned earlier, blocks that need to have more than one source of input require a separate input connector for each piece of information. ExtendSim provides for this by putting *variable connectors* on blocks that might need them. This is usually indicated by a black arrow beneath the connector that can be dragged to display additional connectors.

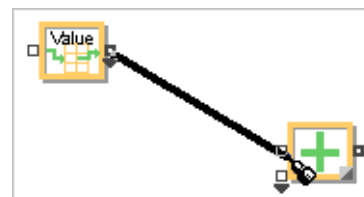
The Math block, for example, has a variable input connector as indicated by the black arrow below its input. However, since the Math block displays two inputs by default, you do not need to expand the variable connector. For more information about using variable connectors, see “Variable connectors” on page 498.

To connect the Lookup Table block to the Math block:

- ▶ Move the cursor to the output connector of the Lookup Table block.

The cursor changes from an arrow to a technical drawing pen: 

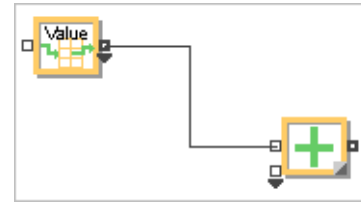
- ▶ Click the Lookup Table’s output connector, then drag a line to the top input connector on the Math block. You can tell when you are over the connector because the line you draw becomes thicker.



Thickened line = successful connection

- ▶ Let go of the mouse button.

If you accidentally release the mouse button before the line has thickened, a dotted red line will appear to indicate the connection has not been made. To remove it, double-click the line so that the entire connection thickens (indicating that you have selected it), then press the Delete or Backspace key, or choose Edit > Clear Connection. (A single click will select one segment of the line only.) You can then make a correct connection.

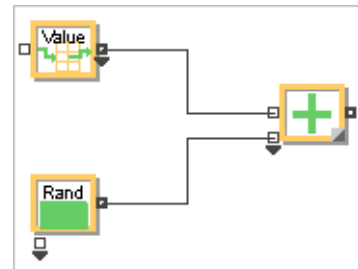


Right-angle connection

### Connecting from the Random Number block

The next step is to connect the second water source, the Random Number block, to the input of the Math block. To do this:

- ▶ Draw a connection line between the output connector on the Random Number block and the second input connector on the Math block, just as you did for the Lookup Table block earlier.



Connect from Random Number block

### Connecting the remaining blocks

Connect the other blocks in the model as follows:

- ▶ Draw a connection line between the output connector on the Math block and the input connector on the Holding Tank block.
- ▶ To monitor the reservoir's level, draw a connection line between the Holding Tank's output connector and the top input connector on the Plotter block.



Connection from Math block



Connection between Holding Tank and Plotter blocks

## Working with block dialogs

Now that all the blocks have been placed and connected in the model, you can enter data and select options in block dialogs. The data for this model comes from a table of values and from a random sample. The sections below will describe the settings used in each block's dialog.

### Rainfall source

The amount of rainfall entering the reservoir is determined by a Lookup Table block that contains each month's expected rainfall. In a real-life situation, for example, these numbers may have been determined by annual recorded averages.

- ▶ Double-click the Lookup Table block to open its dialog.

By default, the Lookup Table block is set to **Lookup the: input value**. In that mode, the block outputs a value that corresponds to the value it receives at its input. However, you want the Lookup Table block to output a value--the amount of rainfall for the month--that corresponds to the current simulation time.

- ▶ Customize the dialog's *Table* tab by setting the behavior of the block to look up simulation time each month:

- ▶ **Lookup the: time**
- ▶ **Output is: stepped** (Default)
- ▶ **Time units: months** (Model Default)

Time means the block will compare current simulation time to a time in the table and output the corresponding value. Stepped means that ExtendSim will use the exact values you enter in the table, not an interpolated amount.

- ▶ Increase the number of rows so the table has room for twelve months of data:
  - ▶ Click the +/- sign in the green square at the bottom right of the table.
  - ▶ Enter **12** for the number of rows and **2** (the default) for the number of columns.
  - ▶ Click OK
- ▶ Enter data into the table as shown in the screenshot at right.

	Month	Rainfall (inches)
0	0	2.8
1	1	4.4
2	2	6.7
3	3	3.4
4	4	1.9
5	5	1.1
6	6	0.7
7	7	0.5
8	8	0.4
9	9	0.7
10	10	2.8
11	11	3.4

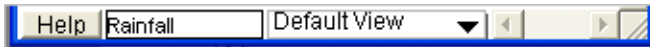
Table settings for Lookup Table

If you click Enter after each value, the cursor will automatically move to the next cell.

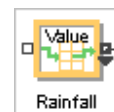
- ▶ Check the **Repeat table every** checkbox and enter **12** in the **months** box.

This causes the determination of monthly rainfall to start over every 12 months.

- ▶ In the label entry box beside the Help button, type in **Rainfall**. Labels can have a maximum of 31 characters, including spaces.



Adding label to the Lookup Table block



Block labeled "Rainfall"

- ▶ On the *Options* tab, enter the following text in the column labels box to give more meaningful headings to the table (be sure to include the semi-colon):
  - ▶ **Month;Rainfall (inches)**
  - ▶ Return to the Table tab

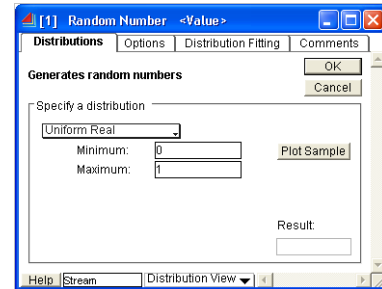
The table's left column now specifies the month and the right column specifies the amount of expected rainfall in inches. At each step, ExtendSim will check the block's table for a time in the first column that is less than or equal to the current simulation time and output the corresponding value (inches of rainfall) to its right. For instance, for the fourth month the block will output 1.9.

- ▶ Click OK to close this block's dialog.

### Stream source

In this model, the Random Number block is used to specify a random distribution of water entering the reservoir from the stream. The distribution is a real number between 0 and 1, indicating that the stream will add between 0 and 1 inches of water to the reservoir's level each month.

- ▶ Open the Random Number block's dialog.
- ▶ By default the dialog's parameters already have the settings you want:
  - ▶ **Distribution: Uniform Real** (Default)
  - ▶ **Minimum: 0** (Default)
  - ▶ **Maximum: 1** (Default)
- ▶ Enter **Stream** in the label field next to the Help button.
- ▶ Click OK.

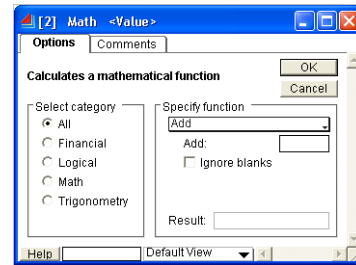


Random Number dialog settings

### Combining the sources

The Math block combines the values from the two water sources. As indicated by the plus sign on its icon when you placed the block in the model, the Math block is set by default to add its inputs. There is no need to change its dialog settings.

- ▶ Although you don't need to for this model, the Math block function can be changed directly in its dialog or by right-clicking a special area on its icon. For the Math, Decision, and Simulation Variable blocks (Value library), the icon's lower right corner has a sensitized area that looks like a partially turned page. You can right-click that area to change dialog settings.

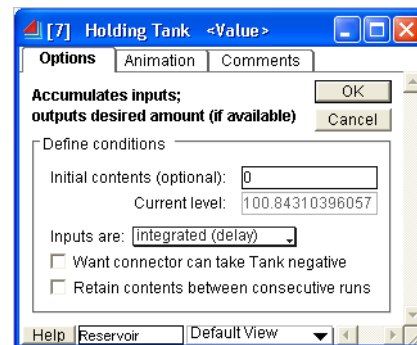


Math dialog settings

### Water in the reservoir

The Holding Tank block represents the level of water in the reservoir. In this model, the Holding Tank has no beginning contents and does not release any of its contents.

- ▶ Open the Holding Tank block's dialog.
- ▶ In the dialog, set:
  - ▶ **Initial contents: 0** (Default)
  - ▶ **Inputs are: integrated (delay)**
- ▶ Enter **Reservoir** in the label field next to the Help button.
- ▶ Click OK.



Holding Tank dialog

- ▶ The Holding Tank should be set to integrate, rather than sum, its inputs. This will output the value at time 1 that has been calculated for the period from time 0 to 1. These continuous simulation concepts are discussed more thoroughly in "Integration vs. summation in the Holding Tank block" on page 610.

### Displaying the results

The Plotter I/O block monitors the total amount of water in the reservoir. (Later in this chapter you will learn different techniques for connecting the Lookup Table and Random Number blocks to the Plotter, to also monitor the amounts flowing in from the rainfall and the stream.)

- ▶ Double-click the Plotter to open its plotter window.

Notice that the first column in the table is titled Contents. The Plotter automatically named it when you connected from the contents output of the Holding Tank block.

To further personalize the plotter window:

- ▶ Click the text label **Value** in the upper left corner of the Plotter's graph to select its text box.
- ▶ Type **Inches** in the text box and click the Tab, Return or Enter key.
- ▶ Using that same process, change other Plotter labels in the graph pane as follows:
  - ▶ Change **Plotter I/O** (located at the top of the graph) to **Reservoir Model**.
  - ▶ Change **Time** (located below the graph) to **Month**.
- ▶ Close the plotter window
- ▶ Save the model.

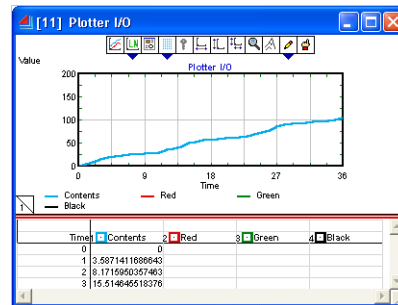
When you save a model, ExtendSim creates a backup, **ModelName.bak**, of your previously saved model. To open a backup file, add the extension **.mox** after the **.bak** so that the file reads **ModelName.bak.mox**. Then Open the backup file from the File menu.

### Running the simulation

Now that you have placed blocks on the model, connected them, and configured their dialogs with data, it is time to run the model.

- ▶ Select Run > Run Simulation or click the Run Simulation button in the toolbar.

A plot similar to the one at right will be displayed. It only shows one line because only the Holding Tank is currently connected to the Plotter block.



### Additional ways of connecting blocks

For this simulation, you need to also connect the rainfall and stream blocks to the Plotter so you can compare their outputs to the total water collected in the reservoir.

When you built the model, you used the right-angle connection line style to connect blocks. This is the default setting for all new models. You can also connect blocks using the straight and multi-segment connection line styles. In addition to using connection lines, you can connect blocks using named connections. These methods are discussed in the following topics.

#### Straight line connection

You draw this line style in exactly the same way as the right-angle style, but it displays differently.

- ▶ Select Model > Connection Lines and select the straight line option (second item).



☞ Changing the setting in the Model menu will only affect subsequent connections in this model. You can change the default setting for all models by choosing Edit > Options > Model tab and unchecking “Default connection line style is right angle”.

- ▶ Draw a line from the output connector on the Lookup Table block to the second input connector on the Plotter.

Both the straight line and right-angle line connections have the disadvantage of running directly over other blocks and connections, making the model more difficult to read.

- ▶ Delete the straight line connection by selecting it and pressing Delete or the Backspace key.

### Multi-segment line connection

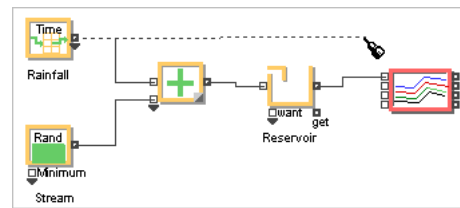
This style has the advantage of letting you draw the lines so that they go around blocks rather than over them.

- ▶ With the straight line option still selected, click the Lookup Table’s output connector and drag your cursor until it is above the space between the Holding Tank and the Plotter blocks.

- ▶ Release the mouse.

This creates the first segment. The cursor remains a technical pen because you are pointing at an *anchor point*.

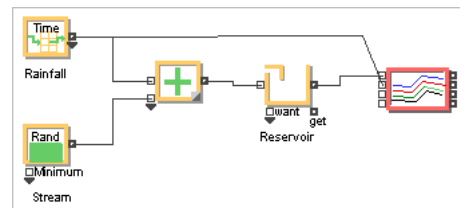
- ▶ Immediately click again and drag the cursor to the second input connector on the Plotter, then release the mouse button.



First segment and anchor point

You now have a *multi-segment connection*. (Note that a right-angle connection is simply a multi-segment connection that is automatically created by the application.)

☞ Anchor points can be moved if the connection did not come out as you intended. Simply move your mouse over the anchor point until the cursor changes into a hand, then click the anchor point and drag it to the desired location.



Multi-segment connection

Although this connection no longer crosses over any other elements of the model, you can see how a larger, more complex model could become very cluttered with so many line segments in the window.

- ▶ Delete the multi-segment connection by double-clicking a segment until the entire connection line thickens, then press Delete or the Backspace key. (To delete just one segment of the line, you would click the segment once and press the Delete key.)

### Named connection

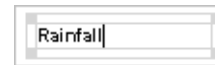
Named connections are text labels that are used to represent one output at many locations in your model. If you have two labels with the exact same text, you can use these to have the flow of data jump from one part of the model to another. Named connections are often used when you do not want to clutter up your model with many lines. You can place the names near the blocks to which they connect and leave much of the area of your model free from connection lines. Named connections are discussed in detail at “Named connections” on page 560.

☞ Named connections are not case sensitive and spaces and returns are ignored, but you must use identical spelling in the text names.

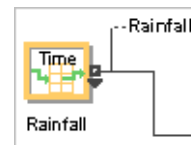
**Creating a named connection between the Lookup Table and the Plotter**

To add a text label for the named connection:

- ▶ Choose Model > Connection Lines and select either the right-angle or straight line style.
- ▶ Double-click in the model window, slightly above and to the right of the Lookup Table block's output connector. This opens a text box.
- ▶ Type **Rainfall** in the text box.
- ▶ When you are finished typing, click anywhere else on the model window.
- ▶ Join the Lookup Table's output connector to the word **Rainfall** by dragging a line from the connector to the text and, when the line thickens, release the mouse.



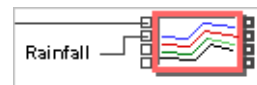
Typing in text box



Lookup Table joined to named connection text

☞ Until you connect the text to an input connector on another block, the line will remain dotted.

- ▶ Click the **Rainfall** text to select it, then choose Edit > Duplicate.
- ▶ Drag the duplicate text to a spot slightly below and to the left of the Plotter and release the mouse.
- ▶ Draw a line between this text and the second input connector on the Plotter. Note that both the connection lines are now solid.



Named text connection joined to Plotter

**Named connection between Random Number and Plotter**

Repeat the process above to create a named connection between the Random Number block and the Plotter:

- ▶ Create a **Stream** text label and place it near the output of the Random Number block.
- ▶ Connect from the Random Number block's output connector to the text label.
- ▶ Duplicate the text label and drag it to a spot below and to the left of the Plotter.
- ▶ Connect from the text label to the third input connector on the Plotter.

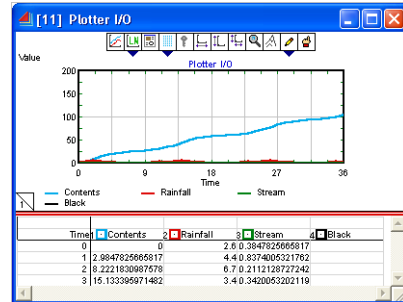
☞ To display the actual connection lines between blocks, choose Model > Show Named Connections.

**Plotting against multiple axes**


- ▶ Click the Run Simulation button on the toolbar.

The Plotter that appears now has three lines on it: the blue one displays the amount of water in the reservoir over time, the red one displays the amount entering the reservoir from the rainfall, and the green line displays the water entering from the stream.


At the end of the simulation run, the Plotter automatically scales its axis to be able to display all values for both columns of data. However, because the total amount of water in the reservoir has a much greater range than the amount entering it each month, the lines representing the rainfall and stream amounts can barely be seen. They simply look like horizontal lines across the bottom of the graph.




To solve this problem, you can add a separate axis (Y2) on the right-hand side of the graph and set the Plotter to display the rainfall and stream values against that axis.

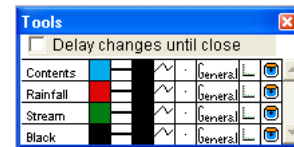
- ▶ If its not already open, double-click the Plotter to open its plotter window.
- ▶ Click the Trace properties button , which is the left-most button in the toolbar at the top of the plot window.

The Tools dialog opens.

- ▶ In the second row, labeled Rainfall, click the Y1/Y2 button , second from the right.

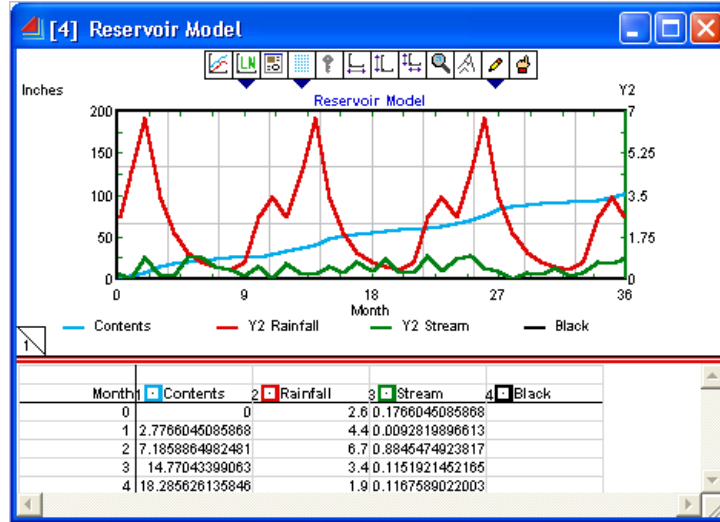
The button changes so the vertical line is to the right of the horizontal line. 

- ▶ In the third row, labeled Stream, click the Y1/Y2 button, causing the stream to also be plotted against the Y2 axis.
- ▶ Close the Tools dialog box.
- ▶ Click the Run Simulation button on the toolbar.



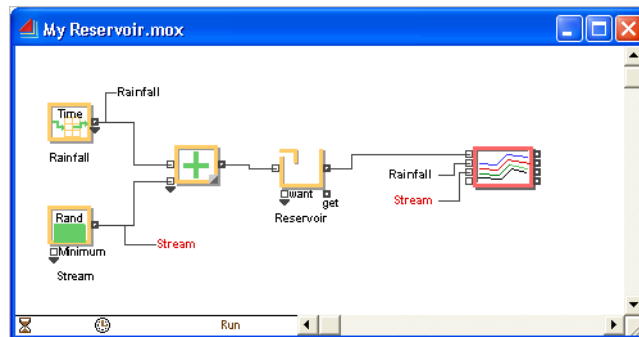
Plotter Tools dialog

The graph now displays the Rainfall and Stream lines using the right-hand axis.



### The final Reservoir model

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



My Reservoir model

### Additional enhancements

Now that you know how to create a model, you can explore some other features, such as hierarchy and cloning, that are easy to do and will greatly enhance your models.

#### Introduction to hierarchy

The Reservoir model displays one block per function performed, i.e. the Lookup Table block outputs the amount of water coming from rainfall, the Random Number block outputs the water coming from a stream, etc. For such a simple model, this works fine. However, models created for real-life simulations can involve thousands of blocks. Building, organizing, and presenting a complex model with all the blocks on one layer of the worksheet would be very difficult.

To help simplify and clarify models, ExtendSim lets you create hierarchical blocks (H-blocks) that group several blocks together into one block while still allowing you to drill down into the lower levels to access the individual blocks.

**Creating a hierarchical block from existing blocks**

In the Reservoir model, you can group the blocks that represent sources of water together into one hierarchical block. This process is extremely easy.

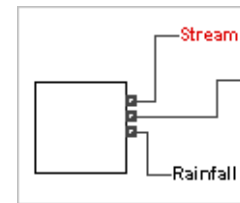
- ▶ Shift-click the Lookup Table, Random Number, and Math blocks to select them.

⚠ Do not select the text labels (Rainfall and Stream) of the named connections!

- ▶ Select Model > Make Selection Hierarchical.

A dialog appears prompting you for a name for the hierarchical block.

- ▶ Enter **Water Sources**.
- ▶ Click **Make H-Block**.



Hierarchical block

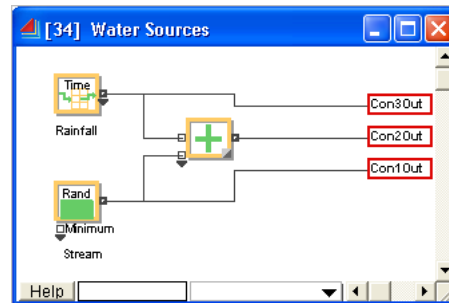
The three individual blocks are replaced with a single hierarchical block with a white rectangle for an icon. By default, hierarchical blocks have drop shadows to distinguish them from other blocks; you can change that option by choosing Edit > Options > Model tab.

Note that there are three connectors, including the Rainfall and Stream named connections, on the hierarchical block's icon.

- ▶ Double-click the hierarchical block to see the *sub-model*, or individual components, inside it.

👉 The hierarchical block and submodel may look slightly different depending on how the blocks were placed.

The window's title bar displays the name of the hierarchical block. Note that connections for transferring data from within the hierarchical block to the outside model are represented in the submodel as named connections with red borders around the text. Those connections correspond to the three connectors on the block's icon.



Water Sources submodel

- ▶ Close the Water Sources window.

ExtendSim provides many more features for creating and using hierarchical blocks, including building hierarchical blocks from scratch, assigning custom icons to them, and providing Help information. To learn more, see "Hierarchy" on page 540.

**The ExtendSim Navigator**

The Navigator is an explorer-like window that can be used for multiple purposes:

- To navigate through the hierarchical structure of a model
- To access any databases used in the model
- To add blocks to the model worksheet, as an alternative to using the Library menu

### Navigating through the Reservoir model

Since your reservoir model now has a hierarchical block (the Water Sources block you created earlier), you can see how the Navigator is helpful for exploring a model.

To open a Navigator:

- ▶ Select Window > Navigator or click the Open Navigator tool in the Toolbar.

By default, the Navigator opens in Model Navigator mode, with the word “Model” selected in the leftmost popup menu. The name of the active model is listed at the top of the window and below the Navigator’s leftmost popup menu, and each block’s icon and information (name, label, and global block number) is displayed.

- ▶ Click the plus sign beside the Water Sources hierarchical block.

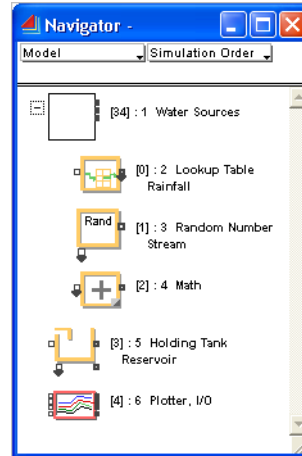
The hierarchical block expands to show the blocks within it.

- ▶ Select the Lookup Table block in the Navigator.

The corresponding block is selected in the model window.

- ▶ Double-click the Lookup Table block in the Navigator.

The block’s dialog opens.



Navigator in Model mode

As you can see, the Navigator is indispensable for exploring complex models, especially for models with many layers or instances of hierarchy. For additional information, see “Navigator” on page 670

### Cloning


In “Running a Model” on page 13 you saw how Notebooks help organize, monitor, and interact with data during simulations. ExtendSim lets you add dialog and plotter items to your Notebook using a technique called *cloning*. Clones are exact replicas of dialog items, behaving exactly like the original. When a cloned value changes, the original dialog item or plot graph also changes.

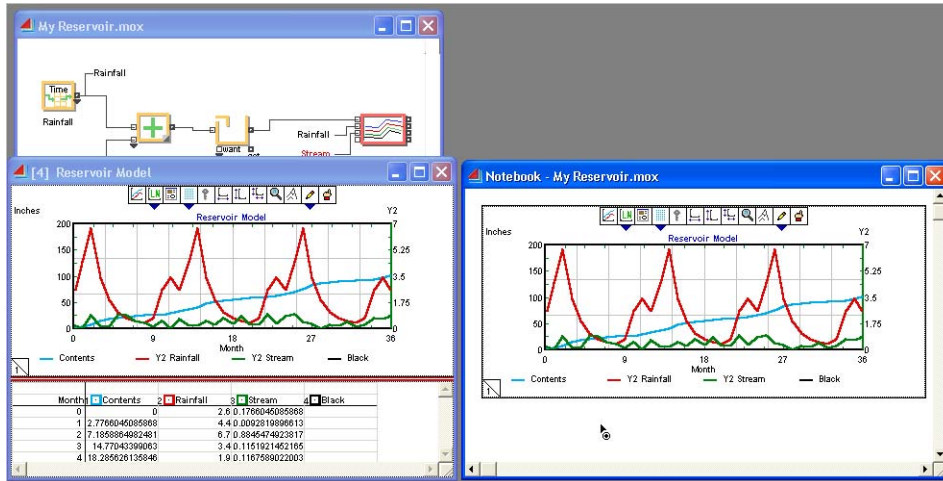
To clone a plot from the Plotter block to the Notebook:

- ▶ Select Window > Notebook or select the Open Notebook tool in the toolbar.

An empty Notebook window opens.

- ▶ Double-click the Plotter block in your reservoir model.

- ▶ Using the **Clone layer** tool  from the toolbar, click the plot (graph portion) of the plotter and drag it to the Notebook window.



Cloning plot onto Notebook

- ▶ Run the simulation again.

The plot in the Notebook is the same as the graph on the Plotter.

For more information, see “Cloning” on page 504.

## Other modifications

Since it is so easy to add and modify elements in ExtendSim models, there are many ways you can enhance them. Here are just a few examples.

- Add blocks to represent more sources of water entering the reservoir.
- Add an additional Plotter to see the various lines on different plots—for example if the scales of the results are very different and you do not want to plot against the Y2 axis. There are four ways to create a second instance of a Plotter:
  - Insert another Plotter block from the Library menu, as you did earlier in this chapter.
  - Insert another Plotter block from the Plotter library’s library window, as discussed at “Library Window mode” on page 671.
  - Copy and paste the existing Plotter.
  - Select the Plotter, choose Edit > Duplicate, and move the duplicate Plotter to the desired location.

You can use these same techniques to duplicate any block in the model. Note that if you copy or duplicate a model’s existing block, any dialog settings for that block will also be copied.

- Configure the Holding Tank block so that it outputs water over time. For an example of this, go to “Tutorial” on page 63.
- Run the model with Show 2D animation enabled to display the water level in the Holding Tank block. Learn more, see “Blocks with built-in animation” on page 551.

- Set delta time to a value less than 1, as discussed on page 83. This calculates output values between the steps, so you can see finer resolution of the model results.

### Next steps

You have learned the basic techniques for running and building models and some additional techniques for enhancing your models. The following are some suggestions on which sections of the manual to explore next, depending on what your own simulation requirements are.

- The Tutorial module's "Simulation Concepts" chapter discusses more general simulation and modeling concepts and how ExtendSim can be used for all types of modeling. It describes the three main modeling methodologies (continuous, discrete event, and discrete rate) as well as modeling approaches such as Monte Carlo, Agent Based, and State/Action.
- If you already know the type of modeling you want to do, the specific modules are:
  - "Continuous Modeling" starting on page 59.
  - "Discrete Event Modeling" starting on page 89.
  - "Discrete Rate Modeling" starting on page 265.
  - "3D Animation" starting on page 389.
- The "How To" module starting on page 488 has several chapters that show how to use ExtendSim to do common modeling tasks (such as creating a user interface, analyzing model results, etc.).



# Tutorial

## Simulation Concepts


Learn about systems, simulation,  
and modeling methodologies

*“It must be remembered that there is nothing more  
difficult to plan, more doubtful of success, nor more  
dangerous to manage, than the creation of a new system.”*  
— *Niccolo Machiavelli*

The first two chapters of the Tutorial showed how to build and run simulation models in ExtendSim. Since you have seen some of what can be accomplished with ExtendSim, now is a good time to explore some modeling and simulation concepts. The following discussion is meant to familiarize you with modeling and simulation terminology and concepts used throughout this guide. This chapter:

- Explains modeling concepts and terminology
- Discusses model types and common approaches to modeling
- Describes the modeling process, including goals and steps
- Shows how to verify and validate a model

After reading this chapter you will have a better grasp of modeling concepts and will be ready to start using ExtendSim for your modeling needs. Note that an in-depth exploration of simulation is beyond the scope of this document. For more detailed definitions and theory, please refer to the numerous books on simulation.

 If you are already familiar with the concepts to be presented in this chapter, skip it and proceed to one of the other modules, as discussed in “About this User Guide” on page 6.

## Systems, models, and simulation

All professions use models of one form or another. But the word “model” does not always have the same meaning to business professionals, managers, scientists, and engineers. Even within a specific discipline, such as manufacturing, modeling has many different definitions. The following discussion serves to clarify what “modeling” means as it relates to ExtendSim.

### Systems

The real world can be viewed as being composed of systems. A system is a set of related components or entities that interact with each other based on the rules or operating policies of the system:

- Entities are the internal components of the system. Entities are involved in processes—activities in which they interact with each other.
- Operating policies—the types of controls and availability of resources—are the external inputs to the system. They govern how the system operates and thus how the entities interact.

Over time, the activities and interactions of entities cause changes to the state of the system; this is called system behavior or dynamics. Systems can be mathematically straightforward, such as a flower growing in the soil and turning towards the sun to maximize photosynthesis. Or they can be more complex, such as supply chain operations composed of planning, selling, distribution, production, and sourcing subsystems.

### Models

A model is an abstracted and simplified representation of a system at one point in time. Models are an abstraction because they attempt to capture the realism of the system. They are a simplification because, for efficiency, reliability, and ease of analysis, a model should capture only the most important aspects of the real system.

Most models can be classified into four basic types:

- A scaled representation of a physical object, such as a 1:18 diecast model of a Ferrari, a clay model of a proposed packaging bottle, or a scale model of the solar system.

- A graphical or symbolic visualization, such as a flow chart of office procedures, the board game Monopoly (which represents the hotels and facilities of Atlantic City), or an architect's plans for a building.
- An analytical or mathematical formula that yields a static, quantitative solution. For instance, an analytic model might consist of several independent sample observations that have been transformed according to the rules of the model. Common examples of analytic models are spreadsheet models or linear programming models.
- A mathematical description that incorporates data and assumptions to logically describe the behavior of a system. This type of model is typically dynamic—it has a time component and shows how the system evolves over time. ExtendSim products are tools for building mathematically-based, dynamic models of systems.

Dynamic modeling is the foundation for computer modeling. Thus, for purposes of this manual, the word “model” will be used to mean a description of the dynamic behavior of a system or process.

ExtendSim models typically have a time component and can show cause and effect and the flow of entities throughout a system (you can also create ExtendSim animations that show spatial relationships.)

### Simulation

The Merriam-Webster OnLine Dictionary defines simulation as “the imitative representation of the functioning of one system or process by the functioning of another.” This means that to determine how an actual system functions, you would build a model of the system and see how the model functions.

Simulations run in simulation time, an abstraction of real time. As the simulation clock advances, the model determines if there have been changes, recalculates its values, and outputs the results. If the model is valid, the outputs of the simulation will be reflective of the performance or behavior of the real system.

Simulation with ExtendSim means that instead of interacting with a real system you create a logical model that corresponds to the real system in certain aspects. You simulate the operations or dynamics of the system, then analyze one or more areas of interest. You do this in order to reduce risk and uncertainty so that you can make informed, timely decisions.

### Modeling methodologies

The formalism you use to specify a system is termed a modeling methodology. The three main modeling methodologies are:

- Continuous
- Discrete event
- Discrete rate.

These methodologies are described, compared, and contrasted in the later topics in this chapter.

In addition to the main modeling methodologies listed above, other modeling approaches are useful and will be discussed in this chapter. These approaches are usually based on one of the three main methods and include:

- Monte Carlo

- Agent-based
- State/Action

For more information, see “Other modeling approaches” on page 47.

As you might expect, you can use different methods to model different aspects of real-world systems. For example, at a chemical plant you could model the chemical reactions as a continuous process, the control logic of the chemical process using discrete event modeling, and the tanks, valves, and flow of the production process with discrete rate.

It is good to note, however, that there is no such thing as “the” model of a system: a system can be modeled in any number of different ways, depending on what it is you want to accomplish. In general, how you model the system depends on the purpose of the model: what type, level, and fidelity of information you want to gather and the amount of detail, or level of abstraction or granularity, of the model. Once that has been determined, you can intelligently choose which type of model to build.

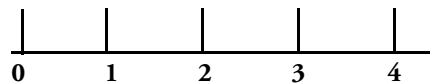
 The types of models that can be built depend on the ExtendSim product that was purchased.

### Comparison of main modeling methodologies

The three main modeling methodologies are continuous, discrete event, and discrete rate. Continuous modeling (sometimes known as process modeling) is used to describe a flow of values. Discrete event models track unique entities. Discrete rate models share some aspects of both continuous and discrete event modeling.

In all three types of simulations, what is of concern is the granularity of what is being modeled and what causes the state of the model to change.

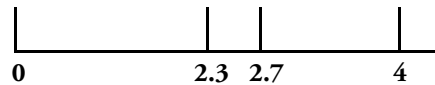
- In *continuous* models, the time step is fixed at the beginning of the simulation, time advances in equal increments, and values change based directly on changes in time. In this type of model, values reflect the state of the modeled system at any particular time, and simulated time advances evenly



Time line for continuous simulation

from one time step to the next. For example, an airplane flying on autopilot represents a continuous system since its state (such as position or velocity) changes continuously with respect to time. Continuous simulations are analogous to a constant stream of fluid passing through a pipe. The volume may increase or decrease at each time step, but the flow is continuous.

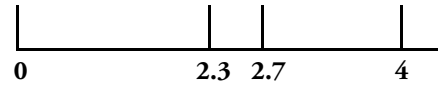
- In *discrete event* models, the system changes state as events occur and only when those events occur; the mere passing of time has no direct effect on the model. Unlike a continuous model, simulated time advances from one event to the next and it is unlikely that the time between



Time line for discrete event simulation

events will be equal. A factory that assembles parts is a good example of a discrete event system. The individual entities (parts) are assembled based on events (receipt or anticipation of orders). Using the pipe analogy for discrete event simulations, the pipe could be empty or have any number of separate buckets of water traveling through it. Rather than a continuous flow, buckets of water would come out of the pipe at random intervals.

- *Discrete rate* simulations are a hybrid type, combining aspects of continuous and discrete event modeling. Like continuous models they simulate the flow of stuff rather than items; like discrete event models they recalculate rates and values whenever events occur. Using the pipe analogy for a discrete rate simulation, there is a constant stream of fluid passing through the pipe. But the rates of flow and the routing can change when an event occurs.



Time line for discrete rate simulation

In some branches of engineering, the term *discrete* is used to describe a system with periodic or constant time steps. Discrete, when it refers to time steps, indicates a continuous model; it does not have the same meaning as discrete event or discrete rate. Continuous models in ExtendSim are stepped using constant time intervals; discrete event and discrete rate models are not.

### Comparison table

The three main modeling methodologies are summarized in the table below.

Modeling method	ExtendSim library	What is modeled	Examples
Continuous time	Value library Electronics library	Processes	Processes: chemical, biological, economic, electronics.
Discrete event	Item library	Individual items	Things: traffic, equipment, work product, people.  Information: data, messages, and network protocols at the packet level.
Discrete rate	Rate library	Flows of stuff	Rate-based flows of stuff: homogeneous products, high speed production, data feeds and streams, mining.

### Table of continuous, discrete event, and discrete rate differences

Although not definitive, the following table will help to determine which style to use when modeling a system.

Factor	Continuous	Discrete Event	Discrete Rate
What is modeled	Values that flow through the model.	Distinct entities (“items” or “things”).	Bulk flows of homogeneous stuff. Or flows of otherwise distinct entities where sorting or separating is not necessary.
What causes a change in state	A time change	An event	An event

Factor	Continuous	Discrete Event	Discrete Rate
Time steps	Interval between time steps is constant. Model recalculations are sequential and time-dependent.	Interval between events is dependent on when events occur. Model only recalculates when events occur.	Interval between events is dependent on when events occur. Model only recalculates when events occur.
Characteristics of what is modeled	Track characteristics in a database or assume the flow is homogeneous.	Using attributes, items are assigned unique characteristics and can then be tracked throughout the model.	Track characteristics in a database or assume the flow is homogeneous.
Ordering	FIFO	Items can move in FIFO, LIFO, Priority, time-delayed, or customized order.	FIFO
Routing	Values need to be explicitly routed by being turned off at one branch and turned on at the other (values can go to multiple places at the same time.).	By default, items are automatically routed to the first available branch (items can only be in one place at a time.)	Flow is routed based on constraint rates and rules that are defined in the model (flow can be divided into multiple branches.)
Statistical detail	General statistics about the system: amount, efficiency, etc.	In addition to general statistics, each item can be individually tracked: count, utilization, cycle time.	In addition to general statistics, effective rates, cumulative amount.
Typical uses	Scientific (biology, chemistry, physics), engineering (electronics, control systems), finance and economics, System Dynamics.	Manufacturing, service industries, business operations, networks, systems engineering.	Manufacturing of powders, fluids, and high speed, high volume processes. Chemical processes, ATM transactions. Supply chains.

Some systems, especially when a portion of the flow has a delay or wait time, can be modeled using any of the three styles. In this case, you would generally choose how to model the system based on the level of detail required. Discrete event models provide much more detail about the workings of these types of systems than continuous models. Continuous and discrete rate models, on the other hand, usually run faster than discrete event models.

Remember that you may combine blocks from different libraries within the same model. For example, it is quite common to use continuous blocks from the Value library when creating a discrete event model. However, the discrete event blocks in the Item library and the discrete rate blocks in the Rate library can only be used in event-driven (non-continuous) models. If you use any discrete event or discrete rate blocks in a model, the timing will change to event driven (time steps will not be periodic) and it will not be a continuous model.

## Other modeling approaches

Although there are several other approaches to modeling, they usually fit within one of the three major categories (continuous, discrete event, or discrete rate) discussed above. For example, System Dynamics and Bond graphs are subsets of continuous modeling, and queueing theory models are subsets of discrete event modeling.

Because of their specialized use, three specific modeling approaches (Monte Carlo, State/Action, and Agent Based) are described below.

### Monte Carlo modeling

Widely used to solve certain problems in statistics, Monte Carlo simulations provide a range of results rather than a single value. This approach can be applied to any ExtendSim model and used wherever uncertainty is a factor.

Monte Carlo modeling uses random numbers to vary input parameters for a series of calculations. These calculations are performed many times and the results from each individual calculation are recorded as an observation. The individual observations are statistically summarized, giving an indication of the likely result and the range of possible results. This not only tells what could happen in a given situation, but how likely it is that it will happen.

You build a Monte Carlo simulation in ExtendSim by incorporating random elements in a model and obtaining multiple observations. There are two ways to do this:

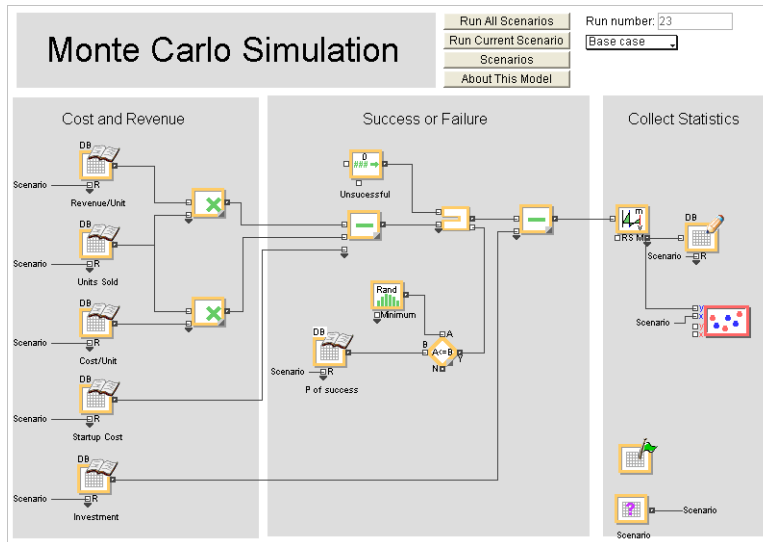
- The classical Monte Carlo method is to take a single mathematical equation or set of equations, then cause the equation to be calculated many times. In this type of simulation, time is not a factor. The entire model is run to completion and evaluated at each step; each subsequent step performs a new calculation. An example is the Monte Carlo model, discussed later in this section.
- An alternative Monte Carlo approach, typically applied in a discrete event model, is to either divide a single simulation run into multiple sections (*batch means*) or run the simulation many times (*multirun analysis*). Monte Carlo is incorporated by adding randomness to the model, running it many times, and analyzing the results. This method can be applied to any continuous, discrete event, or discrete rate model. It is shown in the Queue Statistics model, described later in this section. For more information about using the Statistics block (Value library) for performing batch means or multirun analysis, see “Statistics” on page 564.

### Monte Carlo model

An example of the classical method is the Monte Carlo model. This model determines the expected revenue from a new product. It runs for 10,000 steps, from time 0 to time 9999, and each step results in an observation. This cycle is repeated 24 times, once for each of the cases.

For scenario experimentation purposes, the inputs and outputs for this model are stored in an ExtendSim database.

Tutorial

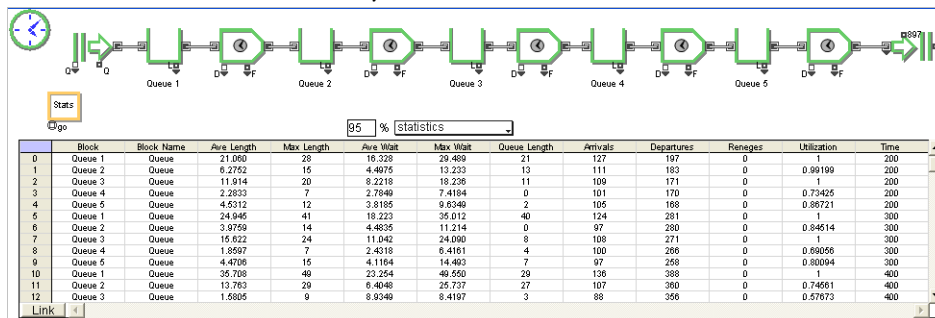


Monte Carlo model

The Monte Carlo model is located at \Examples\Continuous\Standard Block Models.

### Queue Statistics model

The Queue Statistics model is an example of an alternative Monte Carlo modeling approach. It applies batch means analysis to a discrete event model. The model uses the batch-means method to collect multiple observations of the queuing statistics. Every 100 time units a new set of observations are recorded. Information (such as the maximum and average queue length, the number of arrivals and departures, and utilization) is stored and displayed in a table in the dialog of the Statistics block (Value library).



Queue Statistics model

The Queue Statistics model is located in the folder \Examples\Discrete Event\Statistics. It is not available with ExtendSim CP.



## State/Action models

With *state/action* modeling a system is modeled as a collection of discrete states. Sometimes known as a state chart, a state/action model represents a system that responds to an event by transitioning to another state. The model is composed of a series of states where each state depends on a previous state. A state has an associated action and an event that will cause that state to change to another. The transition from one state to the next is not sequential; each state can lead to any other state.

There are rules that govern the communication and transition between the states:

- All states accept events.
- One or more states may create an event as a result of a transition by another state or group of states.
- A group of states can be set to transition conditionally, for instance to only change if another state or group of states achieve a specific stage. These are known as guard conditions.

State/action models are independent of any of the three modeling methodologies (continuous, discrete event, or discrete rate.) They are useful for specification and verification in many areas, from computer programs to business processes.

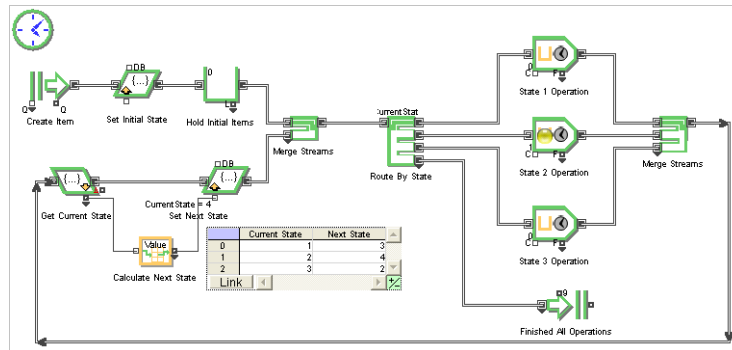
In ExtendSim, the most common ways of creating state/action models are:

- Define one or more discrete event items as objects with behavior that is determined by their states. The information about each state and its next state is stored in a Lookup Table block (Value library) or an ExtendSim database table. This uses ExtendSim's internal event queue and scheduling capabilities to signal and manage events for the item/objects within the system. This method can only be used with discrete event models and is illustrated in the State Action model, described later in this section.
- Store each state, action, event, and next state for the system in rows in an ExtendSim database table. This maps the states for the entire model into one block and works with any type of model. This method can be used with continuous, discrete event, and discrete rate models and is shown in the Markov Chain Weather model, described later in this section.
- Create new blocks that store their current state in a static variable and send messages to other blocks at appropriate state change events. To do this, use ExtendSim functions and its simulation modeling environment to create a custom block. For more information about creating new blocks, see the Developer Reference.

### **State Action model**

In the State Action model, items are created with attributes that determine the item's state. The items are then routed to one of three operations depending on their state. After processing, the

item's state is changed based on entries in a Lookup Table block (Value library). The item continues to be routed to various processes until it reaches state 4, at which point it leaves the simulation.



State Action model

Initially, each item has a CurrentState attribute with a value of 1. The Lookup Table block causes each item with CurrentState 1 to be changed to CurrentState 3 after processing, then to CurrentState 2, and finally to CurrentState 4. The operations are represented by Workstation blocks, which can hold and process the items. After each operation, the item is examined and its state is transitioned accordingly.

Running the simulation with animation on shows the items changing from state 1 (green), to state 3 (red), then state 2 (yellow), and finally state 4 (blue).

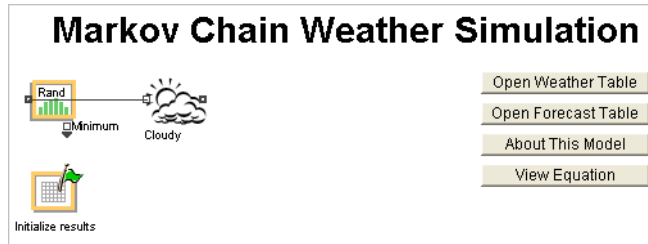
The State Action model is located in the folder \Examples\Discrete Event\Routing. It is not available with ExtendSim CP.

### Markov Chain Weather model

A Markov chain represents a transition from one state to another as defined by a table of probabilities. The Markov Chain Weather model simulates the weather based on a Markov chain. The states are the weather – sunny, cloudy, rainy, and so forth; they are stored in an ExtendSim database named “Weather”. The model runs for 365 days. Each day there is a probability of transitioning from one weather state to the next. For example, if today is sunny, the next day could be sunny (50%), partly cloudy (30%), cloudy (10%), light rain (5%), or rainy (5%). As the model runs, the states move through a probability table, changing the weather for each day.

Most of the calculation in this model is done by a single Equation block (Value library) inside the hierarchical “Weather Forecast” block. In the equation, a random input and the previous state (the output of the equation) are used to lookup a probability for the next day’s weather in the Weather database. The number and percent of days at each weather state is also calculated and recorded in the database. Additionally, if the model is run with animation on, the current weather state is ani-

mated on the icon of the Weather Forecast block. It does this by showing different icon views, depending on the state. (For information about icon views, see page 496.)



Markov Chain Weather model

The Markov Chain Weather model is located in the folder \Examples\Continuous\Standard Block Models.

### Agent-based models

Most of the models discussed in this User Guide represent a system where the behavior of the components of the system are known or can be estimated in advance. With agent-based modeling you usually do not know model dynamics in advance; instead, you obtain that information from the interaction of the agents in the model.

Agent-based models share the following characteristics:

- The identification of individual entities within the model
- A set of rules that govern individual behavior
- The premise that local entities affect each other's behavior

Agent-based modeling is concerned with individual entities (called “agents”) that interact with other agents within their specified locality. All the agents have a set of rules to follow but they also have a degree of autonomy such that model dynamics cannot be predefined. This is because agents can have intelligence, memory, social interaction, contextual and spatial awareness, and the ability to learn.

### Programming for agent-based models

The agents used in agent-based modeling are programmed as ExtendSim blocks. Blocks and their enclosed data have unique searchable identities and locations within the model. ExtendSim functions can find and send messages to blocks that have specific characteristics, locations, and values. This makes it easy to create intelligent behavior, facilitate block-to-block interaction, and cause blocks to be moved in, added to, or removed from, a model.

The Developer's Reference includes several categories of functions that are helpful when creating agents for agent-based modeling:

- Scripting functions are used to build a new model or to add or remove blocks from an existing model. They do this by creating, placing, and connecting blocks, then populating the blocks with specific data. These functions can be called from an ExtendSim block within the model or from an external application.
- Block and inter-block communication functions query the status of a block – its type, label, data, location, size, and connectivity with the rest of the model. They also get information about block dialog values and data table settings.

- Message sending functions can use the results of inter-block communications to send messages globally to unconnected blocks or blocks that are connected in specific ways.
- Animation functions provide a visual indication of block-to-block interaction, such as the influence of one block on another.

For example, in constructing an agent-based model of the robotic clean up of a chemical spill, you could use the inter-block communication functions in a “Controller” block to locate all of the “Robotic Clean Up” blocks in the model. The Controller could send messages to the robots asking them to move towards a spill and clean it up. The robots could send messages back to the Controller stating whether they were available or were currently being recharged, and whether they were too far from a chemical spill or close enough to be useful. The scripting and animation functions would show the robot blocks physically moving around within the model and the spill being removed.

### ***The Game of Life***

The Game of Life was devised by British mathematician John Conway in 1970 and published as an article in Scientific American. It is the most well known example of cellular automata (CA), a type of modeling studied in computability theory, mathematics, theoretical biology, and other fields.

A CA model represents a regular grid of finite state automata (cells) that sit in positional relationships to one another, with each cell exchanging information with the eight other cells to which it is horizontally, vertically or diagonally adjacent. A cell can be in one of a finite number of states and the state of a cell at time  $t$  is a function of the states of its neighboring cells at time  $t-1$ . Every cell has the same rule for updating; each time the rules are applied to the whole grid a new generation of cells is produced.

You interact with the Game of Life by specifying an initial configuration of effects and observing how the CA universe evolves. At each step in time, the following happens:

- A cell is born if it has a specified number of neighbors who act as parents.
- “Loneliness” causes any live cell with fewer than a specified number of neighbors to die.
- “Overcrowding” causes any live cell with more than a specified number of neighbors to die.

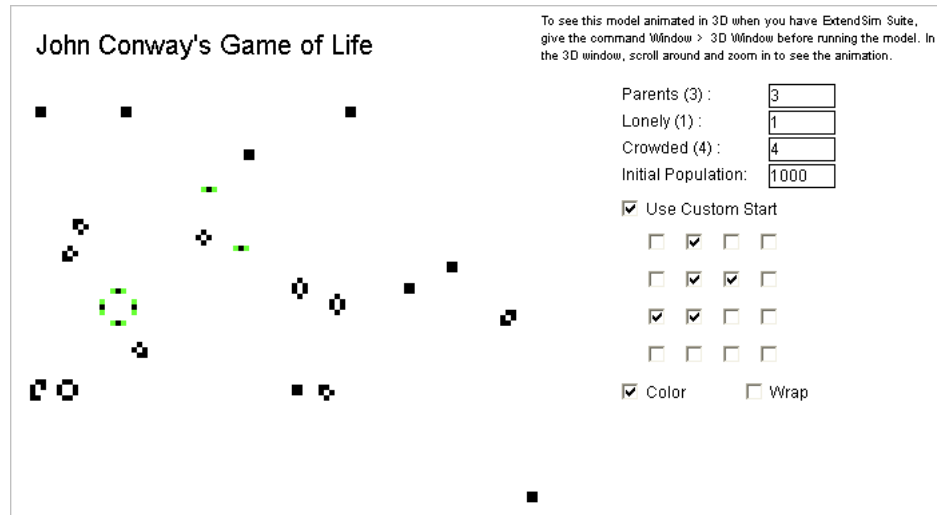
The initial pattern constitutes the first generation of the system. The second generation is created by applying the above rules simultaneously to every cell in the first generation. In other words, births and deaths happen simultaneously. The rules continue to be applied repeatedly to create further generations.

Life has a number of recognized patterns that emerge from particular starting positions, including static patterns (“still lifes” such as block and boat), repeating patterns (“oscillators” such as blinker and toad), and patterns that translate themselves across the board (“spaceships” such as gliders).

### ***The Life model***

The one-block Life model was created using the Life block (Custom Blocks library) that was specifically developed for this model. The code of the block contains the algorithm for Conway's

Game of Life. The block's dialog has fields for specifying initial settings and rules; the dialog items have been cloned to the model worksheet for convenience.



Life model

The concept for this model is that each cell of the grid is defined as living or empty. On each generation, a given cell can give birth to a new life, survive, die, or remain empty. Using the default settings in the Life block, the model adheres to the following rules:

- Count the number of neighbors a given cell has (the maximum possible is 8).
- If an empty cell has 3 neighbors, it will produce a new life (birth).
- If a full cell has less than 1 or zero (loneliness), or 4 or more neighbors (overcrowding), it will die.

Changing the default rule values causes some interesting affects on the population.

There are two ways to set the starting population for the model:

- Define an initial number of cells (1000 is a reasonable starting population for the size of this block.) The cells will be populated randomly.
- Use the Custom Start grid to select up to 16 initially populated cells in specific locations. This is a quick way to begin with a recognized pattern, such as a glider or a blinker.

One feature of the Life block that is not specified in Conway's algorithm is that the color of the cells varies with the age of the cell - new cells are green and older cells vary from light gray to black as they age.

#### Variations

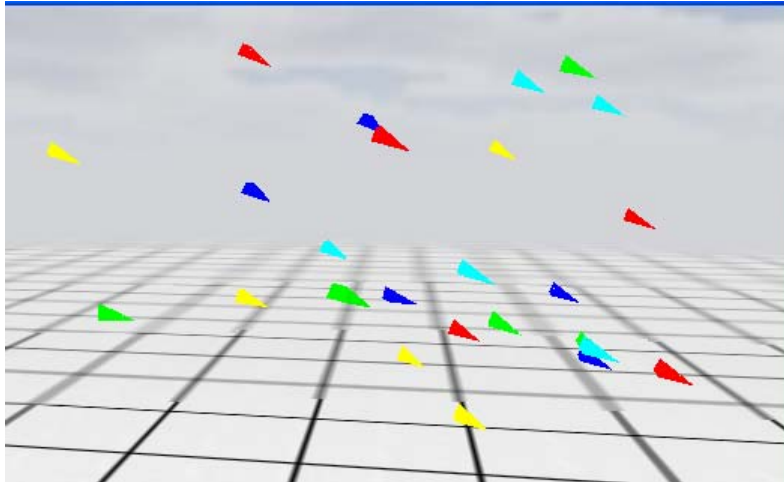
The Life block is open source so you have complete access to the dialog editor and block code. To see the underlying structure of the block, select it on the model worksheet and give the command Develop > Open Block Structure. The procedures that define cell birth, death, or survival are listed at the top of the block's structure window.

The Life model is located in the folder \Examples\Agent Based. The Life block is located in the Custom Blocks library.

### **Boids**

The Boids model is based on an artificial life program, developed by Craig Reynolds in 1986, that simulated the flocking behavior of birds. It can also be applied to schools of fish, herds of animals, or any other type of flocking behavior.

☞ The Boids model requires 3D animation; it requires ExtendSim Suite to run. The model is located in the folder \Examples\3D Animation.



Boids model in E3D window

In the model, each bird is an individual agent that interacts with other local agents based on a set of rules:

- Separation – birds steer to avoid crowding their local flock mates.
- Alignment – each bird steers towards the average heading of its local flock mates.
- Cohesion – birds steer toward the average position of their local flock mates.

### **Other agent-based models**

Additional agent-based models, including “Sheep and Wolves” and “Breakout” are located in the folder \Examples\Agent Based.

## **The modeling process**

An ExtendSim simulation project involves creating a logical model of the system, running the simulation, analyzing the data, optimizing the solutions, and interpreting and presenting the results.

### **Goals of modeling**

The Introduction chapter gave several examples of what you can do with simulation. As stated in *Modeling Tools for Environmental Engineers and Scientists* (N. Nirmalakhandan, CRC Press), the “...goals and objectives of modeling are two-fold: research oriented and management oriented. Specific goals of modeling efforts can be one or more of the following: to interpret the system, analyze its behavior, manage, operate or control it to achieve desired outcomes; to design methods to improve or modify it, to test hypotheses about the system, or to forecast its response under varying conditions.”

### The simulation process

Like all tasks, you can start modeling simply by jumping into it. Also like other tasks, it is usually better to have a plan before starting. ExtendSim makes following a plan for making a model easy. The basic steps to creating a model are:

- 1) *Formulate the problem.* You should define the problem and state the objectives of the model.
- 2) *Describe the flow of information.* Determine where information flows from one part of the model to the next and which parts need information simultaneously.
- 3) *Build and test the model.* Build the system with ExtendSim's blocks. Start small, test as you build, and enhance as needed.
- 4) *Acquire data.* Identify, specify, and collect the data you need for the model. This is usually the most time-consuming step. It includes finding not only numerical data values but also mathematical formulas such as distributions for random events.
- 5) *Run the model.* Determine how long you want to simulate and the granularity of results, then run your model.
- 6) *Verify the simulation results.* Compare the model results to what you intended or expected.
- 7) *Validate the model.* Compare the model to the real system, if available. Or have system experts evaluate the model and its results.
- 8) *Analyze your results.* Draw inferences from the model's results and make recommendations on how the system can change.
- 9) *Conduct experiments.* Implement and test recommended changes in the model.
- 10) *Document.* State the model's purpose, assumptions, techniques, modeling approaches, data requirements, and results.
- 11) *Implement your decisions.* Use the results in the real world.


### Before you build a model

Remember that building a model is an iterative process and each step in the process will require comparing the model to the existing system, analyzing the results, and refining the model. A natural inclination is to immediately start building the model. However, you will end up with more useful models if you begin the model-building process by asking a few basic questions, such as:

- *What is the goal of the model?* It is important to determine the purpose of a model. This will indicate the levels of detail required and will help keep you focused.
- *What are the boundaries of the model and what level of detail should be included?* The model goal should dictate what to include in the model and what to leave out.
- *Where is the required data?* It is useful to start collecting data early in the model building process because it can often take a while to obtain all of the necessary information. You will also need to know if the input data is an absolute value or if the data is from a statistical distribution. Addi-

tional data requirements may surface once the model building process has begun. Your model may, for example, lead you to explore alternatives that had not been considered before.

- *How shall the model be conceptualized?* Before even running ExtendSim, think about what the various components of the system represent. Roughly determine the time delays, resource constraints, flows through the system, and any logical actions that occur in the model. This will help you determine how to build the model.
- *What alternatives will be investigated?* Although the model may lead you into new, unexpected directions, try to think ahead so that the model can be easily changed from one alternative to the next.

 It is common to use a constant or a uniform (integer or real) distribution in the early stages of model building so that modeling problems and variations can be more easily detected. After the model is verified, you can easily change the distributions to correspond to real-world processes.

### Refining models

It is important to remember that models may not give you a single “correct” answer. Instead, they make you more aware of gaps in your thought process. These problems may involve over-simplification in the model, false assumptions on your part when creating the model, or missing connections between parts of a model. Refining your model step by step helps eliminate these and other pitfalls.

Every model can be made more complex by adding assumptions and interconnections. The model-building process commonly begins with the creation of a simple model. After analyzing the simple model, complexity is added, followed by further analysis, the addition of more complexity, and so on. The complexity takes one of two forms:

- Taking one block (a process) and turning it into many blocks (a more complex process)
- Adding a connection between two previously unrelated blocks, usually through a mathematical operation (finding an interconnection between two processes)

At each step, look at your results and make sure they make sense relative to the data. If you can, verify the results in the real world. If one result is way off, check the output from each step to determine where the process went awry.

### Model verification

The process of debugging a model to ensure that every portion operates as expected is called model verification. In the tutorial, you performed part of this verification process by building the model in stages and with minimal detail, then running it at each stage to observe the results. A common verification technique could be termed *reductio-ad-absurdum* (reducing to the absurd), which means reducing a complex model to an aggressively simple case so that you can easily predict what the outcome will be. Some examples of “reducing to the absurd” are:

- Remove all variability from the model, making it deterministic
- Run the deterministic model twice to make sure you get the same results
- Output detailed reports or traces to see if the results meet your expectations
- Run a schedule of only one product line as opposed to several
- Reduce the number of workers to 1 or 0 to see what happens



- Uncouple parts of the model that interact to see how they run on their own
- Run very few or very many items through the model to determine if the model responds properly.

Other methods for verifying models include making sure that you can account for all the items in a model, animating the model or portions of the model, or using diagnostic blocks from ExtendSim's libraries. For more information, see "Debugging Tools" on page 613.

### Model validation

Once the model is verified you need to validate it to determine that it accurately represents the real system. Notice that this does not mean that the model should conform to the real system in every respect. Instead, a valid model is a reasonably accurate representation based on the model's intended purpose. When validating, it is important to make sure that you know what to compare to and that you verify that measures are calculated in the same manner.

For validation, your model should accurately represent the data that was gathered and the assumptions that were made regarding how the system operates. In addition, the underlying structure of the model should correspond to the actual system and the output statistics should appear reasonable. While you would normally compare critical performance measures, it is also sometimes helpful to compare nonessential results that may be symptomatic and therefore show the character of the system.

One of the best validation measures is "Does the model make sense?" Other methods involve obtaining approval of the results by those familiar with the actual process and comparing simulation results with historical data. For example, when validating model performance compared to historical data, try to simulate the past. If you have sufficient historical data, break the actual system performance into various windows of time, where all of the input conditions correspond to the input conditions for multiple runs of your model.

For more information, see "Debugging Tools" on page 613.

## Additional modeling terminology

In addition to the following general information, each of the modules in this User Guide has a section with tips specific to that module. For additional general information about using ExtendSim, see also the How To module that starts on page 488.

### Model parameters, variables, inputs, and outputs

A parameter is any numerical characteristic of a model or system. Parameters describe something about the model and are known or can be estimated.

- An input parameter is a value that is required as part of the model specification.
- An output parameter is a value determined by the input parameters and the operation of the system—output parameters specify some measure of the systems performance or system dynamics.

### Constant values and random variables

You enter parameter values in block dialogs to specify settings for a model. Constant values never change; random values are based on distributions and change each time they are used. Models that have no random input parameters are referred to as deterministic models. Models that are based on one or more variables that are random are said to be stochastic, as discussed below:

- *Deterministic* models contain only non-random, fixed components. No matter how many times a deterministic model is run, unless some parameter is changed there is no uncertainty and the output will be exactly the same. Thus the behavior of the model is “determined” once the inputs have been defined.

The advantage of a deterministic model is that only one run is necessary, since it produces an exact measurement of the model's performance. It is also helpful in when initially building a model since you can be assured that changes in results will be due to changes made to the model and not to randomness. The disadvantage is that these types of models can only accurately be used to model a few types of processes, since real-world systems typically contain some element of randomness.

- Adding randomness to one or more inputs to a deterministic model changes it to a *stochastic* or *Monte Carlo* model. Stochastic models are run repeatedly and then analyzed statistically to determine a likely outcome. Notice that the occurrence of randomness does not mean that the behavior of a process is undefinable or even that it is unpredictable. Random variables vary statistically as defined by a distribution. This means that their range and possibility of values is predictable.

While stochastic models can be applied to very complex systems, a disadvantage is that the output is itself random—the average of the simulation runs provides only an estimate of the model's true behavior.

ExtendSim provides several methods for including randomness in models. For instance, as you saw in the chapter “Building a Model”, the Random Number block (Value library) allows you to select a random distribution or enter a table of values which specifies an empirical distribution of probabilities. For more detailed information about ExtendSim's random number capabilities, see “Random numbers” on page 604.

# Continuous Modeling

## Introduction

Some things to know before you begin  
modeling continuous systems

*“A journey of a thousand miles begins with a single step.”*  
— Confucius

The Continuous Modeling module is focused on building models where time advances in equal steps and model values are recalculated at each time step. It is also a helpful reference if you use continuous blocks in discrete event and discrete rate models.


### How the Continuous module is organized

- Introduction
  - Blocks for building continuous models
  - Application areas in which continuous modeling is commonly used
  - Next steps
- Tutorial
- Application areas and examples
- Continuous concepts, tips, and techniques

### Blocks for building continuous models

To create continuous models you can use:

- Blocks from continuous libraries that are packaged with ExtendSim
- Continuous blocks that you create
- Libraries of continuous blocks developed by third parties

 You can use continuous blocks and the ExtendSim database to build State/Action models, as discussed in “State/Action models” on page 49.

### Using the ExtendSim blocks

You can easily build continuous models in many fields using only the continuous libraries (Value and Electronics) included in every ExtendSim product. The blocks in those libraries allow you to perform complex modeling tasks often with just the click of a button. For example, you can use a popup menu to specify a distribution in the Random Number block or to select a function in the Math block; both blocks are in the Value library.

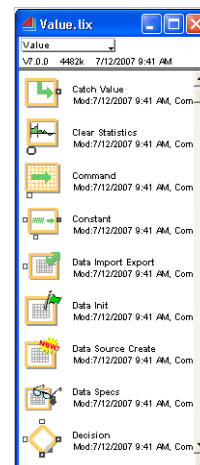
For added power and flexibility, the Equation block (Value library) allows you to enter formulas and equations to calculate values for models. The Equation block gives you access to over 1,000 internal functions. You can also use operators to enter logical statements (if a then b), write compound conditions (if  $a \geq 0.5$  or  $b \leq 0.5$  then  $c = 8$ ), and even specify loops (do task while  $input \geq 5$ ).

 See “Value Library Blocks” on page 715 for a listing and brief description of the blocks in the Value library.

If your model becomes too cluttered with blocks, you can encapsulate portions of it into a hierarchical block, then double-click the hierarchical block to see the submodel. Hierarchical blocks are created using simple menu commands and can be stored in libraries for reuse in other models. They are discussed in “Hierarchy” on page 540.

### Building custom continuous blocks

Continuous simulation is used in a broad number of diverse fields, and it would be impossible to supply one solution that would meet everyone’s needs. To provide complete flexibility, all



Value library window

ExtendSim products include a development environment for creating custom blocks. You can create your own blocks—even your own libraries of blocks—and use them to build models. Because you have the source code for the blocks that are packaged with ExtendSim, you can adapt an existing block to your needs or create an entirely new block from scratch.


ExtendSim has an integrated, compiled development environment, so it is easy to build blocks with custom dialogs and behavior. And because the development environment is optimized for simulation and user-interface design, you can build blocks with less effort and more flexibility than by using a traditional programming language. ExtendSim's development environment has the functionality you need to create blocks that can:

- Process data, perform calculations, and show results in numerical, graphical, and animated form
- Communicate with other ExtendSim blocks and with external applications
- Interact with the user

By programming your own blocks you can:


- Obtain specific behaviors not available in the blocks included with ExtendSim
- Combine the functionality of several ExtendSim blocks into one custom block for increased calculation speed and convenience. (Note that this is different than using hierarchy to encapsulate several distinct blocks as a submodel within one block.)
- Develop a library of blocks for a specific discipline, such as for control systems or paper-making processes
- Design your own modeling architecture

Blocks you create can be saved in libraries and used throughout your models just as you would use any of the standard ExtendSim libraries. And the blocks that are packaged with ExtendSim, such as the Value library, are designed to work well as supplements to any custom blocks you may develop.

 The ExtendSim Developer Reference has all the information you need to program your own blocks.

### Third-party libraries

Third-party developers use the ExtendSim environment to create libraries of blocks customized for specific fields. For more information about third-party libraries, please go to [www.extendsim.com/prtnrs\\_developers.html](http://www.extendsim.com/prtnrs_developers.html).

 Discrete event or discrete rate blocks built by third parties will not run with the ExtendSim CP product.

### Application areas

Computer simulation is indispensable for understanding, analyzing, and predicting the behavior of complex and large-scale systems. It is used to gain an understanding of the functioning of existing systems and to help design new systems by predicting their behavior before they are actually

built. The following table gives some of the most common areas where continuous modeling is used.

Discipline	Fields	Applications
Science	Biology, Biotech, Chemistry, Ecology, Genetics, Mathematics, Medicine, Pharmaceuticals, Physics	Chemical reaction kinetics, thermodynamics, paper making, population dynamics, growth/decay, competition/cooperation, chaos, genetic algorithms
Engineering	Aerospace, Agricultural, Automotive, Control Systems, Electronic, Environmental, Forestry, Material Science, Mechanical, Mining, Nuclear, Petroleum	Hardware design verification, electro-mechanical systems, neural networks, adaptive systems, algorithm validation, signal processing
Business	Finance, Information Technology, Inventory Management, Human Resources, Operations	Forecasting, credit risk analysis, asset pricing, derivatives trading, data flow and sharing, inventory replacement strategies, resource allocation, process flow
Social Sciences	Economics, Gender Studies, Migration, Psychology, Social Dynamics, Urban Studies	Econometric studies, trade policies, relationship modeling, finite capacity planning, economic booms and recessions, immigration policies

Continuous

### Next steps

The next chapter in the Continuous Modeling module provides a tutorial that expands upon the Reservoir 1 model used in the guide's Tutorial module. Chapter 3 describes typical industries and applications for continuous simulation such as scientific, engineering and business; it uses models provided with ExtendSim as illustrations. The final chapter of the Continuous Modeling module discusses concepts specific to continuous simulations and give additional tips when building models.

The How To module that starts on page 488 includes chapters on topics relevant to all types of modeling, including creating a custom user interface, using mathematical and statistical functions, and statistically analyzing models.

# Continuous Modeling

## Tutorial

Building a more complex  
continuous model

*“Be the change you want to see in the world.”*  
— *Mahatma Gandhi*

The Tutorial module showed how to build the Reservoir model. This chapter illustrates some additional modeling techniques to enhance that model:

- Removing content from a Holding Tank block if it exceeds a specified limit
- Using the Equation block to replace the functionality of several blocks
- Adjusting delta time (dt) for more accurate simulation results

If you haven't already done so, it is recommended that you go through the chapters in the Tutorial module (starting on page 13) to familiarize yourself with the basic techniques for building models and running simulations.

Example models for comparison to the Overflow model you will build in this chapter are located in the `ExtendSim7\Examples\Tutorials\Continuous` folder. Reservoir 2 shows a series of blocks that calculate and remove the overflow and Reservoir 3 is the same model using an Equation block to perform the calculations.

## Removing overflow from the Holding Tank

The Tutorial chapter “Building a Model” on page 23 assumed that the reservoir had an infinite capacity. In this chapter, the model is modified so that there is a limit on how much water the reservoir can hold and overflow is removed. To do this you need to add model elements to:

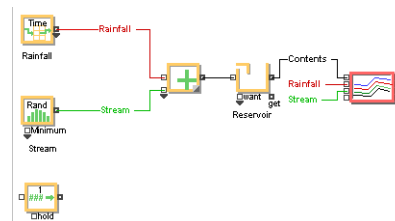
- Establish a maximum reservoir capacity
- Compare the contents of the reservoir to the maximum limit
- Determine if and when the contents exceed the limit
- Calculate the amount to remove if there is any excess
- Remove the excess water

With these changes, the reservoir would behave more like an actual reservoir with a dam where water spills over if it reaches the top.

### Setting the maximum capacity

- ▶ From the `\Examples\Tutorials` folder, open the model **Reservoir 1**.
- ▶ So that you don't overwrite the original model, give the command `File > Save Model As` and save the model as “Overflow”.
- ▶ In the Overflow model, add a Constant block (Value library) at the bottom left of the model.

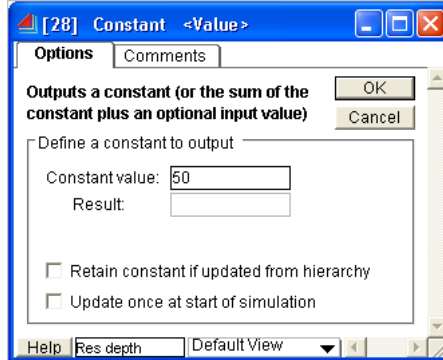
This block will represent the overflow limit for the reservoir, in this case, 50 inches.



Constant block added to Overflow model



- ▶ In the dialog, set the **Constant** value to **50** and enter **Res depth** as the block's label.



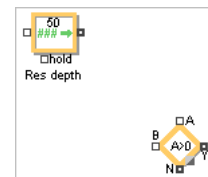
Constant block dialog

- ▶ Close the dialog.

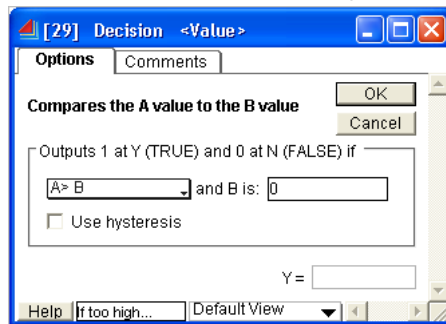
### Determining if there is too much water

Add elements to the model to determine at each step if the capacity is exceeded or not.

- ▶ Add a Decision block (Value library) below the Constant block and to its right.
- ▶ In the Decision block's dialog, choose **A > B** (the default setting) from the popup menu. Label the block **If too high...**



Adding a Decision block

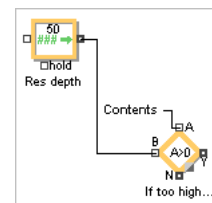


Decision block dialog

- ▶ Close the dialog.
- ▶ Add a named connection (**Contents**, the same as the connection from the output of the Holding Tank block) to the Decision block's **A** input.
- ▶ Connect from the Constant block's output to the **B** input of the Decision block.

When the model runs, the Constant will set the **B** value to **50**.

During the simulation run, the Decision block will evaluate whether or not the value of the Contents (**A**) is greater than the value of the maximum Reservoir Depth allowed (**B**). If yes, it will assign a value of **1** to the **Y** output connector. If no, it will assign a value of **0** to the **N** output connector.



Decision connected

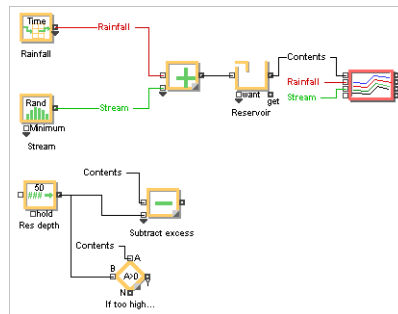
### Comparing contents to overflow limit

To calculate the difference between the capacity limitation and the tank's contents:

- ▶ Add a Math block to the right of the Constant block, set its function to **Subtract**, and label the block **Subtract excess**.

In addition to selecting functions directly in the Math block's dialog, you can choose settings by right-clicking near the sensitized area (looks like a partially turned page) on the lower right of the block's icon.

- ▶ Add a **Contents** named connection to the top input of the Subtract block.



Calculating the excess water

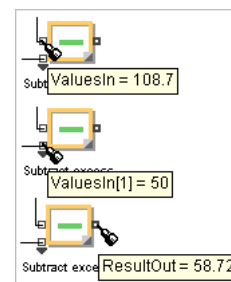
- ▶ Connect the output of the Constant block to the bottom input of the Subtract block.

### Validating intermediate results

When building a model, it is good practice to frequently test if the model is working correctly. Even though this model isn't yet finished, validate that model elements are calculating as you would expect them to.

- ▶ Run the simulation.
- ▶ Click the model window to make it the active window.
- ▶ Hover the cursor over the connectors of the blocks you've added to see the results.

For example, the top input of the Math block shows the current contents from the Holding Tank, the bottom input shows the Constant value (50), and the output connector shows the result of the subtraction. (Note that your results may be slightly different from those at right because the Stream source is random.)



Subtraction results

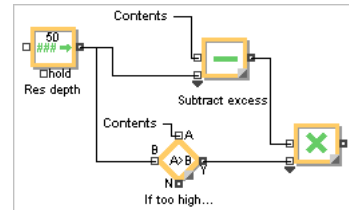
### Calculating how much water to remove

The Decision block's Y connector outputs 1 (one) if the tank's contents exceeds the limit, but outputs 0 (zero) if it doesn't. This information can be used in a calculation.

- ▶ Add another Math block to the model, set its function to **Multiply**, and label the block **Then overflow**.

- ▶ Connect from the output of the Math block labeled “Subtract excess” to the top input of the Math block labeled “Then overflow”.
- ▶ Connect the Decision block’s **Y** connector to the bottom input of the Math block labeled “Then overflow”.

This multiplies the amount of excess water, if any, by the Decision block’s **Y** output (1 or 0) to determine the amount of water that should be removed from the Holding Tank at each step.



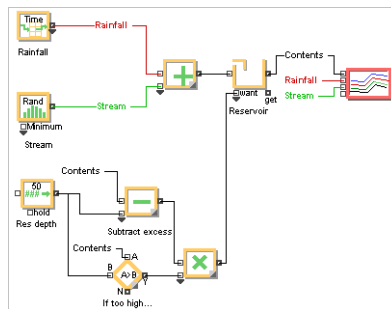
Multiply block connected

You do not need to connect the Decision block’s **N** connector to anything since, if the reservoir’s contents is less than capacity, nothing further needs to be done.

### Removing the overflow

When the contents of the Reservoir/Holding Tank are greater than its capacity, water needs to be removed.

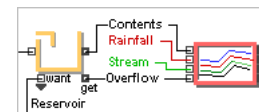
- ▶ Connect the output of the Multiply block to the **w** (want) input connector on the bottom left of the Holding Tank.



Requesting overflow amount

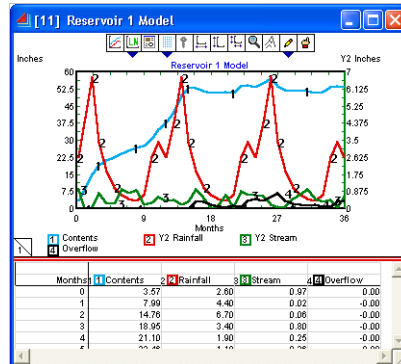
The Holding Tank’s **want** input connector is used to request an amount to be removed. If the tank has that amount, it will be reported at its **get** output connector. If the tank does not have that amount, and “Tank contents can be negative” is not checked, only the amount available will be at the **get** connector. Since this model is only concerned with overflow, the amount requested and the amount available will be the same.

- ▶ Create a named connection called *Overflow* from the **get** output of the Holding Tank block to the fourth input on the Plotter.



Overflow connection

- ▶ Run the simulation.



Simulation results

Scroll through the Plotter's table of data to see the point where the reservoir is beginning to reach its capacity. Column 4 (Overflow) shows the amount of water that overflows. Because the inflows continue even while the excess is being removed, and because there is a calculation delay, it is unlikely that the reservoir will be at exactly 50.

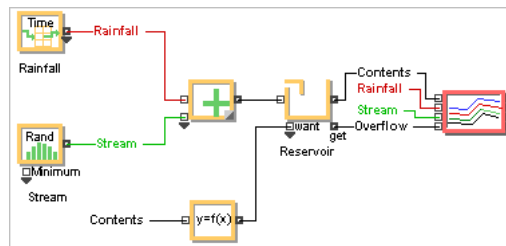
The equivalent model is Reservoir 2, located in the folder \Tutorials\Continuous.

### Simplifying the model

Although the model works perfectly well, the blocks that make up the calculation of the overflow can be a bit confusing to follow. You can easily replace the functionality of those four blocks with a single Equation block that explicitly defines the mathematical expression.

#### Adding an Equation block

- ▶ Delete the Math blocks labelled "Subtract Excess" and "Then Overflow", as well as the Constant and Decision blocks from the Overflow model.
- ▶ Add an Equation block (Value library).
- ▶ Add a **Contents** named connection to the Equation block input.
- ▶ Connect the Equation block output to the **w** connector of the Holding Tank block.



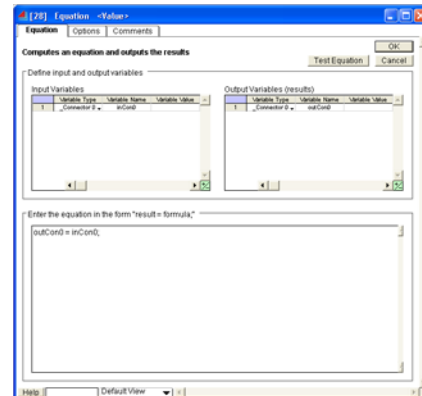
Model with Equation block

### Specifying input variables

- ▶ Open the Equation block dialog.

In the **Input Variables** table at the left of the dialog, note that by default **Connector 0** is selected from the popup menu in the **Variable Type** column. This is the option you want, because the equation will get its values from the top input connector. To give the input a more relevant name:

- ▶ Type **Contents** in the **Variable Name** field to the right of Connector 0.



Equation block dialog

### Specifying output variables

In the **Output Variables (results)** table at the right of the dialog, note that by default **Connector 0** is selected from the popup menu in the **Variable Type** column. This indicates that the results of the equation will be sent to the top output connector, which is the option you want for this example. To give the output a more relevant name:

- ▶ Type **Overflow** as the **Variable Name**.

### Entering the equation

- ▶ In the equation pane, delete the default equation and enter the following code and comments:

```
real reservoirDepth; // define a new variable as a real number
reservoirDepth = 50.0; // set its value

if (contents >= reservoirDepth) // if contents is >= depth...
    overflow = contents - reservoirDepth; // then calculate outflow.
else
    overflow = 0.0; // If the contents is too low, outflow = 0.
```

☞ Comments, which are preceded by //, are optional but helpful for documentation.

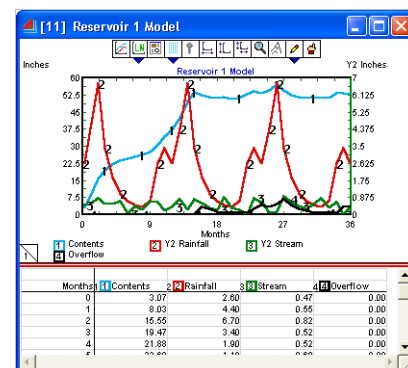
- ▶ Enter **Calc overflow** for the block's label.
- ▶ Close the Equation block's dialog. ExtendSim displays a message that it is compiling the equation; the message may appear too quickly for you to see it.
- ▶ Run the simulation.

Note that the results haven't changed - you've simply substituted the Equation block for the deleted blocks.

For more information, see "Equation-based blocks" on page 601.

### Improving the accuracy of the model

Because the contents of the Holding Tank block is used to calculate the amount of water to remove from the



Final simulation results

Holding Tank, the model incorporates feedback. When a model has feedback, the default delta time (dt) of 1.0 is too long and results won't be accurate. (See "Feedback and delays" on page 84 for more information.)

- ▶ Choose the command Run > Simulation Setup > Continuous tab and change the **Time per step (dt)** value to **0.1**.
- ▶ Close the Simulation Setup dialog.

Notice that the Holding Tank is properly already set to integrate its inputs, rather than sum them. Summation only gives the same results as integration when delta time is exactly 1.0, but integration works with any delta time. Learn more at "Integration vs. summation in the Holding Tank block" on page 610.

When delta time is small, such as for this model, simulation results will be more realistic if Lookup Table inputs are interpolated rather than stepped.

- ▶ In the Lookup Table block (labeled Rainfall), change **Output is:** to **Interpolated**. This setting smooths the change between rows in the table.
- ▶ Run the simulation again.

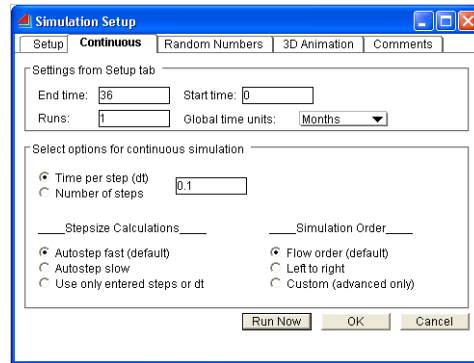
Because this model is small and only includes one feedback loop, there is only a slight different in the results between this run and the previous one with the larger dt value. However, in a larger model where there are many instances of feedback, the second run would produce much more accurate results.

The equivalent model is Reservoir 3, located in the folder \Tutorials\Continuous.

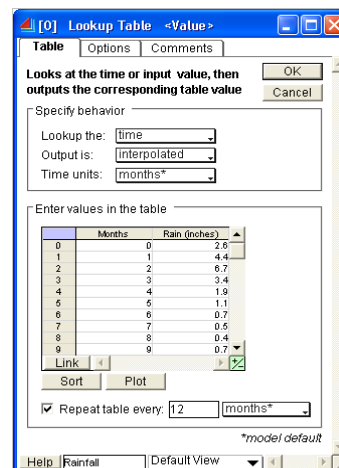
### Next steps

The next chapter describes typical areas where continuous modeling is applied; the final chapter in this module discusses concepts specific to continuous modeling and provides additional tips to help you build and run continuous models.

You should also explore the How To module, beginning on page 487. Those chapters illustrate many more of the features and capabilities you might use when creating models, such as sensitivity analysis, controls, and optimization.



Simulation setup dialog



Lookup Table dialog

# Continuous Modeling

## Areas of Application

Some of the many ways continuous modeling is used

*“(Science is)...the separation of the true from the false by experiment or experience.”  
— Richard P. Feynman*

ExtendSim is used to build continuous models in the fields of physical science, social science, engineering, and business. The first four sections of this chapter describe models created using the ExtendSim Value and Engineering libraries. The last section shows two models built with custom blocks: a physics model called Planet Dance and a Fish Pond simulation that examines predator/prey interactions.

Continuous modeling is very broad and diverse field. While it is possible to create almost any continuous model using the standard blocks (such as the Value library) packaged with ExtendSim, it is more common to use a combination of standard blocks and custom-built blocks. For example, ExtendSim customers have created libraries for proprietary and commercial use in the fields of analytical chemistry, environmental decision making, chemical process control, pulp and paper making processes, and so forth.

## Scientific

Common areas where simulation is helpful in science include biology, chemistry, physics, and ecology. This section describes two models, a classic predator/prey model and a simulation of some factors that affect drug absorption in the human body.

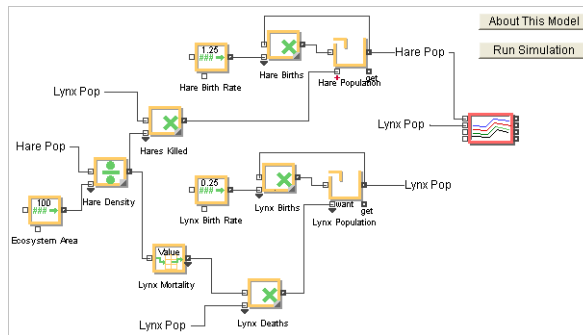
### Predator/Prey

This model shows a small ecosystem composed of hare and lynx. Each population has a direct effect on the other: the lynx feed on hare so the hare population declines, the diminishing food supply results in a decrease in the number of lynx so the hare population grows again and so forth.

#### Model assumptions

The model particulars and assumptions are:

- The 100-hectare ecosystem initially contains 6000 hare and 125 lynx.
- Each hare produces 1.25 offspring per year and each lynx produces 0.25 offspring.
- Hare always have sufficient food supply; their only cause of death is being eaten by a lynx.
- The lynx hunting range is 1 hectare, their only food source is the hare, and each lynx can consume every hare in its area.
- The number of hare killed depends on their density in the ecosystem and the number of lynx who hunt them.
- The lynx mortality rate depends on how many hare they consume in a year.
- There are no outside factors (such as the time of year) which affect the rate at which the populations change.
- The final model runs for 24 years and calculations are done 12 times per year (dt is 0.0833).



Predator\_Prey model

The Predator\_Prey model is located in the folder \Examples\Continuous\Standard Block Models. This model uses blocks from the Value and Plotter libraries.



### Model details

This model illustrates some interesting continuous modeling concepts:

- Because the model has feedback, the Holding Tank blocks are set to *Inputs are: integrated (no delay)*. For more information, see “Integration vs. summation in the Holding Tank block” on page 610.
- The Holding Tank block has a *want* input connector that is used to tell the block how much of the contents you want removed. The amount wanted and the amount that is actually removed (indicated at the *get* connector) may differ, as discussed below.
- By default, the Holding Tank block is set to not allow its contents to become negative. This means that the population of hare cannot be reduced below the available amount, no matter what is requested through the *want* connector. The amount that is actually removed is reported at the *get* output connector. If the tank were allowed to become negative, the amount at the *want* connector and the amount at the *get* connector would be the same. Since it can't go negative in this model, the amount at *want* could be higher than the amount actually removed and reported at *get*.
- Dividing the hare population by the area of the ecosystem (100 hectares) determines how many hares there are per hectare. The lynx can eat every hare in each hectare, but if they eat too many and the hare population decreases, the lynx mortality rate increases. This contributes to the cycles seen in the model.
- Notice that, at each point in time, the result in the Hares Killed block is higher than the hare population shown on the plotter. This occurs because hares are constantly being born and the lynx kill not just the previous step's population but also some of the new births.
- The Lookup Table block is set to *Output is: interpolated*, which means that intermediate values can be used. For example, an input value of 65 (which is halfway between the *Hares Eaten* values of 60 and 70) will cause the block to output 0.175 (which is halfway between the *Lynx Death Rate* of 0.2 and 0.15).

### Further exploration

If you review the initial assumptions for this model, you can probably see several enhancements that could be made.

- The birth rates for both the hare and the lynx could vary based on model conditions or on outside factors. For instance, the birth rate might be dependent on the health of the parents, the level of crowding, or the amount of pollution in the ecosystem.
- You might add a predator or an additional food source for the lynx.
- The model assumes an unlimited food supply for the hare. To a food source, the hare would be considered the predator, so modeling a food source would follow the same logic as adding the lynx predators.
- Hares could have predators other than lynx.
- You could factor in outside conditions, such as the time of year and expected weather conditions. Then examine the effects of those conditions on mortality rates.

### Drug Ingestion

There really can be too much of a good thing. For some drugs, such as blood thinners, it is important that the patient get enough of a dose to cause the desired outcome, but not so much as could

be harmful. One method is to monitor the concentration of the drug in the patient's bloodstream to determine if it is at effective, but safe, level. Many factors influence drug absorption - the amount ingested, the rate of absorption, the patient's diet, and so forth - and simulation is the best method to explore the effect of those factors.

The Drug Ingestion model is located in the folder \Examples\Continuous\Standard Block Models. This model uses blocks from the Value and Plotter libraries.

**Model assumptions**

- Constant blocks specify the dosage amount (1500 mg) and frequency (3 doses/day)
- The stomach's volume is 500ml and the absorption percentage is 0.693
- Blood volume is 5,000ml and the metabolic constant 0.0433
- The model runs for 96 hours; delta time is 0.25

**Model details**

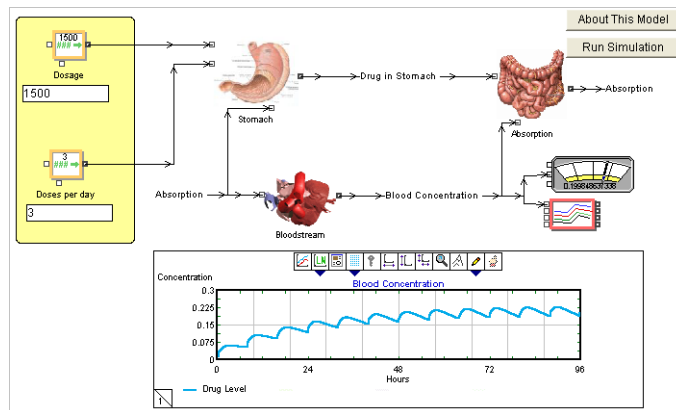
The Drug Ingestion model illustrates the bloodstream concentration for a periodically ingested drug. The drug is taken at even intervals and is absorbed into the bloodstream based on the amount of the drug ingested, the specific rate of absorption for the drug, and the metabolism, stomach volume, and blood volume of the person involved.

Hierarchy compartmentalizes the Drug Ingestion model into submodels representing the stomach, intestines, and

blood stream. (To see the contents of these hierarchical blocks, double-click their icons or use the Navigator in Model mode to drill down to the underlying layers.) In the *Stomach* hierarchical block, the drug is introduced with a Pulse block that generates a periodic pulse based on the value from the Constant block that outputs the number of doses per day. This is then multiplied by the drug dosage and placed in a Holding Tank block representing the amount of drug in the stomach.

In the *Absorption* section, an Equation block calculates the absorption rate by dividing the amount of drug in the stomach by the stomach volume, subtracting the concentration of the drug in the bloodstream, and multiplying that by the absorption constant and the stomach volume constant. The absorption rate is then used to reduce the amount of the drug in the stomach and increase the amount of the drug in the bloodstream (represented by the Holding Tank in the hierarchical block named *Bloodstream*). The concentration of the drug in the bloodstream is then calculated by dividing the amount of the drug in the bloodstream by the blood volume. Finally, the amount of the drug in the bloodstream is reduced by metabolism, calculated in the Equation block that uses the concentration of the drug in the bloodstream, the blood volume, and a metabolism constant. A Plotter I/O reports the concentration of the drug in the bloodstream.

Continuous



Drug Ingestion model

### Variations


Try changing the dosage amount and the frequency of application using the cloned dialog parameters on the left side of the model. Then observe the changes in the concentration of the drug in the bloodstream. Or, to illustrate the affects of different drugs, vary the metabolism and absorption constants.

## Engineering

Simulation is extensively used for modeling electronic, signal processing, control, and mechanical systems, as well as neural networks and other engineering systems. The following electronic signal processing example investigates the performance of a receiver in a digital FM system.

### Noisy FM system

When designing receivers and demodulators, engineers need to balance quality and cost. Simulation helps illustrate the trade-off, for example when the objective is to filter noise without reducing the quality of the reception. Fewer components can result in a product that is cheaper to produce and maintain, but a product with more components might have a better sound.

 The Noisy FM System model is located in the folder Examples\Continuous\Standard Block Models. The model uses blocks from the Electronics and Plotter libraries.

#### Model assumptions

- FM center frequency is 91.6khz
- Antenna tuner uses an elliptical filter
- Demodulator is a passive RC phase locked loop using an exclusive OR comparator

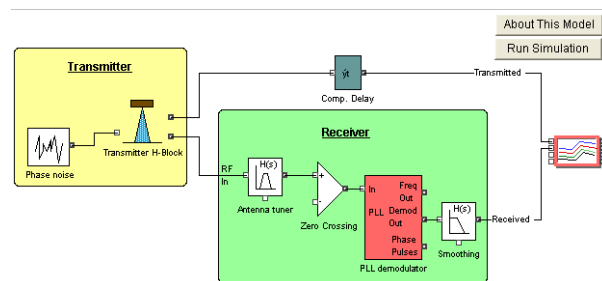
#### Model details

This model shows the performance of a receiver in a digital FM transmitter system, within a noisy environment. The Transmitter is a hierarchical block with a submodel containing a bitstream generator and an FM carrier generator. The bitstream generator is comprised of two Voltage Controlled Oscillator blocks (VCOs) and two Clipper blocks:

- The first VCO outputs a square wave that varies between -1 and +1 volt output.
- The first Clipper limits the square wave signal to 0 and +1 volt, so that it can modulate the second VCO and produce the periodic bitstream for testing.
- A second Clipper limits this square wave output for the Frequency Modulation (FM) generator.

The Clippers can be eliminated, but limiting the VCO input voltage to between 0 and +1 volt makes it easier to enter frequency modulation range parameters.

The Filter-Bandpass block (labeled Antenna tuner) is set to elliptical because of its economy (fewer poles & zeroes yield the fewest components). The Comparator block after the antenna acts as a limiter and detector of the FM signal. The Phase Locked Loop (PLL) block demodulates (separates) the input signal from the FM carrier signal. The output from the loop has a lot of pulses on



Noisy FM model

it, so it is filtered slightly by a simple Butterworth filter. The low-pass filter after the PLL smooths out the switching transients always present in PLL outputs.

### Variations

Because elliptical filters have very large group delays at their edges, a Chebyshev might be better. Try setting the noise to 1.0 volts and running the simulation with the elliptical filter. Then try setting the filter type to Butterworth, Chebyshev, click “Recalculate poles...” and rerun the simulation. The Chebyshev option works better than the elliptical filter, but notice how many poles are generated and how complex (costly) the filter will become.

The PLL's parameters can be easily changed for experimentation. For example, enter different bandwidths, damping, loop types, and phase comparators, and running the simulation again. Playing with the amount of noise on the signal can give interesting results. For example, increasing the amplitude in the Noise Generator block will quickly destroy the signal.

## Business

Finance, economics, inventory management, and marketplace competition all lend themselves to analysis through simulation. The following example explores how to minimize costs while still being able to provide products based on customer demand.

### Inventory Management

A company holding inventory incurs both ordering costs and holding costs:

- Ordering costs include order processing, labor transportation, inspection, and so forth. They are generally stated as a fixed cost per order.
- Holding cost includes such expenses as warehousing, insurance, taxes, obsolescence, and management. They are generally stated as an amount per item per time period or as a percent of unit cost per time period.

These costs are involved in a classic trade-off because as the number of orders per time period increases, ordering cost increases and holding cost decreases. The objective of inventory management is to minimize the sum of the two costs.

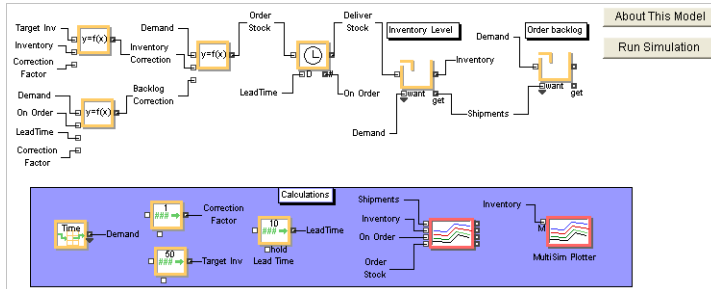
 The Inventory Management model is located at \Examples\Continuous\Standard Block Models. The model uses blocks from the Value and Plotter libraries.

### Model assumptions

- Initially there are 50 units of inventory on hand
- Demand is 10 units of product per week and increases to 12.5 units at week 4
- The lead time from stock order to stock delivery is 4 weeks for the first run
- Sensitivity Analysis automatically varies the lead time by two weeks per run, from 4 weeks on the first run to 10 weeks on the fourth run
- The stock-ordering pipeline is full at 10 units/week for the 4 weeks
- The Correction Factor is 1
- The Target Inventory is 50 (4 week lead time \* 12.5/units expected to ship each week)
- The model runs for a simulated 52 weeks

**Model details**

This model shows inventory stocking/depletion cycles. Its purpose is to stabilize the inventory level so that over- or under-stocking is avoided. The model examines the effect on inventory levels of a 25% increase in product demand. It also uses Sensitivity Analysis to explore the effect of changing the lead times for stock delivery. The MultiSim plotter displays the difference in inventory from one run to the next.



Inventory Management model

**About the model**

Customer demand for product is indicated by a Lookup Table block. The demand amount is the basis for the Customer Order Backlog amount and is also the basis for ordering more stock. Note that stock ordering, inventory replenishment, and shipments are shown as a flow, while the calculations (target inventory, etc.) are shown in a separate section. This helps clarify the model for presentation.

The demand for product determines what is shipped, unless there isn't enough inventory to meet demand. The Correction Factor indicates that a discrepancy between target and actual would be cured in 1 week (the lower the Correction Factor, the faster a discrepancy is cured.) Since you do not have Just In Time (JIT) delivery, you want to order sufficient stock to meet future demand, without over-ordering. You could order just enough to meet current demand, but because there is a delay until stock is received, inventory levels would not necessarily meet future demand (if demand increased, you'd be under stocked. If it decreased, you'd be overstocked). You could also correct the stock order amounts so that they approach target inventory levels (target inventory considers stock lead time as well as current demand).

Because the model is set to run four times for sensitivity analysis, each page of the plotter will show the results of one of the runs. Flipping from one page to another quickly shows the effect of increasing the lead time.

**Variations**

Parameters you may want to change include the Correction Factor, Target Inventory, and/or Lead time.

**Social sciences**

Simulations in the fields of psychology, social dynamics are common. The example that follows investigates the effect of available office space on new business growth in a small city.

**City Planning**

Cities and other governmental agencies forecast tax revenues, infrastructure needs, and operational expenses when setting their annual budgets. Whether predicting population changes, business usage, or housing needs, growth projection models help provide realistic estimates for budgeting

purposes. They are also helpful when developing policies such as environmental protections and residential growth ordinances.

The City Planning model is located at \Examples\Continuous\Standard Block Models. The model uses blocks from the Value and Plotter libraries.

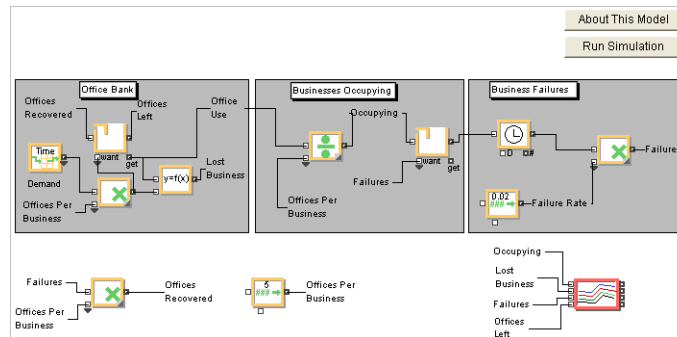
### Model assumptions

The City Planning model represents office occupancy in a city with a limited amount of offices. A projection provides the number of new businesses that will require office space each month over the next 240 months. There is also an estimate of the number of businesses that will fail.

- 4,000 offices are initially available
- Each business requires an average of 5 offices
- 2% of the businesses fail each month

### About the model

The model examines how many of the businesses will be able to occupy offices, the number of businesses lost because offices are not available, and what effect business failures have on the availability of offices. This model is notable because it mainly uses the Math block, rather than the Equation block, to calculate data. That way the relationships and calculations that occur in the model are shown explicitly.



City Planning model

Growth projections (see the Lookup Table block labeled “Demand”) are derived from a study estimating growth over a 20 year time period.

For clarity, this model is separated into sections:

- Office Bank: The Holding Tank represents the number of offices available. The initial amount of 4000 is reduced as businesses occupy offices and is increased as businesses fail. Multiplying the demand for offices by the expected number of offices required per business (5) gives the amount that will be withdrawn from the office bank. If there are not enough offices available, businesses cannot relocate to this city. These “lost businesses” are also calculated.
- Businesses Occupying: The number of offices occupied each month is divided by the number of offices required per business (5) to determine the number of businesses that occupy offices per month. This amount is accumulated to get the total number of businesses occupying offices, so that the number of business failures can be calculated.
- Business Failures: There is a 6-month delay, then 2% of the businesses fail and move out of their offices each month.

Notice that the number of failures is based on the net total number of businesses, not just on the total number of new businesses occupying offices. The failure amount is removed from the “Businesses Occupying” Holding Tank, so that failures are calculated on a net number. The number of

offices recovered due to businesses failing is returned to the Office Bank, making more office space available.

**Variations**

Vary the available office space, office space usage per business, or growth projections to explore alternatives.

**Custom blocks**

As mentioned previously, it is common to develop custom continuous blocks. The advantage is that you can use the full capability of the ExtendSim development environment, including the ModL programming language and dialog editor, to create blocks that behave and look exactly as you want. Then use those blocks to build a model that accesses ExtendSim’s robust simulation architecture.

**Planet Dance**

The Planet Dance model demonstrates the inverse square law of gravity. It uses the Planet and Planet Plotter blocks from the Custom Blocks library. Both blocks were created specifically for this model.

 The model is located in the folder \Examples\Continuous\Custom Block Models.

**About the model**

When you run the Planet Dance model, the three Planet blocks pass information about the planets they represent to the Planet Plotter block. This block plots the location of the blocks on the model worksheet by drawing them as animation objects that move over time.

Each Planet block in the model contains the definition of one planetary object whose mass will affect, and will in turn be affected by, each of the other objects in the model. The dialogs of the Planet blocks contain parameters that define the mass, position, density, and initial velocity of a particular planet. The block’s code contains the math for calculating the gravitational attractions of the objects to each other.

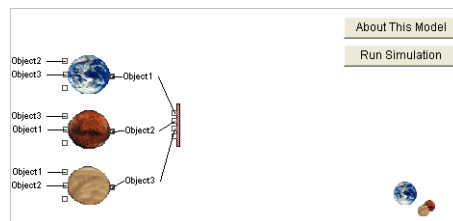
When this model runs the planets display the “slingshot effect.” As one object approaches another, the attraction between them increases, leading to the less massive object being accelerated to a high speed. Scientists sometimes use the slingshot effect to accelerate vehicles on their trips to explore the outer planets in our solar system.

Note that the units of the various parameters are not defined in this model. The numbers entered in the dialogs are just used relative to the other objects in the model.

**Variations**

This model provides lots of room to experiment with the physics of objects:

- Small changes in initial positions or velocities of the objects can cause big changes in model behavior.
- You can either add or remove an object, resulting in a four-body or two-body problem. In each case, make sure that each object has one, and only one, connection to each of the other objects on the worksheet; otherwise the math will get confused.



Planet Dance model

- One interesting problem, which demonstrates the complexities of the physics involved, is to try to modify the parameters of the objects in the three body problem to produce a system of stable orbits like the sun, earth and moon. (Before you get too frustrated trying to set this up, please note that no one has yet been able to get those parameters right.)
- Notice that even light objects will have an effect on the location and velocity of heavy objects. To replicate a relationship like that of the earth and the sun, increase the mass and density of one of the objects. No matter how much you do this, however, the heavy objects will be affected by the position of the light object.

### Fish Pond

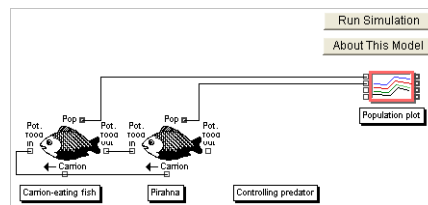
As opposed to the Predator/Prey model shown earlier that uses blocks from the Value library, the Fish Pond model uses a custom-built Fish block to represent both predator and prey.

The Fish Pond model is located in the folder \Examples\Continuous\Custom Block Models. It uses a Plotter I/O block from the Plotter library and the Fish block from the Custom Blocks library. The Fish block was specifically created for this model.

#### About the model

This is a small pond with two types of fish: a predator and a prey. The goal is to balance the pond by reducing the number of predators. The model uses a Fish block from the Custom Blocks library and a Plotter I/O from the Plotter library.

This two creature ecosystem shows how a single block design can model many types of creatures. By entering different parameters and connecting the blocks differently, a more complex ecosystem can be created. The Fish block is used to model two different species: a carrion-eating fish and a natural predator. At the left of the model, carrion-eating fish eat other fish that have died in the pond. The block to the right (a piranha) eats the carrion-eaters. Run the simulation and look at the graph; the Piranha periodically decimate the carrion-eating fish population.



Fish Pond model

#### Variations

Each Fish block added to the model represents another species. You can add another predator to the right of the second block and, based on the default parameters, it will control the piranha population. Try adding a controlling predator to reduce the Piranha population. To do this:

- ▶ Add a Fish block and position it to the right of the “Piranha” block.
- ▶ Connect the “Pot. food in” connector on the new block to the “Pot. food out” connector on the block to the left.
- ▶ Connect the “Carriion” connectors.
- ▶ Connect the “Pop” connector to the plotter.

The addition of a predator to control the Piranha population allows all the species to propagate in cycles. You can also experiment by changing the parameters in the dialogs.



# Continuous Modeling

## Concepts, Tips, and Techniques

Building robust continuous models

*“The work of adult life is not easy. As in childhood, each step presents not only new tasks of development but requires a letting go of the techniques that worked before.”*  
— Gail Sheehy

This chapter discusses some concepts specific to continuous modeling and provides tips to keep in mind when building continuous models. The areas discussed are:

- Simulation timing
- Setting delta time to determine the granularity of calculations
- Feedback and delays in continuous models
- Choosing between integration and summation
- Determining the order in which blocks execute
- Mixing libraries within a model
- Connecting to multiple inputs
- Using plotters as inputs for other blocks
- Using a plot line as reference by comparing new runs to a baseline
- Techniques to unclutter your models

☞ Concepts applicable to all types of models, such as deciding whether a model should have random elements, are discussed in the chapter “Simulation Concepts” that starts on page 41. The following topics are specific to continuous modeling.

### Simulation timing

Most simulations run for a specified time. ExtendSim determines the *duration* of a simulation run based on the values entered in the Run > Simulation Setup > Setup tab; the duration is the period from the start time to the end time.

In continuous simulations, the duration is divided into intervals or *steps* of equal length, where start time is the first step and end time is the last step. The length of time, in time units, for each step is known as *delta time* or *dt*. The delta time setting determines how frequently model data is recalculated.

As the simulation runs, simulation time advances from start time to end time at delta time per step, calculating model data at each step. At the first step, ExtendSim calculates what the status of the model is initially. Then it calculates the changes that take place over the next time step and determines a new set of data points. Model data is generated as a string of successive points corresponding to the steps in time. Notice that, for each step, data is calculated for the entire period from that step up to, but not including, the next step.

Continuous simulations require that either the number of steps or the time per step be specified. As discussed below, in most cases a delta time of 1 is adequate. However, for model accuracy it may be necessary to set a different delta time.

### Delta time

Delta time ( $dt$  or  $\Delta t$ ) literally means the change in time. It is defined as the length of the time interval between the present time and one time interval later. When you select a delta time, you are selecting how finely the total simulation time will be sliced up, i.e. how short the intervals between calculations will be and thus how frequently the computations will take place. Delta time is set in the Run > Simulation Setup > Continuous tab.

For the simulation results to be correct, the delta time for a continuous model needs to be small enough to accurately reflect changes that occur in different parts of the model. In many cases, the default delta time of 1 is adequate. However, for simulation speed or model accuracy it may be necessary to set a delta time other than 1.


### Delta times other than 1

Although 1 is the default setting in the Continuous tab, delta time can be set to any number. For example, to cause the model to run faster and perform calculations less frequently, delta time could be set to 2 or any other number larger than 1.

A delta time less than 1 reduces the step size, causing calculations to be made more frequently. It also results in more steps, so that the simulation takes longer to run in real time for the same simulation run time.

If delta time is not 1, it is most common that it would be less than 1. There are many reasons why delta time would need to be less than 1. Feedback loops and stiff equations in the blocks can require a smaller delta time to ensure that all calculations are reflected in the graph. Simulations that are run with too large of a delta time often show values jumping from very high to very low. This is known as instability or artificial chaos. Examples of models where a delta time of less than 1 may be required are:

- Signal processing models need to have their specific time per step (dt) either entered in the Simulation Setup dialog, or have it calculated by blocks in the model. For example, the Filter blocks calculate the stepsize based on their entered parameters.
- Differential equation models (models with integrators in feedback loops) may need to have a time per step (dt) or number of steps other than 1.
- If you are building custom blocks in process control models, you might set up the blocks so that they have a Stepsize message handler that can calculate the value for the DeltaTime variable, automating this process. See the Electronics library for some examples of blocks that do this.

 If a model contains the Holding Tank block (Value library) and delta time is not exactly 1, you may need to change the Holding Tank to integrate its inputs, as discussed in “Integration vs. summation in the Holding Tank block” on page 610.

### Determining which dt to use

To determine what delta time setting is reasonable, first run the simulation with a delta time of 1 (the default setting). Then run the simulation with a delta time of 0.5 (one half of the original setting). If there is no significant difference between the two graphs, then delta time of 1 is appropriate. If there is a significant difference, reduce the delta time to 0.2 and run the simulation again. Continue halving delta time until you determine a delta time which results in no significant differences compared to the smaller delta time. The main idea is to use the largest delta time that will give accurate results without slowing down the simulation unnecessarily.

### Specifying dt or the number of steps

ExtendSim requires that either the time per step (delta time) or the number of steps be specified in the Run > Simulation Setup > Continuous tab. You can enter delta time directly in the dialog or you can enter the number of steps and ExtendSim will calculate the delta time for you.

A value for the number of steps is automatically calculated based on the setting entered for **Time per step (dt)**. It is computed as:  $\text{floor}(((\text{EndTime} - \text{StartTime}) / \text{DeltaTime}) + 1.5)$ . You can see this by choosing the **Number of steps** radio button after changing the **Time per step (dt)**.

You can also determine the granularity of the simulation run by manually entering a value for **Number of steps**. In most cases, this would be a number equal to the duration (length of the simulation run) so that the model calculates values once for each step, and each step would be one

time unit long. A default value for delta time is automatically calculated based on the number of steps you enter; it is computed as  $(\text{EndTime} - \text{StartTime}) / (\text{NumSteps} - 1)$ . You can see this by choosing the **Time per step (dt)** radio button after changing the **Number of steps**.

#### **Setting the end time when delta time is 1**

Assume you want a continuous simulation to run and calculate values each year for 40 years where start time is 0 and dt is 1. The value you enter for the end time depends on whether there is integration in the model.

- If there is no integration in the model, set the end time to 39. Data will be calculated at each of the 40 steps, starting at step 0 and ending at step 39. Each step's calculation would represent data for the entire year, the period beginning at that step and continuing up to but not including the next step (one delta time unit). Thus the model would calculate 40 years worth of data. If you specified start time as 1 and end time as 40, the duration would also be 40 years.
- If there is integration in the model, set end time to 40. Data will be calculated (but not output) at each of the 41 steps, starting at step 0 and ending at step 40. Each step's calculation would represent data at the beginning of the year, the period beginning at that step. However, blocks that integrate take their inputs at one step and output their results at the next step. Thus the model would calculate 40 years worth of data. If you specified start time as 1 and end time as 41, the duration would also be 40 years.

 Integration is discussed on page 85.

#### **Setting the end time when delta time is other than 1**

If you specify delta time as 0.5, the start time as 0, and the end time as 39, the simulation will run from time 0 to time 39 calculating data for 79 steps, each of which is one half time unit long.

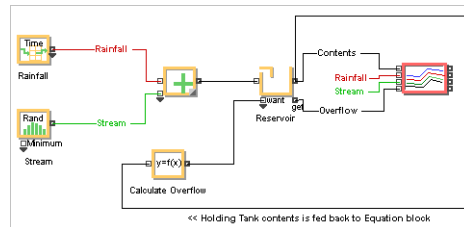
If you specify a delta time which will not result in the duration being divided into equal segments, ExtendSim will adjust the end time to a higher value. This is to insure that the segments are of equal duration, and that the simulation will end at the end time displayed in the dialog. Alternately, you can select a new end time or delta time. For instance, assuming end time is 39, you could specify the Time per step (dt) as 2 years. Data could be calculated every other year, from 0 through 38; however, the last step of the simulation (step 39) would not be calculated. In that case, rather than omit a step, ExtendSim will adjust the end time to 40. This means that model data will be calculated once every two years, starting at time 0 and ending at time 40, for a total of 21 steps.

### **Feedback and delays**

Sometimes it is necessary to create a model that tries to compensate for, or vary itself to match, changes in its inputs. For example, a public address amplifier should output the same sound level even if the speaker's voice varies in loudness. A good technique to accomplish this involves feeding back some of the amplifier's output to control its input, so that the amplifier can effectively monitor itself.

### Feedback

The practice of connecting an output of a model back to an input is called feedback; it is the main factor that causes complex behavior in continuous models. A model with feedback will cause the result of a calculation to be fed back to one of the original variables in the calculation, influencing its own rate of change. Thus feedback causes iterative changes to model variables as the simulation runs. The initial output variable ripples through the calculations such that the effect loops back and re-affects the initial variable.



Reservoir 3 model feeds contents of Holding Tank back to Equation block to calculate overflow

### Delays in feedback loops

ExtendSim is designed to facilitate feedback in models, but using feedback correctly entails some additional modeling techniques. Since continuous models are recalculated at every time step, there will be a delay between when the initial variable first affects the model and when the chain of calculations ultimately and indirectly affects the initial variable. The introduction of a delay can cause the behavior of a model to change significantly, because there is a difference between the effect that instantaneous access can cause and the response that would happen if access is not instantaneous.

When you run a model that uses feedback, the feedback is delayed by one simulation time step. This happens because the output feedback signal has to be calculated before it can be used on the next simulation step, effectively delaying it for one time step. When feedback is delayed too much, it can have a deleterious effect as it arrives too late, causing the system to over- or under-compensate. Reducing the time step or delta time of the model will reduce feedback delay and result in a model with more accurate results. See “Determining which dt to use” on page 83 for more information.

### Integration

Integration is a method of estimating the present value by estimating the past and/or future values of the inputs, calculations, and outputs in a model. It is another way of summing or accumulating values when the input is a rate (expressed as units per time). The advantage of integration over just summing values is that integration will accumulate correctly when delta time is not exactly 1.0, making it especially useful in feedback models that need a smaller delta time to work well. A simple form of integration (Euler) just multiplies the input rates by delta time and accumulates the result.

The Holding Tank and Integrate blocks in the Value library have integration capabilities. You can also use ModL functions to add integration to blocks you create. The Holding Tank block accumulates and, if wanted, releases values. Its dialog gives the option to either sum or integrate the inputs. (Whichever option you choose will have an impact on the accuracy of continuous simulations. To learn more, see “Integration vs. summation in the Holding Tank block” on page 610.) The Integrate block is used to perform the mathematical function of integration in models.

☞ There is no feedback in the My Reservoir model you built in the ExtendSim Tutorial on page 23 and it uses a delta time of 1. For that model, the Holding Tank could be set to either sum its inputs or integrate them (but the end time would need to be adjusted). The Overflow model, described in the “Tutorial” on page 63, has feedback and uses a smaller delta time, so the inputs to the Holding Tank must be integrated.

## Simulation order

The Run > Simulation Setup > Continuous tab allows you to choose the order in which ExtendSim executes block data for continuous models. (Simulation order is also used by a discrete rate system to determine initial bias order settings for Merge and Diverge blocks.)

The choices are **Flow order** (the default), **Left to right**, and **Custom**. To see the order in which blocks are executing, select the command Model > Show Simulation Order before the model is run.

 It would be unusual to change the simulation order from the default choice, Flow order.


### Flow order

During a simulation, the blocks that compose an ExtendSim model perform calculations that generally depend on their inputs. After doing their calculations, the blocks set their output connectors to the results of that calculation so that other blocks may use their results.

In this type of system, there has to be a “first” block: a block that calculates before all of the others that depend on its results. After the first block calculates, the other blocks should calculate in the order and direction of their connections. This order is repeated for every time step of the simulation. To see this order, choose Model > Show Simulation Order.

The following are the rules that ExtendSim uses to derive the order of the block calculations in continuous models:

- Blocks that generate inputs to the simulation go first. For example, Lookup Table or Constant blocks with only their outputs connected to inputs of other blocks would be put first.
- Next, ExtendSim executes blocks that are connected to those first blocks, in the order and direction of their connections.
- Unconnected blocks and bi-directional network blocks (that have only inputs connected to inputs) are executed in left-to-right order.

 The Feedback block (Utilities library) is useful when there are flow-order issues due to feedback in a continuous model.

### Left to right order

 Use caution when changing from Flow order.

If you choose this option, ExtendSim looks at the left/top corner of each block on the worksheet. The left-most block gets executed first, and the next left-most block gets executed second, and so on. Blocks with equal left edges get executed in top to bottom order. If your model flows to the right, and then continues at the left below that flow, this choice will still calculate all the left-most blocks first. If you use this order, be sure that blocks that calculate values are to the left of the blocks which need those values. Otherwise, there will be a one step delay in calculating the values.

### Custom order

 Use caution when changing from Flow order.

In continuous modeling, there are some situations with multiple feedback loops that do not automatically settle into the desired flow order solution. This occurs because there are multiple solutions that solve the DAG (Directed Acyclic Graph) ordering problem, and it is possible for the less desired solution to be picked.

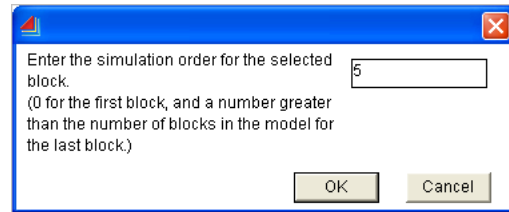
To solve this ordering problem, start with the model set to Flow order and then change the order for a few selected blocks using the Model > Set Simulation Order command.

- To use the Custom order option effectively, the Model > Show Simulation Order command should be checked so that the user can see which blocks need a changed order.

Also, note that the Set Simulation Order command is not enabled unless Custom order is selected and a block is selected.

In the Set Simulation Order dialog, you can enter a new order number for the selected block and it will be moved into that position in the simulation order.

Note that any custom order is lost if either **Flow order** or **Left to right** order is selected again.



Set Simulation Order dialog

Continuous

### Mixing block types

You can use continuous blocks, such as those from the Value library, in discrete event and discrete rate models. You cannot, however, use discrete event or discrete rate blocks in continuous models. All blocks in the discrete event and discrete rate libraries require an Executive block that changes the timing of the model to event timing.

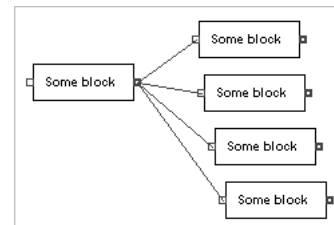
- The types of modeling you can perform depend on your ExtendSim package.

To learn more about what types of connectors can be connected to each other, see “Connector types” on page 498.

### Connections to multiple inputs

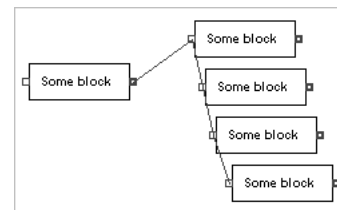
In a continuous model, you can connect from one output connector to as many input connectors as you want. Multiple inputs from one output is useful when many blocks need the results of a preceding block. For instance, if you have one output connector connected to four input connectors, the model might look like the image at top right.

- As discussed in “Event scheduling” on page 258, the connections work differently in discrete event models.



One value output connected to four inputs in a continuous model

For aesthetic reasons, you may want to only have one connection coming from the output connector, for example when the blocks are far away from each other. Instead of four connections, connect to one of the input connectors, then connect the input connectors together (sometimes called *daisy-chaining*) in the image at bottom right. (Note that this method may be confusing to others trying to understand the model.)



Four value inputs connected together

### Using plotters as inputs

The blocks in the Plotter library can store every point that was plotted in the table at the bottom of the plotter. After running a simulation, you can easily access all those points.

There are some instances where you may want to use the plotter data as input for another ExtendSim model. For example, if you have a larger model divided up into many smaller models in different files, you may want to use this technique. To quickly pass data from one continuous model to another, use the Plotter I/O block (Plotter library). That block has four output connectors that correspond to the four input connectors. To get data out of the plotter into another model, simply select the block, copy it to the Clipboard with the Edit > Copy command, select the second model, paste the plotter at the beginning of the second model, and hook the plotter's outputs to the places where you need the data.

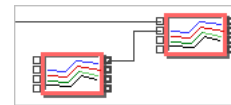
If you want to use the data in the plotter in another computer application, open the plotter, click the columns of data that you want, copy to the Clipboard with the Edit > Copy command, move to the other application, and paste the data there. If you want less than a full column, select the desired cells by clicking and dragging.

### Using a plot line as reference or standard

You may want to plot the current results of a continuous simulation relative to a previous run of a simulation that you are using as a standard. This is fairly easy with the Plotter I/O block (Plotter library). First, run the simulation to get the line that you want as a reference in the Plotter block. Select Edit > Duplicate to create a copy of the Plotter, including all the data contained within it.

Connect the output connector that is associated with the line you want to the input connector on the original plotter block. When you run the simulation again, the data that generated that line will be plotted as a reference line.

For example, assume that you want to replot the line that came from the top input connector of a Plotter. Following the steps above would make your model look like the image at right.



Second Plotter for standard line

You could also paste data directly into a column of a Plotter and use that as a reference. If you want just a straight line for a reference, instead of following the above steps, simply use a Constant block (Value library) to generate the reference line. You can also generate a reference line from a formula by selecting a function and connecting the output of the Math block (Value library) to the Plotter.

Note that these approaches are different than using the Plotter MultiSim blocks in the Plotter library. Those blocks allow you to plot the results of several runs of a simulation on a single plot.

### Uncluttering models

You can use a block to mathematically combine the output of one block with the output of other blocks. You can also use a block to feed the output of blocks into the inputs of several other blocks. However, using individual blocks to represent each function, step, or portion of a complex model can quickly cause models to become too busy. If your models end up being cluttered, combine the functioning of many blocks into a single block. To do this:

- Combine groups of blocks into a hierarchical block. See “Hierarchy” on page 540.
- Use an Equation block (Value library) to replace the calculations of several blocks. See “Equation-based blocks” on page 601.
- Create custom blocks that combine the functionality of many blocks as discussed above. See the ExtendSim Developer Reference.



# Discrete Event Modeling

## Introduction

Read this before you start  
modeling discrete event systems

*“If we all did the things we are capable of,  
we would astound ourselves.”  
— Thomas Edison*

The goal of every company, government agency, and educational institution should be to develop an extremely strong competitive organization. Cost reduction and quality improvement alone are not sufficient to achieve market share. Organizations must also be able to quickly develop and provide innovative new products and services. Invention, innovation, quality, productivity, and speed are the keys to making companies competitive.

One way to maximize competitiveness is to improve operational systems and processes by:

- Eliminating nonessential, non value-adding steps and operations
- Implementing and inserting technology where appropriate
- Managing the deployment and utilization of critical resources
- Identifying key cost drivers for reduction or elimination

In spite of recent and rapid advances in technology, many companies and institutions still suffer from outdated equipment and inefficient work practices. This is due in part to the prohibitive expense and time required to explore alternative methods of operation and try out new technologies on real systems and processes. Simulating a system or process provides a quick and cost-effective method for determining the impact, value, and cost of changes, validating proposed enhancements, and reducing the resistance to change. Simulation models allow for time compression, are not disruptive of the existing system, and are more flexible than real systems. They also provide metrics for meaningful analysis and strategic planning.

ExtendSim's graphical interface and dynamic modeling capabilities are designed to help organizations answer questions about how they do work: what they do, why they do it, how much it costs, how it can be changed, and what the effects of changes will be.

## About the Discrete Event module

The Discrete Event portion of the User Guide shows how to build comprehensive models of industrial and commercial systems so you can analyze, design, and document manufacturing, service, and other discrete processes. Build a dynamic model composed of iconic blocks, run the simulation, and analyze the results. Change aspects of the model and run it again to perform what-if analyses. Whether you model current operations or test proposed changes, the resulting models make it easy to find operational bottlenecks, estimate throughput, and predict utilization.

Discrete event modeling is an integral part of 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. They allow companies to look at their fundamental processes from a cross-functional perspective and ask "Why?" and "What If?"

## How the Discrete Event module is organized

- Introduction
- Tutorial
- Chapters that discuss specific discrete event modeling concepts:
  - Items and their properties
  - Queueing
  - Routing items from several sources and to multiple destinations
  - Processing, travel, and transportation

- Batching and unbatching groups of items
- Resources and shifts
- Activity-based costing
- Statistics
- Tips, techniques, and information about the discrete event architecture

### What the Introduction to the Discrete Event module covers

- Simulating discrete event systems
- Discrete event blocks
- Conventions and terminology for discrete event modeling in ExtendSim:
  - Overview and layout of a discrete event model
  - The Executive block
  - Items and informational values
  - Properties of items, such as attributes and priorities
  - Events
  - Activities
  - Resources
  - Block connectors
  - Closed and open systems
  - Types of item handling blocks
- Application areas in which discrete event modeling is commonly used

📖 For detailed information about discrete event modeling in general, including how it differs from continuous and discrete rate modeling, see “Modeling methodologies” on page 43.

### Discrete event systems and processes

Most systems are composed of real-world elements and resources that interact when specific events occur. The Item library simulates those systems using blocks that mimic industrial and commercial operations and timing that represents the actual occurrence of events. Use blocks from this library to create simulations of business operations, manufacturing processes, networks, service industry flows, information processing, material handling, transportation systems, and so forth.

Discrete event systems have several things in common:

- They involve a combination of elements such as people, 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. Events are what drive most businesses.

Industrial and commercial processes therefore represent the utilization and underactivity of elements and resources driven by events.

## Blocks for building discrete event models

The blocks in the Item library are used to build discrete event models. In addition, third-party developers have created customized discrete event libraries. Plus, you can program customized discrete event blocks.

### 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, transportation, and other discrete systems. They incorporate high-level modeling concepts such as variable batching, conditional routing, and preemptive operations as well as blocks that represent machines, labor, conveyors, and so forth. 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.

As mentioned in the Tutorial, blocks from the Value library are frequently used for data management and model-specific tasks in discrete event models. 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.

See “Item Library Blocks” on page 723 and “Value Library Blocks” on page 715 for a listing and brief description of the blocks in those libraries.

### Third-party libraries

Third-party developers use the ExtendSim environment to create libraries of blocks customized for specific fields, such as semiconductor manufacturing or multi-stage manufacturing systems. For more information about products that can be purchased from third party vendors, please go to [www.extendsim.com/prtnrs\\_developers.html](http://www.extendsim.com/prtnrs_developers.html).

### Creating custom discrete event blocks

Because of the Item library’s extensive capability, it is not likely that you would need to program your own discrete event blocks. If you do want to do this, it is important to note that discrete event blocks use different data structures and programming methods than continuous blocks. Start with an existing discrete event block as a base: either use a *copy* of an Item library block similar to the one you want to build, or use one of the discrete event template blocks in the ModL Tips library; those blocks’ names start with “MYO” for “Make Your Own”. The ModL code of an MYO block is commented to explain how certain features are implemented. Read the Developer Reference

Activity  
Batch  
Catch Item  
Convey Item  
Cost By Item  
Cost Stats  
Create  
Equation (I)  
Executive  
Exit  
Gate  
Get  
History  
Information  
Queue  
Queue  
Equation

Item library blocks

before modifying discrete event blocks so you have a better understanding of how those blocks work internally.


## Terminology and architecture

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 object of the simulation is to determine where there are bottlenecks in the 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 (which is required for all discrete event simulations) must be to the left of all other blocks.

### Executive block

The Executive block controls and does event scheduling for discrete event and discrete rate models. An Executive block must be placed to the left of all other blocks in a discrete event or discrete rate 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 “Executive block” on page 255.

 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.

### 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 can generate items with a random distribution, at a constant rate of arrival, at a fixed schedule, or on demand. The Resource Item block provides a finite pool of items.

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 in “Item properties” on page 94.

*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 plotter in a discrete event model you are plotting information about items, not the items themselves. For example, when the top output of an Exit block (total exited) is connected to a plotter, it displays the time that each item left the model and the number of items that have exited.

### 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 on page 115.

#### 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 on page 122.

#### 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 like the Set block. For more information, see “Quantities” on page 124.

#### 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 on page 255) 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” on page 258.

### 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 chapter “Processing” on page 163.

### 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. In ExtendSim, resources required to be present for a process or activity to take place can be modeled as either item resources that are batched with other items, or as a count of resources (a resource pool) where the count is known, managed, and made available to model processes. See “Resources and Shifts” on page 207 for complete information.

### 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 values.

Blocks in the Item library can contain value connectors as well as item connectors. 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 “Connector types” on page 498.

### 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 where 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.

**Types of item handling blocks**

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

- *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.

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” on page 256.

**Application areas**

Simulation is indispensable for understanding, analyzing, and predicting the behavior of complex and large-scale systems. It is used to gain an understanding of the functioning of existing systems and to help design new systems by predicting their behavior before they are actually built. 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, Finance, Restaurants, Hotels, Insurance, Utilities	Service levels, scheduling, throughput analysis, 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.

Discrete Event



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

**Next steps**

The next chapter is a tutorial showing how to use the Item library to build a discrete event model. Other chapters in the Discrete Event module provide some tips you may find useful when building models and illustrate specific discrete event concepts, such as item generation, assigning properties to items, and activity-based costing.

The How To module that starts on page 488 includes chapters on topics relevant to all types of modeling, including creating a custom user interface, using mathematical and statistical functions, and employing different types of analysis for your models.



# **Discrete Event Modeling**


## **Tutorial**

Building a discrete event model

The key to discrete event modeling is the construction of a flow diagram using blocks to represent the problem's operations and resources. The Item library is designed for building discrete event models of commercial and industrial processes. It is often used with other ExtendSim libraries, especially the Value and Plotter libraries.

The following example shows how to build a discrete event model of a car wash; it will use most of the important blocks in the Item library. Starting with a simple model, then adding complexity and features, this chapter will show how to:

- Model a single waiting line with a single server
- Add a second server
- Animate the model in 2D
- Route items through the model
- Add constraining resources
- Use attributes to characterize items so they can make decisions about which route to take

 This tutorial assumes you have completed the chapters in the Tutorial module that starts on page 14 and that you have read the introductory discrete event chapter that starts on page 90.

## A basic discrete event 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.

### About the model

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


- The model runs for a simulated time of 8 hours (480 minutes)
- Cars arrive approximately every 4 minutes
- There is only one route into the car wash
- There are two bays, one for washing only and one for washing and waxing
- It takes 6 minutes to wash a car; it takes 8 minutes to wash and wax a car
- Approximately 25% of the cars want to be waxed
- Cars have to be driven through the operation by an attendant
- The blocks come from the Item, Value, and Plotter libraries

 The Car Wash models are located in the folder \Examples\Tutorials\Discrete Event\Car Wash.

### Starting a model and setting simulation parameters

The following steps are typical when starting any discrete event model.

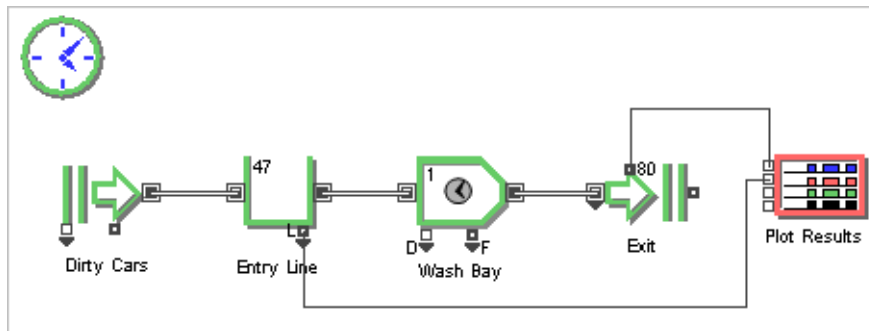
- ▶ Open a new model worksheet
- ▶ Give the command Run > Simulation Setup. In the Setup tab enter the simulation parameters:
  - ▶ **End time: 480**
  - ▶ **Global time units: minutes**
- ▶ If they aren't already open, open the Item, Plotter, and Value libraries
- ▶ Place an **Executive block (Item library)** on the top left corner of the model worksheet

 The Executive block does event scheduling and manages discrete event simulations. It must be present in every discrete event model.

### Start small

In building any simulation model, it is easiest to start with a simple subset of the process and add detail until you arrive at a completed model that approximates the system that's being modeled. This allows you to test at various stages while making the model building process more manageable.




The first step is to model the car wash with one bay that just washes cars. Since there is only one line into the car wash, each car must wait in line for the preceding car to move through the wash before it can enter. When finished with this portion of the tutorial, your model should look like the one shown below.





Modeling a waiting line with a single server


### Modeling a waiting line with a single server

The following table lists the blocks that will be added to the worksheet and their use in the model. Except for the Plotter block from the Plotter library, the blocks in the table are from the Item library.

Name (Label)	Block Function	Purpose in Car Wash Model
Create (Dirty Cars) 	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 (Entry Line) 	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 order.
Activity (Wash Bay) 	Processes one or more items simultaneously. Processing time is a constant or is based on a distribution or an item's attribute.	Washes the cars for a simulated 6 minutes.

Name (Label)	Block Function	Purpose in Car Wash Model
Exit (Exit) 	Removes items from the simulation and counts them as they leave.	Removes the cars from the model.
Plotter, Discrete Event 		Reports the length of the waiting line and how many cars have been washed.

- ▶ Starting at the right of the Executive block, place the blocks on the model worksheet in a line from left to right, based on their order in the table. The model should look like the one shown on page 101.
- ▶ Label the blocks as indicated in the table.

 An easy method for placing blocks on a model worksheet is to access an open library using the Navigator, as discussed in “Library Window mode” on page 671.

#### Entering dialog parameters and settings

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

- ▶ In the Create block’s dialog, 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.
- ▶ 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 Queue.
- ▶ 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 Activity block’s dialog, 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, just enter **Delay (D): 6**, indicating that it takes 6 minutes to wash each car.
- ▶ The Exit block automatically counts and passes items out of the simulation and the Plotter will graph results as the simulation runs. There are no settings to enter for those two blocks.

#### Making connections and running the simulation

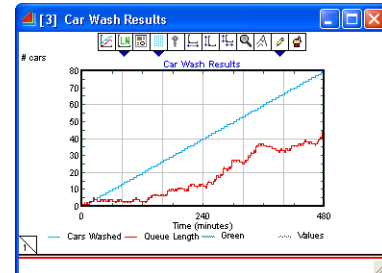
- ▶ To indicate the flow of items, connect the blocks’ *item connectors* as follows:
  - ▶ From the Create block’s item output to the Queue’s item input
  - ▶ From the Queue’s output to the Activity’s input
  - ▶ From the Activity’s output to the Exit’s input
- ▶ So that the model will display results, connect the following *value connectors*:
  - ▶ From the top (total exited) value output on the Exit to the top Plotter input
  - ▶ From the L (queue length) value output on the Queue to the second Plotter input
- ▶ Save the model.

- ▶ Run the simulation.

### Verifying results

This is a good opportunity to verify the results. Because the model has random numbers your results will differ slightly, but the plotter graph should be similar to the picture at the right.

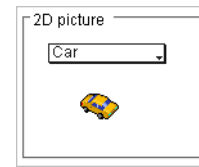
At the end of the simulation, the plotter might show that 80 cars have been washed and that there are around 40 cars waiting to be washed. This would correspond to the information in the Create block, which shows that about 120 cars were generated. These numbers make sense considering that 1 car is generated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



### Animating the model

You don't have to animate the model, but it is sometimes helpful to see the process in action.

- ▶ In the Create block's Item Animation tab:
  - ▶ Choose **Select item animation** from the popup menu
  - ▶ Select **Car** from the animation object popup menu
  - ▶ Click OK to close the dialog
- ▶ Select the command Run > Show 2D Animation
- ▶ Run the simulation



Discrete Event

Now cars are displayed as they flow through the model. With animation on, it is easy to see that cars are arriving faster than they can be washed.

Animation is very useful for debugging models or for making presentations, but it can considerably increase the time it takes a simulation to run. To turn animation off, unselect the command Run > Show 2D Animation or unselect the Animation On/Off tool in the toolbar.

## Adding complexity

Now that you understand how to build the basic model, you can add more details and features.

### Creating a second wash bay

Since long lines deter customers, it would be better to keep the entry line short. There are two ways to model this:

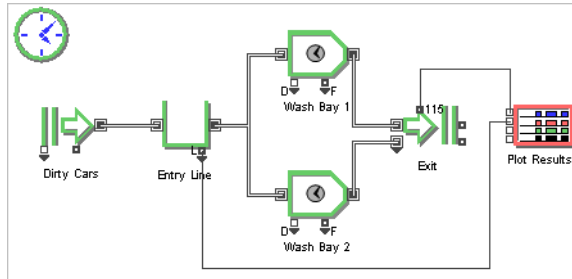
- Increase the processing capability of the Activity block
- Add a second wash line

Both of these are examples of *parallel processing* (described in more detail on page 166.)

To increase the number of cars that the wash bay can process at a time, you would 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.

Since the assumptions state that the final model has two bays, instead of increasing the Activity's capacity you will add a second bay:

- ▶ Copy the Activity block in the Car Wash model and paste it below the original Activity block. Label the new block “Wash Bay 2”.
- ▶ Connect the Queue’s item output to the item input on Wash Bay 2. This creates a parallel connection with the original wash bay.
- ▶ Expand the Exit block’s variable input connector so that it reveals a second input.
- ▶ Connect Wash Bay 2’s item output to the Exit block’s second item input.
- ▶ Save and run the simulation.




Adding a second bay

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

Discrete Event

### Explicit routing

In the model so far, the number exiting from one wash bay is probably larger than the number exiting from the other. (You can see this in the Exit block’s dialog.) 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.

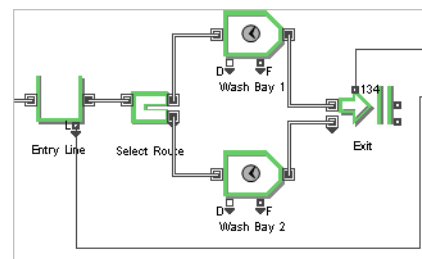
 Unless it is completely unimportant in the model, you should always explicitly state the routing of items using the Select Item In and Select Item Out blocks. Otherwise, the order in which their connections were made will dictate the routing, as discussed in “Implicit routing” on page 151.

To explicitly specify the order in which items go to free inputs, use the Select Item Out block.

- ▶ Insert a Select Item Out block (Item library) between the Queue and the two Activity blocks and label it “Select Route”.
- ▶ Connect the item inputs of the blocks.

This section of the model should look like the screenshot to the right.

The Select Item Output block has many options. You can specify that the item is routed to a random output, to a specific output based on a value at the “select” input connector or based on the item’s attribute or priority, or sequentially. Using the various options to route items is described fully in “Select Item Out dialog” on page 149.



Explicit routing

In the dialog of the Select Item Out block:

- ▶ Choose the option **Select output based on: sequential**
- ▶ Select **If output is blocked: item will try unblocked outputs**

This causes the items to be sequentially routed between the two bays. If the selected bay is blocked, the item won’t wait for it to be free but will instead be routed to the other bay if it is available.



When you run the model, the same number of cars will have been washed as in the previous model, but each wash bay will have been used equally.

☞ The preceding examples have two wash bays and purposefully don't take into consideration the model assumption that 25% of the cars want wax in addition to a wash. You could specify the second bay as also providing waxing, and use the option *Select output based on: random* in the Select Item Out block to route 25% of the cars to that bay. But this tutorial will explore a more powerful method for accomplishing this in the section "Item attributes" on page 106.

### Requiring resources

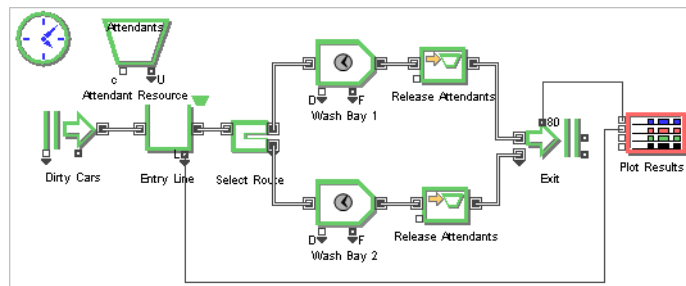
Up to this point, the Car Wash model assumes that you drive the car through the wash. However, many car washes require that an attendant do this. This situation can be easily modeled as a number of resources (the attendants) which are required by and made available to the model. As is discussed in "Modeling resources" on page 209, there are two methods to model resources:

- As a count of resources that are made available to the model as a whole and released when no longer required.
- As items that are joined with other items and flow through the model with them until separated.

In both situations, items cannot continue traveling through the model unless the required resource is available.

- ▶ Add a Resource Pool block (Item library) to the model and place it in any convenient place. Label the block "Attendant Resource". In its dialog, enter **Resource Pool name: Attendants**, and enter **Initial number: 1** (the default).
- ▶ In the Queue block's dialog:
  - ▶ Change the type of queue by selecting **Type: Resource Pool queue**.
  - ▶ In the table, select the Resource Pool named Attendants and set the Quantity to 1. This indicates that each car will now require one attendant to drive it through the car wash.
- ▶ Delete the connections from the Activity blocks to the Exit block.
- ▶ Add a Resource Pool Release block (Item library) after each of the two Activity blocks, and connect from the Resource Pool Release blocks to the Exit block.
- ▶ In each Resource Pool Release block, select to **Release by: name** and choose that the name of the resource pool is **Attendants**.

The model should look similar to the one at the right. Running the model shows that the scarcity of attendants causes a constraint on the process, and fewer cars get processed than when an attendant was not required. In the dialog of the Resource Pool block, you could try increasing the initial number of attendants to 2 or 3 to explore the effect that has on model results.



Requiring resources

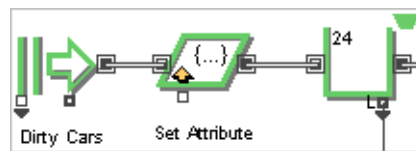
### Item attributes

Most car washes allow cars to have wax applied after the wash. Attributes are a very powerful feature that give items unique properties and characteristics. You can use attributes in this model to indicate that specific cars should or should not be waxed. This is accomplished by adding an attribute to the cars coming from the Create block, then checking for the value of that attribute as the car gets washed.

As discussed in “Attribute types” on page 116, ExtendSim supports both string and value attributes. The following example uses a string attribute type.

#### Creating a string attribute

- ▶ Add a Set block (Item library) between the Create and Queue blocks so they look like the screenshot to the right. Label the Set block “Set Attribute”.
- ▶ In the Set block’s dialog, choose **New String Attribute** from the table’s Property Name popup menu.
- ▶ Name the string attribute **Preference** and click OK.



This causes the Executive block’s Attributes tab to appear. The table in this tab is where the attribute values (and the corresponding strings) for the Preference attribute are declared.

- ▶ In the table for selecting an attribute, enter the strings **Wash Only** and **Wash and Wax** as shown in the screen shot to the right.
- ▶ Close the dialogs of the Executive and Set blocks.

Preference	
1	Wash only
2	Wash and Wax
3	
4	
5	
6	
7	
8	
9	
10	
Link	

The Executive block’s lookup table provides a descriptive text label (string) for each attribute value. That string can then be used in the model in place of the corresponding attribute value, making the model more understandable. But the underlying architecture is that the values for an attribute are still numbers. In this case, the values for the Preference attribute are 1 (for Wash Only) and 2 (for Wash and Wax).

#### Generating the correct types of cars

To specify that items are correctly generated as 75% Wash Only and 25% Wash and Wax:

- ▶ Add a Random Number block (Value library) to the model.
- ▶ Connect its value output to the first value input of the Set block.
- ▶ Choose an **Empirical table** for the distribution in the Random Number block.
- ▶ In the dialog that appears, give the table 2 rows.

Connecting to a Set block causes the Random Number block’s empirical table to be aware of attributes. In this case, connecting the Random Number block to the Set block’s first value input causes the empirical table to be populated with popup menus that relate to the first attribute (Preference) in the Set block’s properties table.

- ▶ In the first row of the empirical table, select **Wash Only** from the popup menu and enter a Probability of **0.75**.
- ▶ In the second row of the empirical table, select **Wash and Wax** from the popup menu and enter a Probability of **0.25**.

- ▶ Close the Random Number dialog.
- ▶ So that there will be sufficient attendants to drive all the cars that are generated, in the dialog of the Resource Pool block enter **Initial number: 2**.
- ▶ Since the bottom bay will now be used for cars that also need waxing:
  - ▶ Change the label of the bottom bay to *Wash/Wax Bay*.
  - ▶ Enter **Delay: 8** in that bay's dialog, to indicate that waxing and washing takes longer than washing alone.

Each item generated by the Create block will now have a Preference attribute. In 75% of the cases the car will be characterized as *Wash Only*; 25% of the time the cars will be designated as *Wash and Wax*.


The next step is to have the model determine which car is which.

### Checking the attribute

In the dialog of the Select Item Out block:

- ▶ Choose to **Select output based on: property**.
- ▶ In the property popup menu, select the string attribute named **Preference**.
- ▶ Check the box to **Display string attributes in table**.

This causes the Preference attribute to be listed in the table's header and puts popup menus for that attribute's strings in the table.

 The Select Item Out block routes items to its outputs based on the selection conditions and internal rules. When attributes are used to select the outputs, the block's top output is referenced as number 0, the second output as 1, and so forth.

To cause cars that don't need waxing to be routed to the top output:

- ▶ In the table's first row, select **Wash Only** from the Preference popup menu.
- ▶ Enter **0** in the *Select Output* column for the *Wash Only* row. That setting will route the *Wash Only* cars to the top output.

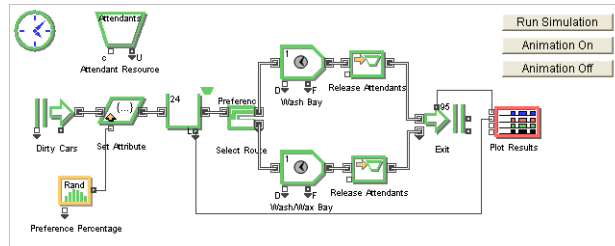
To cause cars that need waxing to be routed to the second output:

- ▶ In the table's second row, select **Wash and Wax** from the Preference popup menu.
- ▶ Enter **1** in the *Select Output* column for the *Wash and Wax* string. That setting will route the *Wash and Wax* cars to the second output.
- ▶ Close the dialog, save the model, and run the simulation.

You may notice that fewer cars pass through this car wash than in the example without attributes or attendants. This is due to the problem of a car with a particular attribute following another car with the same attribute. There is only one entrance to the bays, and the bays now have designated purpose. This means that the second car must wait for the first one to finish even if the other bay is free. Note, however, that every time you run the model, the numbers for the two lines in the Exit block indicate that the cars have been processed in roughly the same proportion as specified in the Random Number block.

**Final model**

For your reference, the completed model, titled Final Car Wash, is located in the folder \Examples\Tutorials\Discrete Event\Car Wash. This model also has buttons for running the model and turning animation on and off. For information about adding buttons to models, see “Creating a dashboard interface” on page 506.



Final Car Wash model

**Further exploration**

There are many other ways to modify the model shown in this section. Some possible variations are:

- Have the wash or wash/wax times be dynamic rather than static. There are several ways to do this:
  - Specify that the delay in the Activity block is from a distribution. This would cause the processing times to be random, rather than fixed as shown in “Random processing time” on page 169.
  - Use a Lookup Table block to schedule the wash or wash/wax times to be dependent on the time of day, as seen in “Scheduled processing time” on page 168.
  - Have an Equation block (Value library) calculate a wash time based on model conditions. Then connect the Equation’s output to the Activity block’s D input.
  - Assign attributes to the cars to represent the expected time to process it, with each type of car requiring a different processing time. This is especially useful in a manufacturing environment where there are several types of products that require different process times.
- Add more wash or wax bays, then use the Select Item Out block to give preference to specific bays rather than just letting cars randomly go to any available bay.
- 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 “Queueing considerations” on page 131.
- Model other aspects of the car wash, such as the limited capacity of a parking lot to hold cars after the wash process. To do this, use the Resource Pool block to represent the total number of parking spaces available. Then set the Queue block as a Resource Pool queue to hold cars waiting for a parking space. The Resource Pool Release block would release parking spaces as the cars pass through it.


# **Discrete Event Modeling**

## **Items, Properties, and Values**

Generating and removing items, and using item properties

As discussed in “Items and informational values” on page 93, items are what flow through the model, properties contain information about items, and values provide information about model conditions. This chapter discusses items and their properties and how information about them is reported as values. It will cover:

- Generating items randomly and by schedule
- The Create block’s Start connector
- Attributes, priorities, quantities, and other item properties

 Most of the models illustrated in this chapter are located in the folder \Examples\Discrete Event\Items and Properties. For other models, location information is provided at the beginning of their respective discussions.

## Blocks of interest

The following blocks will be the main focus of this chapter. The block’s library and category appear in parentheses after the block name.

### Item generating and removing



**Create** (Item > Routing)

Creates items randomly, by schedule, or infinitely. Can also be used to create values randomly or by schedule. Can initialize newly created items with properties, such as attributes or priorities.



**Exit** (Item > Routing)

Passes items out of the simulation. Reports the total number exited and the number that were taken from each input.

### Item properties



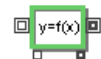
**Get** (Item > Properties)

Displays the value of user-assigned and system level item properties: attributes, priorities, quantity, and item index.



**Set** (Item > Properties)

Attaches user-assigned properties (attribute, priority, and quantity) to items passing through.



**Equation(I)** (Item > Properties)

Can be used to set, modify, or check attributes on existing items. Calculates the equation when the item arrives.



**Executive**

Its Attributes tab is used for attribute management, such as renaming or deleting attributes or locating where they are used in a model. It is also where string/value equivalents are declared for string attributes.

### Property-aware blocks

Item properties include attributes, priorities, and quantities. In addition to the blocks listed above, the following blocks in the Item library provide an interface for viewing, selecting, or modifying existing item properties or for adding new ones:


Activity	Read(I)
Batch	Resource Item
Cost By Item	Select Item Out
History	Shutdown
Information	Throw
Queue	Unbatch
Queue Equation	Workstation
Queue Matching	Write(I)

### Item generation

Items for a model are usually generated using the Create block. While it can also generate values, the Create block can create items:

- Randomly. A random distribution causes items to be generated with a random or constant *interarrival time*. The distribution determines the time *between* item arrivals; a smaller interarrival time indicates that items will arrive more frequently. See the examples below.
- By schedule. Creating items by schedule causes an item to be generated at a specific *arrival time*. The schedule defines *when* the item will arrive and the time between arrivals is fixed. See the examples in “Generating items according to a schedule” on page 114.
- Infinitely. This provides an infinite supply of items that are available *on demand*. For instance, connecting a Create block with this behavior to a Gate block would provide an item to the Gate block each time it opens.

 A Create block is set to *Create items infinitely* should *never* be connected to an infinite capacity queue, since generating an infinite supply of items would overwhelm the system.

 In a model, each item can represent an individual entity or a collection of individual entities. For instance, 50 items coming into a model could represent 50 people or it could represent 50 bus loads of people. How you characterize items is completely up to you.

### Generating items at random intervals

The Create block can generate items that arrive to the model at random times. When set to “Create items randomly”, the Create block outputs items at random intervals; the arguments of the distribution define the interarrival time.

#### Example model

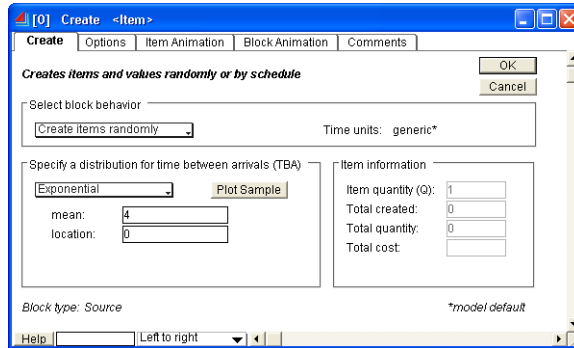
As you saw in the Discrete Event Tutorial on page 100, the Car Wash model is an example of using the Create block to generate items at random intervals.

 The Car Wash model is located in the \Examples\Tutorial\Discrete Event folder.

**Choosing a distribution in the Create block**

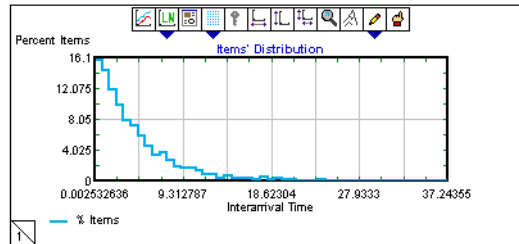
The dialog of the Create block contains several distribution choices in its “Specify a distribution” popup menu, as well as a table for entering data when an empirical distribution is selected. Each distribution is described in the Create block’s Help; they are also discussed briefly in “Probability distributions” on page 606.

Choosing a distribution in the dialog of the Create block defines both the interval between item arrivals (the interarrival time) and the characteristics of the rate of arrival.



Exponential distribution selected; mean is 4

In the Car Wash model, for example, selecting an exponential distribution with a mean of 4 will cause one car to arrive approximately every 4 minutes for the duration of the simulation. This results in an interarrival time of 4. However, the *shape* of the exponential distribution dictates that it is more likely that the time between arrivals will be between 0 and 4 than between 4 and 8.



Distribution of outputs when mean is 4

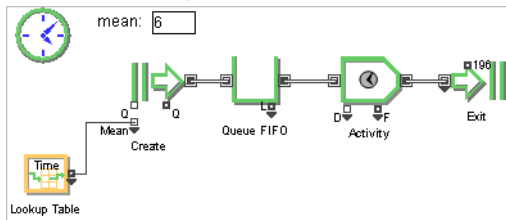
Discrete Event

**Random intervals with dynamic parameters**

You may want the parameters in a random distribution to change as a function of time or model status during the simulation run. The arguments for a given distribution can be controlled dynamically through the Create block’s value input connectors.

**Random Intervals model**

In the Random Intervals model, time dependent arrival rates are modeled by connecting the Lookup Table block (Value library) to the Create block’s value input connectors. A table in the Lookup Table block provides the mean values for an exponential distribution that has been set in the Create block. This causes the timing of item arrivals to be based on the time of day.



Random Intervals model



**Specifying the dynamic parameters**

As seen in the dialog of the Create block, items arrive exponentially. Notice that the exponential distribution has a “Mean” parameter. Connecting the output of the Lookup Table block to the Mean input connector of the Create block causes the mean of the distribution to come from the Lookup Table during the simulation run, overriding any entry in the dialog. This dynamically changes the average interarrival time.

The distribution determines the interarrival time. A smaller mean value indicates that there is less time between arrivals and items arrive more frequently.

In the Lookup Table’s dialog, the mean is smallest from hour 10 until hour 12, causing items to arrive more frequently during that period.

**Choosing time units for the columns**

When the block is set to “Lookup the: time”, the Lookup Table block looks at the current simulation time and outputs a corresponding value. Its table is used to determine the value that is output (by default the Output column; in this model, the Mean column) at a given simulation time (by default the Time column; in this model the Hour column). Its time units popup menu (in this model “hours”) represents the unit of time for the values in the Time/Hour column. However, that does not control what the output of the Output/Mean column represents.

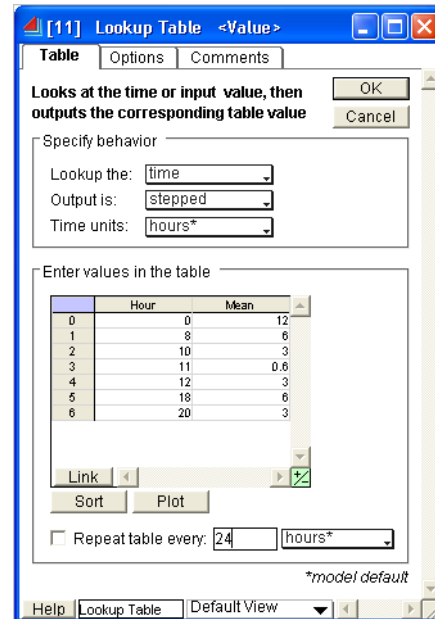
The time unit for the Lookup Table block’s output column is determined by the block that receives its output values.

When a value passed to a block’s input connector is used to set a time parameter (as in this case), the value sent must be defined in the time unit that the receiving block expects. In this model, the values from the Mean column are sent to the Create block and are used to calculate the average time between arrivals. Since the Create block is using minutes as its local time unit, the values in the Mean column of the Lookup Table block also represent minutes. For instance, the value of 6 in the Mean column represents an average of 6 minutes between arrivals, even though the event time in the Hour column is in hours.

Do not set the mean of a distribution to 0. The Create block will warn you if you make this modeling error.

**Making sure the arrival occurs when expected**

To avoid unexpected results, it is important to understand what happens in the Create block when you vary the mean of the arrival intervals over time. The block’s default behavior is to generate an arrival time, called “nextTime”, for the next item based on the current input parameters. When simulation time reaches nextTime, the Create block releases an item and generates a new nextTime based on the current values of the input parameters. For the period of time between releasing items, the Create block will not react to changes in the input parameters. If the inputs change drastically, this can cause unexpected results as discussed in “Cycle timing” on page 254.



Scheduling interarrival time

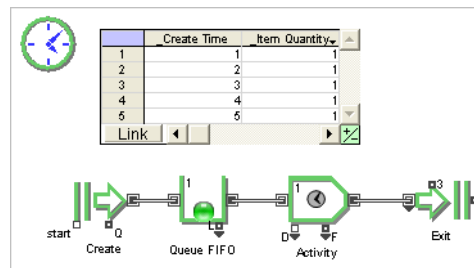
Discrete Event

### Generating items according to a schedule

Scheduling item arrivals can provide more flexibility and precision than having them generated randomly. Setting the behavior of the Create block to “Create items by schedule” allows item arrivals to occur at fixed intervals that are specified in a table.

#### *Scheduled Intervals model*

Assume you want five items generated, one per minute, but that the simulation takes ten minutes to run. Setting the Create block to “Create items randomly” and selecting a Constant distribution would generate one item per minute, but there will be ten items. Generating items by schedule allows you to customize when items will be created and how many items the model will have.



Scheduled Intervals model

In the example model, the dialog of the Create block is set to “Create items by schedule.” The schedule of arrival times, which is cloned onto the model worksheet, indicates that one item will be generated at Create Time 1, another item at Create Time 2, and so forth up to time 5. With this schedule, the model has five arrival events.

The items proceed to a Queue to wait for processing by the Activity, which takes three minutes to process each item. Since the simulation runs for ten minutes, only three items exit; one item is left in the Queue and one item is still being processed in the Activity.

This method is especially useful when the time between item arrivals is known but not regular. For instance, the first item could be generated at Create Time 1, the second item at Create Time 3, and the third item at Create Time 3.5.

Notice that the Item Quantity column has a default value of 1 for each item generated. This means that each item generated represents 1 item. Item quantities are described fully in “Quantities” on page 124.

☞ An alternative method would be to use a Create block set to *Create items randomly*, and select a *Constant* distribution with a value of **1** in its dialog. Then in the block’s Options tab, select *Maximum items generated: 5*. This method is less flexible than the earlier method, since each item would have to have the same interval between arrivals.

#### **The Start connector**

When the Create block is set to *Create items by schedule* it has a *start* value input connector that can be used to control when the schedule is executed. The timing in the Create block depends on whether or not the start input is connected:

- If the start connector is *not* connected, the schedule’s item creation times are synchronized with the simulation run’s absolute time. For example, if the schedule’s first create time is at time 2 and the simulation’s start time begins at time 0, an item would be created when simulation time

reaches 2. However, if the starting time entered in the Simulation Setup dialog is 4, the item scheduled at time 2 is never created.

- If the start connector *is* connected, such as to a Decision block (Value library), the schedule's item creation times are relative to when the connector is activated. For instance, assume simulation starting time is 0 and the first item is scheduled for creation at time 11. If the start connector gets activated at simulation time 5, then the first item will be created at time 16 (5 plus 11).

 Starts are activated whenever the connector receives a message with a True value (defined as greater than or equal to 0.5).

The Create block's Options tab provides choices for how the start connector should behave, as discussed below:

- *Follows schedule.* This is the default option and should be used for most situations. Once the start connector is activated, the entire schedule will be executed. Any new activation signals arriving before the current schedule has completed are ignored.
- *Generates one item per message.* This is an advanced choice for special situations. It is most often used when a custom-created block is connected to the start connector and you want an item to be instantaneously generated for every message. With this choice, the schedule is restricted to one row. Each time the start connector is activated, the row is executed.
- *Generates one item per event.* This is an advanced option for special situations. With this choice, the schedule is restricted to one row. Each time the start connector is activated, a zero-time event is scheduled. Once the Create block gets the zero-time event message, it will execute the schedule.

## Item properties

A property is a quality or characteristic that stays with an item as it moves through the model. Some properties can be assigned to items by the model builder; others are automatically assigned by the system.

An item's properties include:

- User-assigned attributes. These are discussed in the next section.
- Priority. See page 122.
- Quantity. See page 124.
- System-assigned attributes. See page 126.

### Attributes

Because they allow items to be distinguished from each other, attributes play a very important role in a discrete event simulation. They are especially useful for telling an activity-type block how long the item should be processed, or for determining where the item should be routed before or after processing. The following sections describe how to create, use, and manage attributes.

#### Attribute names and values

Each attribute is composed of a name and a numeric value:

- An attribute's *name* identifies some general characteristic of the item such as "size", "route", "CarType" or "tank capacity".

- An attribute's *value* indicates one dimension of the named characteristic. For instance, an item's "size" attribute could have a value of "8" or a value of "12", while an attribute named "CarType" could have a value of "1" (for Ford), "2" (for Toyota), or "3" (for Volvo). An attribute value is not just a number; it can also be the address of data in a database.

Attributes are meant to be unique; if you attempt to add a new attribute with exactly the same name as an existing one, ExtendSim warns you that the name already exists. While attribute names are not case sensitive ("Type" is equal to "type"), spaces are significant and should be avoided.

Attribute names and values are stored in a pair of dynamic, global arrays, described in "Attribute arrays" on page 121.

- ☞ The Car Wash model for the tutorial on page 106 used string attributes. Models with string attributes use text to represent the corresponding attribute value. However, the underlying architecture is that attribute values are still numbers. For more information, see "Attribute types".

#### **Number of attributes in a model**

In a model, each item can contain up to 500 attributes that uniquely describe the item. Every item contains the full set of attributes that have been defined in the model. The Executive block's Attributes tab displays all of the model's attributes.

Each attribute contain a value that can represent either:


- A number that can be used for routing, timing, and so forth.
- The address of data in a database or global array. The data pointed to can contain a single number or an unlimited amount of additional data that describes the item, its route, its properties, and so forth.

- ☞ If you use attributes efficiently, there is almost no limit to what can be represented. If you do approach the 500 attribute limit, consider using DB address attributes (discussed below) to reference information in the ExtendSim database.

#### **Attribute types**

ExtendSim supports three types of attributes:

- A *value attribute* holds a real number as its attribute value.
- The value of a *string attribute* is still a number, but it is represented in the model by a string. With string attributes you enter a descriptive text label (string) for each potential attribute value in a lookup table in the Executive block's Attributes tab. The string can then be used in the model in place of the corresponding number. For example, a string attribute named "CarType" might have three possible values: 1, 2, and 3. Once the lookup table for this attribute has been properly configured, the blocks referencing the CarType attribute will display the strings "Ford", "Toyota", or "Volvo" instead of the numbers 1, 2, and 3.
- The value of a *DB address attribute* contains a single value that represents a location or address in a database. This address is composed of four separate numbers, where each number is an index for an ExtendSim database, table, field, and record. Taken together, the numbers target a specific location in the database. (Incomplete DB addresses are allowed. For example, an item may have a DB address attribute with only the database and table indexes defined.)

 The value of a DB address attribute cannot be used directly. It must be “decoded” using a Get, Read(I), or Write(I) block or by accessing DB attribute functions in one of the equation-based blocks.

**Using attributes**

The following table lists some common attribute-based modeling activities and the blocks that are usually used to facilitate them. All blocks are from the Item library.

To Do This:	Use Block(s)
Initialize newly created items with attributes	Create (when “Create items by schedule” is the selected behavior)
Define default attributes for item resources	Resource Item
Set or modify values for existing attributes	Set, Equation(I), Queue Equation
Check attributes on existing items	Any property-aware block; see the table on page 111.
Route items based on attributes	Select Item Out (when “Select output based on attribute” is chosen)
Sort and release items from queues based on attributes	Queue (when it sorts by attribute value), Queue Matching
Sort items based on attribute values and conditionally release them based on an equation	Queue Equation
Pull in items and batch them based on attribute values	Batch (when “Match items into a single item” is the selected behavior)
Use attribute values to specify a delay or processing time	Activity (when “Delay is: an item’s attribute value)
Define the value/string correspondence for string attributes	Executive (Attributes tab in “Declare string attribute values” mode)
Find which block uses an attribute	Executive (Attributes tab in “Manage all attributes” mode)
Managing attributes and their names, such as renaming or deleting an attribute	Executive (Attributes tab in “Manage all attributes” mode)
Calculate an item’s cycle time	Set an attribute to the current time in a Set block or use the <i>Timing attribute</i> feature in the Create block’s Options tab. Then use the <i>Timing attribute</i> feature in the Information block so that it calculates the difference from start to end time. See “Cycle timing” on page 254.

**Adding attributes to a model**

The Item library blocks that deal with attributes are listed in “Property-aware blocks” on page 111. These blocks provide a popup menu interface for selecting existing item properties or for adding new ones.

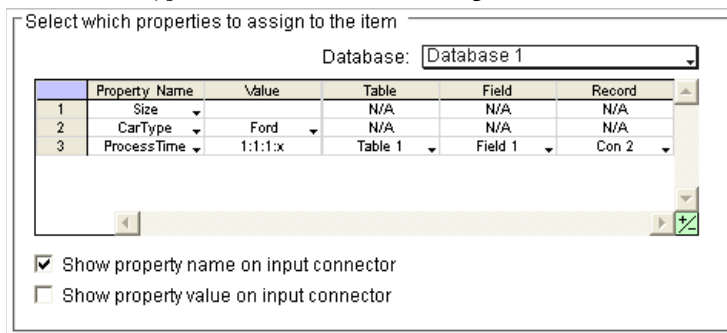
Depending on the block, the popup menu may offer different choices of attribute types to set. For example, the Set block allows creating value, string, or DB address attributes.

Creating a new attribute causes it to appear in the list of properties in the block’s dialog and makes the attribute accessible by every property-aware block in the model.

The following information describes how to create different types of attributes in the Property Name column of the table in the Set block’s dialog:

- To create a new value attribute, click the Property Name popup menu, choose **New Value Attribute**, type the name of the new attribute in the dialog that appears, and click OK. For example, a value attribute might be named “size”.
- To define a new string attribute, select **New String Attribute** from the popup menu in the Property Name column, enter a name, and click OK. This automatically opens the Executive block’s Attributes tab. The table in this tab is where the attribute values (and the corresponding strings) for string attributes are declared. An example of this is shown in “Creating a string attribute” on page 106. An example of a string attribute could be “CarType” and the corresponding string/value combinations might be Ford/1, Toyota/2, and Volvo/3.
- Creating a new DB address attribute requires an existing ExtendSim database. In the Set dialog, select an ExtendSim database from the popup list that appears. To create the DB address attribute, click the popup menu in the Property Name column, choose **New DB Address Attribute**, type the name of the new attribute in the dialog that appears, and click OK. For example, a DB address attribute could be named “ProcessTime”.

With the three different types of attributes, the Set dialog could look like:



After attributes have been created, they must be attached to items in the model and they must have unique values assigned to them.


**Selecting attributes and attaching them to items**

To allow an attribute to be used, define the attribute and assign a value to it. This is done using one of the attribute-handling blocks, such as Set or Create.

The most common method for assigning attributes to an item is to select an attribute in the dialog of a Set block, then pass the items through the block. The value of the attribute can be defined in the Set’s dialog or through its value input connectors.

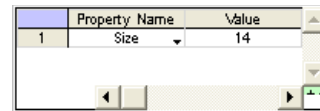
Another commonly used block is the Create block when it is in “Create items by schedule” mode. This is a convenient way to initialize new items with a set of attributes as they are introduced into

the model. In the Create's schedule table, you can select an attribute from a popup menu in one of the columns, then enter a value for that attribute for each Create Time row in the table. An item generated at the specified times will have the attribute name and value indicated in the table. Other blocks, like the Resource Item block, can also be used to attach attributes to items.

 The information that follows assumes that you are using the Set block to assign an attribute to an item and that you have already created the attribute using a method described on page 117.

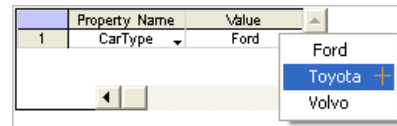
*Value attribute*

To set a value attribute, select an attribute in the Property Name column's popup menu. (Existing value attributes are listed below the New Value Attribute divider.) Then enter a number in the Value column. In the screenshot at right, the Size attribute has been selected and 14 has been entered as the value.



*String attribute*

To set a string attribute, select an existing attribute (listed below the New String Attribute divider) from the popup menu in the Property Name column. Then click the cell in the Value column to bring up a popup menu containing all the string values that have been defined for the attribute. In the example at right, the selected string attribute is CarType and the Value popup menu contains the strings Ford, Toyota, and Volvo. If Toyota is selected, the corresponding value that gets stored on the item for the CarType attribute will be the number 2.



Discrete Event

Connecting a Random Number block (Value library) that uses an Empirical table to a Set block that accesses a string attribute will cause the strings for that attribute to appear as a popup list in the empirical table's Value column. This is shown in "Checking the attribute" on page 107.

*DB address attribute*

Each Set block only points to one ExtendSim database, which becomes one element of the DB address. To set a DB address attribute in a Set dialog:

- ▶ Choose a database from the popup menu.
- ▶ In the Property Name column of the Set block's table, select a DB address attribute from the popup menu or create a new one; existing attributes are listed below the New DB Address Attribute divider.

Once the DB address attribute has been selected, the Set dialog's table enlarges to display the other elements of the address (Table, Field, and Record). The value for the DB address attribute is defined by clicking the appropriate popup menus in the table, selecting whether that element's information should be selected from a list, entered as an index, or accessed from a connector. The screenshot below is an example of the ProcessTime DB address attribute, which gets its value from a record in the Times field of the Processing Time table in the Process database.

You don't need to select every element for a DB address attribute. For example, you may only want to specify the database, table and field indexes and ignore the record index.

Property Name	Value	Table	Field	Record
1 ProcessTime	1:1:1:2	Process time	Times	2 : 5.04

For a DB address attribute, the Value column displays the database address, as determined by the indexes of the settings in the Table, Field, and Record columns. In the screenshot above, the Value notation is 2:1:1:2, where 2 is the index of the Process database, 1 is the table index for Processing Time, 1 is the field index for Times, and 2 is the index for the selected record, which has a value of 5.04.

Once the attribute has been set, the attribute information indicated in the Set block's dialog will be assigned to each item as it arrives to the block. Attribute values may also be defined dynamically using the Set block's value input connectors to override values set in the dialog.

Every model has an internal list of all the attribute names that have been created for use by items in that model. However, not all items in the model will make use of every attribute name. For an item to use an attribute name, the value of the attribute must be explicitly set using an attribute modifying block (such as a Set block).

### Getting attribute values and reporting changes

In order to manipulate an item based on the attribute, usually to route it or process it, you need to get the item's attribute value. The most common method for getting attributes is to select the attribute by name from the list in the attribute popup menu in an attribute-reading block, such as the Activity or Get block.

#### Activity or Workstation blocks

In the dialog of an Activity or Workstation block, you can specify that an item's attribute value be used as its processing time, as shown below.

#### Get block

When items pass through the Get block, it accesses information about the attributes that have been specified in the table in its dialog. It then reports the information in the table and on its value output connectors. What the Get block reports and where, depends on the type of attribute:

- *Value attributes.* The value for the attribute is posted in the Value column of the attribute table and on the value output connector that corresponds to the attribute.
- *String attributes.* The string text is displayed in the Value column of the attribute table and the number that corresponds to the string is posted on the appropriate value output connector.



- ☞ Connecting a Lookup Table block (Value library) that is set to *Lookup the: input value* to a Get block that accesses a string attribute will cause the strings for that attribute to appear as a popup list in the Lookup Table block's leftmost column.
  - *DB Address attributes.* You can get either an individual element of a database address or its entire address. To do this, from the popup menu in the table's "DB attrib reports" column, select which of the 5 components will be retrieved (db index, table index, field index, record index, or db address). The first 4 choices provide individual elements of the address; the "db address" choice provides the entire address. The information will be reported in the Value column and on the value output connector for that attribute.
- ☞ To access all five elements of a DB address attribute, add five rows to the table. Each row should have the same DB address attribute listed in the Property Name column, but different selections for the "DB attrib reports" column. This comes in handy when the Get block is working in conjunction with the Read or Write blocks (Value library). It allows the read or write location to vary based on what information is traveling on the item.

In addition to value outputs for reporting an attribute's value, the Get block has a  $\Delta$  (delta) connector for reporting when an attribute's value changes. The  $\Delta$  connector outputs a 1 when an item's attribute value (for the first attribute specified in the dialog) differs from the previous item's attribute value. Otherwise it outputs 0. This is useful for determining when there is a new type of item or when an attribute value used for processing time has changed. For example, you can have an attribute called "Type" with values that specify the type of item. When the value of Type changes, indicating a new type of item, the  $\Delta$  connector outputs 1. This is shown in "Adding setup time" on page 172.

### Modifying attribute values

The most flexible way to modify the value of an item's attribute or other property is with the Equation(I) block (Item library). This block can look up property information and modify it by applying some mathematical formula, then use the result as the new attribute value for the item. For example, if an item arrives with a value of 5 for the attribute "nextRecord", you could add a 1 to the 5 and create a new attribute value of 6 for that item's nextRecord attribute. The Air Freight model discussed on page 213 is an example of this.

Another way to modify properties is to connect from a Get block's value output to a Math or Equation block (both from the Value library). Then have that block apply some mathematical formula and output the results to a value input on a Set block. The property must be selected in the dialogs of the Get and Set blocks, and the value connectors must be for that property.

### Attribute arrays

Attribute names and values are stored in a pair of dynamic global arrays:

- The one-dimensional *Names array* stores the name of each attribute currently used in the model. Attribute names can be up to 15 characters long. You will receive an error message if you attempt to give an attribute a name greater than 15 characters. Attribute names are not case-sensitive.
- The two-dimensional *Values array* stores the value of each attribute for each item in the form of real numbers.

The following picture represents the attribute arrays:

		attributeX	attributeY	attributeZ	...	number of attributes in model (max of 500)
<b>Names array</b>		type	size	color		
<b>Values array</b>	item A	2	5	1		
	item B	2.34	6.01			
	item C	5		7		
	⋮					
		number of items in model (unlimited)				

As new attribute names are added to the model, new cells (array elements) are appended to the Names array and new columns are appended to the Values arrays, up to a maximum of 500.

As new items are created during the simulation run, new rows are added to the Values array. The number of rows in the Values array is unlimited and will be the same as the number of items in the model. As shown in the above picture, item A has an attribute named “type” that has an attribute value of 2 and item B has an attribute named “size” with a value of 6.01.

Note that each attribute named in the model causes a cell to be reserved in the Values array for every item. However, not every item uses every attribute. To allow an item to use an attribute, you must assign a value to the attribute using one of the attribute-handling blocks (such as the Set block). If there is no value assigned, the attribute is not used by that item. This is shown in the figure above, where item B has no assigned value for the attribute name “color” and item C does not have a value for the attribute “size”.

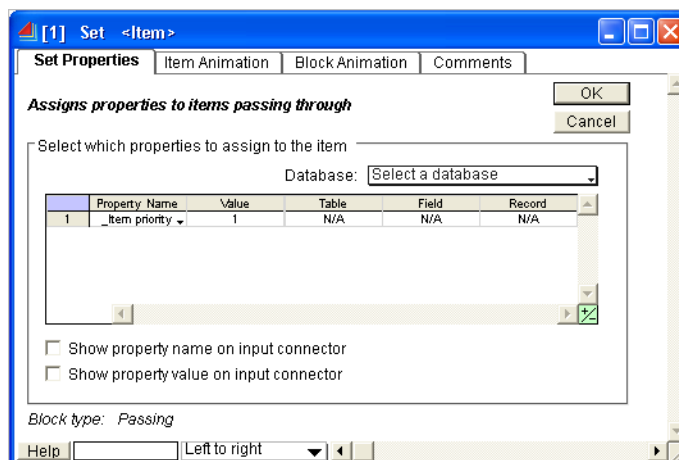
### Priority

Like attributes, a priority is a type of item property that can be assigned to an item. Priorities signify the importance of items. Using the `_Item` priority property, you can assign priorities to items and manipulate them based on their priorities.

Priorities are particularly useful when you want to examine a population of waiting items and determine their processing order. For example, you might have a step in a manufacturing process where a worker examines the pending job orders and chooses the one that is the most urgent.

- ☞ Items can only have one priority. If you need multiple levels of priorities, use attribute values instead.

Discrete Event



When a new priority is added to an item that already has a priority, the new priority prevails.  
When items are batched, the highest priority of the items prevails in the resulting batched item.

☞ The lowest value (including negative values) represents the top priority.

**Setting, getting, and using priorities**

The following table lists some common priority-based modeling activities and the blocks that are usually used to facilitate them. All blocks are from the Item library.

To Do This:	Use Block(s)
Initialize newly created items with priorities	Create (when “Create items by schedule” is the selected behavior)
Define default priorities for resource items	Resource Item
Set, modify, or check priorities on existing items	Set, Get, Equation(I)
Select incoming items based on priorities	Select Item In (when “Select input based on item priority” is chosen)
Sort and release items based on priorities	Queue (when it sorts by priority)
Sort items based on priority and conditionally release them based on an equation	Queue Equation
View an item’s priority	Get, History
Allocate resource pool units to the highest ranked item first	Resource Pool

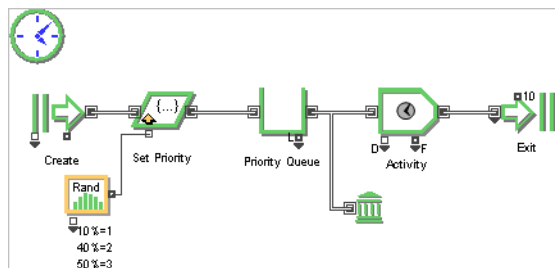
Discrete Event

☞ The Select Item Out block does not assign or use items’ priorities. Instead its output connectors can be prioritized so that an item will be routed to the first available connector that has the highest priority. As shown in “Explicit ordering” on page 155, the Select Item Out block prioritizes the *path* an item will take rather than the item itself.

**Priorities model**

In the example, a Random Number block (Value library) outputs values to a Set block, as follows:

- 10% of the time it outputs a 1
- 40% it outputs a 2
- For the remaining 50% it outputs a 3.



Priorities model

The table in the dialog of the Set block indicates it will assign priorities to incoming items. Connecting from the Random Number block's output to the Set block's ItemPriority value input connector causes the priorities to be set according to the values from the Random Number block. Since the lowest number is the highest priority, 10% of the time items will be assigned the highest priority.

The Queue block is set to sort by priority. This means that the highest priority items held in the block will be made available to the Activity block before other items. A History block, added to the model by right-clicking on the Queue's output connector, shows that only top priority items are processed; the Activity cannot keep up with the demand.

The section "Interrupting processing" on page 177 shows how priority values are used to determine if one item should preempt another.

- ☞ For an item to be ranked by priority, there must be other items in the group at the same time. For example, items will only be sorted by priority in a Queue block if they have to wait there with other items.

### Quantities

*Quantity* is another type of property that can be assigned to items. Each item can be a single entity or a group of duplicates. As is true for priority, an item can only have one quantity assigned to it at a time; the default quantity is 1. If the quantity property for an item is 1, it represents one item. If the quantity is other than 1, it represents a group. Item quantities are typically set in the Create and Set blocks.

- ☞ An item's quantity can be any number, including a negative number. An item with a quantity of 0 or less disappears when it reaches a queue.

For most purposes you would not want to change the quantity of an item from its default value of 1. However, to model a change of shift consisting of five workers going off duty at the same time, to simulate the delivery of a box of 300 pieces of mail to a mail room, or for similar situations, set the quantity of the item to be other than 1.

#### **How blocks treat items with quantities other than 1**

Items with quantities other than 1 are treated differently depending on the nature of the block processing them. They will travel together as a unit, being processed essentially as one item, until they reach an Exit, a Queue, a Batch, or a Resource Item block, or are sent into a universal connector (such as Change, Demand, Select, or Start.)

- When an item with a quantity other than 1 reaches an Exit, Queue, Batch, or Resource Item block, it is decomposed into separate identical items. For example, when it enters a Queue, an item with a quantity of 10 will become 10 distinct items, each with a quantity of 1 and each with the same properties (attributes, priority, and so on) of the original item. An item with a quantity of 0 will disappear when it reaches a Queue.
- When items with quantities other than 1 are sent into a universal connector (such as *demand* or *select*), they are treated as one item, but the quantity of the item may be used by the block as control information. See below for more information on how universal connectors deal with the incoming items that have quantities other than 1.

All the other blocks deal with items that have quantities other than 1 as a single item, ignoring the quantity associated with it.

- ☞ The Activity block accepts an item with a quantity greater than 1 as a single item and process it as one unit. To have the items be processed separately, precede the Activity block with a queue, since queues decompose items with quantities greater than 1.

### Setting an item's quantity

A quantity can be assigned to an item in the Create, Set, or Equation(I) blocks.

#### Set block

In the table in the Set block's dialog, select **\_Item quantity** in the **Property Name** column and enter the quantity in the **Property Value** column or input a value to the Block's **\_Item Quantity** value input connector. Each item that passes through the Set block will be assigned that quantity.

#### Create block

The default setting in the Create block is that one item is input to the model at each arrival event; this is the most common case when building models. How you specify that a multiple number of items be released at each event depends on which behavior is selected for the Create block:

- Create block is set to "Create items randomly". Change **Item quantity (Q)** in the Options to an integer number other than 1. Or input a value to the Create block's **ItemQuantity (Q)** value input connector.
- Create block is set to "Create items by schedule". Enter values in the table's **Item Quantity** column for each **Create Time** field that has arrival times.

For example, assume you want to show that one item arrives randomly approximately every 4 minutes. To do this, use the same settings as in the Car Wash model from the discrete event tutorial: in the Create block select the Exponential distribution and enter **Mean: 4**; in its Options tab leave **Item quantity (Q): 1**. To show that 2 car/items arrive every 4 minutes, keep the settings at Exponential with a mean of 4, but enter **Item quantity (Q): 2**. The block will now output one item with a quantity of 2 approximately every 4 minutes.

As discussed in "How blocks treat items with quantities other than 1" on page 124, the blocks that follow the Create block determine how an item with a quantity greater than 1 is treated. For instance, if an item with a quantity of 2 goes directly into a Queue or Resource Item block, it will be split into two items each with a quantity of 1. However, if the item goes directly into an Activity block, it will be treated as a single item with a quantity of 2. In most cases, you will want to follow the Create block with a Queue, which will decompose the item into two separate items.

- ☞ In most cases, you probably will not want to generate more than one item at each event. For example, rather than inputting 2 items every 4 minutes as discussed above, you would probably want to generate 1 item every 2 minutes. This is because, unless they are inside a container of some sort, it is not common to see two items arrive at exactly the same time; items are more likely to arrive at slightly different times.

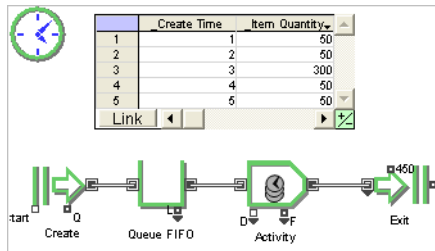
### Quantities model

For example, assume you will receive 500 items a week, but that almost all of them are received on Wednesday. In this case, there are five arrival events (one event each on days 1 through 5), each with an item quantity of either 50 or 300.

This Quantities model is similar to the Scheduled Intervals model from page 114, except each item the Create block generates has a quantity greater than 1 and the Activity processes 5 items at a time.

The dialog of the Create block is set to **Create items by schedule**. The arrival times (**Create Time**) and the number of items arriving at the scheduled time (**Item Quantity**) are entered in the table, which has been cloned onto the model worksheet.

The table indicates that on the third day (Wednesday) 300 items arrive but that 50 items arrive on each of the other days.



Quantities model

Discrete Event

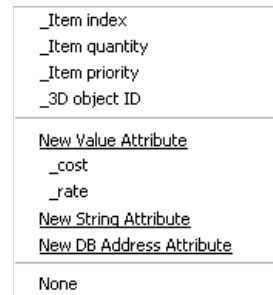
The Create block outputs an item with a quantity greater than 1 as if it were a group of items all arriving at the same time. When the item goes to a Queue block, it becomes multiple copies of itself. In this example, as each item is sent from the Create block to the Queue block, it will become either 50 or 300 units, depending on its quantity.

Running the model shows that the Create block creates 5 items, but that 500 items have arrived to the Queue.

**Other item properties**

In addition to the item properties discussed above, such as the item quantity or user-defined attributes, ExtendSim can assign properties to items. As shown in the property popup menu below, these system properties are preceded by the “\_” character and include:

- **\_Item index.** This property is available in the Get, Equation(I), and History blocks and points to where the item is located in the item arrays stored in the Executive block. It is used by block developers for debugging.
- **\_3D object ID.** When you select a 3D animation object to represent an item, this property stores the index of the object.
- **\_Cost or \_Rate.** If there is an entry for cost somewhere in the model, ExtendSim will add the **\_Cost** and **\_Rate** attributes to property popup menus. For more information, see “Working with cost data” on page 231 and “Combining multiple cost accumulators” on page 236.



History block Properties menu

# **Discrete Event Modeling**

## **Queueing**

Storing items in buffers or waiting lines

A queue provides a buffer or waiting line to store items awaiting further processing. Queues can have simple behavior, such as holding items in first in, first out (FIFO) order, or more complex behavior, such that items are held and released in groups based on their attributes. You can also set an option in the Queue block's dialog to specify how long an item will wait until it reneges, or prematurely leaves.


This chapter covers:


- Queueing disciplines: LIFO, FIFO, Priority, Attribute, and User-Defined
- Queue/server systems
- Blocking, balking, and reneging
- Sorting items using the Queue Equation block
- Least dynamic slack, minimizing setup, and maximizing service levels
- Using the Queue Matching block to match items into groups based on their attributes
- Viewing and initializing queues with the Queue Tools block
- Displaying queue contents through animating


 This chapter's examples are located in the folder Examples/Discrete Event/Queueing.


### Blocks of interest


The following blocks are the main focus of this chapter. Each block's library and category appears in parentheses after its name.

 **Queue** (Item > Queues)  
Stores items until there is downstream capacity. As a sorted queue, holds items in FIFO or LIFO order, or sorted by their priority or attribute value. As a resource pool queue, holds items in FIFO order.

 **Queue Equation** (Item > Queues)  
Stores items. Calculates an equation when it receives an item or when it is triggered by a value connection. When there is downstream capacity, releases items based on the results of the equation.

 **Queue Matching** (Item > Queues)  
Has a specified number of internal queues for holding items in separate groups. Releases a group when there is downstream capacity and the group requirements have been met. This block is useful for matching one type of item with another.

 **Queue Tools** (Utilities > Discrete Event Tools)  
When connected to the L (length) output of a queue, views and initializes the queue's contents. Displays information about item properties in a table. Can add an initial number of items, with specified properties, to a queue.

 In this chapter the focus is on using a Queue block to represent a sorted queue. For information about using the Queue block as a resource pool queue see "Resource pool blocks" on page 208.

### Queueing disciplines

ExtendSim supports several scheduling algorithms, also known as *queueing disciplines*, through the queue blocks.



- *FIFO*. When set to be a sorted queue, the Queue block can represent a first in, first out (FIFO) queue, also known as a first come, first served queue. When set as a resource pool queue, the Queue block represents a FIFO queue for resource pool units. The “MM1 model” on page 130 is an example of a FIFO queue and most of the models in the Discrete Event module use a Queue block in FIFO mode. For more information about resource pools and how the Queue is used as a resource pool queue, see “Resource pool blocks” on page 208.
- *LIFO*. When set to be a sorted queue, the Queue block can represent a last in, first out queue. As is true when the Queue is set to FIFO mode, the Queue block automatically takes care of LIFO sorting.
- *Priority*. As a sorted queue, the Queue block can read priorities and pass items with the highest priority (lowest number) out first. For this to happen, the arriving items must have a priority. Items that have not been assigned a priority in the model have a default priority with a Blank value; they get relegated to the end of the waiting line. To see a Queue sorting items based on priorities, see “Priority queues” on page 130 or “Animating queue contents” on page 140.
- *Attribute*. As a sorted queue, the Queue block can use attribute values to sort items in the queue. In addition, the Queue Matching block allows you to define custom scheduling algorithms based on item attributes. It groups items based on certain attributes and releases them as a group once requirements are met. For this sorting rule, items must have attributes assigned to them before entering the Queue. Items that have not been assigned an attribute in the model have a default attribute with a Blank value; they get routed to the end of the waiting line. The process for having a Queue sort items based on attributes is similar to the process for sorting using priorities.
- *User-Defined*. The Queue Equation block allows a user-defined equation to decide the sorting order for items it holds. This can be used to specify any user-defined criteria for sorting, including Least Dynamic Slack, Minimize Setup, Maximize Service Level, and any other combination of sorting rules. A discussion of these ranking rules and example models start on page 133.

☞ It is important to remember that, except for a FIFO queue, there must be other items in a queue at the same time to allow the queueing disciplines to work appropriately and affect the order of the items. For example, if you set a Queue block to sort by priority, and there is never more than one item in the block at a time, the effect of queueing based on priority is negated.

## Queue/server systems

Queue/server systems involve the creation of items which then wait in a queue until they can be processed by one or more servers. The following blocks in the Item library are used to represent queue/server systems:

- The *Create* block is used to provide items at exponential interarrival times (and many other interarrival times as well).
- A *Queue* block, set to sort in FIFO, LIFO, or some other order, holds the items and releases them in the designated order. It can have a maximum queue length specified in its dialog.
- The *Activity* block represents servers: you can specify an exponential or other distributional service time within its dialog or by connecting a Random Number block (Value library) to its D (delay) connector.

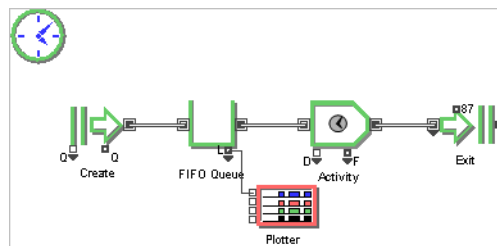
### M/M/1 queues

A standard notation often seen in queueing theory is M/M/1. This is a basic construct, representing a single server queue. The notation translates to: *exponential interarrival times/ exponential service times/ single server*. It is also common to see the designation M/M/1/∞, where the ∞ translates to *unlimited queue length*, or the designation

M/M/1: ∞/∞/FIFO, which translates to *exponential interarrival times/ exponential service times/ single server: unlimited queue length/ infinite population/ first in, first out service*.

#### MM1 model

A typical M/M/1 system expressed using ExtendSim blocks, with the addition of a plotter and an Exit block, would look like the screenshot below.



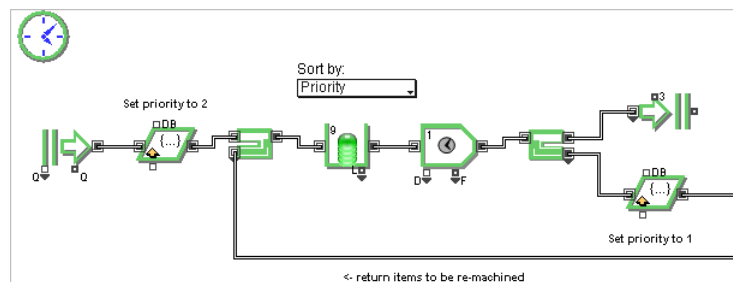
MM1 model

### Priority queues

As is true when any other sorting rule is used, a Queue block that sorts by priority will hold items until there is downstream capacity. Once the downstream block can accept an item, the Queue searches through the contents of the queue and releases the item with the highest priority. For the Queue to work properly in this mode, items that enter should already have their priority set; items without a priority are assigned a default Blank priority and get sent to the end of the waiting line.

#### Priority model

In the Priority example, items enter the model and immediately have their priority set to 2. They then enter a Queue block set to **Sort by: priority**. After the machining processes, each item is inspected for flaws. If the item does not pass inspection, its priority



Priority model

is re-set to 1 and it is sent back to the Queue block where it waits to be re-machined. When the machine can accept a new item, the Queue block will release the item with the highest priority. In this case, any item waiting to be re-machined will be released first.

Run the model with animation turned on to watch the items with a priority of 1 (red circles) bypass items with a priority of 2 (green circles) while waiting in the Queue.

## Queueing considerations

Once items are generated for the model, it is common that they will be held in a queue, typically a Queue block. In addition to the queueing disciplines discussed above, queues and the items in them can exhibit other behaviors.

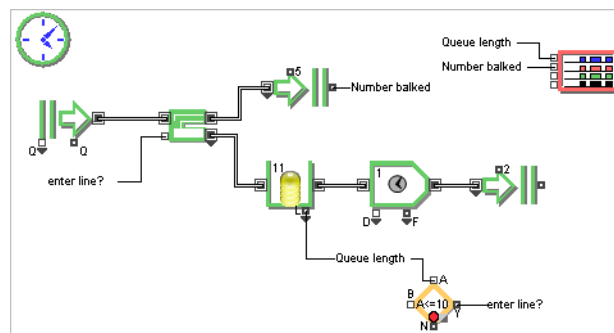
### Blocking

*Blocking* occurs when an item is prevented from leaving a block because there is a downstream capacity constraint. Blocking is common in serial operations where there are several activities in a row without queues in between; each activity has the potential for blocking arriving items. It also occurs when activities are preceded by queues with finite capacity, causing backups in the preceding activity. Blocking increases the waiting time for items in queues and is added to the calculation of their utilization.

The examples in the sections “Processing in series” on page 165, “Sequential ordering” on page 154, and “Machines that can only process certain types of items” on page 161 illustrate potential blocking situations.

### Balking

Sometimes customers enter a facility, look at the long line, and immediately leave. This is an example of *balking*. Balking is typically represented by having a Decision block (Value library) look at a queue’s length or wait time. If the line meets certain conditions (is too long, takes too long to move, etc.), a Select Item Out block routes the item out of the model before it enters the queue.



Balking model

In the Balking model, the Decision block (Value library) monitors the length of a Queue. If the queue length is less than or equal to the threshold defined in the dialog of the Decision block (10 items), its Y connector will output a 1. This instructs the Select Item Out block to route the item through its bottom output connector. If the queue length is greater than the threshold, the Decision block’s N connector will output a 0 and the Select Item Out block will route the item out its top connector to the Exit block.

### Reneging

*Reneging* occurs when an item, having entered a queue, leaves before it reaches the output. An example of this is telephone callers who, after being put on hold, will hang up without getting help if they feel they have waited too long for assistance.

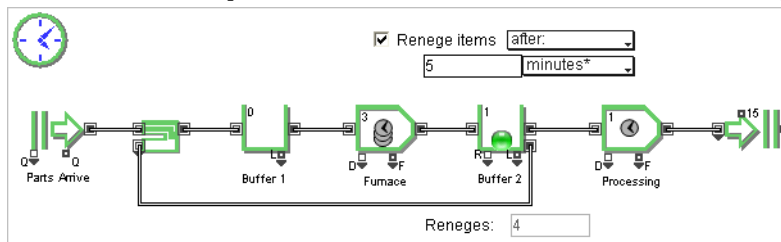
To simulate reneging, select an option in a Queue block’s Options tab. The choices are:

- Renege items after a specified number of time units. The number of time units can be set in the block's dialog or through its R (renege) connector.
- Renege items immediately when the R (renege) connector gets a true value (0.5 or greater).

When either of these choices is checked, an alternate item output appears on the right of the Queue block. Items that renege leave through that Renege output. They can be routed back to the original line (as in the example below), routed elsewhere in the model, or they can exit the model.

### Reneging model

In the Reneging model, parts wait in the first buffering queue until they can be heated by a furnace, then wait for processing in a second buffer. If too much time passes before a part is processed (such that it cools down), the part is sent back to the first buffer to wait for reheating.



Reneging model

The Options tab of the Queue that represents Buffer 2 specifies that a part will wait 5 minutes before it must be returned for reheating. The relevant information has been cloned onto the model worksheet. As seen in its Results tab, the Queue block automatically counts and reports how many items have reneged.

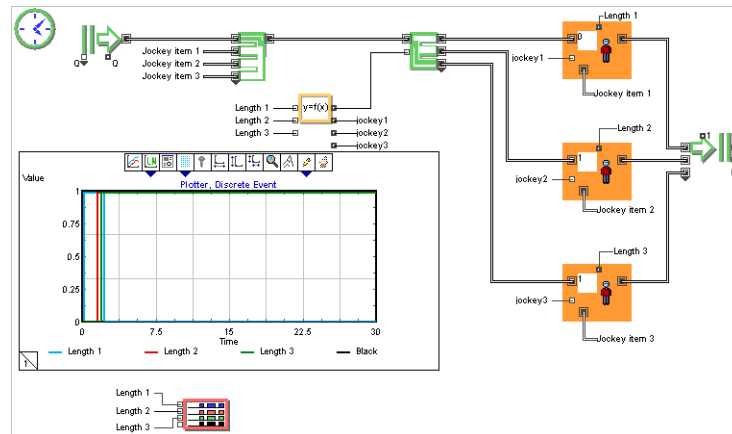
### Jockeying

*Jockeying* is when items move from one waiting line to another in an attempt to gain some advantage. To see a good example of jockeying, go to any supermarket and watch as people leave the end of a slow moving cashier's line to try and get onto a faster line.

The reneging feature on the Options tab of a Queue block is useful for building a model of this type of behavior. Normally, items renege if they have spent too much time in a queue. But the Queue block has a connector that can force reneging of the last item in the line. The Jockey.mox model is a good example of this.

### Jockey model

In this model customers arrive from the Create block and are routed through the Select Item Out block to the shortest of three possible queues.



Jockey model

As customers wait in the queues it is possible for the lines to move at different speeds. The last customer in each queue has the option to move to another queue if a shorter line opens up.

### Sorting items using the Queue Equation block

The Queue Equation block uses its equation to determine each item's position in the queue. So that item positioning can be properly determined, every time there is a potential change in the ordering the equation gets evaluated once for each item currently in the queue. Consequently, this calculation takes place every time an item enters or leaves the queue, or when one of the block's value input connectors get a new value.

The equation in this block has several different types of output variables; they are listed in the table on page 134. One of the output variables, "Queue rank", is used to assign a position to each item in the queue.

☞ At least one **Queue rank** output variable must be defined for this block to function properly.

Once each item has been assigned a ranking, the items are sorted in the queue according to the "ranking rule" that has been selected in the block's Options tab; the ranking rules are listed on page 135. The block's internal data structure keeps track of the ranking assigned to each item and items with the "best" rank are placed towards the front of the queue.

☞ If an item has been given a Blank ranking, the item is never allowed to leave the queue.

While only one Queue rank output variable is required to be defined, you may also define additional ("secondary") Queue rank variables. They are useful if you are concerned with tie breaking for two or more items that have the same rank. What constitutes a tie can also be defined on the Options tab with the +/- parameter. For an example of how a secondary Queue ranking variable is used, see "Combined rules" on page 137.

As shown in the example models later in this section, the Queue Equation block is useful for calculating sorting rules for least dynamic slack, setup minimization, service level maximization, and other complex queueing situations.

**Variables and rules**

The Queue Equation block has several types of input and output variables as well as some pre-determined *ranking rules*.

**Input variables**

As seen in the popup menu of the block's input variables table, a number of input variables can be used to determine the ordering.

<b>Input Variable</b>	<b>Uses</b>
Attribute	An attribute on the item currently being evaluated
Last item to exit	Provides the chosen attribute value on the last item to exit the block
Item quantity	The quantity of the item currently being evaluated
Item priority	The priority of the item evaluated
Item index	The current item's index value
3D object ID	The ID of the object used to represent the item in the 3D window
DB value	Access a value from an ExtendSim database table
DB address	The address of a specific location in an ExtendSim database
DB index	The index of an ExtendSim database, table, field, or record
Static variables	Variables that maintain their value from one equation calculation to the next
Arrival time	The time that the item arrived to the queue
Best result	The best (highest or lowest) equation result (ranking) so far
Connector	A variable number of input value connectors are available

**Output variables**

Once the block has determined which item will be the next one to be released, a number of output variables can be calculated for that item. They are shown in a popup menu in the Queue Equation block's output variables table and described below.

<b>Output Variable</b>	<b>Uses</b>
Attribute	Store attribute information on the released item
Item priority	Change the priority of the released item
3D object ID	Change the released item's 3D ID
DB Value	Write information to the ExtendSim database
Queue rank	Defines each item's rank (position) in the queue. At least one of the output variables must be of this type.
Connector	A variable number of output value connectors are available

Output Variable	Uses
Select con	Designed to be connected to the Select connector on the Select Item Out block, this connector is used to route the released item

As mentioned earlier, you can select one or more output variables but at least one of the output variables has to be of the type “Queue rank”. You can also have more than one ranking variable; the secondary ranking variable will be used to arbitrate in the case of tied ranking.

These variables are only available for the item that is currently being released.

**Ranking rules**

The Queue Equation block’s Options tab provides the following selections to determine which items should be released first:

- Items with the highest rank value
- Items with the lowest rank value
- Items with the first TRUE rank value

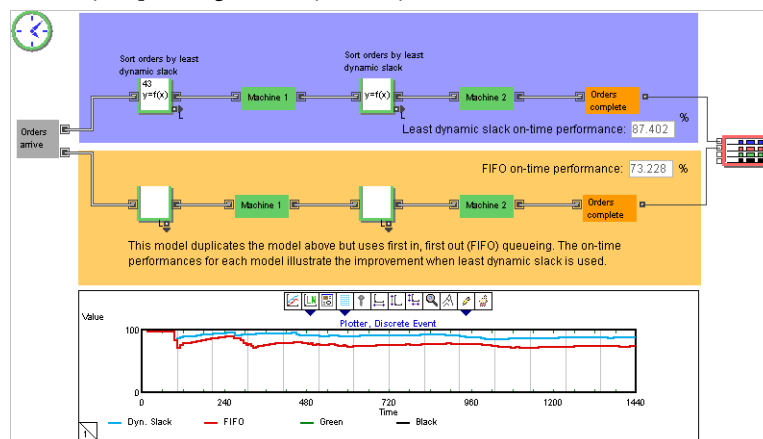
Each item’s rank value is calculated by the equation using the input variables. The items with the best rank will be released first.

**Least dynamic slack**

Least dynamic slack is a ranking rule for queues that tends to reduce the “lateness” of a sequence of orders. Slack is defined as the due date minus the remaining processing time. In essence, it is the “float” that is available before the item is due to be completed. When using slack to sort items, priority is given to those items that are closest to being late. In a model that represents orders for services or goods, choosing the order with the least dynamic slack tends to minimize the number of late orders.

**Least Dynamic Slack model**

The example model Least Dynamic Slack illustrates the improvement in on-time performance that can be achieved by sequencing orders by least dynamic slack instead of first in, first out ordering.



Least Dynamic Slack model

The two models are identical, except the top model uses Queue Equation blocks with least dynamic slack calculations and the bottom model uses Queue blocks and typical FIFO ordering. In the model, the equations in the Queue Equation blocks calculate the dynamic slack for each item. The item with the smallest dynamic slack (least amount of time before being late) will be selected first. As seen on the plot which has been cloned onto the model worksheet, on-time performance is higher using least dynamic slack (top line) compared to FIFO (bottom line).

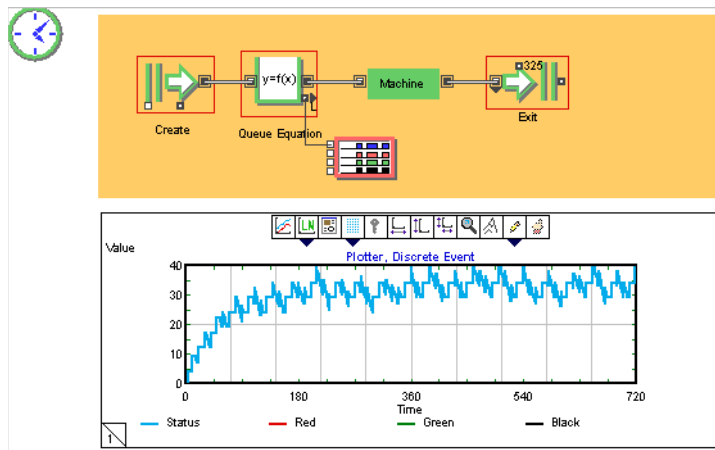
**Minimizing setup**

In some systems, setup time (the changeover from one product to another) can add significant delay to the processing of items. If this is the case, it may be useful to process the same item type until there is no longer any of that item type in the queue. Only when a particular type of item has been exhausted will another type of item be processed. Giving priority ranking in a queue to the same type of product that has just exited the queue reduces the number of setups or changeovers between products. Like least dynamic slack, minimizing setup time is another type of queue ranking rule.

**Minimize Setup model**

The model Minimize Setup compares the Product attribute for each item in the Queue Equation block to the Product attribute on the last item to leave the queue. The first item with its Product attribute value equal to the item that has just exited is released first. If no item in the queue can be found with an attribute value that matches the last exited item, the first item in the queue is selected. The plot shows the effects of this rule: the queue builds up initially until it can combine enough batches together to gain an efficiency from minimizing the setup time. (The example includes a second model, with a FIFO queue instead of a Queue Equation block, for comparison purposes.)

Discrete Event



Minimize Setup model

**Maximizing service levels**

In a service system, the service level can be defined as the number of customers served within a certain time period. For instance, technicians might be rated on the percentage of customer requests fulfilled within a certain time period. To maximize this, a queue that applies the maximize service level rule gives priority to those customers who waited less than the service level time, leading to dramatic improvement in the service level. However, this type of system might not be popular with



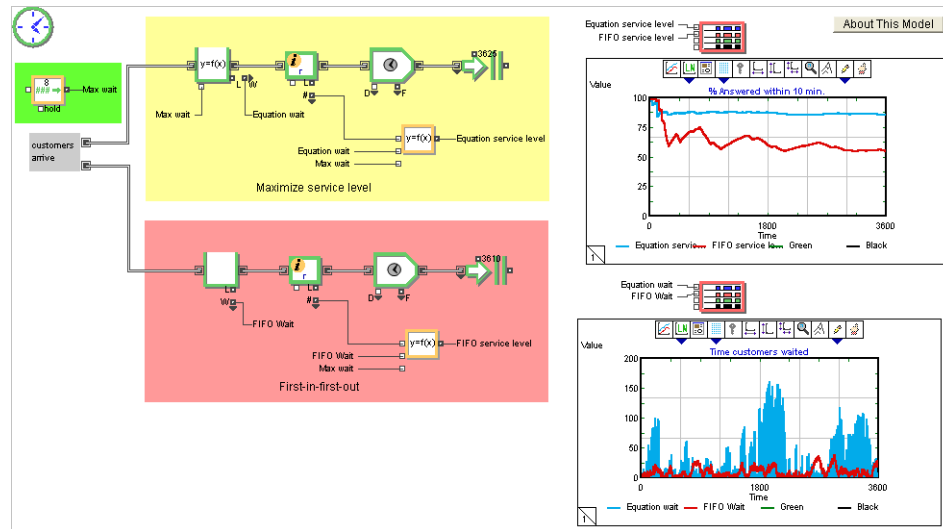
real-life customers since some of them may have to wait a very long time while other customers who arrived later would wait a much shorter time.

**Maximize Service Level model**

The top section of the Maximize Service Level model uses a Queue Equation block to sort the queue into two priority levels:

- The service priority is the customer who has spent the longest time (but less than 10 minutes) in the queue
- Customers who have waited longer than 10 minutes are placed at the back of the line

The lower section of the model is exactly the same as the top, except it uses a Queue block in FIFO sorted queue mode. As the model is run, two plotters show the effect on service level and wait time. Comparing the service level of the upper model to the bottom model, it is obvious that there is a dramatic improvement if the sorting rule is used. However, the second plotter makes it equally as obvious that some customers have a much longer wait when the service level is maximized.



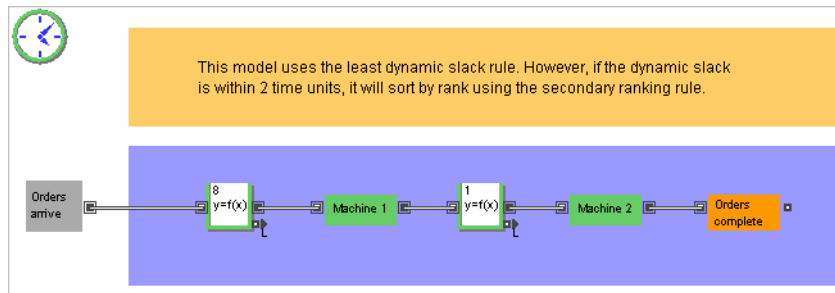
Maximize Service Level model

**Combined rules**

Because of its tie-breaking capabilities, the Queue Equation block can be used to model situations where two items are considered equal using the primary sorting rule but a secondary rule is used to determine the item with the higher priority.

### Combined Rule model

The Combined Rule model uses the least dynamic slack as the primary rule. However, if the least dynamic slack is within 2 time units, the rank (order of the items in the queue) is used.



Combined Rule model

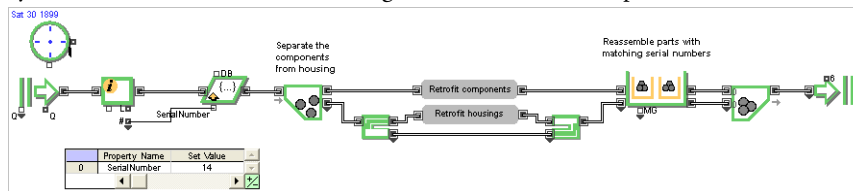
### Matching items using the Queue Matching block

The Queue Matching block has a variable number of item connectors where each connector represents a separate internal queue. Within each queue, the block sorts items into different groups based on each item's match attribute value. Items are released from a group only when the required number of items are present in each group in each queue.

This block is especially useful for making sure that items have a particular characteristic at a specific time. For instance, you would use this queue to reassemble parts in the correct order or to insure that subassemblies are correctly matched with each other.

### Queue Matching model

In the following example, electronic systems arrive from the field, are separated into their individual components for refurbishing, and are reassembled. In this operation, all of the components but only 40% of the housings need to be reworked. In addition, it is important that the housings for each system be reassembled with their original (refurbished) component set.



Queue Matching model

The Information block counts each electronic system as it arrives and outputs the total number that have passed through the block. The Set block uses this value to set a Serial Number attribute for each electronic system. After refurbishment, the system is reassembled using its original parts.

### Other models that use the Queue Matching block

The folder Examples\Discrete Event\Queue Matching contains additional models that use the Queue Matching block. Those models explore more advanced topics such as modeling fixed and variable requirements for specific items.

## Advanced queue topics

This section discusses viewing, initializing, and animating the contents of a queue. These techniques are useful in creating a more exact model as well as for debugging and/or validating a model.

### Viewing and manipulating queue contents

A powerful ExtendSim feature is the ability to manipulate the contents of a queue. The Queue Tools block (Utilities library) allows you to view items in a queue, manually manipulate the ordering of those items, and initialize the queue's contents. To use this block, connect from the L (length) output of a Queue or Queue Equation block to the value input connector of a Queue Tools block. The block has two tabs, View and Options, as discussed in the following sections.

#### View tab of Queue Tools block

The Queue Tools block's View tab is used to manipulate items in a Queue or Queue Equation block and display information about the items. When the model is run, every item in the attached queue will have an entry in the table.

Popup menus at the top of the columns are for selecting which of the item's properties (attribute, quantity, priority, and so forth) to view. There are also buttons (Up, Down, Destroy) on the View tab that can change the rank of an item in the queue or delete an item from the queue. For instance, in the screenshot to the right an item with a priority of 4 has been moved in front of other items with priorities of 1.

	Item priority	Arrival	Type
1	4	1.48769776981	4
2	1	0	1
3	1	11.2430571554	1
4	2	0	2
5	2	6.19883014399	2
6	2	7.51626158604	2
7	3	0	3
8	3	1.52333520755	3
9	3	3.78966463848	3

Manipulating a queue

To manipulate the items in a queue, run the model, pause the simulation at the desired point, select an item, and use the buttons in the dialog to move or destroy it. (To pause a simulation run, click on any cell in the Queue Tools table or use the Pause button in the ExtendSim toolbar or the Pause menu command.) Leaving this block open (or having a clone on the model worksheet) while the simulation is running will slow the model down; it is best to close it when not needed.

The View tab of a Queue Tools block is cloned onto the worksheet of the Initializing and Viewing model, discussed in the next topic.

### Initializing a queue

Sometimes it is useful to introduce items into the model at the start of the simulation run. The Options tab in a Queue Tools block (Utilities library) can be used to preload a queue with items at start time. Situations where queues might be initialized with items include:

- Reducing start-up bias. By placing items in queues at the start of a simulation, the model begins in a state that is closer to steady-state.
- Importing current system status in a scheduling model. When using simulation to model a detailed schedule, it is necessary to start the simulated system with the same work-in-progress as the real system.

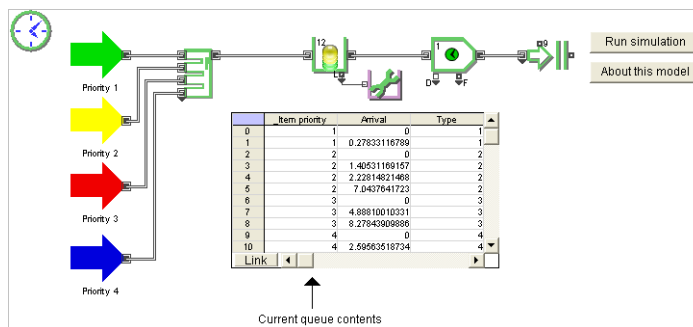
To use this block, connect from the L (length) output on a queue to the value input connector on a Queue Tools block. The block's Options tab has three choices:

- **No queue initialization:** Items are not added to the queue at the start of the simulation.
- **Initialize queue:** The queue is initialized with the number of items entered in the number field and property values as specified in the Properties table.
- **Initialize from global array:** The number of items and property values are read in from a specified global array. Because global arrays can themselves be initialized from a number of sources including Excel, a database, or the Internet, this is a very useful way to import the contents of a queue from an external source.

### Initializing and Viewing model

The Initializing and Viewing example uses the Options tab of a Queue Tools block to introduce items into a Queue block at the start of the simulation. A clone of the block's View tab has been placed on the model worksheet.

Discrete Event



Initializing and Viewing model

In this model, ten items are added to the queue at time 0. As seen in the Properties table in the Queue Tools's Options tab, these initial items have their item priority set to 1, their item quantity set to 1, their Type attribute set to 1, and their Arrival attribute set to 0. The tab also indicates that those items are represented by a cyan circle for animation. After the initial 10 items, items from the four processes are animated as circles with the same color as the arrows.

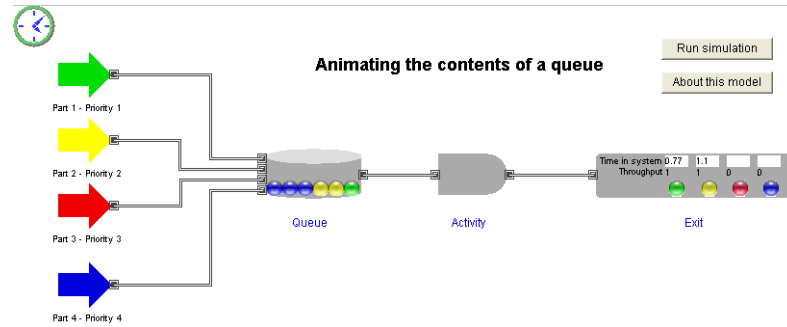
### Animating queue contents

By default, the number of items in a Queue block are displayed on its icon. However, a more detailed animation showing an animation picture for each item can be obtained by putting the Queue inside a hierarchical block and animating the hierarchical block's icon. To add this type of animation to a model:

- ▶ Encapsulate a Queue inside a hierarchical block (right-click the Queue block and select **Make Hierarchical**.)
- ▶ Open the hierarchical block's structure (right-click the hierarchical block and select Open Structure) and add a number of identically sized animation objects from the icon tools in the toolbar.
- ▶ On the Item Animation tab of the Queue block, enter the first and last animation object numbers in the **Animate H-block objects** fields.

**Animating Queue Contents model**

This example model animates a priority queue. There are four possible priorities, as indicated by the colored arrows on the left. Green circles represent items with a priority of 1, yellow circles indicate an item has a priority of 2, and so forth.



Animating Queue Contents model

Items with a lower number for their priority (a higher priority) will move to the front of the queue. The animation shows this happening as new items with lower priorities will pass other items.

In this model, there are four rows and six animation objects per row on the icon of the hierarchical block to the right of the arrows. The Item Animation tab of the Queue block inside that hierarchical block is set to **Animate H-block objects: 1 to 24**. This causes all 24 objects on the hierarchical block's icon to be animated, based on what is happening in the Queue.

For more detailed information about animating hierarchical blocks, see “Animating a hierarchical block’s icon” on page 554.



# **Discrete Event Modeling**

## **Routing**

Handling items from several sources;  
sending items to multiple destinations

When building models, you will frequently encounter situations where you want to manipulate items coming from several sources or send items to several possible destinations. Depending on the purpose, there are several methods for accomplishing this. This chapter will cover:

- Items arriving from multiple sources
  - Merging items from several streams into one stream
  - Balancing waiting line lengths
  - Using the Throw Item and Catch Item blocks
- Items going to several destinations
  - Simple routing to one of several streams
  - Scrap generation
  - Successive, explicit, and conditional ordering of routes
  - Routing based on item properties

 The models illustrated in this chapter are located in the folder \Examples\Discrete Event\Routing.

### Commonly used blocks

The following blocks will be the main focus of this chapter. The block's library and category appear in parentheses after the block name.

#### Blocks that route items



**Catch Item** (Item > Routing)  
Receives items sent remotely by a Throw Item block.



**Select Item In** (Item > Routing)  
Selects an input and outputs its item.



**Select Item Out** (Item > Routing)  
Sends each item it gets to a selected output.



**Throw Item** (Item > Routing)  
Sends items remotely to a Catch Item block

#### Blocks that affect the flow of items



**Decision** (Value > Math)  
Can be used with Item library blocks to control the flow of items in a portion of the model.



**Gate** (Item > Routing)  
Controls the flow of items in a portion of the model (area gating) or based on model conditions (conditional gating).





**Math** (Value > Math)

Performs a mathematical operation, such as addition or subtraction, that can be used with Item library blocks to control the flow of items in a portion of the model.



**Max & Min** (Value > Math)

Outputs the minimum or maximum value found among its input connectors. Can be used with Item library blocks to control the flow of items in a portion of the model.


**Items from several sources**

Depending on your modeling needs, you may want to merge different streams of items into one stream of individual items, select one item from several for routing or processing, or join separate items into a single item.

- To *merge* streams of items from several sources into one stream, where each item remains separate and retains its unique identity, use the Select Item In or Throw Item and Catch Item blocks. You then typically direct the single stream into a queue. For example, you can use this to represent traffic merging into one lane or people accessing one hallway from several offices. A Select Item In block is used to:
  - Merge streams of items in the “Merging Inputs model” on page 147.
  - Direct items requiring more processing in “Cumulative processing time: time sharing” on page 171.
  - Reroute preempted items in “Preemption” on page 178.

Those models use the Select Item In block to route items. The section “Throw Item and Catch Item blocks for merging item streams” on page 148 illustrates using a Catch Item block to merge multiple streams of parts into one stream.

- To *select* an item for processing from several sources based on some decision, use the Select Item In block. The decision can be a logical decision (choose every other item to route to the top waiting line) or it can be based on some characteristic of the item (get the item with the highest priority). The specifications for the decision are determined by the entries you make in the dialog of the Select Item In block and are modified by blocks connected to its “select” input connectors. Using the Select Item In block to choose specific items is shown in “Balancing multiple input lines” on page 147.
- To *join* items from various sources and process them as one unit, use a Batch block, as described in “Batching” on page 194. This is most common when modeling manufacturing processes or packaging operations where subassemblies are joined together. It is also used when two or more items need to be temporarily associated with each other for processing or routing, such as a clerk processing an order. Note that batching differs from using the Select Item In and Select Value In blocks, which only merge streams of items so that items remain separate and are processed separately.

 To merge streams of items from one hierarchical layer into one stream at a different hierarchical layer, you can add connectors to the hierarchical block or use the Throw Item and Catch Item blocks, as shown on page 148.

### Select Item In dialog

The Select Item In block chooses an item from one of its input connectors and sends that item to its output connector. The selection is based on settings and options in the block's dialog.

#### Selection options

The Select Item In block has several rules for selecting an item from its input connectors:

- *Item priority.* Selects the input connector that has an available item with the highest priority (the lowest numerical value for its priority.) For example, you could use this option to select from a group of queues to a single activity. The queue with the item that has the highest priority will be selected. This option always starts and restarts its selection search at the top input.
- *Random.* The inputs are selected randomly based on probabilities entered in the block's selection table. Enter probabilities in decimal format. For example, enter 0.75 for 75%. If the entered numbers do not equal 1.00, the actual sum will appear in red in the bar below the Probability column. If *Select from: all inputs* is chosen, an input will be randomly selected whether or not an item is available at that input. This situation can potentially cause starving, as discussed below. If *Select from: only inputs with available items* is chosen, the block will only select from inputs with available items.
- *Select connector.* The value received at the *select* connector determines which input is chosen. The block's dialog has an option for setting which value chooses the top input; the default is 0. The lower connectors will be numbered sequentially after the top connector. That is, if the top input is chosen by a select value of 1, the second input will be numbered 2, the next lower input would be numbered 3, and so forth. In that case, a value of 3 at *select* would cause the item from the third connector from the top to be selected. Note that, even if items are available at the other inputs, the block will wait for an item at input 3, potentially causing starving as discussed below.

☞ See "Item library blocks" on page 255 for some precautions when using this option with a Get block.

- *Sequential.* Selects the inputs in strict sequential order starting at the top; this is also known as a "round robin" selection. This option could cause starving (discussed below), since the block will wait for an item to become available at each selected input.
- *Merge.* Items are taken as they become available through any input. Generally, this option is used to combine the flows of items where there is no blocking of items arriving at the Select Item In block. Inputs are selected in a "round robin" fashion starting from the top; once a selection has been made the selection search will resume at the next lower input.

#### Starving conditions

If an item is not available from the selected input of a Select Item In block, the following options will cause a starving condition:

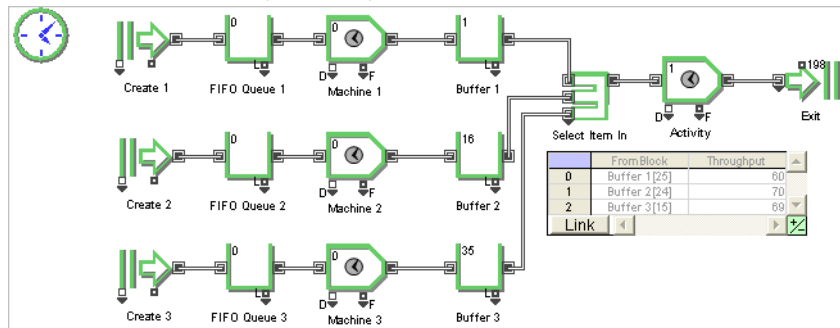
- Random (if *Select from: all inputs* is chosen)
- Select
- Sequential

#### Merging several item flows into one stream

The Select Item In block can combine the inputs from any number of sources into one stream of output items.

### Merging Inputs model

In the Merging Inputs example model, the Select Item In block will accept items from any of the three inputs. Its dialog is set to *Select input based on: merge*. If the Select Item In's output is blocked, the block will force items to wait in the Queues (labeled Buffers 1-3). When the Select Item In becomes unblocked, it will check each input in turn to try to pull an item through for processing by the Activity. As you can see in the table that has been cloned from the Select Item In block, when it is ready to process items, the Activity gets whichever item is available. This can cause some queues to have longer waiting lines than others, as you can tell from their Results tabs.



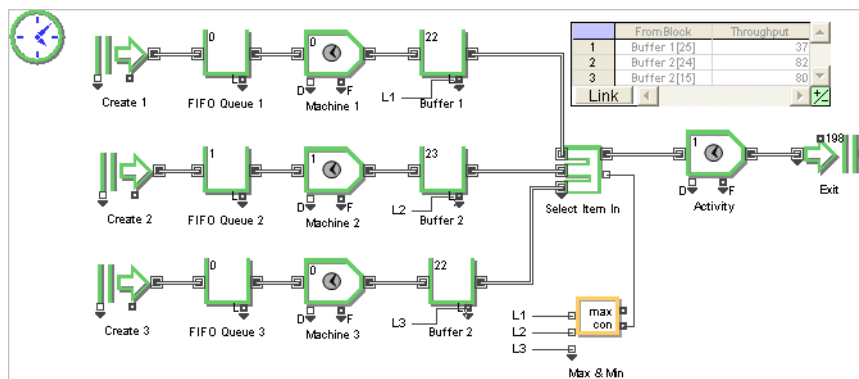
Merging Inputs model

### Balancing multiple input lines

To even out the queue lengths of multiple input lines, use a Select Item In block controlled by a Max & Min block (Value library), which checks the length of each queue. An example of this would be three loading docks that fill up as trucks unload, and you want items to come first from the dock that is most full.

### Input Line Balancing model

The Input Line Balancing model is the same as the Merging Inputs model, except a Max & Min block looks at the length for each of the queues and sends that information to the Select Item In block.



Input Line Balancing model

On the Max & Min block, the *con* output connector tells which of the inputs has the largest value, in this case it indicates the longest queue. This tells the Select Item In block which queue to

retrieve the next item from. In the dialog of the Select Item In block, *Select input based on: select connector* and *Top input is chosen by Select value: 1* have been selected. As you can see from the cloned Throughput table, items are drawn in a balanced manner from each line, and the queue lengths are almost equal, as opposed to what happened in the Merging Inputs model, earlier.

### Throw Item and Catch Item blocks for merging item streams

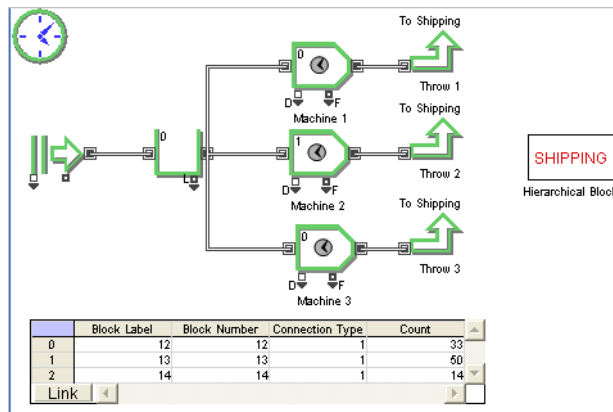
The previous examples discussed routing items using connections to blocks that are nearby and at the same level of hierarchy. Sometimes, especially in large models, it is necessary to send an item to a different hierarchical layer. The Throw Item and Catch Item blocks are especially useful when there are items from various locations in a model (even from various hierarchical levels) that need to be sent to one place. Note that these blocks are used as an adjunct to routing, not a replacement for the methods described previously.

Throw Item and Catch Item blocks pass items without connections and can even be used deep within nested hierarchical blocks to send items to other hierarchical blocks. For that reason, they are sometimes used instead of the Select Item In and Select Item Out blocks.

☞ Throw and Catch blocks should only be used when named connections will not be sufficient. For instance, to pass items through different levels of hierarchy or to use the routing features on the Throw and Catch blocks.

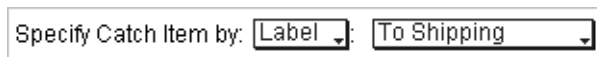
#### Throw & Catch model

The Throw Item and Catch Item blocks can also be used to merge several item flows into one stream. In the following example, three Throw Item blocks route items to Shipping, which is a hierarchical block containing two blocks, a Catch Item and an Exit. The Catch Item block is labeled *To Shipping* and is designated as belonging to *Catch group 1*.



Throw & Catch model

A popup menu in the Throw Item block's dialog displays the labels of the possible Catch Item blocks. You have the option of routing all items to the Catch Item block specified in the popup menu (as shown to the right) or routing items to different Catch Item blocks depending



Selecting Catch Item block labeled "To Shipping"

Discrete Event

on the value of a specified attribute or priority (see “Throw and Catch Attributes model” on page 156).

- ☞ You must enter text in the label field at the lower left corner of each Catch Item block’s dialog. Only labeled Catch Item blocks will appear in a Throw Item block’s popup menu.

### Catch Item groups

If you are working with a large number of Catch Item blocks, you may want to organize them into groups. To do this, select or create a group name using the *Catch Item group* popup menu in the Catch Item block, shown at right. Then use the *Catch Item group* popup menu in a Throw Item block to select the desired group. Once a group is selected in the Throw Item block, the block’s *Specify Catch Item by: Label* popup menu will only contain the labels for the Catch Item blocks in the selected group.

Catch Item group:

Catch Item group popup menu

- ☞ Groups can only be defined in a Catch Item block.

## Items going to several paths

In many cases, you will need to route items from one stream to one of many different streams:

- Taking a stream of items and routing them to different activities or operations is called *parallel processing*. In parallel processing, each item is handed off to one of several activities, such as an Activity block. The logic that determines which operation the item is routed to can be simple (the part is machined at the first available station) or it can be complex (bottle type A is filled at Machine 3). Different methods of routing items to parallel processes are described in detail throughout this chapter. See also “Processing in parallel” on page 166.
- For situations where one item is *unbatched* or separated into its component items, use the Unbatch blocks. For example, you might receive a shipment of furniture consisting of 8 desks, 20 chairs, and 7 typewriter returns, or a mail cart with 1000 pieces of mail. You use an Unbatch block to disassemble that item into its individual components, then route the items to appropriate destinations, as described in “Unbatching” on page 201.
- To *select the path* an item should go on, use the Select Item Out or Throw Item blocks. The Select Item Out block is useful for routing a stream of items to several paths based on some decision. For instance, you can send all the parts that need rework to a rework station, and ship the remaining parts. Or direct patients requiring immunizations to the Injection Clinic. The use of these blocks is described in “Sequential ordering” on page 154, “Explicit ordering” on page 155, “Routing decisions based on properties” on page 155, and “Select Item Out dialog” below.

### Select Item Out dialog

The Select Item Out block is appropriate for routing items onto one path or another. Its dialog contains several options for determining which route an item should take.

#### Selection options

The logic in the dialog of the Select Item Out block chooses which output connector an input item should be routed to. The selection options are:

- *Property*. The appropriate output is determined using the item's property—its attribute or priority. Values to represent the outputs are entered in the table's Select Output column; the default is that 0 selects the top output. For each item, the block finds the value of the specified property in the table's second column (which is named for that property), and determines the corresponding output connector for that value in the Select Output column. Since the block will hold the item until there is downstream capacity, this option can cause blocking.
- *Connector priority*. An attempt is made to send the item out each connector, in the order of the connector's priority, until the item is accepted by a connected block. The priority for each connector is entered in a table in the block's dialog. The top priority is the lowest number, such that an output with a priority value of 1 has a higher priority than an output with a priority value of 3.

☞ Note that this is different from assigning a priority to an item and selecting the output based on the item's priority, as can be done with the block's *Property* option. With the *Connector priority* option, the Select Item Out block essentially prioritizes the output path, not the item.

- *Random*. Outputs are selected randomly based on settings in the block's probability table. Enter probabilities in decimal format. For example, enter 0.75 for 75%. If the entered numbers do not equal 1.00, the actual sum will appear in red in the bar below the Probability column. When the option *If output is blocked: item will try unblocked outputs* is chosen, the block will randomly try to find an output that can accept the item. When *If output is blocked: item will wait for blocked output* is used, the block will select an output and the item will wait until that output is able to accept the item; this can cause blocking.
- *Select connector*. The value received at the *select* connector determines which output is chosen. The block's dialog has an option for setting which value chooses the top output; the default is 0. The lower connectors will be numbered sequentially after the top connector. That is, if the top output is chosen by a select value of 1, the second output will be numbered 2, the next lower one would be numbered 3, and so forth. In that case, a value of 3 at *select* would cause the item to go to the third connector from the top. Since the block will hold the item until there is capacity downstream from connector 3, this option can cause blocking.

☞ See "Item library blocks" on page 255 for some precautions when using this option with a Get block.

- *Sequential*. Outputs are selected one after the other in sequential order starting from the top; this is also known as a "round robin" selection. When the option *If output is blocked: item will try unblocked outputs* is chosen, the block will try the next connectors sequentially. When *If output is blocked: item will wait for blocked output* is used, the block will select an output and the item will wait until that output is able to accept the item; this can cause blocking.

☞ The Select Item Out block expects integer values for comparison and will truncate non-integer values. For example, if *select connector* is chosen as the selection condition, the numbers 0.001 and 0.999 received at the SelectIn input would both be truncated to a 0.

### Blocking conditions


The Select Item Out block is a decision-type of block; its default is to pull in the item and then determine the path that the item will take. In some situations, the selected output path may be blocked and the selected item will have to wait to leave. Some selection conditions can cause the items behind a selected item to be blocked:

- Property
- Random (when *If output is blocked: item will wait for unblocked output* is chosen)
- Select
- Sequential (when *If output is blocked: item will wait for unblocked output* is chosen)

For the *random* and *sequential* selection conditions, the ability to choose what happens if the output is blocked is useful for certain modeling problems. For instance, in the Simple Routing model shown below, if the top Queue block is designated to get the item, but it is blocked, the Select Item Out block will route the item to an unblocked Queue.

**Predicting the path of the item before it enters the block**

As mentioned above, an item can be pulled into a Select Item Out block but not be able to proceed because the downstream path is blocked. An alternative to this situation is to cause the item to wait in an upstream queue, rather than in the Select Item Out block. This is accomplished by checking *Predict the path of the item before it enters this block*. When this is enabled, the Select Item Out block will query upstream to determine the properties of the next item to arrive. It then checks to see if the appropriate downstream path is clear. Only if the item can be sent out the desired output will the item be pulled in. This guarantees that the item will not get “stuck” in the Select Item Out block.

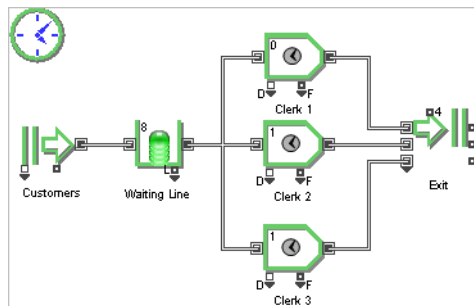
 This setting requires that any properties used to make the selection have to be set before the item begins to move into the Select Item Out block. For example, a Queue is necessary between a Set block and a Select Item Out block.

Discrete Event

**Implicit routing**

The simplest, but not necessarily the best, way to route items is by creating connections between the output of the collection point and the inputs of each activity-type block. This causes ExtendSim to pass items to the first available activity.

However, if more than one activity-type block is free when an item is ready, it is not obvious which block will get the item. For instance, a Queue that holds items for three Activity blocks would look like the model below.



Simple Connections model

If two or more machines are free when an item comes out of the queue, the machine that was *first connected* will get the item. With these types of simple parallel connections, even just disconnecting and then reconnecting a connection line could change the order of activities getting items. This implied routing may not be reflective of the actual system and is usually not what you would want.

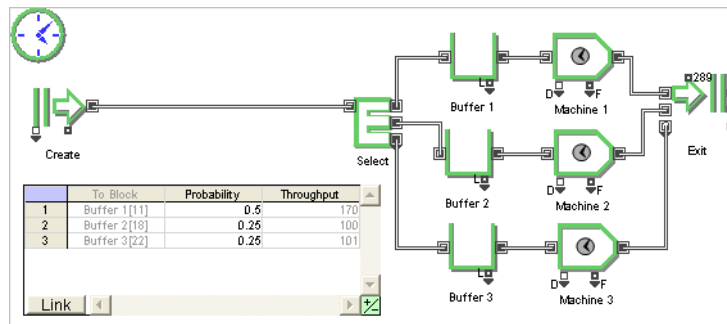
**!** Unless it is completely unimportant in the model, you should always use the Select Item In and Select Item Out blocks to explicitly state how items should be routed. Otherwise, the order in which their connections were made will dictate the routing.

### Simple routing

It is most common to route a random number of items to one section of the model, while the rest are routed to another. An example of this is an intersection where a number of cars will turn to the left, some will go straight, and some will turn right.

#### Simple Routing model

In this model, items are routed randomly to one of three machines, as indicated by the setting *Select output based on: random*. The probability table in the dialog of the Select Item Out block indicates that 0.50 (50%) of the items will go to the top Queue while the remainder will be distributed equally between the remaining two Queues.



Simple Routing model

Notice that there are queues in front of the machines. If you omitted the queues, there is a possibility that items arriving from the Create block could be blocked. For example, if the second item were destined for the top machine, but that machine was still processing the first item, the other machines would have to wait for items until the top machine finished processing and pulled in its item.

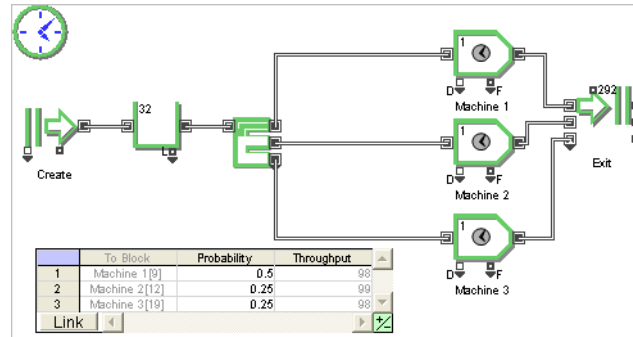
**!** The previous model routed items based on probabilities. To distribute an input item to *any* available output, in the Select Item Out dialog choose *Select output based on: sequential* and *If output is blocked: item will try unblocked outputs*.

#### Simple Routing One Queue model

A model similar to Simple Routing would be if there were only one queue and it was placed before the Select Item Out block. As mentioned above, however, if two sequential items are destined for the same activity they will block items that arrive behind them. The option *If output is blocked:*



*item will try unblocked outputs* allows the Select Item Out block to try other outputs if the first choice is unable to accept the item.



Simple Routing One Queue model

As seen in the model, even though 50% of the items should be going to the top machine, item distribution is almost even. Because all the machines process items for the same amount of time, the top machine is often busy and, rather than cause the system to be blocked, its intended item is routed to a different machine.

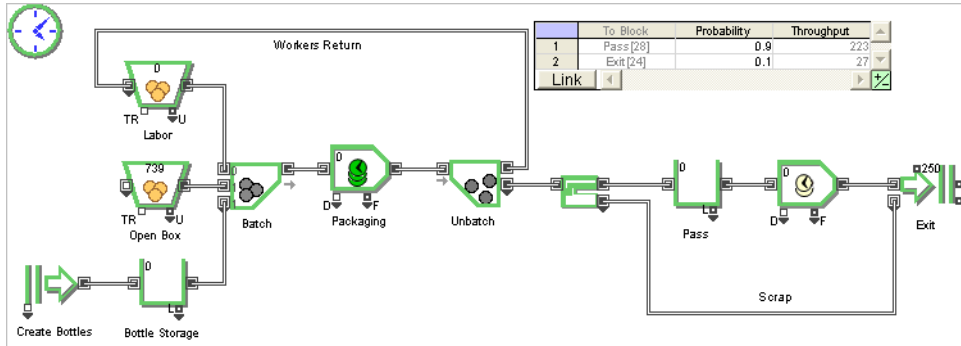
### Scrap generation

An important aspect of some systems is modeling the generation of *scrap* or simulating a *yield rate*. For instance, many manufacturing processes create an expected but irregular quantity of waste or bad items. This can be accomplished in ExtendSim by randomly routing some items out of the normal processing stream.

### Scrap Generation model

This example is similar to the model that is described in “Simple unbatching” on page 202, except it has a Select Item Out block that determines whether items coming from the Unbatch block should continue for processing or be discarded, and the Activity block processes two items at a time. The Select Item Out block is set to *Select output based on: random*. By setting a probability

that 0.90 of the items will exit through the top connector and 0.10 through the bottom (scrap) connector, the Select Item Out block causes one out of every ten items to become scrap.



Scrap Generation model

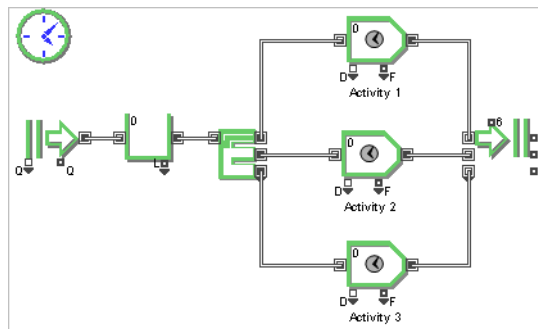
As an alternative, you can also set and check attributes to represent items that need to be scrapped. This will be shown later in this chapter.

### Sequential ordering

To hand items to operations in successive order regardless of whether another operation is free, use the Select Item Out block set to *Select output based on: sequential*.

#### Sequential Ordering model

With the sequential setting, the Select Item Out block will choose outputs in successive order starting from the top output. The block's dialog is set to *If output is blocked: item will wait for blocked output*.



Sequential Ordering model

The first two activities are set to process for 1 time unit while the third activity takes only 0.5 time units. For this model, even if the third activity is the first one ready to accept an item, it will only get an item after the first and second activities have pulled in an item. Also note that an item is only pulled from the Queue block when an activity has finished processing, potentially causing blocking in the system. The example “Balancing multiple output lines” on page 158 shows a solution for this.

- To distribute an input item to any available output, choose *Select output based on: sequential* and set the block to *If output is blocked: item will try unblocked outputs*.

Discrete Event

### Explicit ordering

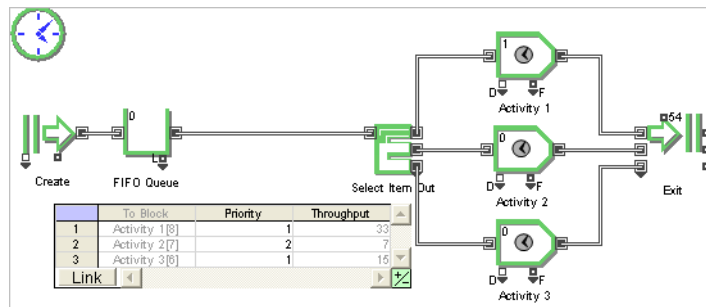
If there are several operations, and you prefer certain ones to be active more than others, you can explicitly state which operations have a higher priority for items. This is common when you want to avoid using an operation because it is not as efficient (such as an older piece of equipment) or because it is an uneconomical use of a resource (having a supervisor wait on customers.)

Choosing the selection condition *connector priority* in its dialog allows the Select Item Out block to be used to specify the priority of each output.

- Note that this is different from assigning a priority to an item, since the Select Item Out block essentially prioritizes the output path, not the item.

### Explicit Ordering model

For example, assume you want Machines 1 and 3 to get most of the items for processing, and Machine 2 to only get items if Machine 1 and 3 are busy. The model is:



Explicit Ordering model

The dialog of the Select Item Out block is set to *Select output based on: connector priority*. In the table, the highest priorities (which are the lowest numbers) are assigned to the top and bottom outputs, and the next lowest priority is assigned to the middle output. In this case, Activity 1 has first priority on items. If Activity 1 is busy, Activity 3 will get the item. Only if Activity 1 and 3 are busy will Activity 2 get the item.

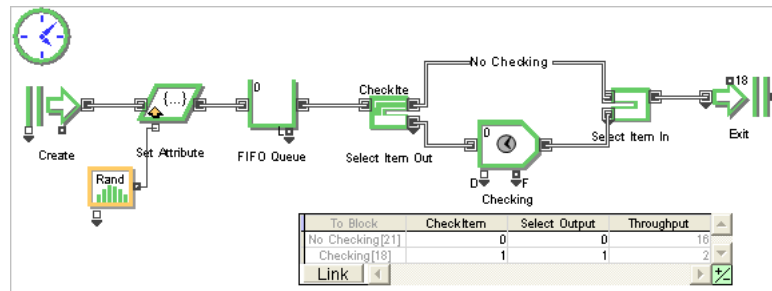
- As seen in this model, multiple outputs can have the same priority. However, the item will go to the topmost output that has the highest priority and is free. If that output is not free, the next lower output with the same priority will be checked to see if it is free, and so forth. If this is not what you want, set the priority values explicitly. For instance, you could set the output priorities to 1, 2, and 3 rather than to 1, 2, and 1 as was done in the example model.

### Routing decisions based on properties

You may want to route items based on some characteristic of the item, such as its priority, size, quality, age, or state. To do this, assign an attribute or priority to the item and read that property value to route the item.

### Attributes for Routing model

To specify whether or not an item must have a process performed on it, set an item's attribute to a yes-or-no value using the Random Number block (Value library) as shown in the Attributes for Routing model, below:



Attributes for Routing model

Discrete Event

The Empirical distribution in the Random Number block specifies that 75% of the items do not require checking (0 value for the CheckItem attribute) and 25% do (1 for attribute value). The Select Item Out block, set to *Select output based on: property*, reads the attribute value to determine which of two routes the item will take, one through the checking line (value = 1) and the other around it (value = 0). The Select Item In block is used to combine both lines into one stream, exiting the simulation.

This method is especially useful if the checking process takes more than one step. For instance, you may need to transport the item to the checking station using transportation blocks, but only if checking is needed. All those steps would be between the Select Item Out block and the Select Item In block.

With this model, an item that needs to be checked can be pulled into the Select Item Out block but not be able to advance because there is already an item in the Activity. To prevent this, you can cause the block to predict the path of an item before it enters, as discussed page 151.

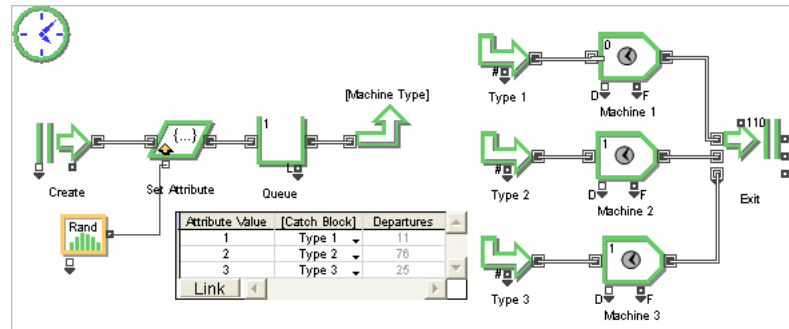
☞ An item that requires checking that is blocked in the Select Item Out block will also block other items that arrive after it, even if they do not need to be checked. If this is not how your process works, insert a Queue before the Activity to hold items that need checking.

The example “Machines that can only process certain types of items” on page 161 is another instance of using attributes to route items. For a very different approach, the DB Job Shop model located in the folder Examples\Discrete Event\Routing uses information from the ExtendSim database to route items.

### Throw and Catch Attributes model

As described in “Throw Item and Catch Item blocks for merging item streams” on page 148, the Throw Item block can be used to route items to a specific Catch Item block that is identified by its label. Throw Item blocks can also be used to route items to different Catch Item blocks depending on the value of an item's attribute or priority. A modification of the Attributes for Routing exam-

ple, built using Throw Item and Catch Item blocks rather than the Select Item Out block, is shown below.



Throw & Catch Attributes model

In this example, The Throw Item block is set to *Specify Catch block by: Property: Machine Type*, where Machine Type is a value attribute. The Throw Item block reads the Machine Type attribute and routes the items to the appropriate Catch Item block according to the table in the throwing block's dialog, which is cloned onto the model worksheet.

To cause an attribute or priority value to be associated with a specific Catch Item block, type the value into the "Property Value" column and select the appropriate Catch Item block using the popup menu in the "[Catch Block]" column.

### State Action model

Another routing example is the State Action model where items are routed to operations depending on their state. For complete information, see "State/Action models" on page 49.

### Conditional routing

Sometimes you will want to route items based on the current conditions of the model. For example, monitoring queue lengths to determine whether or not an activity will be brought on-line or balancing the use of parallel waiting lines.

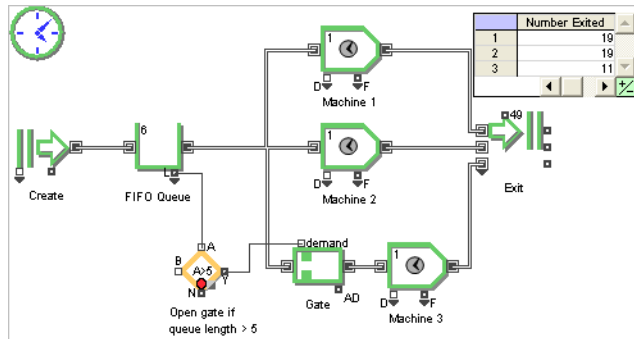
### Bringing a system on-line

Most of the examples in this manual show items being passed to operations where all the operations are on-line and running. In many situations, particular operations are only started when they are needed. You can bring another system on-line based on the time of day (such as in "Scheduling activities" on page 173) or based on some other factor such as the backlog of work.

*Conditional Routing model*

For example, you might have a factory where most of the processing is done by two machines but excess work is handled by a third machine. ExtendSim can simulate this easily using the Decision block (Value library) and the Activity block (Item library).

The *L* output of a queue that is feeding one or more machines outputs the number of items waiting to be processed. If this value is greater than a certain threshold, you can route some of the items to another machine or activate another process.



Conditional Routing model

In the model, the dialog of the Decision block specifies that the Y connector outputs a true value (1) when the value at the A input is greater than 5. This activates the Gate block's *demand* connector so that it lets items through to the third machine (until then, it will not accept items). When the Queue holds 5 or fewer items, the Gate closes.

You can also model this situation in the opposite manner, by having all the operation blocks process items and then shut one or more of them down under certain conditions. If you do this, items may be trapped in the shutdown operation until processing resumes.

When you bring a system on line, it may cycle on and off too frequently. See “Bringing an activity on-line” on page 173 for some methods for avoiding this.

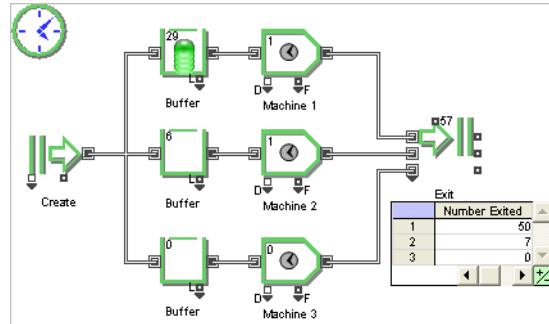
**Balancing multiple output lines**

Operations are often preceded by queues before each operation, such as a staging area for each machine (as compared to the single staging area for all machines as in the “Explicit Ordering model” on page 155.) The location and ordering of placement of queues in a model can affect how the model performs.

Discrete Event

*Buffering Operations model*

A model with queues before each operation could look like:



Buffering Operations model: Parallel processing with buffering

In the model, the queues have a maximum queue length of 30 each. The first queue that was connected will receive all the items until that queue reaches its maximum, then the next queue will start to fill (unless the first machine kept up with the flow of items, in which case the next queue will never receive any items). In the Buffering Operations model, for example, most of the items go to the top queue and get processed by Machine 1 and none of the items go to the bottom queue to be processed by Machine 3. This is rarely what you want.

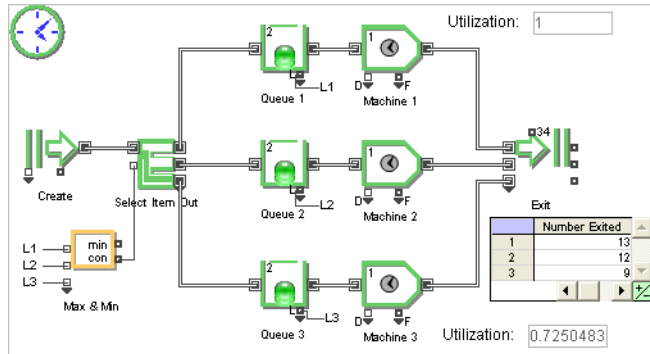
To even out the use of the machines, use a Select Item Out block set to select machines sequentially, as shown in the “Sequential Ordering model” on page 154, and place a queue before each machine. Note however, that if the machines work at different speeds, this will cause the queue of the slowest machine to fill more rapidly than the other queues.

- ☞ A green bar across its top indicates that a Queue is set to something other than infinite capacity. For instance, the Queues in this model are set to hold a maximum of 30 items and therefore have green bars across their tops.

*Output Line Balancing model*

A better method than the Buffering Operations example would be to check the length of the waiting line in each queue and give the next item to the queue that is shortest, causing the queue lines

to be balanced. The Max & Min block (Value library) connected to a Select Item Out block is excellent for this.



Output Line Balancing model: Choosing the shortest queue

In the model, the Max & Min block tells the Select Item Out block which queue line is shortest and thus which queue to hand the next item to. The Min & Max block is set to *Output the: minimum value* and *Top input connector # is: 1*. With these settings the block's top input (L1) is number 1 and the **con** output reports which of the inputs (L1, L2, or L3) has the lowest value, indicating the shortest queue. The dialog of the Select Item Out block is set to *Select output based on: select connector* and *Top output is selected by select value: 1*.

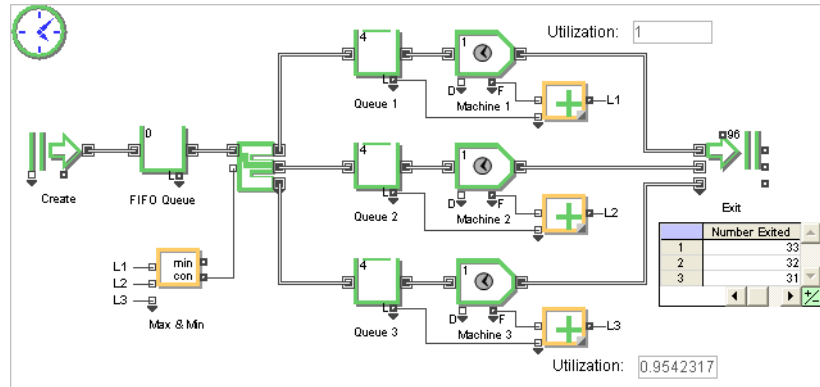
Compared to the Buffering Operations model shown earlier, in this model the number of items in the queues tend to be more balanced. However, the system is not as efficient as it could be since an item often goes to the queue for Machine 1 even though Machine 3 is idle. This happens because if all the queue lengths are equal, the Max & Min block will report the first connector as having the shortest queue length. In this case, Queue 1 has first priority for items and the Max & Min block is just looking at the queue lengths and is not considering whether or not the Activity is occupied.

#### *Line Balance with Activities model*

The previous model showed how to balance the queues. A more useful model would be to include information about the items being processed when selecting the shortest queue. You can do this by adding the value of an Activity's F (full) connector to the queue length. The full connector is 1 when the activity is at capacity and 0 otherwise. The resulting model, Line Balance with Activities,



prevents items from going to a queue followed by an occupied activity when other activities are idle.



Line Balance with Activities model

As in the previous model, the queue lines tend to be more balanced than the Buffering Operations model; but this method makes more efficient use of the Machines.

Yet another option would be to use a Workstation block to replace the Queue and Activity blocks. The Workstation can report all of the items in its internal queue and activity through its Length connector.

### Machines that can only process certain types of items

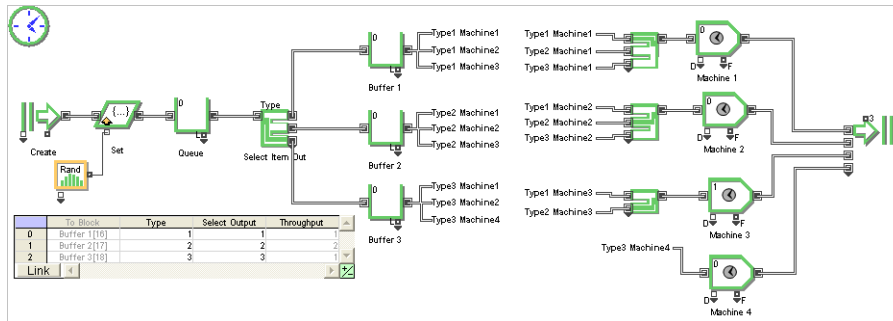
A typical assembly line can handle more than one type of item at a time. For example, you may have three stereo models being assembled on a single line. Most of the assembly is identical, but a few different parts are used at different points. Unfortunately, some machines in such a heterogeneous assembly line cannot work on particular models being assembled. The method for accomplishing this is a combination of splitting items into different paths to establish the different “types” of items, then recombining the paths appropriately for the different machines. The Select Item Out block and the Select Item In block are handy in such situations.

#### Processing by Type model

Assume there is an assembly line where each item has an attribute called *Type* that is either 1, 2, or 3, depending on the type of item it will be. At one step of the assembly process, there are four machines. Two of the machines can work on all three types, but one of the machines is old and can

162 | **Routing**  
Items going to several paths

only work on types 1 and 2, and the other machine can only work on type 3. The model is shown below:



Processing by Type model

A Set block assigns a Type attribute to each item. The empirical table in the Random Number block (Value library) indicates that there is a 50% probability that the item will be Type 2, and 25% probability that it will be either Type 1 or 3.

The Select Item Out block is set to *Select output based on: property*. It looks up the value of the Type attribute (1, 2, or 3) and selects the appropriate output (1, 2, or 3) based on entries in the block's options table, shown at right.

To Block	Type	Select Output	Throughput
0 Buffer 1[16]	1	1	1
1 Buffer 2[17]	2	2	2
2 Buffer 3[18]	3	3	1

Options table in Select Item Out block

Notice the use of the queues as buffers in the above model. They are used to store the items by type, with the top queue for Type 1, etc. Without the queues, the whole line could be blocked, depending on the order in which items arrive. For example, if the first three machines are all processing a Type 1 item, and a Type 1 item is the next to exit the Select Item Out block, blocking occurs until one of the machines is finished with its item. The fourth machine will not be able to pull in an item until one of the other machines finishes processing and pulls in the new Type 1 item. Even then, it will have to wait until a Type 3 item is output before it can process anything.

- Named connections are used to simplify the look of this model. Without these, there would be a spaghetti of connection lines connecting the Queue blocks to the Select Item In blocks. Another option for organizing the model would be to use Throw Item and Catch Item blocks to route the items to the appropriate machines.


# **Discrete Event Modeling**

## **Processing**

Using activity-type blocks to cause and control processing

This chapter will discuss different ways to use activity-type blocks and how to control processing time and the availability of items and resources. It will cover:

- Processing in series and in parallel
- Setting the processing time
- Bringing an activity on-line
- Interrupting processes: preemption and shutdowns
- Multitasking
- Kanban systems
- Material handling and transportation blocks

 The models discussed in this chapter can be found in the folder Examples\ Discrete Event\Processing. That folder also contains some subfolders, as indicated in the relevant sections of this chapter.

### Commonly used blocks

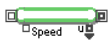
The following blocks will be the main focus of this chapter. The block's library and category appear in parentheses after the block name.

Discrete Event



**Activity** (Item > Activities)

Processes one or more items simultaneously; outputs each item as soon as it is finished. Can also be used for multitasking.



**Convey Item** (Item > Activities)

Moves items on a conveyor from one block to another. Has dialog settings to define whether the conveyor is accumulating or not, what the length of the item is, and how far and how fast the item moves.



**Create** (Item > Routing)

Creates items or values randomly or by scheduled. Can be used to control shutdowns for the Activity block.



**Shutdown** (Item > Resources)

Used to control shutdowns for an Activity (Item library) or Valve (Rate library). Useful for setting random or constant time-between-failures (TBF) and/or time-to-repair (TTR).



**Transport** (Item > Activities)

Moves items from one block to another. Has dialog settings for defining how fast and how far the item moves.



**Workstation** (Item > Activities)

Acts as a FIFO queue combined with an Activity block, holding and processing items. Takes in one or more items at a time, holds them in FIFO order, processes them simultaneously, then outputs each item as soon as it is finished.

### Systems and processes

Systems encompass one or more processes, which are a series of activities that achieve an outcome based on the inputs. Some examples of processes, the events that might drive them, and the items that flow through or are consumed by the process are:

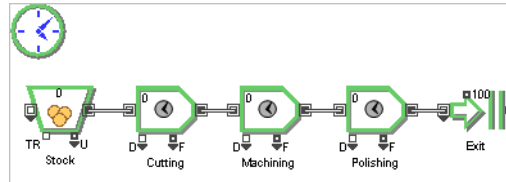
Process	Event	Item(s)	Activity
Planning strategic directions	Plan implementation	Decisions	Planning meetings
Developing a new product	Employee has a new product idea	A prototype	Document the specifications
Manufacturing a product	Receipt of raw materials	Parts, labor	Assemble the parts
Sales fulfillment	Customer orders goods	An order, or the goods themselves	Process the order, ship the goods
Call center support	Customer calls on telephone	Telephone call	Route call to technical support
Processing an insurance claim	Claim is received (or accident occurs)	Claim	Review the claim
Emergency room admitting	Accident	Patients, medical personnel	Assess incoming patients (triage)
Regulating traffic	Traffic light changes	Cars, pedestrians	Cross the street
Computer network	Packet is transmitted	Packet of data	Communication
Material handling	Arrival of AGV	Parts, AGVs	Load part onto AGV
Hiring employees	Company wins contract	Employees	Interview potential candidates
Completing an expense report	Employee finishes trip	Report	Prepare and file report
Writing a contract proposal	Request for proposal is issued	Proposal	Research the requirements
Approving a loan	Customer submits application	Application	Review credit history

☞ The following discussions most often refer to the Activity block for processing. However, the Workstation block can often be used instead of a Queue and an Activity, and the Convey Item and Transport blocks are also useful for simulating processing.

### Processing in series

Serial processing occurs when items flow from one activity to another, where each activity performs one required task on the item, out of a series of required tasks. This is most common in manufacturing activities, order entry, or service-intensive situations. A simple example of serial

processing is an assembly line, where several processes are performed on one part prior to shipment.



Serial Processing model

Since there are many machines in series without buffering queues between them, it is possible that items will not be able to leave one machine because the next machine will still be busy; this is known as *blocking* (as discussed in “Blocking” on page 131). Serial processes can cause the entire operation to be slowed to the speed of the slowest activity. This will cause utilization to increase by the amount of time that the item is blocked. If this doesn't accurately represent your process, put a queue in front of each machine to represent a holding area, as shown in “Select Item Out dialog” on page 149.

## Processing in parallel

It is common in industrial and commercial systems for there to be multiple activities working in parallel, each representing the same task being performed. For example, you might have five machines that can each process parts arriving from the stockroom. Or three bank tellers who are available to wait on customers. With the blocks in the Item library, there are many ways to route items to parallel activities.

Remember that, unless items are purposefully duplicated in the model, they can only follow one path at a time.

### Parallel processing using one block

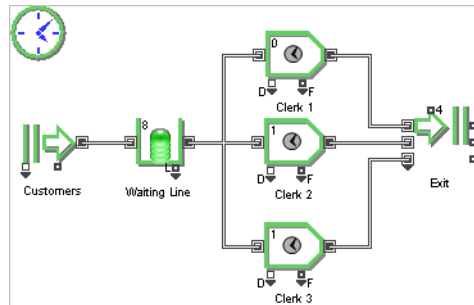
When you do not need to show each activity as a separate block, you can choose that the Activity block represent several operations that occur in parallel. This is accomplished by entering a number greater than one for the *Maximum items in activity* field in the block's Process tab.

The Activity block can take in items (up to the specified maximum) and process them for a specified time starting from when they arrive. The item with the shortest time in the block (based on the item's arrival time and how long it takes to process) is passed out first. For example, you could use the Activity block to represent a supermarket where customers arrive at different times and take varying amounts of time to shop. Customers who arrive early or who only shop a little will leave first; customers who arrive later or shop a long time will leave later.

### Simple parallel connections

You can also use multiple Activity blocks in a model, each of which represents a process that can accept items in parallel with the other Activity blocks. The simplest way to hand out items to separate parallel activities is by creating connections between the output of the collection point and the inputs of each Activity. This causes ExtendSim to pass items to the first available Activity block. However, if more than one block is free when an item is ready, it is not obvious which block

will get the item. For instance, a Queue that holds items for three Activity blocks would look like the model below.



Simple Connections model

If two machines are free when an item comes out of the queue, the machine that was first connected will get the item. With simple parallel connections, even just disconnecting and then reconnecting a connection could change the order of activities getting items.

Unless it is completely unimportant in the model, you should always explicitly state the ordering for parallel activities using the Select Item In and Select Item Out blocks. See “Items going to several paths” on page 149 for examples of how to control the flow of items to parallel processes.

### Setting the processing time

Activities involve a processing time or delay that is the amount of time it takes to perform a task. Processing time can be static or can vary dynamically depending on model conditions. It can be random, scheduled based on the time of day, customized depending on the item that is being processed, or any combination of these.

You model the processing or delay time explicitly using the Activity or Workstation blocks or implicitly by specifying the length and speed in the Transport or Convey Item blocks. The Activity block is most frequently used to represent a process or operation, and is illustrated in most of the examples for this module

The models discussed in this section can be found in the Examples\Discrete Event\Processing\Time folder.

#### Processing time for an Activity

In the Process tab of the Activity block you can select that the delay is:

- *A constant.* This uses whichever number is entered in the Delay (D) field. See also the discussion of “Fixed processing time” on page 168.
- *From the D connector.* The processing time is the value at the D input, overriding any value initially entered in the Delay (D) field. For example, see “Fixed processing time” on page 168 and “Scheduled processing time” on page 168.
- *An item’s attribute value.* With this option, you can select an attribute to control the processing time. This is illustrated in “Custom processing time” on page 170.
- *Specified by a distribution.* This choice provides a random processing time, based on the distribution and its arguments selected in the dialog. An example of this is shown in “Random processing time” on page 169.

- *From a lookup table.* This choice allows you to use an attribute value to specify the parameters of a random distribution. With this option, two popup menus and a table appear. The first popup is for choosing a distribution; the second is for selecting an attribute. The table is where the processing time for each type of item is characterized, with each row containing a different set of arguments for the selected distribution. As each item enters the block, its attribute value identifies which row in the table the item is associated with, and thus what its processing time is based on. This option is illustrated in the “Simulate Multitasking Activity model” on page 184.

### Processing time for other activity blocks

The Workstation block works much like a Queue combined with an Activity block. It has most of the same options (shown above) for setting the processing time as the Activity. (Because of its internal queue, the Workstation does not have a D input connector.) The Convey Item and Transport blocks calculate an implicit processing time based on the settings for speed and length entered in their dialogs. They are discussed in “Transportation and material handling” on page 185.

### Fixed processing time

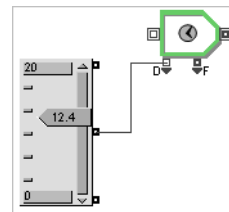
Set the delay in the Activity dialog if the delay doesn't change and you know how long it is. For example, if Machine A always takes 5 minutes to process parts, enter the value 5 as the processing time in the Activity's dialog. This is most common in the early stages of model building when you use constant parameters to get repeatable results.

Another method for having a fixed or constant processing time is to connect to the D input of the Activity block. For instance, you can connect from a Constant block (Value library) or a Slider control, as shown at right. If you use the Slider, you can manipulate it with each simulation run, or within a simulation run, to see the effect of various processing times.

Connecting to the Activity's D connector overrides any manual entry in the Delay (D) field.

### Scheduled processing time

If a process takes a specific amount of time under most conditions, but takes another amount of time if the conditions are different, you can schedule the processing time. This is common when simulating worker performance, where output could be a factor of time.

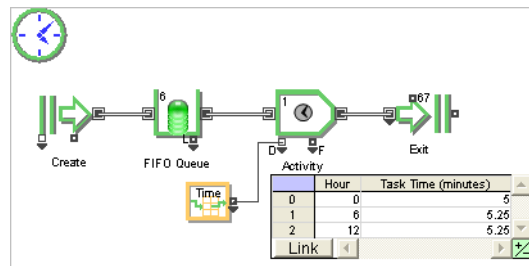


Slider control used to set Activity's processing time



### Scheduled Time model

For example, assume that a worker normally takes 5 minutes to perform a task, but takes 5.25 minutes to do the task after doing it for 6 hours. To do this, connect a Lookup Table block (Value library) to the D input of the Activity block as shown in the following model.



Scheduled Time model

The Lookup Table block is set to *Lookup the: time*. Data that represents the worker's day is entered in the *Hours* column; the time to perform the task is entered in the *Task Time* column. As indicated in the Lookup Table's dialog, a portion of which is cloned onto the model worksheet, the time to perform the task changes at hour 6.

#### Lookup Table block's time units

Notice that the time unit for the Hours column in the Lookup Table's dialog is hours and that the time the worker takes to perform the task (Task Time column) is in minutes. The time units for the model are hours, so the first column must be in hours. However, since the output of the Lookup Table block is connected to the D connector of the Activity block, the Task Time column should be defined in the same time units (minutes) that are used for the Activity's Delay parameter. For this model, at a particular hour of the day (Hour column) an activity will take a specified number of minutes to perform the task (Task Time column). Time unit consistency is discussed in "Choosing time units for the columns" on page 113.

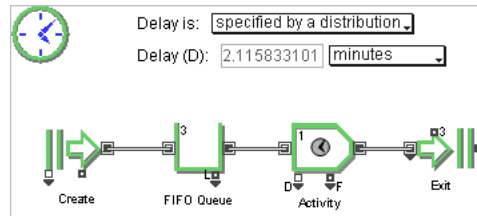
### Random processing time

A common requirement for activities is to set a random processing or delay time. This is easily accomplished by selecting a distribution in an Activity block.

#### Random Activity model

In the dialog of the Activity block, select the appropriate distribution, for example Normal, and specify the value of the parameters, such as a mean of 2 and a standard deviation of 0.2. The pro-

cessing time will then be normally distributed and the Activity block will process each item for approximately 2 time units.



Random Activity model

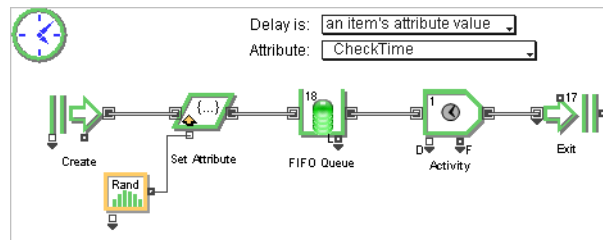
For more information, see “Constant values and random variables” on page 57, “Random numbers” on page 604, and “Probability distributions” on page 606.

### Custom processing time

Attributes can be used to specify how long a specific item will be processed. This is a very powerful feature since the Activity block can recognize each item’s processing time and behave accordingly.

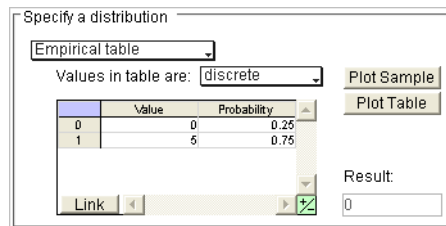
#### Custom Time model

In the simplest case, set an item’s attribute value to the desired amount of processing time, then use the Activity block to read the attribute value and process the item for that period of time.



Custom Time model

The Custom Time model uses the Set block to set the value of an attribute called CheckTime to the amount of time it takes to check the item. Items that need a final check have a CheckTime attribute value of 5, for instance, and items that ship unchecked have an attribute value of 0. That value (0 or 5) is provided by the Random Number block (Value library) using an Empirical distribution where 25% of the items have a value of 0 and 75% have a value of 5.



Custom processing time

All items then go through the checking step, easily represented by an Activity block. In its dialog, this block indicates that the *Delay is: an item’s attribute value* and the attribute is *CheckTime*.

Although items with a CheckTime value of 0 will not be processed by the Activity block, they may be delayed in the Queue (which is set to FIFO order) while a preceding item undergoes checking.

### Implied processing time

Some ExtendSim blocks allow you to specify distance, speed, or other factors that indirectly result in a processing time. For example, the Convey Item block allows you to enter an item length and the length (in feet or meters) and speed (in feet or meters per time unit) for an accumulating or non-accumulating conveyor.

These settings in Convey Item and Transport blocks result in delay times for items. For example, if you set the Transport to be 10 feet long with a speed of 1 foot per time unit, it will take 10 time units for an item to travel to the next block.

For more information about using the Convey Item and Transport blocks, see “Transportation and material handling” on page 185.

- ☞ If the *Metric distance units* preference is selected in the Edit > Options dialog, length units in these blocks are set by default to meters. Otherwise, their length units are in feet.

### Cumulative processing time: time sharing

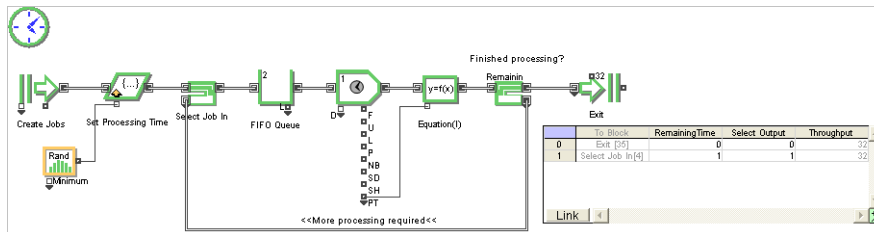
The prior section discussed setting an item’s attribute to the time required by a specific activity. You can also set its attribute value to the total processing time required, then route the item to a series of activities, each of which performs one part of the processing, until the attribute value is reduced to zero and the item is fully processed. This is common when there are several stations with different processing times, any of which can process the item. Or when there is one machine that processes each item for a specified time, then passes the item to another section for further processing, and the item must be returned to the original machine for finishing.

You can do this using an activity block, building the model such that the attribute value is decreased by the amount of processing time. In this situation, each activity subtracts its processing time from the attribute value, so that the value represents the remaining processing time. Use a Get block after each activity to determine if the item was fully processed or not, and therefore whether it should proceed to the next activity or be routed out of the line.

*Time sharing* occurs when an activity processes an item, sends it back to a queue for a short period, then processes it again until the required processing time is completed. This is common for computer networks and telephone communication systems. In these systems, time is specified in small fractions of a second, there are a lot of jobs that must be processed at the same time, and there are only a limited number of processors to do the work. In time sharing, instead of each job being processed sequentially, all jobs are processed at what appears to be the same time. However, each job is processed a small bit at a time, and a given job may have periods in between where nothing is happening to it. Since the time units are so small, the periods when there is no processing of a specific job are typically not noticed, and each job appears to be processed continuously.

### Cumulative Time model

A simple time-sharing model has one activity that processes each job for a short period of time, then sends the job back to the queue so it can be processed again, until the total required processing time for that job has elapsed.



Cumulative Time model

In the model, items (jobs) are generated randomly and the Set block attaches a RemainingTime attribute to each job generated. The Random Number block determines the initial value of that attribute, 1, 2 or 3 milliseconds, which represents the total processing time required for each job. The Activity block processes the job for a fixed time (1 millisecond). An Equation(I) block then subtracts the amount of time spent processing (Process Time, or PT) from the RemainingTime attribute.

Once the remaining processing time has been calculated, the decision to route the item back to the Queue or to the Exit is made in the Select Item Out block. Jobs with a RemainingTime of 0 are routed through the top output and exit the simulation; items that have 1 or more millisecond of processing time left are routed back to the Queue for further processing. Notice that the table in the Select Item Out block (shown above) only indicates explicitly what happens if the remaining time is 0 or 1. However, the block's dialog is set to *Invalid Select value: chooses bottom output*. With these settings, an item with a RemainingTime value of 0 will exit the top output. If the value is 1 the item will exit the bottom output. And any value other than 0 or 1 will also exit from the bottom output.

	To Block	RemainingTime	Select Output	Throughput
0	Exit [32]	0	0	32
1	Select Job In[4]	1	1	32

Table in Select Item Out block

- ☞ The time units for this model are integers representing milliseconds, because the Select blocks expect integer values for comparison and will truncate non-integer values. (For example, the value 0.003 would be truncated to a zero.) When using non-integer values for the processing time, convert the attribute values to integers before they go to the Select Item Out block. You can do this with a Lookup Table block (Value library).

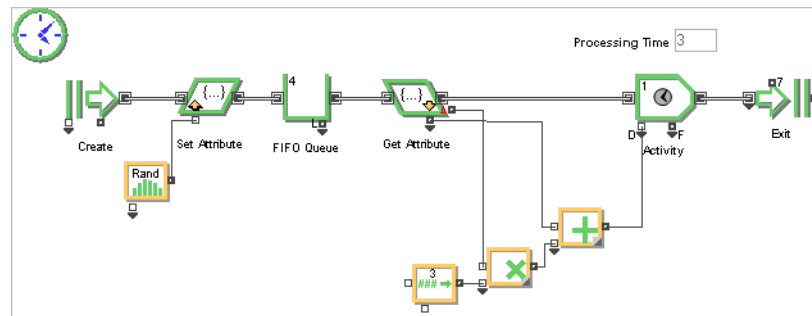
### Adding setup time

Every process is not 100% efficient. For instance, it is common in manufacturing for a machine to be reconfigured when the type of item it is processing changes. This reconfiguration usually takes additional time beyond the normal processing time. In the example below, the processing time (and the part type) is determined by the attribute values, similar to what was shown in “Custom processing time” on page 170. However, this model requires an additional *setup time* whenever the type of item changes.

- ☞ Setup time can add significant delay to the processing of items. For an example showing how to minimize setup time, see “Minimizing setup” on page 136.

### Setup Time 1 model

In the Setup Time 1 model, a Process attribute is assigned to each item. The attribute value represents how long items should be processed for – either 1, 3, or 5 minutes, depending on probabilities entered in the Random Number block (Value library). The  $\Delta$  (delta) output connector on the right of the Get block signals when the value of the Process attribute has changed, indicating that the current item is of a different type than the previous item. This information is used to add a setup time to that item's processing time, resulting in a longer total delay time for the first item of a new type, each time the type changes.



Setup Time 1 model

The  $\Delta$  (delta) connector outputs 0 (for False) as long as the value of the Process attribute stays the same and outputs 1 (for True) when the attribute value changes, indicating the arrival of a new type of item. The Constant block (Value library) specifies a setup time of 3 minutes. As long as the attribute value does not change, the Constant block is multiplied by 0, adding nothing to the normal processing time. When the attribute value changes, the Constant value is multiplied by 1, and the 3 minute setup time is added to the value of the Process attribute to determine the processing time for that new item. You can see this if you run the simulation – the processing time is cloned onto the worksheet and will be 1, 3, or 5 minutes for most items but 4, 6, or 8 minutes for the first item that is of a new type.

Notice that each item still has its original attribute value. You do not change the attribute value in this model, it is only used to determine whether the item type has changed and thus whether the item requires a setup time. The processing time (whether equal to the attribute value or equal to the attribute value plus the setup time) is input at the D connector. The Activity block processes based on the value at the D connector, not directly based on the attribute value.

- While the model above shows the mathematics explicitly, the Equation block (Value library) can also be used to specify the setup time. This is illustrated in the model Setup Time 2.

### Bringing an activity on-line

As discussed in the following sections, many systems, activities, or operations can be brought on and off-line based on a schedule or on the current conditions of the system.

- The models discussed in this section can be found in the Examples\Discrete Event\Processing\Bring On-Line folder. The Shift block is discussed starting on page 218 and models using the Shift block are located in the folder Examples\Discrete Event\Resources.

### Scheduling activities

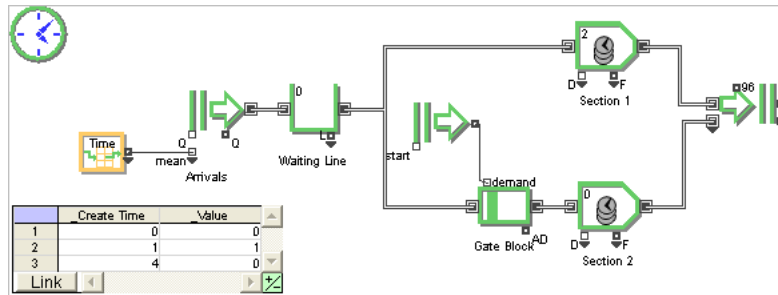
Activities don't always occur randomly; they can be scheduled. This is common when you bring an activity on-line based on the time of day. This could be represented by a schedule in the Create

(Item library) or Lookup Table (Value library) block connected to and controlling a Gate block (Item library), by using a Create block to schedule the capacity of an Activity, or by using a Shift block to control an Activity. As seen below, the Scheduling Activities 1 model uses a Gate block and the Scheduling Activities 2 model schedules an Activity's capacity. Shift blocks are discussed in "The Shift block" on page 218.

So that you can compare both the Scheduling Activities 1 and Scheduling Activities 2 models, they have the same random seed, as seen in their Run > Simulation Setup > Random Number tabs.

### Scheduling Activities 1 model

The following example shows how to schedule the availability of a portion of an operation using a Gate block. The Scheduling Activities 1 model represents a diner that opens at 10 AM and closes eight hours later. Customers arrive exponentially throughout the day, with most customers arriving during the lunch period, which is from 11 AM until 2 PM (from hour 1 to hour 4). There is 1 dining section that can service 5 people at a time throughout the day. A second dining section that can also service 5 people at a time is available only for the lunch shift.



Scheduling Activities 1 model

The scheduling of random customer arrivals is accomplished by entering values in a Lookup Table block (Value library), set to *Lookup the: time*. The output of the Lookup Table block is connected to the *mean* input of a Create block, which generates customers exponentially. As seen in the table to the right, the Lookup Table outputs a smaller value from time 1 to time 4. The output represents the average time *between* arrivals, so arrivals will occur more frequently from time 1 to time 4.

Time	Output 1
0	0.1
1	0.05
2	0.04
3	0.08
4	0.2
5	0.2

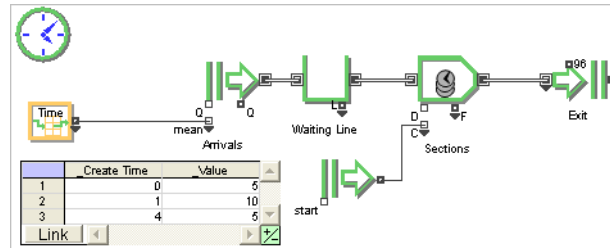
Customer arrivals

Opening the second dining section is accomplished by connecting a Create block to a Gate block, allowing customers access only during certain hours.

The Gate block is set to *Mode: conditional gating with values*, so that it only allows items through when its demand connector is activated by a value. This is accomplished by connecting a value connector, such as the one on the Create block when it is in *Create values by schedule* mode, to the demand connector. As long as the value connector is true (outputs 1), the Gate stays open; when the value is 0, for false, it closes. Running the model with animation on shows that, even though the queue length is increasing, the Gate shuts down after three hours.

### Scheduling Activities 2 model

As an alternative to the preceding example, you could have used the C connector on the Activity block to control its capacity, simulating the opening of the second dining section.



Scheduling Activities 2 model

Each dining section has the capacity to serve five customers at once. By doubling the capacity of one Activity block during the period between 11 AM and 2 PM you can model both dining sections being open. This is accomplished by connecting the value output of a Create block, set to *Create values by schedule*, to the C input on the Activity. In the model, the portion of the Create block that controls the capacity of the Activity is cloned onto the model worksheet.

### Shift block used to schedule

Yet another approach to scheduling an Activity would be to use a Shift block to control it. For instance, the Shift block could contain the same information as the Create block that is connected to the C input on the Activity in the Scheduling Activities 2 model, above.

For more information, see the section titled “The Shift block” on page 218.

## Controlling the flow of items to an activity

As discussed in “Scheduling activities” on page 173 you can have an activity start based on the time of day. Some methods of adding an additional activity to a model can cause the new activity to cycle on and off frequently. You may not want this to occur, as it can result in higher start up costs, increased machine wear and scrap production, and excess energy consumption. Instead, you can add some *hysteresis* and have the activity stay on to process a number of items, or stay on for a period of time.

When bringing a system on-line, there are two main ways to control the flow of items to an activity:

- Specify the number of items that will be processed
- Specify the amount of time the activity will be on-line

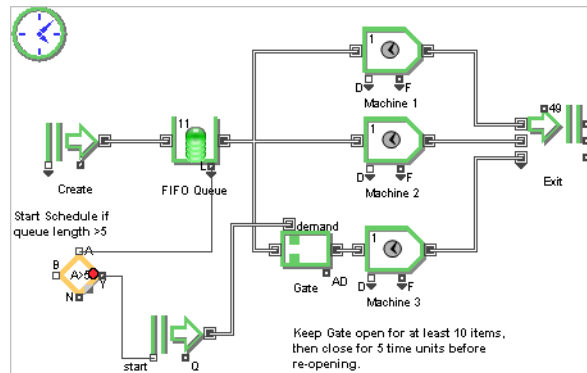
### Fixed number of items

Instead of having a system cycle on and off, you may want to keep the optional activity running. For instance, you can keep a machine on to process a particular number of items, even if the waiting line for the other machines is below the threshold that originally activated it. This reduces the number of times the machine turns on and off.

### Fixed Items model

To keep an activity running until it has processed a fixed number of items, add a Create block between the Decision and Gate blocks to the model discussed in “Conditional routing” on

page 157. (The Conditional Routing model is located in the Examples\Discrete Event\Routing folder.) The resulting model is named Fixed Items, and is shown here.



Fixed Items model

In this model, the optional machine will be brought online if/when the queue exceeds five items. The Y output of the Decision block (Value library) is connected to the start connector of a Create block, which is set to *Create items by schedule*; the schedule is shown at right. The Decision block outputs a true value (1) at Y if the queue length is 5. When that happens, the Create block (Control the Gate) starts its schedule and puts out a single item with a quantity of 10 to the demand input on the Gate block. This causes the Gate to stay open until 10 items pass through the block.

Enter a schedule of arrival times

	Create Time	Item Quantity
1	0	10
2	5	0

Link

Repeat the schedule every 10

Activating "demand" with an item

There are two items of special note in this model: how the *start* and *demand* connectors are activated.

- Since the Create block's *start* connector is connected, and since the block is in *Create items by schedule mode*, the block's schedule runs in relative simulation time (begins its schedule relative to when start is activated), as explained in the section "The Start connector" on page 114. Once start is activated (gets a value  $\geq 0.5$ ) it causes the entire schedule to happen; messages from the Decision block to the start connector are ignored until the schedule is complete. The second line of the schedule means that the Create block will also ignore any start messages for 5 time units after the schedule has been completed. This provides hysteresis and allows the machine to process items (and hopefully reduce the buffer length) before the sequence is activated again. If the schedule did not include this pause, the Activity block could be activated constantly.
- The demand connector is activated when it gets an item, causing the Gate to open and allow items through. The number of items allowed through before the Gate closes is determined by the quantity of the item at *demand*. For instance, each item with a quantity of 10 creates a demand for 10 items before the Gate will close.

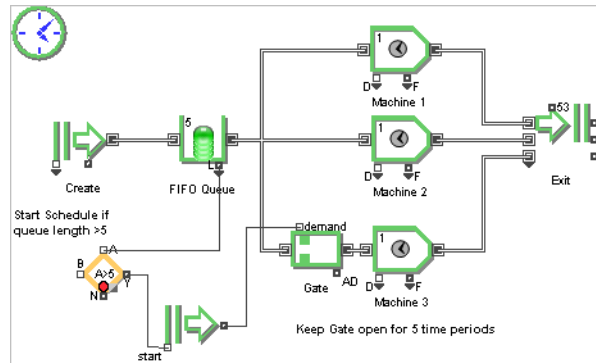


### Fixed period of time

You may want to keep the optional machine on for a particular length of time instead of for a certain number of items.

#### Fixed Time model

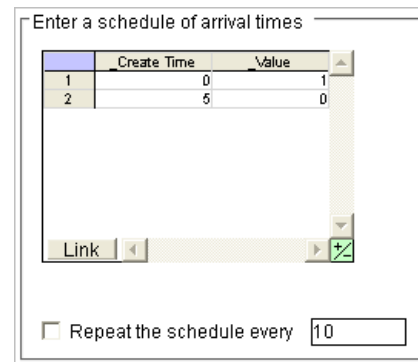
To do this, use the Create block, set to *Create values by schedule*, to output *values* to the demand connector on the Gate block.



Fixed Time model

In the table in the Create block's dialog, the first line has 0 for the output time and a value of 1. The second line has the time you want to turn off the optional machine, and a 0 for the value. For example, to keep the optional machine on for five minutes, you enter the values as shown at right.

Once start is activated, the demand connector will receive a value of 1 (True). After 5 time units have passed, the value at demand will change to 0 (False). Because you are connecting a value output to demand, the Gate block will stay activated as long as the value the demand connector receives is greater than or equal to 0.5, which in this example is for 5 time units.



Activating "demand" with a value

### Interrupting processing

In discrete processes, it is common for interruptions to occur. This could happen for any number of reasons, such as the arrival of an item that has a higher priority for processing, random machine failures, planned shutdowns, the occurrence of a higher priority event, and so forth. Interruptions are of two kinds, preemption and shutdown.

- *Preemption* occurs when an Activity block is told to prematurely end one or more items' processing. When this occurs, the Activity immediately sends the preempted items out of the block through an alternate item output connector.
- *Shutdown* occurs when processing is suspended for one or more items currently in an Activity block. Items that have been shut down may or may not have their processing completed when the shutdown ends. In any case, they are either discarded or leave through the normal item output connector.

The Activity block has a Preempt tab for specifying what to do when there is preemption and a Shutdown tab for controlling what happens when the block gets a message to shutdown. The Convey Item block can be shutdown by reducing its speed to zero. The Shutdown block is most commonly used for shutting down an activity.

☞ Preemption and shutdown are discussed in the following two sections. The models for those sections are located in `Examples\Discrete Event\Processing\Preemption and Shutdown` folder.

### Preemption

Preemption occurs when a signal is received at an Activity block's PE (preempt) input, prematurely ending an item's processing by forcing it to leave through an alternate output.

In the Preempt tab you can specify that preemption occurs only if the block is already processing its maximum number of items and that the preempted item's remaining processing time be stored as an attribute for subsequent processing. Once preempted, the item's processing can be finished by another Activity, finished later by the original block, or never finished at all, depending on how the item is routed in the model.

#### PE input connector

Once preemption is enabled in the Preempt tab, the PE (preempt) universal input connector and an alternate item output connector appear on the Activity's icon. The connection to the PE connector can either be a value input or an item input, depending on what is selected in the Preempt tab. The type of connection to the PE input determines how preemption is controlled:

- Value connection. Based on which of the first four preemption options (discussed below) is chosen, the selected item or items will be preempted whenever a true (0.5 or greater) value is received at the PE input.
- Item connection. When a "preemption item" arrives at the PE connector, the Activity looks up the specified attribute value on the preemption item. The Activity then searches all items currently in processing, and any of those items with an attribute value equal to the one on the preemption item will be required to leave the block.

#### Preemption options

As discussed above, preemption occurs when a signal is received at an Activity block's PE (preempt) input. Settings in the block's Preempt tab determine which item or items must leave. Depending on whether the preempt signal is sent by a value or an item connector, the preemption options are:

- The item that is closest to finishing
- The item that is furthest from finishing
- The item with the lowest priority
- All items currently being processed
- Only items with a particular attribute value

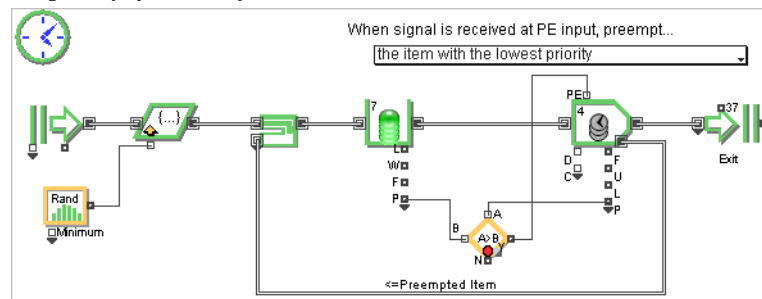
The first four options are only available when a value connection is made to PE; the last option is only available, and is the only choice, if an item output is connected to PE.

#### Preempting model

For instance, an Activity and Queue (set to *Sort by: priority*) can be used in conjunction with a Decision block (Value library) in such a way that lower priority items being processed *may* be preempted to make room for higher priority items as they arrive at the Queue. Note that it is not

guaranteed that a lower priority item will be preempted. If “Preempt only if block is full” is checked, items will be preempted only if the Activity block is full.

In the Preempting model, the Queue reports the priority of the item that is about to leave and the Activity reports its lowest priority item; this information is sent to the Decision block. If it is determined that there is a higher priority item in the Queue than is being processed by the Activity, a True signal (a value greater than 0.5) is sent to the Activity’s PE input. Notice that the Activity’s dialog is set to *When signal is received at PE input, preempt... the item with the lowest priority* and to *Preempt only if block is full*.



Preempting model

Unless they are preempted, items arriving to the Activity block are processed for the time indicated in the block’s dialog. In block’s Preempt tab, *Store remaining time in attribute: remainingTime* is selected. If an item is preempted, the Activity attaches the remaining processing time to the item as an attribute named *remainingTime*. Since the Activity also has *Use this attribute as delay* checked, when the preempted item returns to the Activity block it will process only for the time indicated by the *remainingTime* attribute.

### Shutting down

Employee breaks, equipment maintenance, inventory-taking closings, and tool failures all involve interruptions in activities for a period of time called *downtime*. If interruptions are significant, models should include provisions for shutting down activities to avoid overly optimistic predictions.

Shutdowns involve a temporary or permanent halting to the processing of items currently in the Activity block. The block’s Shutdown tab has settings to determine which items should have their processing shut down, how long to interrupt the processing, and what to do with the items in an Activity when the shutdown occurs.

You can shut down activities at a scheduled time, such as for vacations or machine maintenance, or it can be a random occurrence, such as for equipment failures or emergency leaves. Activities can also be shut down based on some factor in the model, for instance when a downstream Queue is full. Like shutdown occurrences, the duration of the downtime can be a constant value or a random number. A shutdown can also be used to block the entry of additional items while the shutdown is in effect.

### SD input connector

Once shutdown is enabled in the Shutdown tab, the SD universal input connector appears on the Activity’s icon. It is common to connect from a Create or Shutdown block to the SD input but connections can be made from other blocks as well. The input to the SD connector can either be a

value connection or an item connection, depending on what is selected in the Shutdown tab. The type of connection to the SD input determines both which items are shut down and for how long:

- Value connection. This acts like an on/off signal. The entire block will be shutdown whenever a true (0.5 or greater) value is received at the SD input. This suspends the processing of all items in the block and stops new items from entering it. The Activity will stay shut down until the SD input gets a false (less than 0.5) value.
- Item connection. When a “shutdown item” arrives at the SD connector, the Activity will shut down the item or items currently being processed, as specified by the shutdown options discussed below. The duration of the shutdown is determined by an item’s attribute or quantity as specified in the Shutdown tab; the value of that property on the shutdown item determines how long the shutdown will be in effect.

The Create block is used to schedule the shutdown for the “Scheduled Shutdown model” on page 181; the Shutdown block provides random shutdowns for random durations for the “Random Shutdown model” on page 182.

### **Shutdown options**

As discussed above, shutdown occurs when a signal is received at an Activity block’s SD (shutdown) input. Settings in the block’s Shutdown tab determine which items currently in processing will be shut down. Depending on whether the shutdown signal is sent by a value or an item connector, the shutdown options are:

- All items currently in processing
- A randomly chosen item
- Items whose attribute matches the attribute at SD
- Entire block

The first three choices are only available if an item connector is connected to SD; “Entire block” is only available, and is the only option, if a value output is connected to SD. If “Items whose attribute matches the attribute at SD” is selected, the Activity looks up the specified attribute value on the shutdown item. The Activity then searches all items currently in processing, and any of those items with an attribute value equal to the one on the shutdown item will be shutdown.

### **Item options**

Once a shutdown is in affect, the shutdown items are handled in one of four ways, as specified in the Shutdown tab of the Activity’s dialog:

- The item can be discarded, such as when food is spoiled by the machine going down.
- The Activity can resume processing the item after the shutdown ends.
- An Activity can restart processing the item after the shutdown ends.
- The Activity can finish processing the item prior to shutting down, such as when the shutdown is part of scheduled maintenance and can wait until the item is finished.

Items that are not discarded leave through the Activity’s normal item output.

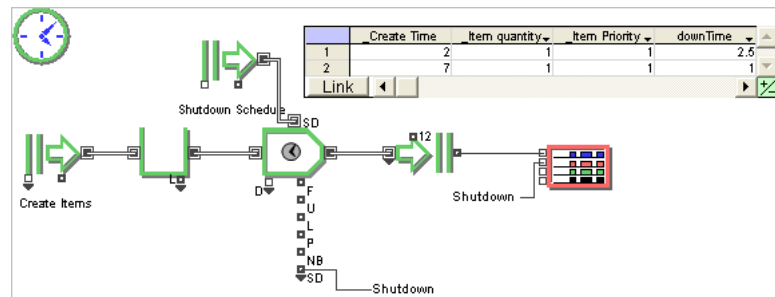
### **SD output connector**

The Activity block’s variable output connection contains an SD connector that can be used to relay shutdown status. Depending on how the Activity has been configured, this connector either outputs a 1 (one) while the Activity is down and a 0 (zero) when it is up, or it outputs the number of

items that are currently shutdown. If a value connection has been made to the SD input, the SD output connector is set to 1 or 0; if an item connection has been made to the SD input, the SD output connector reports the number of items currently shutdown.

**Scheduled Shutdown model**

The Create block is often used to schedule an Activity to shut down. For example, the Scheduled Shutdown model schedules downtime for a machine by connecting a Create block's item output to the SD input on an Activity.



Scheduled Shutdown model

The “Shutdown Schedule” from the Create block's Shutdown tab is cloned onto the model worksheet and shown at right. It indicates that two items will be created, one at time 2 and one at time 7. Each item has a downTime attribute as seen at the top of the fourth column; the attribute's value is 2.5 for the first item and 1 for the second item. In the Create block's dialog, this maintenance schedule has been set to repeat every 10 time units.

Create Time	Item quantity	Item Priority	downTime
1	2	1	2.5
2	7	1	1

Schedule in Create block

The Activity block is set to process one item at a time. Its Shutdown tab, shown at right, indicates that *all items currently in processing* will be shutdown when a “shutdown item” is received at SD, *downTime* is the name of the attribute that determines the duration of the shutdown event, and the block will *keep items, resume process after shutdown*.

Enable shutdown

SD (shutdown) input is from:

When signal is received at SD input, shutdown...

Shutdown duration specified by:

When activity shuts down:

Settings in Activity block

With this information and the settings in the Activity's Shutdown tab, processing will shut down for any item that is already in the Activity at time 2 and that item will not resume processing until time 4.5. Likewise, any item in the Activity at time 7 will be shut down and not resume processing until time 8, and so forth.

When shutdown is triggered by an item connection to SD, the selected items being processed by the Activity block will be shut down at the time and for the duration specified. However, because

the block supports parallel processing, if items arrive to the block after the shutdown has been triggered, those items will be processed normally. For complete shutdown of the block, use a value connection instead of an item connection.

When the model is run, the plotter will show both the number of items processed (obtained from the Exit block) and the timing and duration of the shutdowns (obtained from the SD output on the Activity.) The SD output gives a value of 1 while an item is shutdown and a value of 0 while it is not.

### The Shutdown block

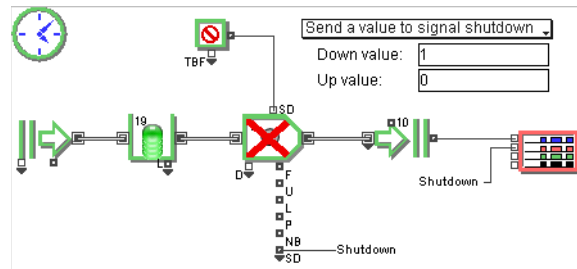
Time between failures (TBF) and time to repair (TTR) are common ways of determining how frequently shutdowns occur and how long they last, respectively. You can use the Shutdown block to specify a fixed or random TBF and/or a fixed or random TTR.

The Shutdown block has been specifically designed to work with the SD input on the Activity block and it is capable of generating value or item-based shutdowns according to TBF and TTR distributions you select.

- When set to *Send a value to signal shutdown*, the Shutdown block generates values that represent a down or up (off or on) state.
- When set to *Send an item to signal shutdown*, the Shutdown block randomly generates one item for every shutdown event. The item contains attributes that indicate to the Activity both which items will be shut down and for how long that will happen.

### Random Shutdown model

While the Scheduled Shutdown model on page 181 used a Create block to schedule when an Activity would be down, the Random Shutdown model uses the Shutdown block to halt processing on a random basis for a random amount of time. It does this by connecting the Shutdown block to an Activity block's SD input.



Random Shutdown model

In this model the dialog of the Shutdown block is set to:

- **Send a value to signal shutdown**
- Output a **Down value: 1** and an **Up value: 0**
- Use an **Exponential** distribution with a **Mean: 9** for the **Set time between failures (TBF)** parameter
- Use a **Triangular** distribution with the settings **Minimum: 1**, **Maximum: 3**, and **Mostly likely: 2** for the **Set time to repair (TTR)** parameter.

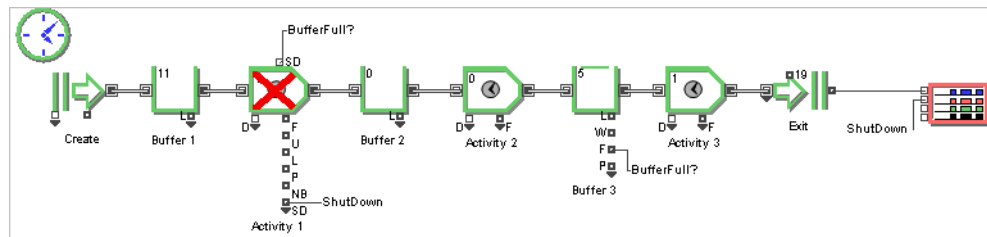
With these settings both the TBF and TTR are random. A true signal (a value of 1) from the Shutdown block will cause the Activity block to shut down the entire block, halting all items currently in processing and blocking any new items from entering. It is only when the signal switches from true to false (a value of 0) that the block will resume processing. In this case, the Activity will remain down for approximately 2 hours before coming back online.

**Model-related shutdown**

The Scheduled Shutdown and Random Shutdown examples presented earlier showed how to shut down a process in isolation from other events in the model. You can also shut down activities based on model factors such as the length of a waiting line or whether another activity is in process.

*Explicit Shutdown model*

The Explicit Shutdown model represents a buffer downstream from a machine that reaches a limit. So that the queue length will control shutdown events, the *F* connector from the Queue block is connected to the *SD* input connector on the Activity block.



Explicit Shutdown model

Discrete Event

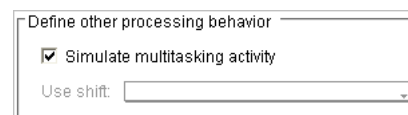
For this model the Queue’s limit is specified in its dialog; whenever the Queue is full, the *F* connector outputs 1. The Activity’s Shutdown tab indicates that the *SD* (shutdown) is: *value input connection* and that therefore the *Activity stays down until SD value < 0.5*. This causes the Activity to stay shut down for as long as the Queue is full.

This is an example of how to have downstream factors affect upstream activities. If you examine the model closely, you see that the last machine is processing so slowly that Queue 3 quickly reaches its limit of 5 items. Since a Queue cannot take in any more items while it is full, the middle Activity is blocked (cannot process a new item until an item is removed from Queue 3). However, the first Activity continues to process items, filling Queue 2. By explicitly shutting down the first Activity, you affect where items are stockpiled and which Activities are shut down when one of them is blocked.

👉 Please also see “The Shift block” on page 218 for examples of activity and resource allocation that are tied together and scheduled as “Shifts.”

**Multitasking**

The Activity block has a checkbox in its Process tab that allows the block to simulate multitasking. Choosing this option means that the block’s available time to process items (its Delay time) must be divided between each of the items in the block. This causes each item to take longer to finish processing and leave. Examples of multitasking include computer processors or a person who is working on multiple tasks at the same time.



Choosing multitasking in an Activity

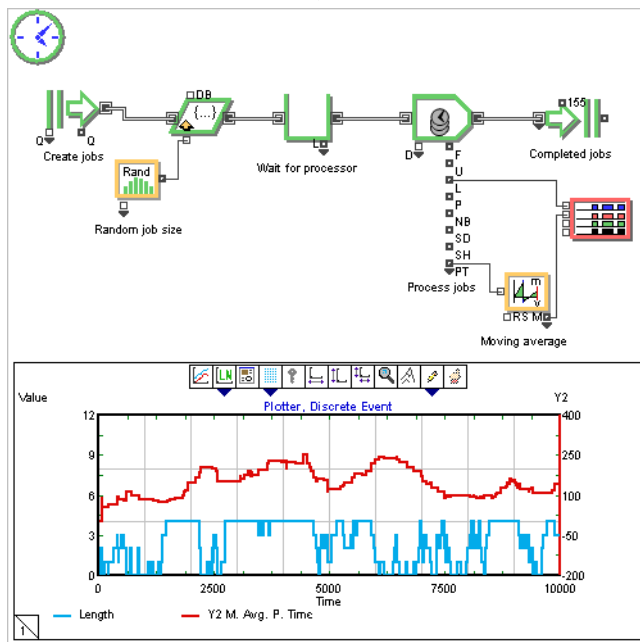
With multitasking, if only one item is in the block the actual processing time will be exactly the same as the original delay time specified by the block. If two items are in the block, each item will take twice as long as the original specified processing time. If three items are in the block, their processing times will be multiplied by three, and so forth. This is equivalent to situations where a single server or operator has to divide their available time between multiple customers or tasks.

The changes to the processing time occur dynamically as items enter and leave the block. When new items enter the Activity, the remaining delay times for all of the items in the block will become progressively longer. As processed items leave the block, the delays for the remaining items will become shorter.

### Simulate Multitasking Activity model

In this example model, there are three kinds of jobs that are being processed on a computer. The computer must share its time between all the jobs it is simultaneously working on.

Discrete Event



Simulate Multitasking Activity model

The Simulating Multitasking Activity creates a random number of each type of job (small medium and large), as determined by probabilities entered in the Random Number block (Value library). Each job has a Job Size attribute with the string value small, medium, or large. The job's processing time is defined by a distribution and entries in a table in the dialog of the Activity block, which is set to *Delay is: from a lookup table*. (For a description of this setting, see "Processing time for an Activity" on page 167.)

Running the model shows that the moving average processing time increases as the number of items in the Activity increases.

☞ Changing the capacity of the Activity changes the number of items allowed in the block, but it does not change the calculation for the delay time. Thus, if "Simulate multitasking activity" is



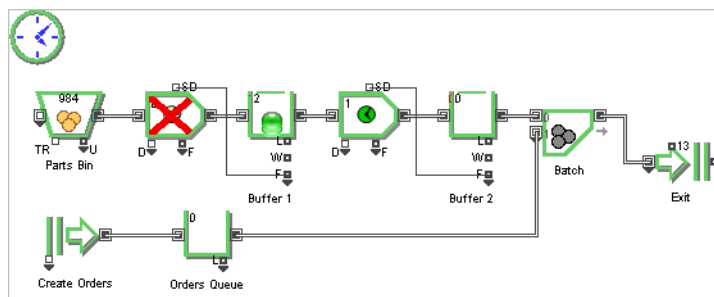
enabled, increasing the capacity will not necessarily increase the throughput rate of the Activity block.

## Kanban system

A kanban just-in-time (JIT) inventory system limits the amount of inventory between processing stations with a controlling “kanban” card. In this type of system, a station is only authorized for processing if a kanban for that part is available. When processing is complete, the kanban moves with the part to the next station. As the next station consumes parts, it returns the kanbans to the previous station to authorize additional processing.

### Kanban model

A Kanban system is modeled in ExtendSim by monitoring the queues between machines and having that information regulate processing. To do this, set the Queue block capacity to the number of kanbans and connect the *F* (full) output from the block back to the preceding Activity’s *SD* (shutdown) output connector. When the queue has remaining capacity, its *F* connector will output 0 (zero) and the preceding Activity block will be authorized to produce parts. If the Queue block is full, its *F* connector will output 1 and the preceding machine will be shut down until the queue length is reduced.



Kanban model

If you run this model with animation on, the Activity blocks will be struck through in red while they are shut down.

## Transportation and material handling

The following Item library blocks are used to represent transportation and material handling:

- Transport
- Convey Item
- Resource Item

The Convey Item and Transport blocks represent ovens, conveyors, and so forth to provide fixed-path routes with a specified travel time for items. The Resource Item and Transport blocks are used with the Batch and Unbatch blocks to simulate AGVs and other independently moving vehicles.

The models discussed in this section can be found in the Examples\Discrete Event\Processing\Material Handling and Transportation folder.

### Travel time

In a discrete event model, items move from block to block as dictated by the connections. These connections indicate the direction of movement, but they don't provide any delay for the items. If travel time is significant, it is common to either:

- Increase the delay time of destination blocks to compensate for the travel time.
- Specify a minimum wait time in a Queue block's Options tab to simulate travel time.
- Set an explicit travel time in a Convey Item or Transport block, as shown below.

### Transport blocks

Transport blocks (Item library) move items from the start of a path to the end based on distance and speed information. When the model is animated, they can display multiple items travelling a certain distance simultaneously.

Transport block: Behavior tab

While you can choose either feet or meters as the distance unit in the block's dialog, the default is feet. If *Metric distance units* has been selected in the Edit > Options command, the default will be meters.

### Travel time options

The block's Behavior tab has options that specify how to calculate travel time – the time it will take an item to travel from the starting point to the ending point. Travel time can be based on:

- Move time. Each item will take the amount of simulation time that is entered in the *Move time* field or received at the D input connector. This acts just like a delay in an Activity block; length and speed are ignored.
- Speed and distance. How fast the item is traveling, and how far the item must travel to reach its destination, are entered in the fields in the Behavior tab or received at input connectors. The calculated move time is displayed in the dialog. This option is most often used if the transportation pathway is centered around the block.
- Speed and calculated distance. The item's speed is entered in the *Item speed* field or received at an input connector. The distance is determined from information entered in the frame labeled "Select From and To locations for calculated distance", as discussed below. This travel time option is most commonly used for 3D animation when you want the location of the transportation pathway to be independent from the block.

If *Speed and calculated distance* is selected, the starting and ending locations (which determine the distance) must be defined in the tab.

### Calculated distance

If *Speed and calculated distance* has been selected as the travel time, a frame appears at the bottom of the dialog for entering the distance information. If the *from* and/or *to* locations are set to anything other than *Entered X and Y location*, the relative

Frame for entering information to be used to determine the distance

Discrete Event

positions of the blocks in the model, and their connections, determine the distance.

The factors considered in the calculation are: the “from” location, the “to” location, how the distance is calculated, and the distance ratio.

*From location options*

- Previous non-passing block (the default)
- Entered X and Y location
- Block location
- Enclosing hierarchical block
- Previous block

(See explanations following the To location options.)

*To location options*

- Next non-passing block (the default)
- Entered X and Y location
- Block location
- Enclosing hierarchical block for next block
- Next block

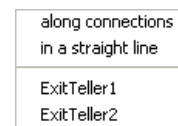
Note: In the Item library *non-passing blocks* are either residence or decision types of blocks, as described in “Types of item handling blocks” on page 96. If *Entered X and Y location* is selected, the numbers can either be entered in the dialog or defined by the location of block in the model; remember that the 3D coordinates are expressed in meters. The *block location* option means the current block; this choice is especially helpful when the *from* location is a previous block (non-passing or not).

 If the selected *from* and/or *to* locations are blocks, the distance starts at the *from* block’s output connector and ends at the *to* block’s input connector.

*Calculate distance options*

The popup menu provides two methods for calculating the distance from the start to the end of the path:

- Along the connections between the *from* and the *to* locations
- In a straight line between the *from* and the *to* locations



More options are available if the E3D window is open and you have created custom paths for 3D animation. In this case, the paths will appear at the bottom of the popup menu as shown.



Button on Transport Animation tab

The block’s Transport Animation tab has a button, shown at right, that will calculate the length of the currently selected custom 3D path. It then puts that length into the distance parameter field on the block’s Behavior tab.

Notice that the shape of the conveyor will not visually change with these choices, but the information is included in the calculation of the conveyor’s length. For instance, if there is a series of right-angle connections between the *from* and the *to* locations, the conveyor’s length will be longer than if the straight line option had been selected.

*Distance ratio option*

This popup menu is for specifying the ratio between pixels in the 2D model and meters in the E3D window. It will control how the distances defined in the Behavior tab affect 2D or 3D animation. The choices are:

- Use speed and distance directly. Specifies a ratio of 1 pixel to 1 meter.
- Use 3D distance ratio. Uses the distance ratio defined in the Run > Simulation Setup > 3D tab; the default is 20 pixels per meter.
- Use distance ratio of: This is for entering the ratio directly. A value of “x” means a ratio of “x” pixels to “x” meters or feet.

☞ These options are explained more in the comprehensive example discussed in “How the length is calculated” on page 189.

**Convey Item blocks**

The Convey Item block (Item library) moves items along a conveyor, oven, cooling unit, moving walkway, or any other type of moving path. The items travel along the length of the conveyor, from its start to its end. The Behavior tab is similar to that for a Transport block, discussed above. The differences are:

The screenshot shows a dialog box titled "Specify conveyor behavior and how item length is determined". It contains several input fields and dropdown menus:
 

- A dropdown menu for "Accumulating conveyor".
- A dropdown menu for "Travel time" set to "move time".
- Input fields for "Move time" (value: 1) and "time units".
- Input fields for "Length" (value: 70) and a dropdown for "feet".
- Input field for "Capacity" (value: 4).
- A dropdown for "Item length is" set to "a constant" and an input field for the value "1".

Convey Item block: Behavior tab

- The Convey Item block can be:
  - Accumulating. If the block’s ability to pass items through exceeds downstream demand, any items delayed from exiting will begin piling up at the outflow end of the block, up to the *Capacity* setting.
  - Non-accumulating. If downstream demand exceeds the block’s ability to pass items through, the conveyor will stop until the item at the end moves into the next block.
- Instead of the Transport block’s reference to the *distance* from one point to another, the Convey Item block is concerned with its *length*.
- If *Travel time: move time* is selected, a *length* entry is also required. In combination with the item length (discussed below), the conveyor’s length determines how quickly items can move onto the conveyor.
- The length of the items that pass through this block must be defined. They can be defined as:
  - A constant
  - From an attribute
  - Based on length and capacity. (For instance, if the conveyor’s capacity is 1,000 units and its length is 50 units, the length of each item will be 50/1000, or 0.05 units.)

Discrete Event

Item length does not visually change the item picture or object but it is included in calculations for accumulation, capacity, and the timing of when items are pulled onto and released from the block.

### How the length is calculated

When *Travel time: speed and calculated length* is selected in a Convey Item block, the relative positions of blocks in the model, and their connections, determine the path's length. The following example explains how the length is determined.

The bottom portion of the Behavior tab for a Convey Item block is seen at right. In this frame:

Select From and To locations					
	2D	3D		2D	3D
From X location:	211	10.55	To X location:	344	17.2
From Y location:	165	-8.25	To Y location:	165	-8.25
From location is:	previous non-passing block				
To location is:	next non-passing block				
Calculate length:	in a straight line				
Distance ratio:	use 3D distance ratio				

Length calculations for Convey Item block

- The 2D *From x location* is 211 pixels (the position of the previous non-passing block) and the *To x location* is 344 pixels (the position of the next non-passing block).

- By default, the length is calculated *in a straight line* between the *from* location and the *to* location. (The alternative is to calculate the distance along the connections.) The shape of the block's object will not visually change with either choice, but the choice affects the determination of the path's distance. For instance, if there were a series of right-angle connections between the *from* and the *to* locations, and *along connections* was selected, the distance would be longer than if the straight line option had been selected.

- The *Distance ratio* (pixels per meter) is based on the 3D distance ratio set in the command Run > Simulation Setup > 3D Animation tab. The default is 20 pixels per meter, as shown at right.

Distance:  pixels per meter

Distance conversion ratio

In addition, the block's Block Animation tab is set by default to stretch the object to the conveyor's length. This will cause the Conveyor object in the E3D window to automatically resize to reflect the calculated length.

Stretch 3D object to conveyor's length

Setting in Block Animation tab

#### How the settings affect the length

The distance between the previous non-passing block (the *from* location) and the next non-passing block (the *to* location) is calculated in a straight line between the output connector of the *from* block and the input connector of the *to* block. This is 133 pixels (344 pixels - 211 pixels).

The distance ratio of 20 pixels per meter is used to convert the 133 pixels into meters. The result is the length of the conveyor: 6.65 meters. as can be seen in the block's Behavior tab. (This is also reflected in the 3D information, where the *From x location* is 10.55 meters and the *To x location* is 17.2 meters, a difference of 6.65 meters.)

- ☞ If the *from* and/or *to* locations are blocks, the determination of the length starts at one block's output connector and ends at the other block's input connector.

Moving this Convey Item block along the connection line between the *from* and the *to* blocks will not have any affect on the conveyor length. This is because the distance between the *from* and *to* locations stays the same. However, moving the *from* block away from the Convey Item will change the length of the conveyor. This is because the *Next non-passing block* is now further away from the *Previous non-passing block*.

For information about how these blocks are used for 3D animation, see “Adding 3D behavior to an existing model” on page 406 and “Animating a bank line” on page 416.

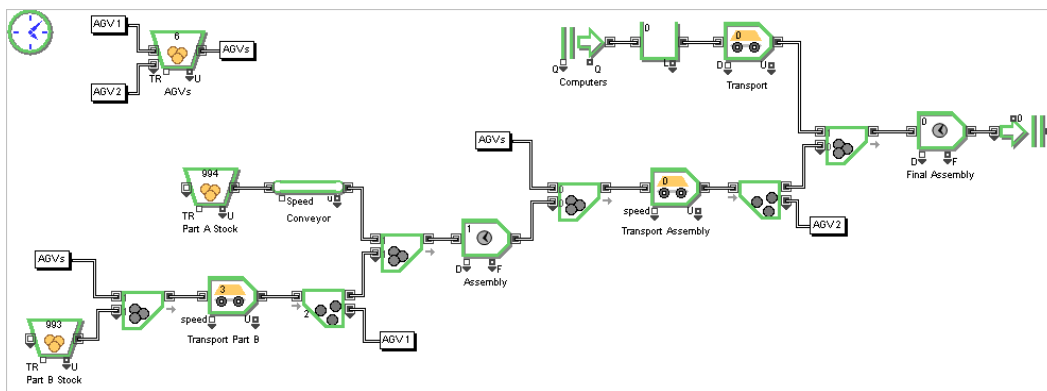
### Transportation models

The following models use Convey Item, Resource Item, and Transport blocks to simulate item movement along fixed paths and material handling using AGVs.

#### Transportation 1 model

To model vehicles such as AGVs in ExtendSim, use a Resource Item block to provide items that represent the vehicles, a Batch block to attach the vehicle with whatever it is transporting, one or more Transport blocks to provide the transportation delay, and an Unbatch block to separate the vehicle from its load at the end of the route.

Discrete Event



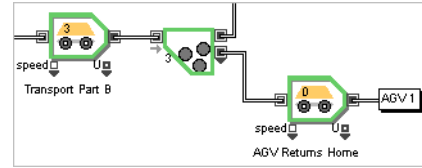
Transportation 1 model

The Transportation 1 model shows how two parts are assembled, inserted into a computer, then moved to a loading dock. Part A is moved by a Convey Item block to the assembly machine while Part B is moved there by a Transport block. The two parts are joined by a Batch block for processing. After assembly, they move by a Transport block to the machine that will put them into the computer. Since the computer is heavy, it is moved to the final assembly machine by a Transport block that represents a small crane.

To simulate movement, AGVs from a Resource Item block are batched to the parts/items and then moved via the Transport block. If the AGV is released at the end of each route, as it is in the model above, the part is left behind where it can then be processed before moving on to another section of the model. To model a situation where the Resource Item transports the item to an activity, then waits there to continue transporting the item again, don't release the Resource Item until the last stop in the route.

**Transportation 2 model**

In the Transportation 1 model there is no time associated with the return of the AGVs to the Resource Item block. To model how long the return path takes, insert Transport blocks after the Unbatch blocks and enter a distance and speed for the AGV return trip. This is shown in the model segment shown at right.



Adding time for AGVs to return





# **Discrete Event Modeling**

## **Batching and Unbatching**

Joining items or separating them

In a discrete event model, items pass through the system and something is done to or with them. The process often involves temporarily or permanently joining, or *batching*, resources or other items with the original item. For instance, in a manufacturing plant, precursors of the final products come into the process as raw materials, subassemblies, and packaging that are joined in various combinations. During the manufacturing process they are often batched with other precursors and require additional resources such as pallets and workers for processing. These batched items move through the process together.

These same concepts apply to other discrete processes. For example, in an emergency room model, doctors are temporarily batched with their patients during medical diagnosis. In the same model, a technician, a diagnostic machine, and a patient would be batched for the duration of x-ray treatment. In a communication system, multiple packets might be batched together to create a single message. For a retail store model, customers could be shown arriving and selecting merchandise, then be temporarily batched with a sales person to make the purchase.

This chapter discusses:

- Blocks for batching and unbatching
- How to batch and unbatch items
- Dealing with item properties when items are batched or unbatched
- Delaying the batching of items until all requirements are present
- Preserving unique item properties as items are batched and unbatched

☞ The models discussed in this chapter can be found in the folder \Examples\Discrete Event\Batching.

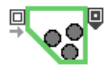
### Blocks of interest

The following blocks will be the main focus of this chapter. The block's library and category appear in parentheses after the block name.



#### **Batch** (Item > Batching)

Joins multiple items into a single item for use in the model. This causes the original input items to be destroyed and replaced by one output item. A batched item may be unbatched at a later point in the model, but that is not required.



#### **Unbatch** (Item > Batching)

Outputs multiple items for each input item. Depending on selections in the dialog, this block can separate items that were previously batched or make duplicates of items that were never batched.

### Batching

Batching allows multiple items from different sources to be joined as one new item for simulation purposes (processing, routing, and so on). The Batch block accumulates items from each source up to a specified count, then releases a single item that represents the batch. In this process, the original input items are destroyed and replaced by one new output item.

☞ Items may have properties, such as attributes, before they are batched. To specify what will happen to the properties of items that have been replaced by a new batch item, see “Properties when items are unbatched” on page 204.

The number of items required for a batch is called the *batch size*. In some situations you know in advance how many of each item is required to make one item; in other situations the number of items batched depends on model factors and changes dynamically.

Items can be permanently batched together as one new item that flows through and exits the model, or they can be temporarily joined for some specific purpose and unbatched at a later point in the process. For example, two manuals could be batched with three promotional pieces and one CD to make a software package that is shipped as one product. Or a ship attempting to dock might be temporarily batched with two tugboats resources to guide it through the docking process, after which the tugboats and the ship are sent on separate paths.

### Batch dialog

The Batch block has three tabs for determining when items should be batched, how many items to include in a batch, and what to do with the properties of items that are batched. The Batch and Options tabs are discussed below; options in the Properties tab are described on page 199.

#### Batch tab

The top of the Batch block's Batch tab has a popup menu with two options that determine how the block behaves. These are summarized below and illustrated in models later in this chapter.

- **Batch items into a single item** creates a batch using items on a first-in, first-used basis as they arrive at the item input connectors. The quantity of items required from each input is entered in a table or (if this choice is selected on the Options tab) determined by the value at a BatchQuantityIn connector.
- **Match items into a single item** creates a batch of items that have a common attribute value. For instance this could represent a process where items were combined together based on a serial or order number. For this choice, it doesn't matter which input connector the items arrive from.

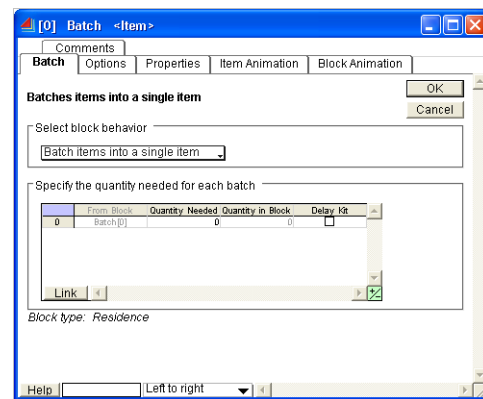
The table in the Batch dialog is for entering the number of items from each input that are required to make a batched item; the size of the batch can also be set using value input connectors as discussed later in this chapter. The first column shows the block label or name the input is connected to. The Quantity column is for specifying the number of items required from the input and the next column reports the number of items available. Checking the fourth column determines if Delay Kit is activated; as discussed on "Delaying kits" on page 201, this causes the specified item or items to not be pulled into the block until certain conditions are met.

#### Options tab

Among other choices, this tab has options for setting the size of a batch through value input connectors and determining when to start a batch.

##### BatchQuantityIn connectors

Checking *Use quantity input connectors* on the Batch block's Options tab enables the BatchQuantityIn variable connector. Each BatchQuantityIn value input connector corresponds to an item input connector and controls the batch size for that item connector. If a value input con-



Batch tab in Batch block

connector has been connected, it will set the number of items required at its adjacent item input connector. If a value input connector is not connected, the number of items for the adjacent item connector will be set by the value in the Batch dialog.

When *Use quantity input connectors* is checked, there are two options that affect the batch size. With either option, the *initial* size of the batch is the value at the BatchQuantityIn connector when the first item on its corresponding item input connector arrives to the Batch block. The options determine what happens if the input value changes:

- **Dynamically as batch is created.** If the value at a BatchQuantityIn connector changes before that item connector's batch is released, the number of items required for that batch will change as well. This enables the size of a batch to be changed dynamically.

☞ The number of items to be batched from each input connector can never be less than the number of items that have already arrived to the block from that input. That is, if 10 items have already been pulled in through an item input connector, and the BatchQuantityIn connector changes to 8, the batch size for that item input connector will be set to 10, not 8.

- **By first item at each connector.** The size of the batch does not change after the first item for the batch has arrived, even if the value of the BatchQuantityIn connector changes. Once that batch is released, a new batch size can be set.

#### *The demand connector*

To enable the demand connector, check the *Show demand connector* box in the Batch block's Options tab. Then choose one of the following options:

- **Start batch when value at demand  $\geq 0.5$ .** No items are brought into the Batch block until the value of the demand connector equals or exceeds 0.5. For instance, while the Batch block sees a 0 at demand, no items will enter for batching. When it sees a 1 at demand, the required items currently available will enter the Batch block to be joined together. Depending on how the model is constructed, selecting this behavior can cause blocking of upstream items.
- **Create batch when value at demand  $\geq 0.5$ .** Items are allowed into the Batch block as they are available, up to the required number. However, the batched item will not leave the block as long as the demand connector has a value  $< 0.5$ . When the demand connector becomes  $\geq 0.5$ , the batched item leaves the block. With this option, a batch can consist of fewer items than the number in the Quantity Needed column, because the batched item will have been created by joining whichever items were available when the demand connector got a value  $\geq 0.5$ .

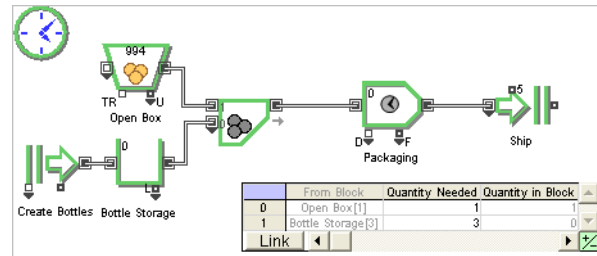
A value output connected to the Batch block's demand input is used as a true/false indicator, triggering batching. The actual value from the value connector is ignored; what is considered is whether or not it equals or exceeds 0.5.

### **Simple batching**

The simplest batching method is to cause multiple items to be joined as one new item, replacing the original items in the model. The batched item may or may not be unbatched at a later point, depending on model requirements.

### Simple Batching model

The model shown at right joins one “Open Box” with three “Bottles”; they then travel as one item to be shipped. For this model, the Batch block’s behavior is set to *Batch items into a single item*. The items are joined by the Batch block according to settings in the table in its dialog. The Batch block will not release the batched item until it has received one item (a box) from the top input and three items (bottles) from the bottom input.



Simple Batching model

### Batching by matching items

Selecting the *Match items into a single item* option in the Batch dialog allows you to specify an attribute the items must match and a different attribute that determines the batch size. With this option, each batch will be composed of items whose matching attribute value is the same; the batch size attribute of the first item in the batch determines how many items are in the batch. With this option, it does not matter which input connector the items arrive at. As items arrive to the Batch block they are segregated based on their matching attribute value until the total number of items in that group equals the batch size attribute. When this occurs a batch is created and the item representing the batch leaves the block.

Discrete Event

### Matching Items model

The Matching Items model shown below simulates a refurbishment process for police cars. As the cars arrive, they are assigned a consecutive “serial number” by the Information and Set blocks. The Information block counts each car in order and outputs that number to the Set block, which assigns the count number as the value of the Serial Number attribute. For instance, this causes the Serial Number attribute for the second car to be assigned a value of 2.

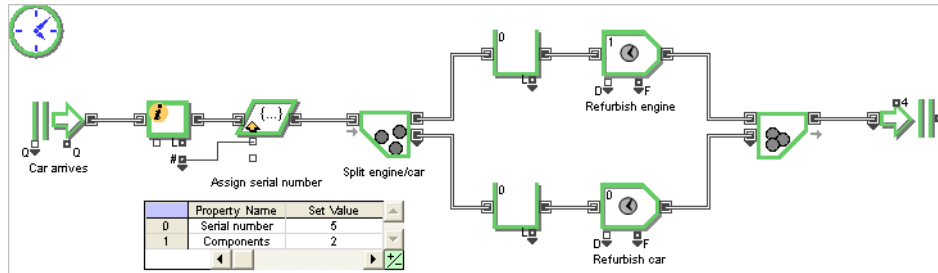
The engine is then separated from the car by an Unbatch block. (When items are unbatched, you can specify what the block should do with their properties. This is accomplished by selecting an Action for item properties in the Unbatch block’s Properties tab, as shown at right. Actions are discussed in “Properties when items are unbatched” on page 204.)

	Property	Action
0	_Animation	Batched value
1	Components	Batched value
2	Serial number	Batched value
3	_Item priority	Batched value

Properties tab of Unbatch block

These two components (the engine and the car) are refurbished individually. When both components are finished being refurbished, the engine is reassembled into its original car by matching them together in a Batch block set to *Match items into a*

single item, Match on attribute: Serial Number, and Get batch size from attribute: Components.



Matching Items model

For a similar model that uses the Queue Matching block to match items based on an attribute value, see “Matching items using the Queue Matching block” on page 138.

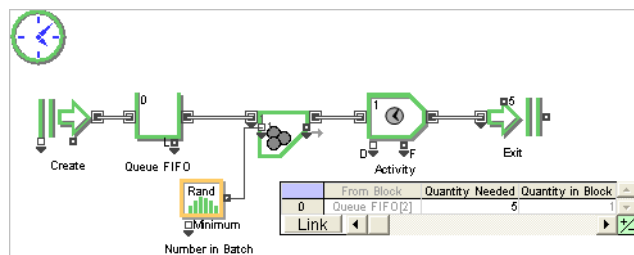
### Batching a variable number of items

Sometimes the number of items required to create a batch changes during the course of the simulation. For instance, outside factors could determine how many items go into each batch, or you may want batches to be made in a time-dependent fashion. As shown in the following two models, the Batch block’s Options tab allows you to manipulate the size of batches through quantity input connectors or through a demand connector.

For additional examples of dynamically setting batch size, see also the “Batch and Unbatch Variable model” on page 203 and the advanced batching models “Equation(I) Controls Batch” and “Queue Eqn Controls Batch” that are located in the folder \Examples\Discrete Event\Batching.

#### Batching Variable model

This example model uses a Batch block’s quantity input connector to determine the size of a batch. A Random Number block (Value library) sets random batch sizes of two to five items. In the Batch block’s Options tab, *Use quantity input connectors* is checked to enable the BatchQuantityIn connector and *Set batch size: dynamically as batch is created* is selected. The output of the Random Number block is connected to the BatchQuantityIn connector on the Batch block. These settings cause batches to be created that require a variable number of items. The information about the size of the batch can be saved on an attribute specified in the block’s Options tab.



Batching Variable model

The BatchQuantityIn connectors will not be visible on the Batch block’s icon unless *Use quantity input connectors* has been checked in the block’s Options tab.

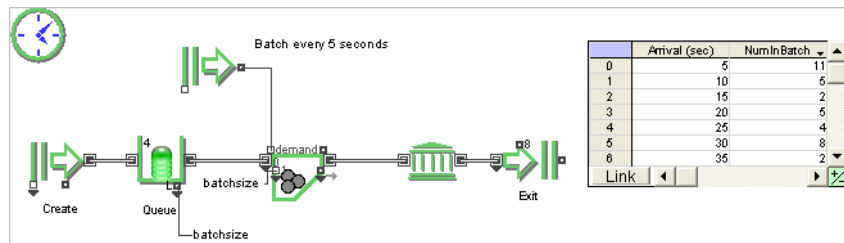
**Batch on Demand model**

You may want batches to be made in a time-dependent fashion, such as based on the time of day or on a periodic basis. The Batch block’s demand connector can be used to control when items are pulled into or sent out of the block. For example, if the model represents filling a truck with boxes, you can signal the demand connector to stop the batching at the end of the day or when another truck arrives at the loading dock. The batched item (the truckload of boxes) is then created and released. In this example model:

- The top Create block sends a value that triggers the Batch block’s *demand* connector at scheduled times, every 5 seconds.
- The Options tab of the Batch block is set to *Use quantity input connectors. Set batch size: by first item at each connector* and *Start batch when value at demand ≥ 0.5*.
- The Batch block’s quantity input (BatchQuantityIn), is connected to the **L** (length) output on the Queue block.

With these settings and connections, the Batch block will create a batch every 5 seconds, the size of the batch is dependent on how many items are available to the Batch block when it creates the batch, and items will be allowed into the block only when demand is triggered.

The information about when items arrive and the size of the batch is recorded by the History block and displayed in its cloned table.



Batch on Demand model

By default the History block clears its information each time the model is saved. A choice on the block’s dialog allows you to *Save item history with model*, but that option can cause the model to become quite large.

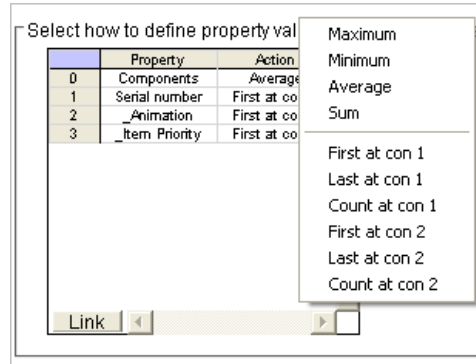
**Properties when items are batched**

As described in “Item properties” on page 110, items can have properties such as attributes and priorities. When items are combined into a batch, their properties need to be combined as well.

**Property options**

A table in the Batch block's Properties tab allows you to define what happens to item attributes and priorities. The table's first column lists every property for the items in the model. The second column, Action, gives potential options that can be taken for that property's values. This allows you to select how properties get transferred from the original items to the batched item.

The Properties tab for the Batch block in the Matching Items model looks like the screenshot at right. It shows two user-defined attributes (Components and Serial Number) and the item properties *\_Animation* and *\_Item Priority*.



Action options for properties

*Attributes and priorities*

The options that can appear in the Action column for attributes and priorities are:

- Maximum. Sets the value of the property to the largest number found on any of the items that formed the batch.
- Minimum. Sets the value of the property to the smallest number found on any of the items that formed the batch.
- Average. Sets the value of the property to the average value of that property for all of the items that are part of this batch.
- Sum. Sets the value of the property to the sum of that property's values for all of the items that are in this batch.
- First at con X (the default). Sets the value of the property to the value of that property of the first item that entered on connector X. Con 1 is the topmost item connector.
- Last at con X. Sets the value of the property to the value of that property of the last item that entered on connector X. Con 1 is the topmost item connector.
- Count at con X. Sets the value of the property to the number of items that entered on connector X. Con 1 is the topmost item connector.

*Other item properties*

An Item's quantity property (*\_Item quantity*) determines its count toward the batch size. For example, if an item arrives with a quantity of 2 and two items are required at that input, then that input is full and no further items are required for that batch from that input. In most cases, the item quantity of the items going into the batch will be 1.

Some models have an animation attribute (*\_Animation*) that stores the indexes of the 2D and 3D animation pictures for the items moving through the model. Note that the animation attribute can only be set to *First at con X* or *Last at con X*.

*Batch size attribute*

By creating a new attribute or selecting an existing attribute for *Store number of items in batch in attribute*: in the Batch block's Properties tab, an attribute can be set to the total number of items in the batch as it is released.

Discrete Event



### Delaying kits

You might not want to begin forming a batch until some or all of the items required for each part of the batch are available. This is most common when you do not want a resource item from the Resource Item block to flow into the Batch block until all the items requiring that resource are available. A good example is a manager who does not want to wait for everyone else to arrive at a meeting.

The Batch block's *Delay Kit* feature restricts specified items from entering the block until all of the other input connectors have the items they need. Delay Kit is enabled through checkboxes in the fourth column of the table in the Batch dialog. Each item input for which Delay Kit is checked will have its items wait outside until all required items for the unselected inputs are in the block.

 Delay Kit is only available when the Batch block's behavior is set to "Batch items into a single item."

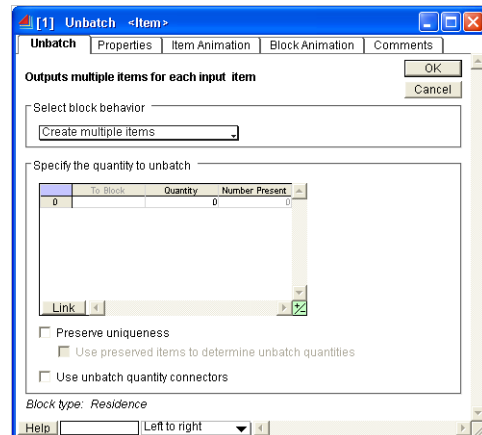
### When the kitting starts

If the number of items required at a Delay Kit input is one, and all the other required items have been pulled into the block from their input connectors, the batched item will be created as soon as the Delay Kit item is available. If the number of items required at a Delay Kit input is greater than one, the block will start a kit as soon as all the other required items are available and there is at least one of the Delay Kit items available. This causes items with Delay Kit to be pulled into the block as they become available; the batched item will not be created until all the items with Delay Kit are available.

### Unbatching


Unbatching can be used to separate items that were previously batched or to duplicate items that have not been batched. Some examples of when you would use unbatching are:

- Returning an item resource to the Resource Item block
- Ungrouping items that had been temporarily grouped so that they could be processed at the same time
- Creating items based on a single "seed" item
- Creating a logical item to trigger some action in the model while allowing an item that represents the physical part to continue processing



Unbatch dialog

If items were previously batched with *Preserve uniqueness* enabled in the Batch block's Options tab, the Unbatch block can be used to restore the items that formed the batch with their original properties. This is accomplished if *Preserve uniqueness* is also checked in the Unbatch block's dialog. For information about this feature, see "Preserving the items used to create a batch" on page 204.

 Be careful when using any property-setting blocks in the path between a Batch and an Unbatch block. Those property modifications could be lost, depending on selections in the Unbatch block's Properties tab, as discussed on page 204.

The top section of the Unbatch tab in the Unbatch block has two options that determine how the block behaves when costing is involved. Each option causes the block to output a number of items, specified in the dialog, for each item that is input. The options are:

- *Create multiple items.* This is the default choice. If the model has costing, costing attribute values are distributed to the output items as specified by settings in the table in the Unbatch block's Properties tab. The section "Simple unbatching" on page 202 shows how to use the block to separate batched items.
- *Release cost resources.* Resources can have a cost associated with them. If an item is batched with a resource, cost information is maintained with it. If the model has costing and the *Release cost resources* option is selected, the block releases the resources out of the same connector that they were originally batched and updates costing information for the items accordingly. For more information, see "Combining resources with cost accumulators" on page 234.

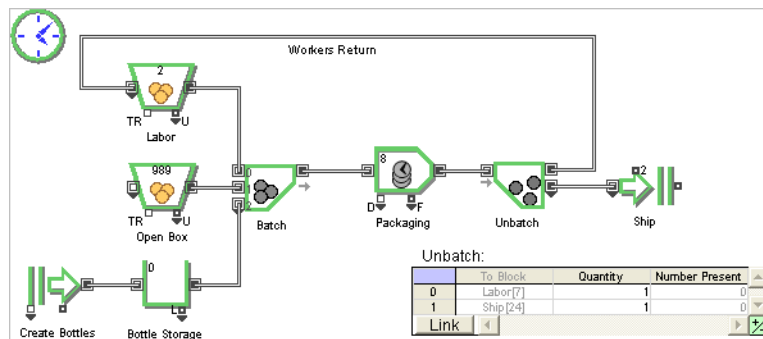
☞ The Unbatch block will not behave differently when either of these two options are selected unless the model calculates costs.

The Unbatch tab also has a table for specifying the number of items that will be sent through each output connector. The first column displays which blocks the Batch's output connectors are connected to. The Quantity column is for entering the number of items that will be output for each input item the Batch block gets, while the next column displays the number of items present.

⚠ When preserving items that have been batched or when releasing cost resources, it is important to physically match where items enter a Batch block with where they leave an Unbatch block. For instance, items that arrived to the Batch block on the top input should be released from the top output of an Unbatch block.

### Simple unbatching

The Batching and Unbatching model is an extension of the example "Simple batching" on page 196, with the addition of laborers and unbatching. Since there is no costing in this model, the Unbatch dialog is set to *Create multiple items* (the default option).



Batching and Unbatching model

In this model, the packaging process must be performed by a laborer. There are 10 laborers available and each is represented by an item in the Resource Item block. Each worker item is temporarily batched with the bottles and cartons to represent the requirements of the packaging process.

Discrete Event

The dialog of the Batch block (shown at right) indicates that one laborer is needed for each package assembly. Since *Delay Kit* is checked for that row, the labor resource will not be drawn into the Batch block until the other items required for the batch (1 box and 3 bottles) have arrived.

	From Block	Quantity Needed	Quantity in Block	Delay Kit
0	Labor[7]	1	0	<input checked="" type="checkbox"/>
1	Open Box[1]	1	1	<input type="checkbox"/>
2	Bottle Storage[3]	3	2	<input type="checkbox"/>

Binding a worker

The worker is returned to the pool when the task is finished, while the boxed bottles exit the simulation. This is modeled by the Unbatch block, which takes a single item (the output from the Activity) and creates two items, as shown on the right. One item represents the worker who is sent back to the Resource Item block through the top output and the other represents the assembled package that is shipped.

	Property	Action
0	Animation	Batched value
1	Item priority	Batched value

Unbatching 1 worker and 1 package

### Variable batching and unbatching

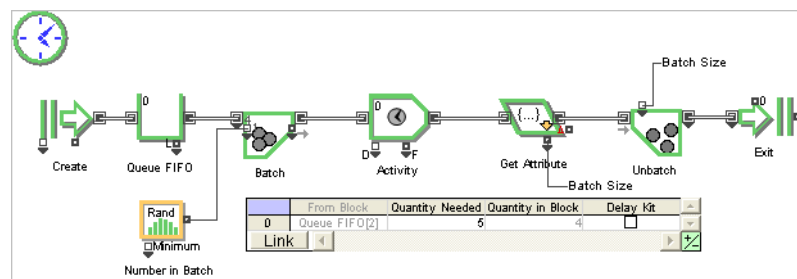
The previous model showed how to batch and unbatch a fixed number of items from each input. This example shows how to batch a variable number of items and unbatch that same number of items.

To keep track of the number of items a batch is composed of, select an attribute in the Batch block's Properties tab to store the number of items in the batch. When the batch is created this attribute's value will be the number of items that arrived to create the batch.

- As discussed later in this chapter, if both the Batch and Unbatch blocks are set to preserve the uniqueness of items, batch size does not need to be saved in an attribute. Instead, just check *Use preserved items to determine unbatch quantities* in the Unbatch block's dialog.

### Batch and Unbatch Variable model

An example of batching a variable number of items and unbatching that same number, is shown in the model below.



Batch and Unbatch Variable model

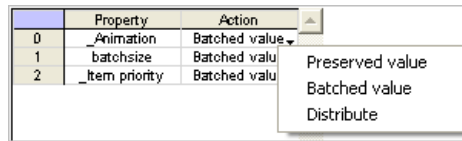
In this model, the Batch block's Options tab indicates that the size of each batch is stored on an attribute named *batchsize*, which is accessed by a Get block. Attaching the Get block's value output to the Unbatch block's *UnbatchQuantityIn* connector sends information about the size of the batch to the Unbatch block, causing each batch to separate into its original number of items.

- Since the Batch block is set to *Set batch size by first item at each connector*, the size of the batch is locked when the first item for that batch arrives in the Batch block.

### Properties when items are unbatched

As happens when items are batched, when items are unbatched you can specify what the block should do with their properties. This is accomplished by selecting an Action for item properties in the Unbatch block's Properties tab.

For example, assume a Batch block combines six bottles that have a Weight attribute, and that the bottles are then filled with liquid such that the batch weighs 12 pounds. When these bottles are subsequently unbatched, you can select one of the following actions for the Weight attribute:



Unbatch block property actions

- Preserved value. This option causes the bottles to retrieve their preserved value, if *preserve uniqueness* is turned on. (See the following topic for more information about preserving uniqueness.) In this case the weight of each bottle will be what it was before batching and the 12 pounds of weight acquired after batching is discarded.
- Batched value. With this choice, the 12 pounds of weight will be copied to each of the resulting bottles.
- Distribute. The 12 pounds of weight will be divided among each item equally, 2 pounds to each bottle.

Discrete Event

### Preserving the items used to create a batch

Before items are batched they may have properties such as attributes attached to them. By default, a reduced set of those properties is transferred to the new batched item, according to actions selected in the Properties tab of the Batch block. (For more information, see “Property options” on page 200.)

If it is important in your models to retain the attributes and priorities of the items that were batched, select *Preserve uniqueness* in the Batch block *and* in the Unbatch block. This marks the items in the batch as unique so that an Unbatch block can restore all of the items' properties.

For those properties that you want to retain, select the Preserved Value action in the Properties tab of the Unbatch block, as discussed in “Properties when items are unbatched” on page 204.

**⚠** Do not select *Preserve uniqueness* unless the items have unique information attached to them, the items are not just temporarily batched, and you need to restore the items and their properties at a later point. Preserving uniqueness requires a lot more memory and slows processing time.

### Both blocks choose to preserve uniqueness

The consequences of selecting, or not selecting, Preserve uniqueness in both the Batch and Unbatch blocks are:

- If the Preserve uniqueness option *is* checked, the batch is temporary. The original members of the batch are stored when the batch is created and they can be restored when the item representing the batch is unbatched. Examples include batching a group of parts together to process them as a group and later unbatching them to continue processing individually, or batching an item with a resource and later returning the resource item with an Unbatch block.
- If the Preserve uniqueness option is *not* checked, the items used to make up the batch are destroyed when the batch is created. The batch, represented by a new item, is permanent and

the original items cannot be restored. Examples include batching items together into a box for final shipping, or batching an order with the required inventory.

- ☞ If a Batch block is set to preserve uniqueness, the unique identity of the items will only be restored upon unbatching. While batched, the attributes of the unique items will be combined into one set of attributes, as specified by settings in the Batch block's Properties dialog.

### Either block chooses to preserve uniqueness

To restore the items with their properties intact, the option to "Preserve uniqueness" must be selected in both the Batch and Unbatch blocks. There are special outcomes if "Preserve uniqueness" is selected in *either* the Batch *or* the Unbatch block, but not in both blocks:

- If Preserve uniqueness is selected in the Batch block but not in the Unbatch block, the preserved items travel with the first item that leaves the Unbatch block's top output connector. All the other items leaving the Unbatch block will be identical and not contain any information about the preserved items.
- If Preserve uniqueness is not selected in the Batch block but is selected in the Unbatch block, it is the same as if preserve uniqueness is not checked at all. Because the original information about the batched items was lost when they were batched, the Unbatch block will unbatch identical copies of the items that arrive to it.

Neither of these conditions is typically desirable.

### Additional models

The folder located at \Examples\Discrete Event\Batching contains additional batching models not discussed in this chapter:

- Equation(I) Controls Batch
- Queue Eqn Controls Batch

Both models show advanced concepts for dynamically changing the size of batches.



# **Discrete Event Modeling**

## **Resources and Shifts**

Modeling resources and controlling them with shifts

Items will sometimes require resources before they can proceed to the next step in a process. For example, a car might need an attendant to drive it through the car wash, a vendor's invoice could require a receiving report before payment is made, or parts might need to be assembled by a worker. Resources provide a service to the items in a model; their availability or lack thereof can cause constraints on the flow of items.

One of the main reasons to model a process is to analyze resource availability and utilization and to determine the impact of resource constraints on the system's capacity. This tells how efficiently current resources are being used and what happens if they will not be available or when there is a wait for them to become available. Often the objective is to try to improve resource utilization without causing overly long waiting lines or to determine how to reduce waiting lines without adding more resources.

This chapter discusses:

- Modeling resources with the Resource Pool block
- Modeling resources using the Resource Item block
- Other methods for modeling resources
- Closed and open systems
- Ways in which resources can be scheduled
- Controlling resources and activities with the Shift block

 The models illustrated in this chapter are located in the folder \Examples\Discrete Event\Resources and Shifts.

## Blocks of interest

The following blocks are the main focus of this chapter. Each block's library and category appears in parentheses after its name.

### Resource pool blocks



#### **Resource Pool** (Item > Resources)

Stores a count of resources for the model. The resources are taken by the Queue block (in "resource pool queue" mode) and released by the Resource Pool Release block at some later point in the model.



#### **Queue** (Item > Queues)

When the Queue type is set to "resource pool queue", items wait here for required resource pool units from the Resource Pool block. Once the needed resource units are available, the block checks for downstream capacity before releasing items.



#### **Resource Pool Release** (Item > Resources)

Releases the specified number of resource pool units, making them available for re-use and causing the count in the Resource Pool block to increase.



### Other resource blocks

**Resource Item** (Item > Resources)

Unlike the resource pool method, this block stores resources as items for use in the model. Resource items are usually batched with items that require them; they may or may not be unbatched at some later point in the process.

**Shift** (Item > Resources)

Generates a shift schedule that can be used to change the capacity or stop the activity of other blocks in the model.

### Modeling resources


Resources are the means by which process activities and operations are performed. Different parts of a model can share the same resource, just not at the same time. While a particular resource is being used in one place in a model, it is not available for any other part of the model. Thus the availability or lack of availability of resources causes constraints in a model.

#### How to model resources

As seen in the following sections, there are many ways to model resources when building models. Resources can be modeled explicitly using specialized resource management blocks; this has the advantage of direct access to features like automatic costing and utilization calculations. In some situations, however, it could be simpler or provide more control to model resources just as any other item in the model or by limiting block capacity.

ExtendSim's discrete event architecture supports two explicit ways to model resources:

- *Resource Pool method.* As a *count* of the resources that are available in a pool. By keeping track of the available resource pool units, this method controls the flow of items that require the resources. This is accomplished using the resource pool blocks (Resource Pool, Queue [in *resource pool queue* mode], and Resource Pool Release), as shown in “Resource Pool method” on page 209.
- *Resource Item method.* As one or more *resource items* that are available to another item. This method involves batching resource items from the Resource Item block with the items that require them and, typically, unbatching the resource when it is no longer needed, as described in “Resource Item method” on page 213.

 Resource-type blocks should only be used in a model if their presence is required for the system. In addition to the two explicit methods listed above, ExtendSim provides additional ways to model resources, as discussed in “Other methods for modeling resources” on page 215.

#### Resource Pool method

The resource pool blocks in the Item library (Resource Pool, Queue, and Resource Pool Release) cause restraints to be placed on the flow of items in the model based on the availability or lack of resources. The Resource Pool block maintains a count of the number of resources that are currently available for use. When an item enters a Queue block that is in resource pool queue mode, the Queue will query the resource pools to determine if the required number of specified resources are available. If so, the number of resources currently available will be decremented in the appropriate Resource Pool block, and the item that requires the resource will be released from the

Queue. If the required number of resources are not available, the item will wait in the Queue until resources become available.

In a closed system, the resources are returned to the Resource Pool block by passing the “resourced” item through a Resource Pool Release block. In an open system, such as for a consumed resource, the resource is not returned to the pool but is removed from the system when the item exits. Closed and open systems are discussed on page 216.

☞ Since resources are not returned to the originating block in an open system, statistical calculations such as utilization cannot be accurately determined.

**Advantages and disadvantages of using resource pools**

*Advantages*

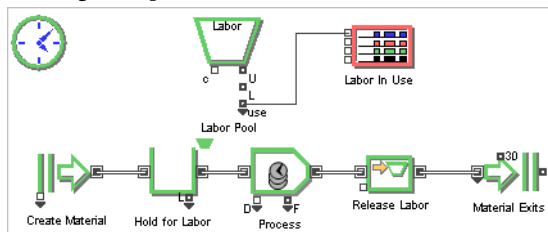
- The Resource Pool block does not require any connections to other blocks in a model. Because of this, using resource pools to model resources (as opposed to using the resource item/batching method that will be described later) is more flexible when the same resource can be used in many different places or when an item can use any one of a group of resources.
- The resource pool method does not require complex routing of resource items because the resources are not actual items but merely constraints on the flow of items through the model.
- When items wait for resource pool units, they can be ranked by priority or FIFO order. The Resource Pool block is able to globally allocate the resource pool unit to the highest ranked item.

*Disadvantages*

- The resource pool method does not allow the use of attributes to track information about the individual resources. To use attributes, you must use resource items; this is shown in “Resource Item method” on page 213.
- It is more difficult to control the complex scheduling of competing resources across a number of different queues using resource pools.

**Simple Resource Pool model**

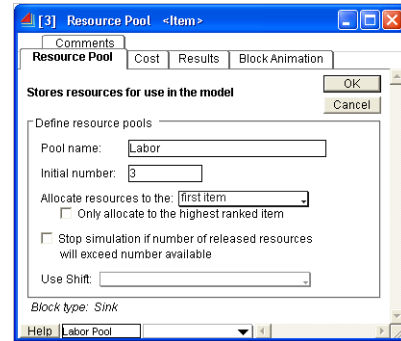
The discrete event tutorial on page 105 showed how to use and release resources from resource pools, as does the following example:



Simple Resource Pool model

Discrete Event

The Simple Resource Pool model represents a flow of material where each piece requires one laborer for processing. In the dialog of the Resource Pool block (labelled Labor Pool), the pool of resources is called *Labor* and the initial number of Labor units is 3, as shown at the right. One piece of material is generated by the Create block about every two minutes. Since this model uses resource pool units, the Queue block's type is set to *resource pool queue*. This causes generated material to wait in the Queue until the required Labor is available.



Resource Pool dialog

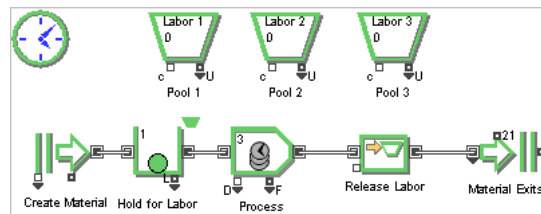
In the Activity block, processing takes 5 minutes and the capacity is infinite, so any number of pieces can be worked on at a time. Within the Queue and Resource Pool Release blocks, the quantity of resources required/ released is 1 and the name of the resource pool (Labor) is listed.

Running the model shows that the amount of processing that can occur is constrained by the number of laborers available. Although the Activity has an infinite capacity, the cloned plotter graph shows that there are not enough workers available for much of the simulation.

So that the focus is on the constraining effect of labor resources, the Activity block is set to infinite capacity. This causes the availability of labor, but not the processing of material, to affect the flow of items in the model.

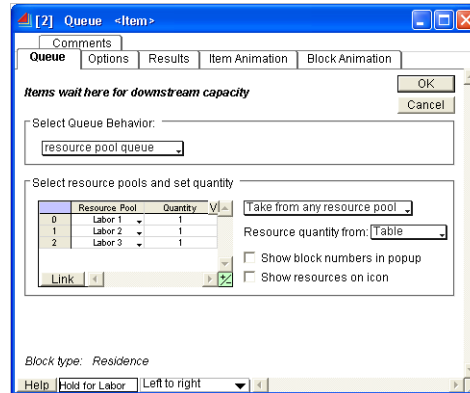
### Resources required from different pools

In the Multiple Pools example, there are three Resource Pool blocks, each with their own labor resource. Items require *either* Labor 1, Labor 2, *or* Labor 3.



Multiple Pools model: One laborer per piece required from any of the three pools

Because it will hold items that require resource pool units, the Queue's type is set to *resource pool queue*. As shown in the Queue's dialog, the three resource pools have been selected from popup menus in the table and the block is instructed to take the resource from any one of those pools. When an item enters the Queue, it will query each of the resource pools in the order that they are listed in the table (from top to bottom). In other words, if no resources are available in Pool 1, the Queue block will try Pool 2, and then Pool 3. If a resource is still not found, the material will be held in the Queue until the resource requirement is met by whichever pool first has an available resource.



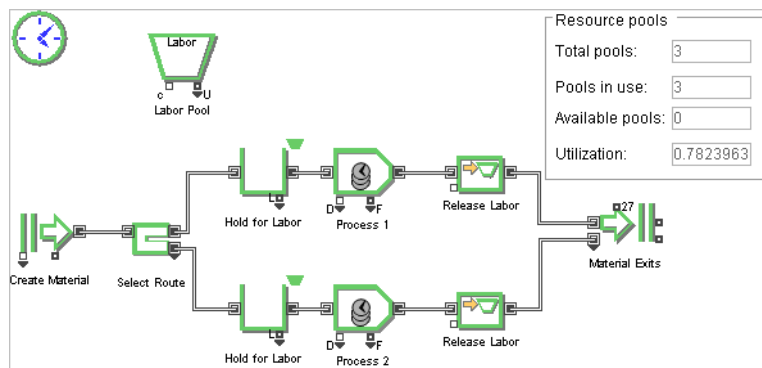
Queue block dialog

Note that the Queue block (Options tab) stores the information about which pool the resource came from, and how many resources were used, in an attribute that attaches to the items processed. In this model, the information is stored in the attribute "Resource Name." This attribute is used by the Resource Pool Release block to inform the appropriate pool when a resource is no longer in use.

The Multiple Pools model is similar to the Simple Resource Pool model in that each piece of material requires 1 laborer. However, in this model the laborer can come from one of three pools rather than from a pool of three laborers. As in the earlier example, modeling this process using the resource item/batching method would require complex logic to correctly route the resources.

### Same resource used in multiple places

The Multiple Uses model has two parallel processes, each requiring laborers. Items waiting in both Queues (set to *resource pool queue*) require a labor resource from the same Resource Pool block, which has 3 laborers initially available. When an item enters one of the queues, a request is sent to the Resource Pool block for a labor resource. The requests are satisfied in the order in which they were received (or in order of the requesting item's ranking as specified in the Resource Pool dialog).



Multiple Uses model

Discrete Event

### Resource Item method

Another method for explicitly modeling resources is by using the Resource Item block. With this method, each resource is represented by an item whose purpose is to provide a service for other items in the model. The number of resources that are initially available are entered in the dialog of a Resource Item block. For an item in the model to use this type of resource, the item must be batched with the resource (see also “Batching and Unbatching” on page 193.) While the resource is batched with an item, it cannot be used elsewhere in the model. If a resource is not available, the batch will not be able to be completed, and the item will have to wait until a resource becomes available. As with the resource pool method, the movement of items in the model is restrained based on the availability or lack of resources.

If you are modeling a closed system, the resource must be unbatched from the item when it is no longer being used. Once it is unbatched, it should be routed back to the resource-type block so that it may be used again. In an open system, for example where the resource is a consumable product, the resource can stay batched with the item. Closed and open systems are discussed on page 216.

### *Advantages and disadvantages of using resource items*

A resource item can have properties such as attributes, a priority, and a quantity like any other item. For this reason, this method of modeling resources is preferred if you need to track information about resources.

☞ See “Items, Properties, and Values” on page 109 for a complete explanation of attributes and other item properties.

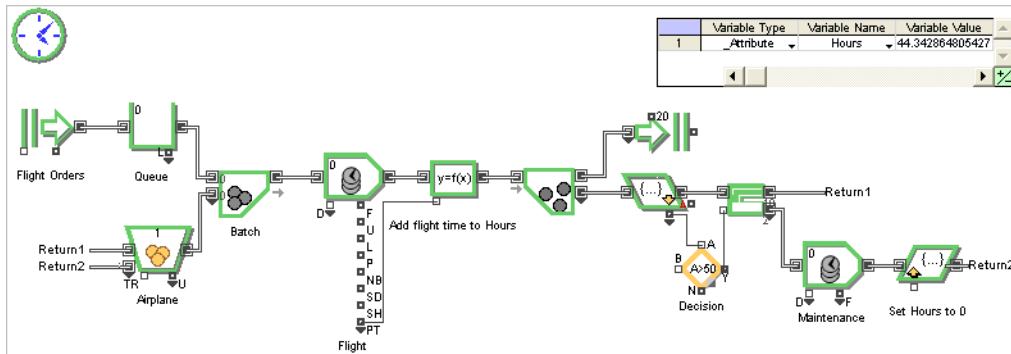
Some limitations of using resource items are:

- The Resource Item block must be connected in the model and the connection must be such that the resources it outputs can be batched with the items that require them.
- The resource item cannot “see” the items waiting for it. You must use routing blocks to direct the resource item to the correct Batch or Unbatch block.

### *Air Freight model*

An example of using the attributes of resource items to track information is shown in the following model of an air freight company. An airplane receives orders for flights, but regulations require that airplanes must undergo maintenance after every 50 hours of flight time. Thus the model needs to track the airplane’s accumulated flight time (hours). Once an airplane accumulates 50 hours of

flight time, it is sent for maintenance and the accumulated hours are reset to 0. The model looks like:



Air Freight model

Discrete Event

The Create block generates orders which are batched with an airplane from a Resource Item block. While the airplane is batched with a flight order, it is not available for other flight orders; the order will wait in the Queue (set to *type: sorted queue*, and *Sort by: first in, first out*) until the plane becomes available.

The Resource Item block attaches an *Hours* attribute to the airplane. Settings in the Batch block's Properties tab, shown at right, cause the airplane's Hours attribute (the first attribute from the item arriving at the second connector) to be attached to the batched item.

Property	Action
0 Hours	First at con 2
1 Animation	First at con 2
2 Item Priority	First at con 1

Batch block properties

For more information about item attributes, see "Attributes" on page 115.

An Activity block (labelled Flight) uses a random distribution to determine how long each flight will take, then outputs the flight time to its Process Time (PT) connector. The Equation block gets the airplane's flight time and adds it to the plane's Hours attribute.

After the flight, the airplane is unbatched from the order; the airplane returns to the Resource Item block for reuse and the order exits the simulation. The Get block reads the value of the airplane's Hours attribute and the Decision block determines if the accumulated flight time is greater than 50 hours. If it is, the airplane will be routed to the maintenance group for processing. After maintenance the Set block re-initializes the Hours attribute to zero. If accumulated time is not greater than 50, the airplane is returned to the Resource Item block where it will wait for another flight order.

When the simulation is run, a clone of the output from the Equation block's dialog shows the value of the airplane's Hours attribute. With animation on, it is easy to see that once the airplane has an Hours value greater than 50, it is routed to the maintenance group.

For this model it is essential to be able to track information about the airplane's flight time. Therefore the ability to assign attributes to the airplane resource is critical.

**Stripping attributes from resource items**

As described in "Properties when items are unbatched" on page 204, an item returning to a Resource Item block after batching may have many attributes that are irrelevant to the returning item. The Resource Item block provides the option of stripping attributes or keeping them with

resources that are recycled; the default is to strip them. When tracking resource information using attributes (as in the above model), you will not want to strip the attributes, so the block is unchecked. However, in cases where you are not concerned with attribute values after the item has been recycled, you may want to strip the attributes so that the item will be “clean” when it comes out of the Resource Item block again.

- ☞ The Queue Matching block (Item library) holds different types of items until the requirements for each type have been met. This can be useful when modeling the release of resources and items into a Batch block. For more information, see “Matching items using the Queue Matching block” on page 138.

### Other methods for modeling resources

The Resource Pool and Resource Item methods described earlier in this chapter use specialized blocks to explicitly represent resources. The ExtendSim architecture provides many additional methods for modeling resources. For example, it might be simpler or provide more control to imply a lack of resources by limiting capacity in some blocks or to model resources just like any other item in the model.

#### *Implicit resources*

A resource can be implied in a model by restricting or scheduling the capacity of residence type blocks like the Activity and Queue. These blocks are useful for implicitly modeling resources.

For instance, the “Simple Resource Pool model” on page 210 illustrates how to model resources using the resource pool blocks. A simpler method would be to use the Activity block to represent a limited resource, without using explicit resource blocks. In this case, you would remove the Resource Pool and Resource Pool Release blocks from the model, set the Queue as a FIFO sorted queue, and set the maximum items in the Activity block to three. The Activity’s capacity limitation would have the same constraining effect as the Resource Pool block in the original Simple Resource Pool model.

Another advantage of modeling resources implicitly is that the Activity block can be shutdown and brought back online using the Shutdown block, as shown in “Shutting down” on page 179. This is common when modeling random failure.

#### *Conceptual resources*

The concept of what is a resource is not limited to the explicit (resource pool and resource item) or implicit (capacity-constrained) methods of representing resources. Theoretically, a resource is anything where its availability can restrict items flowing from point A to point B. Some examples are:

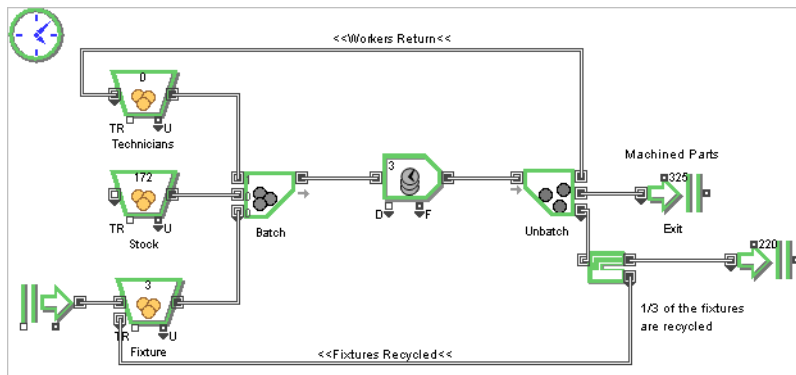
- Any item can conceptually represent a resource. For example, batching a “bus” item with “people” items, where the bus is required before the batched bus/people item can be released from a Queue (see “Delaying kits” on page 201). Note that the bus is created as any other item, not as a resource item from the Resource Item block.
- Using the ExtendSim database or global arrays to track resource availability, limiting the flow of items in a model. For example, item availability would be regulated by how data in the database changes during the course of a run.
- Using block combinations to control item movement as model status changes over time, such as a Queue followed by a Gate that is connected to a Read block.

Your cleverness and knowledge of ExtendSim can probably lead to even more ideas.

## Closed and open systems

As discussed on page 95, blocks that provide a finite number of resources can be part of closed or open systems. In a closed system, resources are recycled back to the originating block. In an open system, resources are not recycled but instead exit the system. Systems can also be partially closed, for example when some of the resources are recycled back and others are not.

The following model uses three Resource Item blocks to illustrate a closed system (Technicians), an open system (Stock), and a partially closed system (Fixtures).



Closed and Open Systems model

The model assumes that about one third of the fixtures are consumed in the process; they are restocked at periodic intervals by the Create block.

## Scheduling resources

To accurately characterize the impact of resources in a simulation model it is common to model resource scheduling logic, both in terms of where and when a resource should be assigned. For example, if one resource item is required in multiple places, such as an item that could be routed to two or more Batch blocks, then scheduling logic needs to be added to the model.

There are several ways resources can be scheduled. Some methods apply to using either resource pools or resource items and some apply only to scheduling resource items.

### Scheduling resource pools and resource items

As discussed in the following sections, there are two systems available to schedule resource pools and resource items:

- Use the *TR* (total resources) connector on individual resource blocks, as described in the following section.
- Use Shift blocks to control aspects of one or more resource blocks. This is discussed in “Resources model” on page 221.

### Using the *TR* (Total resources) connectors

Blocks that provide resources (whether as items or as resource pool constraints on items) have value connectors labeled *TR* (total resources). You use the *TR* input connector to change the total number of resources available. This change can be scheduled, such as when workers take breaks, or unscheduled, such as an equipment failure.

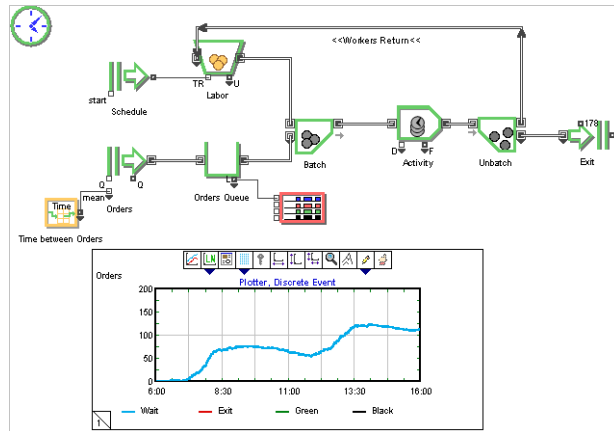


The value at the TR connector determines how many resources the block has and can result in an increase or a decrease in resource availability. For example, if the initial number in a Resource Item block is 10, and the block gets a value of 3 at its TR input connector, the block will eliminate 7 resources from its availability list. If the block doesn't have enough resources to dispose at the time of the change, it will dispose of them as they return.

*Scheduling Resources model*

It is common to schedule the availability of resources based on some factor in the model, typically time. For example, in the model discussed in "Scheduling activities" on page 173, you could have scheduled workers in the diner depending on the time of day using the Resource Item and Create blocks.

The new model is shown at right. It assumes there are three workers initially available when the coffee shop opens at 6am. Two additional workers arrive after 5 hours and remain just for the lunch period, from 11am to 2pm (1400 hours). The coffee shop closes after 10 hours.



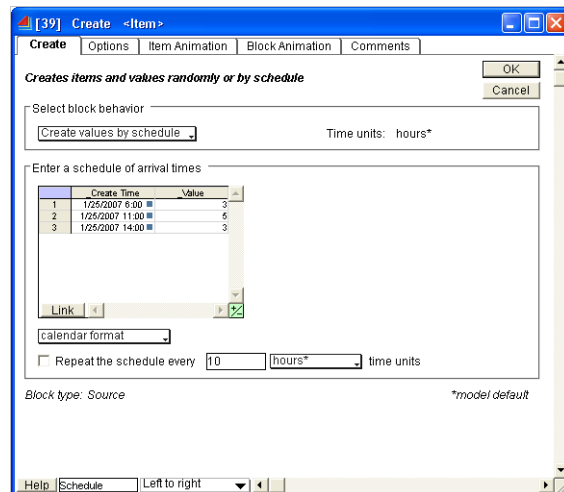
Scheduling Resources model

Discrete Event

The schedule for workers is entered in the dialog of the Create block, as shown at right. The block is set to *Create values by schedule*, and its value output is attached to the TR (Total resources) input of a Resource Item block.

This model is set to use Calendar dates. The Simulation Setup dialog and block options (such as selecting *calendar format* when a Create block is set to schedule its outputs) facilitate the display of times and dates in Calendar format. For more information, see "Calendar dates" on page 528.

In this model the workers are part of a partially closed system. Some are recycled back to the Resource Item block through its item input connector, while other workers are added to or removed from the block through its TR connector.



Arrival times for workers

**Scheduling resource items**

There are at least four additional ways to schedule resource items in ExtendSim:

- 1) A Resource Item block followed by a Gate can control when resource items are released.

- 2) A Resource Item followed by a Select Item Out block controls where resource items are routed.
- 3) A combination of the Gate and Select Item Out blocks can be used to control both where and when items are scheduled.
- 4) The Queue Equation block is useful for controlling both where and when items are scheduled for use when the scheduling logic is more complex. With this block, ModL logic statements can intelligently control the scheduling of items based on their properties, information in the ExtendSim database, or even the status of other sections of the model. For more information, see “Sorting items using the Queue Equation block” on page 133.

### The Shift block

The Shift block is used to schedule both the magnitude and availability of capacity in other blocks in a model. This is useful for simulating situations where a system’s resources follow a pattern of coming on and off line over time. For example, the Shift block could be used to model workers in a factory following a repeated daily pattern of reporting to work in the morning, taking a break for lunch and going home at some point in the evening.

Each Shift block represents a named shift and its schedule that can be referenced by other Item library blocks. The Shift controls the capacity of the blocks that reference it, based on the schedule that is defined in its dialog table. If a shift schedule is changed, all blocks using that named shift will receive the same modified shift pattern. In addition, shifts may be repeated at regular intervals if the *Repeat schedule every* checkbox is selected. This is useful for modeling repeated shift patterns, e.g., an eight-hour workday each day of the week or breaks that occur every four hours.

It would be quite easy to define a complex shift schedule that includes all breaks, holidays, week-ends, and so forth. However, before adding such complexity to a model, carefully consider whether such detail adds to the validity of the model. If, for example, nothing at all happens during the weekend, a better solution would be to simply assume one week is 5 days long (or specify that in the Simulation Setup dialog) rather than adding a Shift block to model the weekends. Shifts should only be used when they will make a significant difference in the results of the simulation.

### Shift types and what they control

A Shift can schedule capacity in a number of Item and Rate library blocks. For example, it can turn an Activity on or off, or specify a maximum number of items the block can process at a time. The schedule depends on whether the selected Shift type is On/Off or Number.

- An *On/Off* type of shift acts like a binary switch that turns associated blocks on or off at specific points in time. For example, an On/Off shift might be used to shut down an Activity block during lunch and evening hours, to open or close a Gate block, or to turn a Valve (Rate library) on and off according to a schedule.
- The *Number* shift type explicitly defines the size of a block’s capacity over time. For instance, it might set the size of a Resource Pool to 3 items for the morning shift, 0 over lunch, 5 for the afternoon shift, and 0 overnight.

The following table shows which Item library blocks can be controlled by a Shift block, which types of shifts those blocks support, and what aspect of a block, if any, the Number type of shift controls:

Item Blocks That Can Use Shifts	Shift Type: On/Off or Number	Number Type Setting on Shift Controls This Aspect of Item Library Block
Activity	Both	Maximum number of items in the Activity at a time.
Convey Items	On/Off	N/A
Create	On/Off	N/A
Gate	Both	In “area gating” mode, the number of items allowed in the gated area at a time.
Resource Item	Both	Total number of resources available, per the TR (Total resources) connector.
Resource Pool	Both	The total count of resources available, per the TR (Total resources) connector.
Transport	Both	Block capacity.
Workstation	Both	Maximum number of items in process.

In the Rate library, the Convey Flow, Interchange, Tanker, and Valve blocks can use a Shift set to On/Off type. See the Rate module for more information about using the Rate library.

- ☞ If a Shift block is used in a model, statistics in the Queue blocks will probably not accurately reflect utilization, etc.

### Status connectors

The Shift block’s value input connector (*StatusIn*) can be used to override the shift schedule. If the input connector is less than 0.5, the Shift is considered off shift; this will override any value found in the Shift block’s dialog table. If the input connector is greater than 0.5, the shift schedule from the dialog table is used.

The Shift’s value output connector (*StatusOut*) reports the current shift status (ON = 1 or OFF = 0 for On/Off type Shifts, or the number for Number type Shifts).

### Shift models

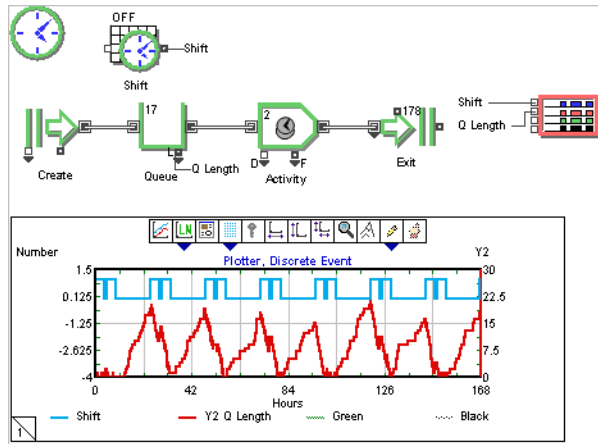
The following models show how to use the Shift block in typical modeling situations. They also illustrate how the two types of Shift, On/Off and Number, are used in simulations.

- ☞ The models are located in the \Examples\Discrete Event\Resources and Shifts folder.

**On/Off type example**

In the “Shift On and Off” model at right, an On/Off type of shift turns the Activity block on for 4 hours in the morning, off for 1 hour at lunch, back on for 4 hours in the afternoon, and off again in the evening until the next morning. The schedule, named “Day Shift”, is set in the Shift block; the dialog of the Activity block is set to *Use Shift: Day Shift*, which causes the Activity to go on or off according to that schedule.

The model contains a cloned graph from the plotter. On the graph the upper (blue) line reflects this 24-hour shift cycle for seven days, a total of 168 hours. Observe how the lower (red) line, which is plotted against the Y2 axis, reflects the queue length’s dynamics during the various on and off shift periods.



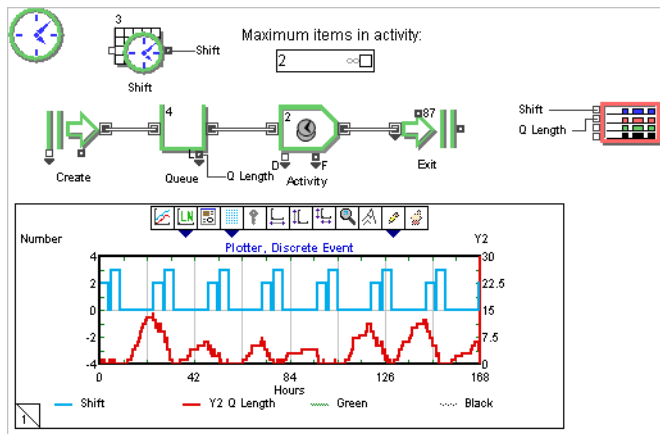
Shift On and Off model

**Number type examples**

*Shift Capacity Change model*

In the model “Day Shift Capacity Change”, shown at right, a Number type of shift is used to control the maximum number of items in the Activity block. As determined by the Shift block, the Activity is limited to 2 items during the morning shift, 0 during lunch, 3 items during the afternoon shift, and 0 overnight. The schedule, named “Day Shift”, is set in the Shift block; the dialog of the Activity block is set to *Use Shift: Day Shift*. Since this model uses a number type shift, the Shift block’s table controls the maximum number that can be allowed in the Activity at one time.

When the model is run, the cloned Activity dialog item “Maximum items in activity” reflects the Shift table’s schedule of allowed items. Shift behavior over the course of seven days (168 hours) is reflected in the plotted upper line of the graph. The lower graph line reflects the queue length’s growth when activities are limited.



Shift Capacity Change model

Discrete Event

*Resources model*

The next example also uses a Number type of shift. It illustrates a Shift block controlling Resource Pool availability so that workers start their shift, work 4 hours, take a lunch break, work 4 more hours, and then leave for the day.

When workers are not available, the backlog starts building up in the Queue.

**Complex patterns**

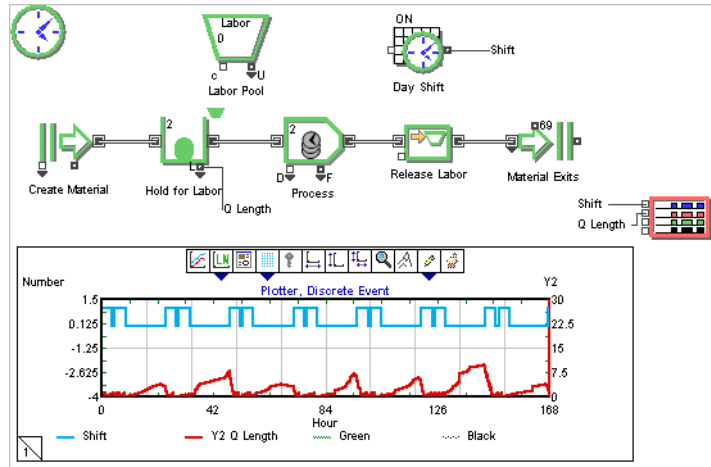
Shift blocks may be configured serially, controlled by other blocks, or used in other patterns to create more complex shift patterns.

*Weekly and Daily Shifts model*

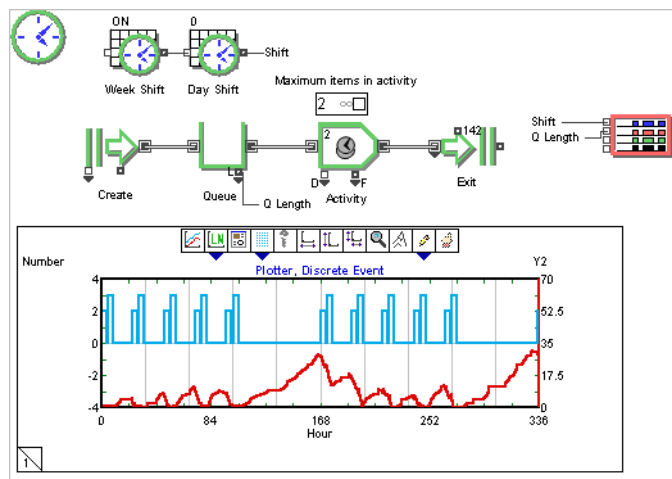
In the “Weekly and Daily Shifts” model, shown at right, a 40-hour work week with two days off during weekends is modeled by feeding a Week Shift (set to On/Off) into a Day Shift (set to Number).

The Week Shift is On for the five weekdays and Off during weekends. Consequently, during weekdays, the Week Shift block sends a value of 1 to the StatusIn connector of the Day Shift block, allowing it to observe its own daily schedule. However, during weekends, the Week Shift block overrides the Day Shift schedule by sending it a value of 0. The plotted line in the cloned graph shows two work weeks (336 hours); the first and second weeks are separated by a weekend.

The cloned dialog item shows the same daily schedule as in the preceding model, with activity halted during lunch and at night.



Resources model

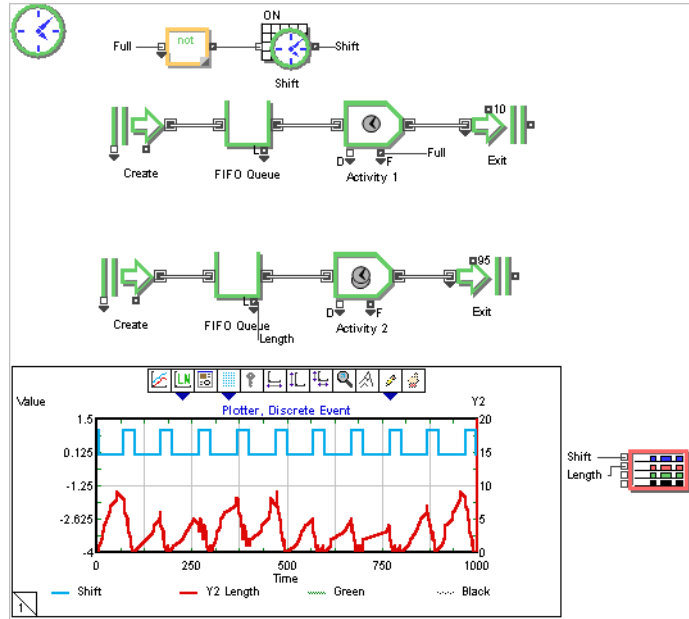


Weekly and Daily Shifts model

Discrete Event

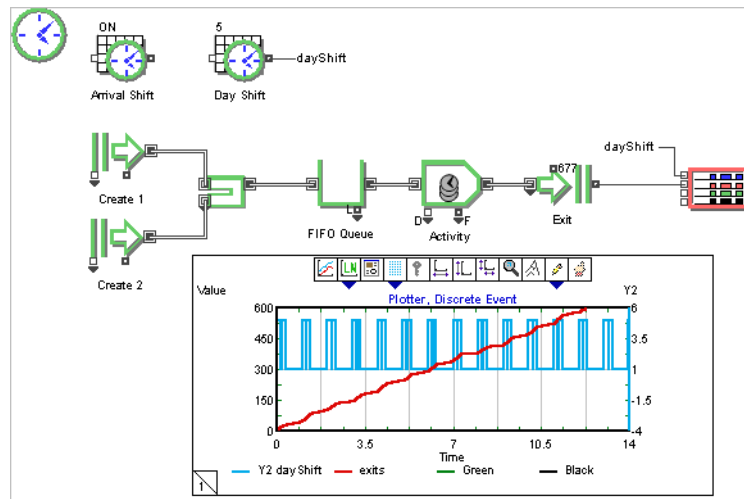
*Controlling Shifts model*

Another example of a more complex shift structure is the Controlling Shifts model where a Math block (Value library) monitors the processing at the Activity 1 block. While Activity 1 is processing, the shift is turned off, stopping any processing in Activity 2. An example of this would be if a worker is required to monitor two processes, but can only monitor one process at a time and must stop the second process while the first process is active.



Controlling Shifts model

The Arrivals and Activity model illustrates how Shift blocks can shut down portions of a model without connections. This example shows the day shift dynamically setting the maximum capacity of an Activity to either 1 or 5 items and the Arrival shift turning the top-most Create block on and off at specified times.



Arrivals and Activity model

Discrete Event

# Discrete Event Modeling

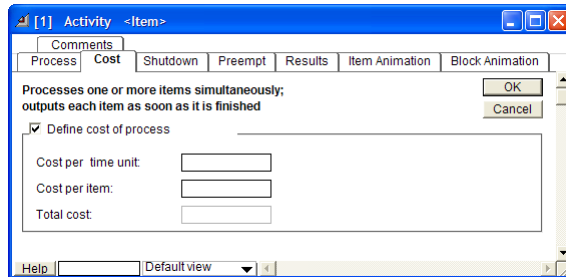
## Activity-Based Costing

Identifying and tracking fixed and variable costs  
to determine operating costs

*“The cost of a thing is often more  
than the sum of the cost of its parts.”  
— The Rev. P.N. Wallis*

Activity-based costing (ABC) is a method of identifying and tracking the operating costs directly associated with processing items. It is the practice of focusing on some unit of output, such as a purchase order or an assembled automobile, and attempting to determine as precisely as possible its total cost based on the fixed and variable costs of the inputs. ABC helps identify, quantify, and analyze the various cost drivers (such as labor, materials, administrative overhead, rework, etc.) and determine which ones are candidates for reduction.

Once a model has been built, the discrete outputs of the system, as well as the processes and resources that are involved in creating those outputs, have already been identified. To add ABC to models, you enter costing information into dialogs of blocks in the model. Blocks that generate items or provide resources, and blocks that process items, have fields and Cost tabs for specifying costing data. Enter variable cost rates per time unit and fixed costs per item or use. After the cost information has been defined, costs will automatically be tracked as the items in the model change state.



Cost tab of Activity block

This chapter covers:

- Identifying cost accumulators and resources
- Defining fixed and variable costs
- How ExtendSim tracks costs

The models illustrated in this chapter are located in the folder \Examples\Discrete Event\ABC.

### Blocks of interest

The following blocks are the main focus of this chapter. Each block's library and category appears in parentheses after its name.



**Cost by Item** (Item > Information)

Calculates the cost of every item in the model, as well as the average and total cost of the process.



**Cost Stats** (Item > Information)

Records the input costs and total cost generated in each costing-based block. Determines total model cost based on a specified confidence interval.

In addition to the two Cost blocks, many of the Item library blocks have cost fields or a Cost tab for entering and reporting cost information, or have cost-handling capabilities, as shown in the table below:

Blocks with Cost tabs or fields	Blocks with cost-handling capability
Activity	Batch
Convey Item	Get



Blocks with Cost tabs or fields	Blocks with cost-handling capability
Create	Equation(I)
Queue	Set
Resource Item	Unbatch
Resource Pool	
Transport	
Workstation	

The Interchange block (Rate library) can also define costing information for items in discrete rate models.

### Modeling with activity-based costing

To include activity-based costing in models, you need to know how to define cost rates, how to properly combine cost resources with items, and how to gather and work with cost information. Although an understanding of ABC is important, most of the work will be done by the ExtendSim architecture.

#### Item types

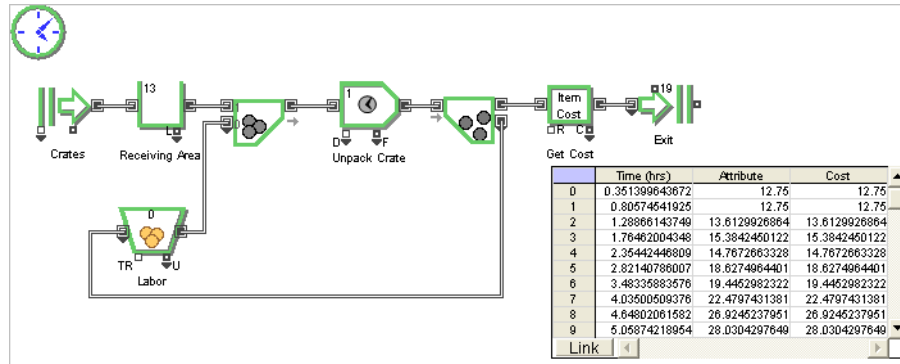
For purposes of costing, every item in a model can be categorized as either a *cost accumulator* or a *resource*. Understanding the difference between cost accumulators and resources is important, because ExtendSim treats them differently, as you will see in “Combining multiple cost accumulators” on page 236.

- ☞ As is true of items that are resources, non-item resources from the Resource Pool block do not accumulate their own costs.

#### Cost accumulators

You perform ABC to determine the costs associated with storing or processing an item. The item being stored or processed is called the *cost accumulator* and will accumulate costs as it waits, gets processed, or uses resources. Cost accumulating items can be introduced into a model using the Create and Resource Item blocks, as you will see in “Costs for cost accumulators” on page 227. The following example shows how costs are assigned to cost accumulating items.

Receive Inventory model



Receive Inventory model

Assume you want to determine the cost associated with receiving crated inventory at a warehouse. There is a one-time docking fee of \$3.00 for every shipment that is received, and it costs \$0.15 an hour any time the crate waits for processing (such as in the receiving area.)

As each shipment arrives, a labor resource takes an average of 30 minutes to unpack the crate and stock the contents on the appropriate shelves. In this case, the crate is being processed and is therefore the cost accumulator. The half-hour processing time and the hourly wage of the laborer is used to automatically calculate the cost of unpacking and shelving. That cost is then added to the accumulated cost being tracked with the crate. As the crate progresses through the steps of being received and unpacked, ExtendSim will add the cost incurred at each step to the accumulated cost.

**Resources**

As discussed in “Modeling resources” on page 209, resources can be modeled using resource pool blocks or by batching resources from a Resource Item block with other items. In the resource pool method, resource units act as constraints on the flow of items throughout the model. In the batching method, resource items are required to be batched with other items before the items can proceed to the next process.

Whether resource pool units or resource items, resources provide a service for the items in a model; they do not accumulate their own costs. Whenever a cost accumulator uses a resource, the resource’s cost rates are used to calculate costs which are then added to the total cost of the cost accumulator. (Cost rates are discussed on page 228.) For instance, in the Receive Inventory model described above, the laborer is a resource item that is batched with the crate. Since the laborer is a resource, it will not accumulate its own cost. Rather, the cost rate of the laborer (the hourly wage), and the time it takes the laborer to unload the crate, is used to automatically calculate the cost of unpacking the crate.

The default is that the Resource Item block outputs resource items. However, you can select in the block’s Cost tab to output items as cost accumulators. This is discussed at “Resource Item block” on page 228.

**Defining costs and cost rates**

To include ABC in a model, simply enter information in the costing section of the dialogs of cost-aware blocks. There are two types of cost information:

- A direct or fixed cost, which is entered as the *Cost per item* or *Cost per use* in block dialogs.

Discrete Event

- The variable cost rate, entered in block dialogs as the *waiting cost/time unit* or *processing cost/time unit*.

You do not need to define all cost information in order to perform ABC. However, if even one cost field is defined as a positive, non-zero number, ExtendSim will automatically track costs when the simulation is run.

Cost accumulating items have their own fixed costs and variable cost rates. As they use resources, wait for processing, and are processed, they acquire additional costs from resources, queues, and activities.

The following information describes how and where to define costs.

### Costs for cost accumulators

You specify costing information for a cost accumulator in the cost section or tab of the block that originates the item. Each cost accumulator can have a fixed cost per item, such as its direct materials cost, and a variable waiting or processing cost rate, which causes it to accumulate costs as it is stored or waits for processing.

Cost accumulators are usually generated by the Create block. They can also be provided by a Resource Item block, depending on a setting in its Cost tab.

#### Create block

Costing information is entered differently in the Create block depending on whether the block is set to *Create items randomly* or to *Create items by schedule*.

- In the Receive Inventory model described on page 226 the Create block generates crates randomly. The *Waiting cost/hour* and the *Cost per item* for each crate are defined in the cost section of the block's Options tab, as shown above.

Cost section of Options tab; block set to "Create items randomly"

- When the Create block generates items by schedule, you must explicitly set *\_cost* and *\_rate* system attributes (discussed in "Working with cost data" on page 231) for each cost accumulating item. The value of the *\_cost* attribute should be set to the cost accumulator's fixed cost. The

value of the `_rate` attribute should be set to the cost accumulator's variable cost rate (waiting cost per time unit), as shown below.

	Create Time	Item Quantity	cost	_rate	None	None
1	0	5	5.25	2.5		
2	10	15	3.8	0.95		
3	20	10	4.75	1.27		
4	30	5	2.75	2.75		

Repeat the schedule every   time units

Create tab; block set to "Create items by schedule"

**Discrete Event**

The `_rate` attribute must be defined using the same time unit as the model's default global time unit. In the example, the time unit is hours, so the `_rate` attribute is the hourly rate.

*Resource Item block*

Cost accumulators can also be provided for a model using the Resource Item block.

To do this, you must choose that the block *Provide items that calculate costing as: cost accumulators* in the block's Cost tab, a portion of which is shown at right.

Define Activity Based Costs

Provides items that calculate costing as:

Waiting cost:  / time unit

Cost per use:

Total cost:

Cost tab of Resource Item block, providing cost accumulators

**Costs of resources**

The costs that will be assigned to items that require resources are defined in the Resource Pool and Resource Item blocks. The Cost tab in those blocks has fields for entering the following information:

- The cost per time unit rate, used to calculate and assign a time-based cost to the cost accumulator while it uses the resource.
- The cost per use is a one-time cost assigned to the cost accumulator for the use of that resource, such as a fixed service charge.

In the Receive Inventory model described earlier, cost rates for the laborer are defined in the Resource Item's Cost tab. The block's dialog indicates that items stored in the block are resources, as seen at right.

Cost tab of Resource Item block, providing resources

☞ For the Resource Item block to provide resources, the Cost tab must be set to *Items are: resources*, the default choice. Otherwise the items will be cost accumulators, as discussed earlier.

### Activities

You can also define cost information in activity-type blocks; those costs are accumulated by each item the block processes. The activity-type blocks are the Activity, Convey Item, Transport, and Workstation. Within the Cost tab of these blocks, enter a cost per time unit and a cost per item:

- The processing cost per time unit is used to calculate the time-based processing cost of each item that passes through the block.
- The cost per item is a fixed cost added to every item that passes through the block.

### Combining resources with cost accumulators

As discussed in “Modeling resources” on page 209, there are two ways that items can use resources. One method is to batch a resource item with another item. While batched, the resource item is in use and cannot be used by another item until it is unbatched and returned to the Resource Item block. The second method is to use the resource pool blocks, which act as a constraint on the flow of items throughout the model. This section discusses how to properly use these two methods when performing ABC.

### Batching and unbatching resources with cost accumulators

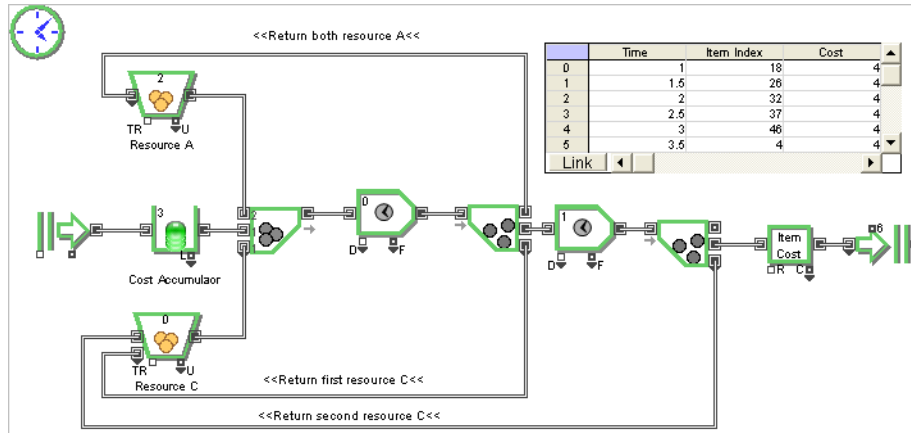
The Resource Item block holds resources for use in the model. When batching a resource item with a cost accumulator, the resource's cost rates are automatically stored with the cost accumulator and used in any subsequent cost calculations.

☞ A maximum of two separate **types** of resource items can be combined with a cost accumulator at a time. For instance, one or more worker resources from a Resource Item block and one or more cart resources from a different Resource Item block.

To unbatch a resource and remove its cost rate information from the cost accumulator, you must select *Release cost resources* as the Unbatch block's behavior. This choice tells the block to modify the information stored with the cost accumulator to indicate that the resource has been released.

☞ If the Unbatch block's behavior is set to *Create multiple items*, the items released by the block will be identical to the item which entered the block. In other words, there would be multiple copies of the cost accumulator, and each copy would still be joined with the resource.

*Multiple Resources model*



Multiple Resources model

Discrete Event

In the example model, the cost accumulator is initially batched with 2 of Resource A and 2 of Resource C. When multiple resources are batched with a cost accumulator, they may be released all at once (as with Resource A), released incrementally (as with Resource C), or remain with the cost accumulator. Whenever a resource is batched or released, the cost array of the cost accumulator is updated to reflect the current number of resources in use. (The cost array is described in “Combining resources with cost accumulators” on page 234.)

There are three things to remember when batching resource items with cost accumulators:

- 1) A resource will be released from the output connector that corresponds to the input connector originally used to batch it to the cost accumulator. For example, if the resource entered the Batch block through the “ItemsIn(2)” connector, it will be released through the Unbatch block’s “ItemsOut(2)” connector.

As in the Multiple Resources model, this could mean that one or more of the Unbatch block’s outputs will be unconnected and you will need to define that there will be zero items output through that connector.

- 2) If an item is simultaneously batched with different types of resources, you must use different connectors for each resource type when creating the batch. In the Multiple Resources model above, Resource A uses connector “ItemsIn” and Resource C uses connector “ItemsIn(2)”.
- 3) When performing ABC, you are limited to two different *types* of resource items batched with a cost accumulator item at one time. This limitation is not true, however, when modeling resources using the resource pool blocks, as you will see below.

**Cost accumulators and the resource pool blocks**

As described in “Resource Pool method” on page 209, as cost accumulating items pass through a Queue block in Resource Pool mode, resources are allocated to the items. When this happens, the cost rate of the resource pool unit is automatically stored with the cost accumulator and used in any subsequent cost calculations.

When a resource pool unit is released using the Resource Pool Release block, the information stored with the cost accumulator is modified to indicate that the resource has been released.

Unlike what happens where resource items are batched with other items, there is no limit to the number of different types of resources a cost accumulator can use when using the resource pool blocks. Furthermore, the two methods of modeling resources (batching resource items and resource pools) may be used in conjunction with each other.

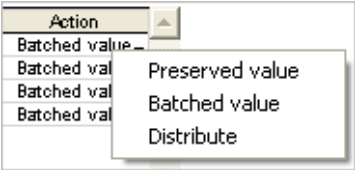
**Combining cost accumulators**

The Batch block can be used to combine cost accumulators arriving from one or more paths. This may be used in conjunction with an Unbatch block, for instance to combine cost accumulators for processing and separate them after processing has been completed. In the Properties tab of both the Batch and Unbatch blocks you can specify what the block should do with the cost values. This is accomplished by selecting an Action for the `_cost` and `_rate` attributes.

**Costing attributes when items are unbatched**

For example, assume a Batch block combines three cost accumulators together and that while batched, these items accumulate an additional \$9.00 due to processing. When these cost accumulators are unbatched, you can select one of the following actions for the `_cost` and `_rate` attributes in the Properties tab of the Unbatch block:

- *Preserved value.* This option causes the cost accumulators to retrieve their preserved value, if “preserve uniqueness” is turned on. In this case, the \$9.00 is discarded.
- *Batched value.* With this choice, the \$9.00 will be copied to each of the resulting cost accumulators.
- *Distribute.* The value will be divided among each item equally. In this case, \$3.00 to each.

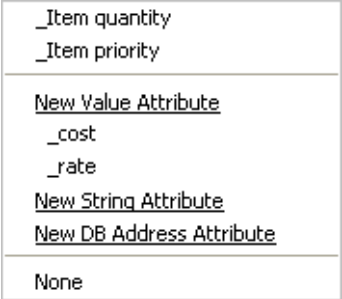


For more information, see “Preserving the items used to create a batch” on page 204 and “Properties when items are batched” on page 199.

Discrete Event

**Working with cost data**

To provide access to cost information, ExtendSim creates two attributes (`_cost` and `_rate`) for every item in models that have costing. Since these are automatically created, they are considered *system* attributes. If a cost is defined somewhere in the model, these attributes will appear in attribute popup menus, shown below:



Attribute popup menu

The information that is stored in these attributes depends on whether the item is a cost accumulator or a resource, as described in the following table:

<b>Item type</b>	<b>_cost attribute</b>	<b>_rate attribute</b>
Cost accumulator	The accumulated cost of the item	The item's waiting or storage cost (cost per time unit defined using the model's global time unit)
Resource	The cost per use of the resource	The resource's cost per time unit (defined using the model's global time unit)

The attribute handling blocks in the Item library (Get, Set, and Equation(I)) can be used to read, set, or manipulate these attributes. In addition, two statistics blocks in the Item library (Cost By Item and Costs Stats) can be used to gather cost data.

**Viewing Cost Data**

You can use a Get block to read the `_cost` and `_rate` attributes of any item, then plot the data or use the attribute value to perform additional calculations. For example, you can use a Get block to read the `_cost` attribute of cost accumulators and connect the Get block's `_cost` output connector to a Plotter Discrete Event to plot the accumulated cost of each item that passes through. This is shown in the model discussed in the following section.

**Changing Cost Data**

In most cases, it is sufficient to define the cost rates of the various cost drivers in a model and allow ExtendSim to automatically calculate and track costs. However, there may be times when you need to manipulate the cost values generated. The attribute handling blocks in the Item library (Get, Set, and Equation(I)) can be used to accomplish this.

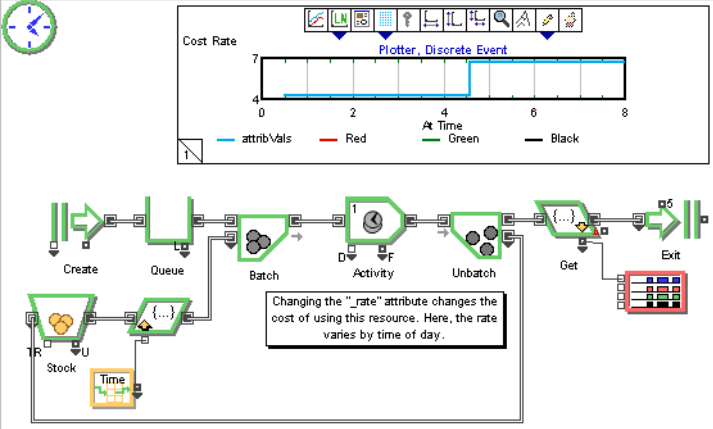
*Change Rate model*

For example, suppose the cost rates of a resource vary throughout the day. During peak times the demand for the resource is high and the cost per time unit increases. This can be modeled using

Discrete Event



the Set block (Item library) and the Lookup Table block (Value library) to explicitly set the `_rate` attribute of the resource as it exits a Resource Item block, as shown in the model below:



Change Rate model

The table in the Lookup Table block has a different rates for the period between hour 4 and hour 6.

A change in the rate will only affect resources as they exit the Resource Item block. Resources currently in use will not be affected until they are recycled back through the Resource Item and Set blocks.

In the above model, a Get block reads the `_cost` attribute before the items exit the model. The accumulated cost of each cost accumulator is then plotted. The plot (cloned onto the worksheet), shows that the cost of the items increases during the period of time that the resources' cost rates are higher.

**Gathering and Analyzing Cost Data**

The Create, Resource Item, and Resource Pool blocks, as well as queue and activity-type blocks, are capable of generating costs that get tracked with cost accumulating items. Additionally, each cost-generating block displays the total cost it generated in its *Total Cost* dialog item.

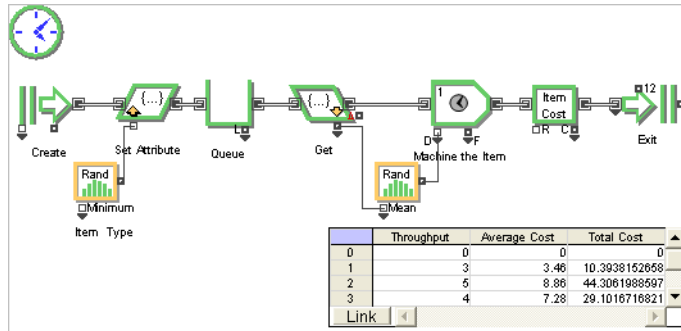
You can also use the Cost By Item and Cost Stats blocks (both in the Item library) to gather summarized cost information. The Cost By Item block reads and stores the `_cost` and `_rate` attributes of all the cost accumulating items that pass through it. The Cost Stats block collects and displays the total cost for each cost-generating block in a model.

*Cost By Item block*

Depending on selections in its dialog, the Cost By Item block lists the accumulated cost of each item that passes through it, the time the item passed through the block, and the total and average cost of all the items that have passed through. This block can also be used to list the cost of the items sorted by type.

In the Sort By Type model (shown below) three different item types are generated by randomly assigning a *Type* attribute of 1, 2, or 3. It costs \$5.00 per hour to run the machine. The machine's processing time for, and therefore the cost of, each item varies by type. The Cost By Item block

lists the costs of the items sorted by the Type attribute. As an item passes through the block, the row corresponding to the value of the Type attribute (1, 2, or 3) is updated.



Sort by Type model

*Cost Stats block*

The Cost Stats block is useful to determine which blocks are contributing the most to the total cost of the items being processed. See the help text of the block for a detailed description of how to use the Cost Stats block.

**How ExtendSim tracks costs**

The previous sections discussed how to perform ABC in ExtendSim. This section provides a more detailed look at how ExtendSim tracks costs and is included mainly for informational purposes.

**Setting the `_cost` and `_rate` attributes**

When you define the cost or cost rate for a cost accumulator or resource (as discussed in “Defining costs and cost rates” on page 226), ExtendSim will assign the value to the appropriate costing system variable for that item. The fixed cost of the item is assigned to the `_cost` attribute and the variable cost rate is assigned to the `_rate` attribute.

**Combining resources with cost accumulators**

Whether you use the resource item and batching method or the resource pools method to model resources, two things happen when a resource is attached to a cost accumulator:

- 1) The value of the resource’s fixed cost is automatically added to the cost accumulator’s `_cost` attribute.

If using the batching method, the resource’s fixed cost comes from the resource’s `_cost` attribute. If using resource pools, the resource’s fixed cost comes from the *Cost per use* dialog item of the corresponding Resource Pool block.

- 2) The resource’s variable cost rate and the number of resources used are stored in an internal program structure called the *cost array*.

The cost array stores costing information for each cost accumulator in the model. ExtendSim uses the data in the cost array to calculate the time-based cost contributed by any resources that are combined with the cost accumulator. If using the batching method, the resource’s variable cost rate comes from the resource’s `_rate` attribute. If using resource pools, the resource’s variable cost rate comes from the *Cost per time unit* dialog item of the corresponding Resource Pool block.

Discrete Event

When a resource is released by an Unbatch or Resource Pool Release block, the information stored in the cost array is updated to indicate that the resource is no longer combined with the cost accumulator.

### Calculating costs

As previously mentioned, the Create block and activity, queue, and resource-type blocks are all capable of generating costs. As these blocks process cost accumulators, they will automatically calculate the cost and add it to the item's `_cost` attribute. In addition, each cost-generating block will update its *Total Cost* information. This dialog item displays the total cost contributed by that particular block only. The following sections briefly discuss how these calculations are performed.

#### In the Create block

When a cost accumulator is generated, ExtendSim will add the fixed cost (*Cost per use*) of the Create block to the cost accumulator's `_cost` attribute.

For each cost accumulator generated, ExtendSim also will add the fixed cost of the Create block to its *Total cost* dialog item.

#### In activity-type blocks

When a cost accumulator enters an activity-type block, ExtendSim will add the activity's fixed cost (*cost per item*) to the cost accumulator's `_cost` attribute. In addition, it will calculate the variable time-based cost (the *processing* or *transportation* cost of the activity and the waiting cost of any resources currently combined with the cost accumulator), and add it to the `_cost` attribute of the cost accumulator.

For each cost accumulator that passes through the block, ExtendSim also will add the fixed and variable cost contributed by that activity-type block (not including costs contributed by any resources combined with the cost accumulator) to that block's *Total cost* dialog.

#### In queue-type blocks

Queue-type blocks have a checkbox labelled "Calculate waiting costs". If that checkbox is selected when a cost accumulator enters a queue-type block, ExtendSim will calculate the time-based cost. This is composed of the waiting or storage cost of the cost accumulator as calculated from the cost accumulator's `_rate` attribute and the variable cost of any resources currently combined with cost accumulator. The time-based cost is added to the `_cost` attribute of the cost accumulator.

For each cost accumulator that passes through a queue-type block, ExtendSim also will add the waiting cost calculated from the cost accumulator's `_rate` attribute (not including costs contributed by any resources combined with the cost accumulator) to that block's *Total cost* dialog item.

#### In resource-type blocks

The Resource Item block is capable of providing items that are either cost accumulators or resources, depending on selections in its Cost tab, as shown at right.

If the block is providing cost accumulators, it will generate costs similar to a queue-type block.

If the block is providing resources, the total cost of using the resources is calculated and displayed in the block's *Total cost* dialog item. The calculations are based on the resource's utilization rate and cost rates defined in the block's Cost tab.

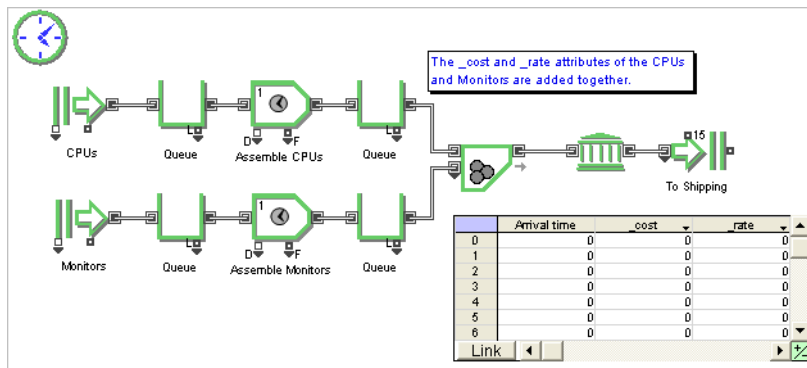
Cost options in Resource Item block

### Combining multiple cost accumulators

In manufacturing processes, different parts of the product may be worked on in parallel, then combined later to form the final product. In these cases, multiple cost accumulators will contribute to the cost of the final product.

#### Multiple Cost Accumulators model

For example, when a computer manufacturer prepares a system for an end user, the CPU and the monitor must each be assembled then combined into one shipment. The CPU and monitor may be worked on in parallel, then combined using a Batch block, as shown in the model below:



Multiple Cost Accumulators model

When the two cost accumulators, the CPU and the monitor, are batched together, two things will happen:

- The `_cost` and `_rate` attributes of the input items are added together. The resulting cost accumulator will have an accumulated cost equal to the combined accumulated cost of the input items and a waiting cost rate equal to the combined waiting cost rates of the input items.
- Any resources, whether from batching or from a resource pool, that are combined with the input cost accumulators will be combined with the cost accumulator that is output from the batching block. Note that any rules or limitations associated with batching resources with items will apply to the resulting cost accumulator (see “Batching and unbatching resources with cost accumulators” on page 229).

Discrete Event

# **Discrete Event Modeling**

## **Statistics and Model Metrics**

Statistically analyzing discrete event models

Remember that, by itself, simulation does not provide exact answers or optimize a system. Instead, a well-built model will capture important data and report statistical results. These metrics should provide the information needed for the analysis and decision-making process.

This chapter discusses specific methods for statistically analyzing discrete event models, such as:

- Gathering statistics for specific types of blocks
- Clearing statistical accumulators after a warm-up period
- Using the History block to get item information
- Using attributes to accumulate information about items
- Determining cycle time by timing the flow of items
- When to use time weighted statistics

For a more generalized discussions of statistical analysis, see also the following chapters:

- “Math and Statistical Distributions” starting on page 599
- “Analysis” starting on page 563

 The models illustrated in this chapter are located in the folder \Examples\Discrete Event\Statistics.

### Commonly used blocks

The following blocks will be the main focus of this chapter. The block’s library and category appear in parentheses after the block name.



**Clear Statistics** (Value > Statistics)

Clears statistics in other blocks, eliminating the statistical bias of the warm-up period.



**Display Value** (Value > Outputs)

Displays and outputs the value that is input.



**History** (Item > Information)

Records information about items and their properties, such as the value of an attribute, the item’s arrival time, its priority, and so forth.



**Information** (Item > Information)

Reports item statistics such as a count of the number of items, the throughput rate, cycle time, and the time between item arrivals.



**Mean & Variance** (Value > Statistics)

Calculates a mean, variance, standard deviation, and confidence interval.



**Statistics** (Value > Statistics)

Summarizes statistics for a particular type of block, such as activities or queues. Reports results in a table. Information is calculated using a specified statistical method, which can be customized.

### Gathering statistics

The Statistics block (Value library) accumulates data and calculates statistics for a particular type of block using a specified statistical method. In addition to the block number, block name, and the

time the information was observed, this block displays metrics that are specific to the block type, such as utilization or average wait time for activity-type blocks or the mean, variance, and standard deviation of all the Mean & Variance blocks in the model.

The Queue Statistics model, located in the folder \Examples\Discrete Event\Statistics, uses the Statistics block to gather information about queues.

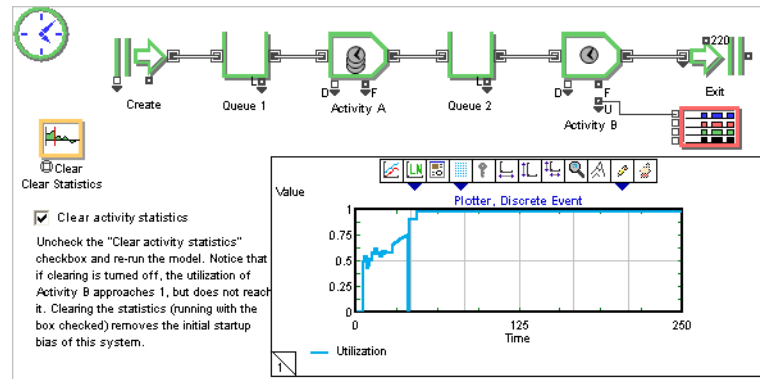
Since the Statistics block is used to gather information in continuous, discrete event, and discrete rate models, it is discussed fully in “Statistics” on page 564.

### Clearing statistics

At the start of a simulation run the queues are often empty and operations have nothing to process. After the model has been running for a while, it gets to the point where it is functioning more like the real system at normal operating levels. The interval from when the model starts to when it is functioning in a steady or normal state is called the *warm-up period*.

The Clear Statistics block (Value library) is used to reset statistical accumulators for the blocks specified in its dialog, eliminating the statistical bias of the warm-up period. For more information about this block, see “Clear Statistics” on page 566.

### Clearing Statistics model



Clearing Statistics model

In the Clearing Statistics model, statistics are cleared after 40 seconds, removing the warm-up period for the model. This is seen by the utilization of 1 for Activity B when the model is run. Unchecking the *Clear activity statistics* checkbox on the model worksheet causes the utilization of Activity B to approach, but never actually reach, 1. This is due to the effect of the initial idleness of the Activity B block at the start of the simulation run.

### Using the History block to get item information

When building a model, it is important to start small, verify that the section you have built is working as expected, then enhance that model section. The History block is particularly useful for verifying model data because it provides important information about each item as the simulation runs.

There are two ways to add a History block to a model:

- Connect it in series by dragging a History block from the Item library and connecting it between other blocks so that items pass through it.

- Connect it in parallel by right-clicking an item output connector and selecting *Add History block*. This automatically connects a History block to the original block's item output connector. If a History block is added in this manner, only its input connector is used. (Caution: Be sure there is an Item library block connected to the original block's item output connector, otherwise its item will have no place to go.)

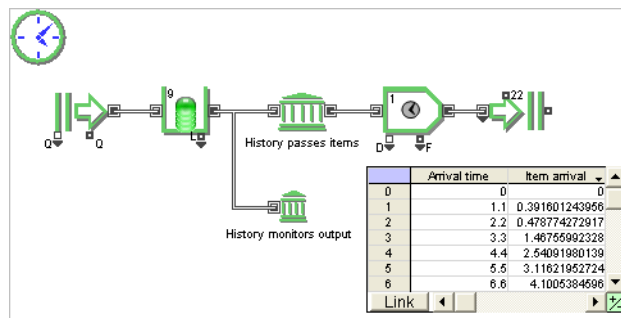
Each item that passes through the block (if it is connected in series) or is viewed by the block (if it is connected in parallel) is allocated a row in the History block's table. The table's first column displays the item's arrival time. Popup menus at the top of the other columns are for selecting additional information to display, such as the value of an attribute, an item's property, and so forth. You can choose to save item history with the model, show string attributes, and display Calendar dates.

Since the History block can use a lot of memory, put it in the model during testing, then remove it when you have verified that the section is working as expected. To automatically remove all of the blocks that have been added by right-clicking, right click one of them and select "Delete all auto-created History blocks".

### History model

The History model shows two History blocks: one has been physically placed in series between a Queue and an Activity and one has been auto-created and placed in parallel to the Queue block.

Both blocks report the same information (the item's arrival time and the value of an attribute called *Item arrival*), as shown in the cloned table in the model window.



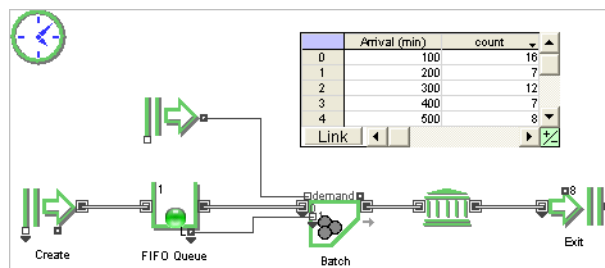
History model

If a History block has been auto-created and placed in parallel to another block, there must be subsequent blocks that can pull the items in. This is discussed at "Pulling and viewing" on page 247.

### Verifying Information model

The example shown at the right illustrates the use of the History block to verify that batches are created at the correct time with the correct number of items.

The topmost Create block (labeled Schedule Batches) schedules when the batch is created and the Queue's length (L) output determines the batch size. This causes all items within the Queue to be batched.



Verifying Information model

Discrete Event



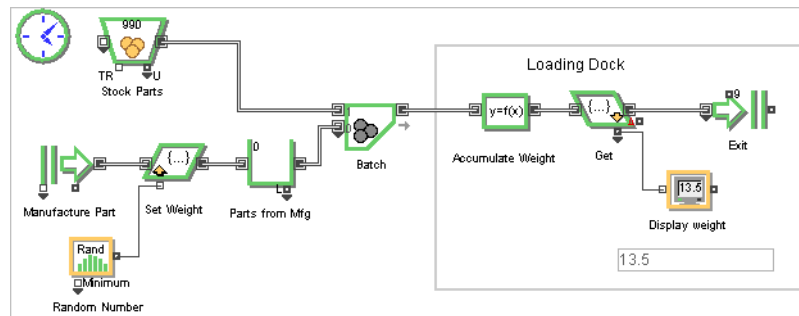
### Accumulating data

There are various methods you can use to accumulate data. Attributes can be used to hold cumulative values, such as the total weight of an assembly or the number of parts in a box. And the Holding Tank block (Value library) can accumulate total processing time to determine equipment refurbishment schedules. Data can be accumulated at any step in the model, even when the item is not being processed.

**!** It is important to not make the error of assuming that you can combine attribute values and then accumulate them. See “Using the Holding Tank block to accumulate values” on page 252 for more information.

### Non-Processing model

In the Non-Processing model, one part from Stock and another from Manufacturing are combined into an assembly. The stock part weighs 10 pounds and the manufactured part weighs between 1 and 3 pounds. The model uses an attribute called **Weight** to track the weights of the separate parts.

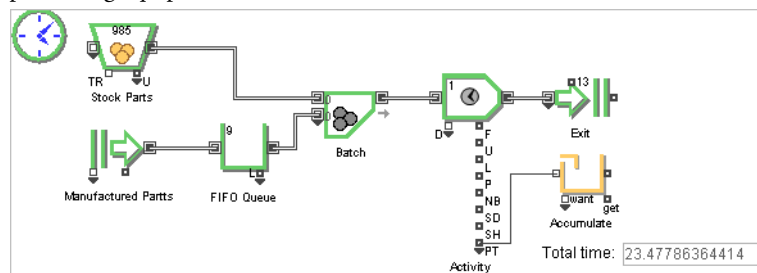


Non-Processing Model

The Properties tab of the Batch block is set to *sum* the values of the Weight attribute for the completed assembly. After the parts are batched, an Equation(I) block increments the Weight attribute by 0.5 pounds. At the loading dock, the weight of the current item is displayed as it leaves.

### Processing model

If the data to be accumulated is dependent on processing, you can accumulate values using a Holding Tank block (Value library) connected to the *PT* (process time) connector on an Activity block. For example, to accumulate the total amount of processing time parts required, as an indication of when the processing equipment needs to be refurbished.



Processing model

In this model, each time an item leaves the Activity, its processing time will be added to the value in the Holding Tank.

### Time weighted versus observed statistics

The Mean & Variance block (Value library) can calculate either a time weighted or observed statistic:

- If *Use time weighted statistics* is checked in the Mean & Variance block, the mean, variance, and standard deviation are calculated by weighting the input value based on the simulation time. This is derived by dividing the input value by the duration of that input value and then summing these over the course of the simulation.
- If *Use time weighted statistics* is not checked, the sum of the input values will be divided by the total number of input values received, resulting in an observed statistic.

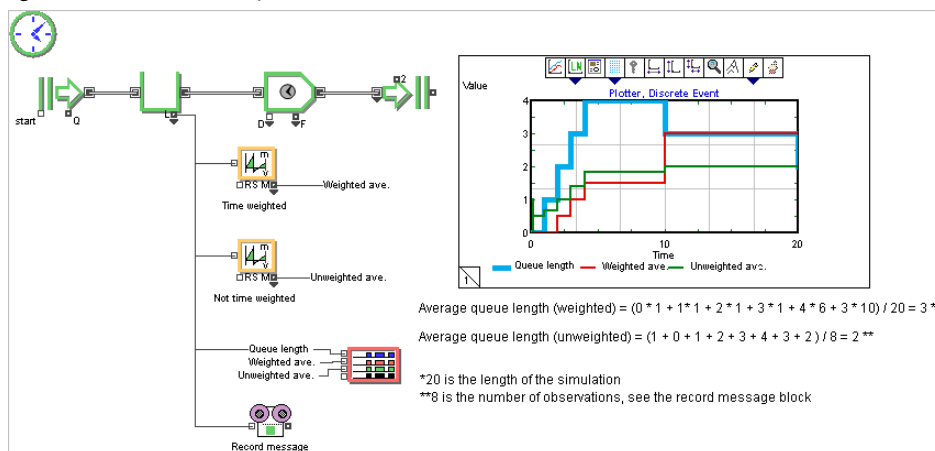
When using the Mean & Variance block, carefully consider the type of statistics that you want to calculate. Some guidelines for whether or not to select the time weighted statistics option are:

- If the value that you are collecting statistics on has a value at every point in the simulation, enable time weighed statistics. A good example of this is the number of items in a block (reported by the L connector). At any point in the simulation, there are a certain number of items in a block. To determine the average value, weight it over time.
- If the value that you are collecting statistics on only has a value at specific events, do not use time weighted statistics. An example of this is the W or wait time connector. This connector only has a value when an item leaves the block, which is a specific event. In that case, time weighted statistics should not be used.

Discrete Event

### Time Weighted Statistics model

The Time Weighted Statistics example shows the difference between the two methods of calculating statistics and how they are calculated.



Time Weighted Statistics model

Comparing the weighted and un-weighted approaches to the average queue length reported in the Queue block's Results tab, it is clear that not using time weighted statistics would give an incorrect answer for this model.

### Timing the flow of items in a portion of the model

In addition to performance information that is directly available in a model, you may want to determine *cycle time* – how long it takes an item to go from one part of the model to another. For example, you may want to know how long a customer waits in line to place an order, or how long it takes that customer to get served once the order is placed.

To see how to determine cycle time in a model, see “Cycle timing” on page 254.



# **Discrete Event Modeling**

## **Tips and Techniques**

Helpful information for when you build discrete event models

This chapter provides some tips, techniques, and information you may find helpful when building discrete event models. The chapter covers:

- Moving items through a simulation
  - How items move: holding and pushing, viewing and pulling
  - Implications of connecting to multiple item inputs
  - An item's travel time
  - Using scaling for a large number of items
  - Preprocessing
  - Restricting items in a system
  - Connecting to the *select* connector
- Issues for continuous blocks in discrete event models
  - Setting time-based parameters using a Random Number or Lookup Table block
  - Varying a distribution's arguments for the Create block
  - Accumulating values using a Holding Tank block
- Cycle timing
- Item library blocks
  - The Executive
  - Types of blocks: residence, passing, decision, and non-item
  - Common connectors on Item library blocks
- Event scheduling
- Messaging in discrete event models

 The models for this chapter are located in the folder \Examples\Discrete Event\Tips.

## Moving items through the simulation

In general, item input connectors on discrete event blocks will pull an item in, do something with it, wait for the block connected to the item output connector to pull the item out, then pull in another item. For example, the Activity block will pull items from preceding blocks, process those items, and hold them to be picked up by another block.

It is important that you understand the ExtendSim discrete event behavior so you can avoid making modeling errors.

### How items move through the simulation

It is important to understand how items move through specific blocks so that you can avoid two rare but possible pitfalls: losing items from the simulation and having items stop moving in the simulation.

To avoid the problems discussed below, you should probably connect Create blocks to queues so that items do not get lost and connect the History block in parallel with other blocks that will actually pull in the items.

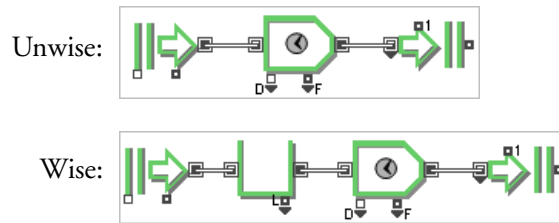
### ***Holding and pushing***

Item library blocks treat their output items in one of two ways:

- Most blocks hold the item and it leaves only when another block pulls it in.
- When set to create items randomly, by schedule, or infinitely, the Create block pushes the item from the block when it is generated, regardless of whether it will be picked up by another block. The Create block has to push items out, because those items are created within the block and are *arrival time* related.

*Avoid this pitfall*

When a Create block pushes an item and it is not picked up, the item disappears from the simulation. Generally this would only be used in certain very specific types of models. In most situations where the Create block is set to create items randomly, by schedule, or infinitely, follow the block with a queue to collect the items and hold them, so that all the items generated are available for the rest of the model.



Of course, if the Activity has an infinite capacity, it is not necessary to place a queue after the Create block.

**!** A Create block set to *Create items infinitely* should *never* be connected to an infinite capacity queue, since generating an infinite supply of items would overwhelm the system.

**Pulling and viewing**

There are two ways a block's item input connector can have access to an item: it can pull an item from the preceding block (as most connectors do), or it can simply *view* an item that is waiting at the item output of the preceding block. If an item input connector pulls an item in, it has access to the item for processing. However, if an item input connector only views items, it does not have direct access to them, it can only sense their presence at the preceding output connector.

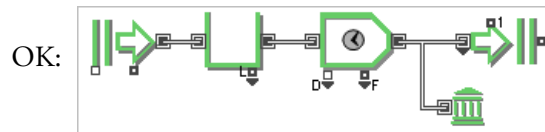
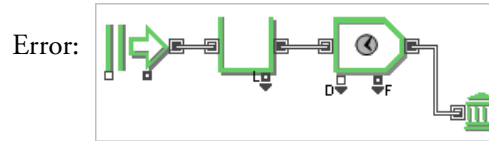
The particular connectors that only view items (not pull them) are:

- The Gate block's *sensor* connector when it is set to *Type: area gating* or its *demand* connector when the block is set to *Type: conditional gating with items*.
- The item input connector on the History block, if the block has been added in parallel to another block. This is shown below and described in "Using the History block to get item information" on page 239.

*Avoid this pitfall*

When one block holds an item and that item is *only* viewed by another block, the item does not move through the simulation and is blocked. This is never desired.

For example, both screenshots to the right show a History block that has been auto-created and is viewing items in an Activity block. In the top (error) screenshot, after the Activity block has finished processing its first item, the item will have nowhere to go since it is blocked.




**Connections to multiple item input connectors**

You can connect from one item output connector to as many item input connectors as you want. However, since items can only be in one place at a time, the first connector to pull in the passed item gets it and the other connectors do not.

Furthermore:

- If more than one input on a single block is free, the item will arbitrarily go to any available input. (Note that the selection is arbitrary – not random.)
- If more than one block is free to accept the item, the item will go to the block that was first connected in the model. This is shown in “Implicit routing” on page 151 and “Simple parallel connections” on page 166.

It is more typical that you would want to specify which input connector, or which block, would get an item. For more information, see “Items going to several paths” on page 149.

 Unless it is completely unimportant in the model, you should always use the Select Item In and Select Item Out blocks to explicitly state how items should be routed. Otherwise, the order in which their connections were made will dictate the routing.

**An item’s travel time**

In a discrete event model items travel from block to block as dictated by the connection lines. The lines between blocks indicate the path of the movement, but they don’t provide any delay to the items.

In most cases, travel time is insignificant and can be safely ignored. Where an item’s travel time is significant to the model, you can:

- Increase the delay time of destination blocks to compensate for the travel time
- Specify a minimum wait time in a Queue block’s Options tab
- Explicitly set a travel time in a Transport or Convey Item block, as discussed in “Transportation and material handling” on page 185. (You can easily add a Transport block between two blocks by right-clicking the leftmost block’s output connector and selecting “Add transport block”.)



### Using scaling for large numbers of items

In discrete event modeling problems, the number of items that need to be processed through the simulation may be quite large. This will slow down the execution of the model and increase the amount of memory required. It is often possible (and non-destructive to the validity of your results) to scale down the number of items passing through the model. For example, if there is one item representing a single log in a simulation of a lumber mill, the same model could quite possibly run faster, and equally well, with one item representing ten logs.

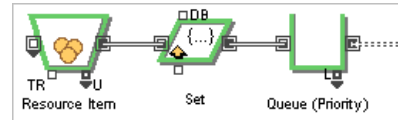
When you make scaling changes to a model of this nature, it is very important to reflect the changes everywhere in the model. Thus, if an activity that represented a saw in the lumber mill was set to take one time unit to process an item (one log) before, it must now take ten time units to process the same item (ten logs) after the scaling.

While scaling can sometimes be a useful approach, the Rate library is specifically designed to model high volume and/or high speed systems. In most cases, using the Rate library is superior to item scaling. The Rate library is available with the ExtendSim AT and Suite products.

### Preprocessing

You sometimes want to have all the items available at the beginning of a simulation instead of generating them as the simulation proceeds. For instance, if you need some random orders presented to the model in sorted order, you might want to sort them before the simulation starts. This is difficult under normal circumstances since the first order would begin traveling through the simulation as the second one was being created. There is an easy method that will cause ExtendSim to create lots of items, store them in a queue, and release them.

Set the initial value in a Resource Item block to the number of items you want to generate. Connect the Resource Item block to a Set block where you attach item properties (priority, attribute, etc.). Then connect to a Queue that sorts based on the desired item property.



Preprocessing

When the model is run, all the items will travel from the Resource Item block to the Queue on step zero. This makes the items, with all their properties, available to the rest of the model at the start of the simulation.

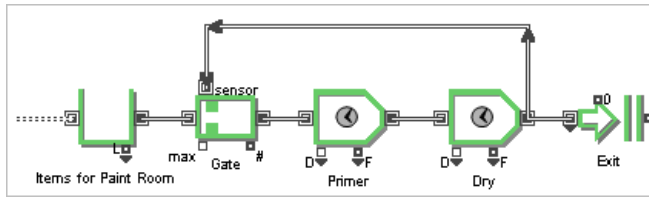
If there are many items in the Resource Item block, the status bar may show the phrase *Initializing Data*. As soon as the preprocessing is done, the timer will settle into a more useful number.

### Restricting items in a system

As part of a model you may want to have a section composed of a group of blocks in which only one item (or a limited number of items) can be anywhere in the section at a time. For example, assume you are modeling a manufacturing process with a paint room. There are many blocks that represent the steps in the paint room but only two items are allowed in the entire paint room at a time. New items must be restricted from entering the room until one or more items leave.

When set to *Type: area gating*, the Gate block is perfect for this because its *sensor* connector tells it each time an item has reached the end of a system. The number of items allowed are set in the Gate block's dialog; in this case, two. Put the Gate block at the beginning of a system; at the end of the system, run a parallel connection from the output of the last block to the Gate block's sensor connector.

In this example, the paint room is represented by two Activity blocks. The Gate block will pass the first two items it receives into the paint room, then only let a new item pass when it sees the first item at its sensor connector. As each item leaves the paint room, a new item can enter. Note that the sensor connector doesn't accept any items; it only views them as an indicator of their position in the model.

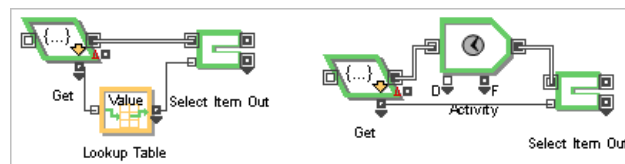


Restricting items

Another, more flexible, approach is to use a Resource Pool block to restrict items in a section of the model. This is useful when you need to track statistics on utilization, or if you have multiple flows of items accessing the same physical space.

### Connecting to the select connector

The *select* connector is used to control the behavior of the Select Item In and Select Item Out blocks. If the select connector gets its value from a Get block, you should avoid putting Set blocks, activities, and queues between the Get block and the Select block.



Safe and unsafe connections to the select connector

These blocks can alter the value sent to the *select* connector or delay the item so that the Select block routes the wrong item.

For instance, the model segment shown at the left of the above screenshot will work properly. The one shown to its right may not work correctly, because the item to be routed may still be in the Activity block.

### Continuous blocks in discrete event models

Value library blocks can be considered passive blocks in discrete event models. In most cases, blocks from the Value library will not recalculate unless told to do so by an Item library block. This has important ramifications for the behavior of Value library and other continuous blocks in discrete event models.

When an Item library block needs a new value at one of its value input connectors, it will send a message out that connector to the connected Value library block, requesting a new value. Likewise, when an Item library block has calculated a new value at one of its value output connectors, it will send a message to the connected Value library block notifying it of the change. Typically these messages will cause the Value library blocks to recalculate. The messages are then propagated to all other connected Value library blocks.

In discrete event models, most blocks from the Value library typically neither post events to the Executive nor receive event messages from the Executive. Furthermore, Value library blocks do not recalculate on each time step in discrete event models as they do in continuous models. Rather, they are only alerted to recalculate if they receive a message from an Item library block. And most Item library blocks are triggered to action only by the arrival of an item. Complications can arise if a Value library block does not get a message from an Item library block when you expect or want it to recalculate.

☞ Some Value library blocks, such as the Clear Statistics and the Lookup Table, do generate events in a discrete event model because they need to perform a specific action at a scheduled time.

To prevent modeling errors, it is helpful to understand this relationship between Item and Value library blocks. Common situations where this is important include:

- 1) Setting time-based parameters using connections from a Random Number or Lookup table block. This is described on page 251.
- 2) Varying an argument for a Create block's distribution with a Lookup Table where there is the possibility of a message being ignored. This can cause a lot fewer items to be created than expected, as discussed on page 252.
- 3) Using a Holding Tank block to accumulate the result of a calculation performed on two or more values coming from Item library blocks. If not modeled properly, the Holding Tank can get duplicate messages and will have incorrect results. This is described on page 252.

☞ For a detailed discussion about messaging between discrete event and continuous blocks, see "Value input and output connector messages" on page 261.

### Setting time-based parameters using connectors

Some time-based parameters can be set using a connector value. In these situations, the value sent to the input connector must be defined in the time unit specified in the receiving block. The following examples illustrate issues you should be aware of.

#### Random Number block

Assume you want the delay for an Activity block to be approximately 30 minutes and you connect a Random Number block to the Activity's D input connector. If the local unit of time for the Activity is minutes, you would set the Random Number block to generate numbers with a mean of 30. However, if the Activity block used hours as its local time unit, the Random Number block should be set to generate numbers with a mean of 0.5.

☞ It is a modeling error to expect the Random Number block to create random values at each event in a discrete event model. The only time this Value library block will be activated to output a new value is when it receives a message on one of its connectors. In the above example, the Random Number block will get a message each time an item arrives to the Activity block, so each item will get a random delay time. For more information, see "Value input and output connector messages" on page 261.

#### Lookup Table block

It is possible that the numbers in one column of a block are based on the time unit for that block, and the numbers in another of its columns are based on the time unit for a second block. An example of this is described in "Choosing time units for the columns" on page 113.

### Varying a distribution's arguments

It is common to use another block to specify the arrival intervals by varying the argument (such as the mean) of a distribution in the Create block. To avoid unexpected results, it is important to understand what happens in the Create block when you do this. The Create block's default behavior is to send a message to the Executive block giving an arrival time, called "nextTime", for the next item based on the current input parameters. When simulation time reaches nextTime, the Executive block sends a message to the Create block. The Create then releases the item and generates a new nextTime based on the current values of the input parameters. For the period of time

between releasing items, the Create block will not react to changes in the input parameters. If the inputs change drastically, this can cause unexpected results as shown in the following example.

**Lookup Table example**

Assume you connect a Lookup Table block to the *mean* input connector of a Create block, varying the interarrival mean according to the schedule in the table at right.

	Input Value	Output 1
0	0	12
1	6	0.5
2	14	12
3	24	12

Lookup Table's table

For this example, both the Time and Output 1 columns of the Lookup Table block are defined using hours as the time unit. The mean for the interarrival time is 12 hours except for the period between hours 6 and 14 where the mean is 0.5 hours. Because the mean is only an average, it is possible for the Create block to generate an item at time 0 with an arrival time of 14 or more. If that happens, the Create block will get the message that the mean should have changed to 0.5 between hours 6 and 14, but it will ignore it. In this case, the number of arrivals will be much less than expected.

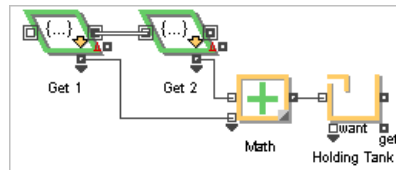
The Options tab of the Create block has a check box labeled *Interarrival time changes occur immediately*. When checked, it will cause the Create block to immediately respond to changes to any input parameter. In the case of the above example, the Create block would recognize that the mean had changed from 12 to 0.5 at time 6. It would then generate a new random number for the arrival time using the new input values.

**Using the Holding Tank block to accumulate values**

When accumulating data in a model, it is important to not make the error of assuming that you can add attribute values and then accumulate them.

**Incorrect approaches**

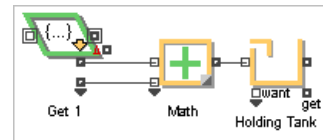
Assume you want to accumulate the sum of two attribute values. Your first intuition might be to add the two attribute values together and send the result to a Holding Tank block (Value library).



Incorrect approach #1

Two incorrect approaches to do this are shown at right. In the first case the attribute values are obtained from two Get blocks; in the second case the attribute values are captured from one Get block. In both cases the Holding Tank will give incorrect results.

Each time an item passes through a Get block, a message is sent out each value output connector. The way this been constructed, the passing of one item will result in two additions being contributed to the Holding Tank block.

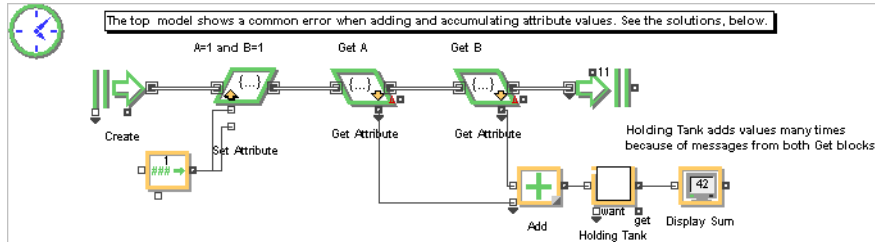


Incorrect approach #2

*Attributes Error model*

The Attributes Error model illustrates the modeling problem and some potential solutions. The problem with this model is clear if you look at the numbers, 11 and 42 respectively, displayed by the Exit and Display Value blocks. Since the values of attribute A and attribute B are both 1, the accumulated total displayed

on the Display Value block should only be twice the value displayed in the Exit block; clearly this is not the case in this model.



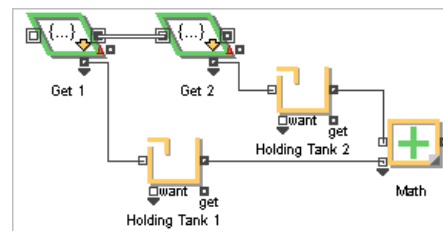
Attributes Error model: Problem

The reason for the modeling problem shown above involves the message passing system in discrete event models. Individual items travel through the Get blocks sequentially. As an item passes through the first Get block, the block sends a message and the value of the attribute to the Math block (Value library). The Math block then recalculates and sends a message and the value to the Holding Tank block (Value library). When the item moves to the second Get block, it will send a message to the Math block again. This causes the Holding Tank block to get two messages and two values for each item that passes through the system. This kind of problem will occur in any discrete event system where there are multiple connections to a Holding Tank block (either directly, or indirectly as shown above) or if one Get block was used with two outputs.

The Attributes Error model includes four examples: the problem and the three solutions discussed below.

**Solution #1: two Holding Tank blocks**

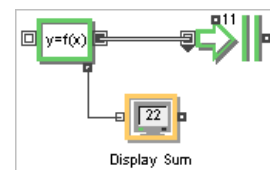
One way to solve this problem is to accumulate the attributes' values separately using two Holding Tank blocks. The contents of the Holding Tanks are then added together. This prevents the "double counting" of the previous example, because each Holding Tank block receives only one message and value per item that passes through the Get block it is attached to.



**Solution #2: the Equation(I) block**

Another solutions is to perform the calculation in the Equation(I) block (Item library). The Equation(I) sums up and accumulates the attributes in one step, so it avoids the double messaging problem altogether. The equation entered is:

```
Accumulate = Accumulate + a + b;
Result = Accumulate;
```

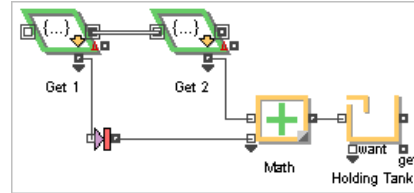


where *accumulate* is a static variable and *a* and *b* are the two attribute values from the item entering the Equation(I) block. The equation adds the two attribute values to the accumulated value, then sets the output to the accumulated value.

As an alternative, instead of both summing the attributes and accumulating, the Equation(I) could just sum the attributes and output that value to a connected Holding Tank.

**Solution #3: the Stop Message block**

You can also use the Stop Message block (Utilities library) to prevent the Math and Holding Tank blocks from receiving two messages for every one item. This block stops messages from being passed through a value connection; it is designed to solve problems of this nature.



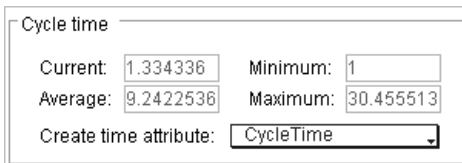
The Stop Message block is connected between the value output of Get 1 and the value input on the Math block. This will prevent the first message from reaching the Math block but will allow the value to be passed.

**Cycle timing**

The amount of time one block takes to process an item is known as the *delay* or *processing* time. *Cycle time* is the time an item takes to travel through a group of blocks. If there is no blocking in a model (that is, if all items leave their blocks exactly at the end of their delay time), the cycle time is the sum of the delay times for the section being measured. In most situations, this would rarely occur, and cycle time is usually more than the sum of the processing times. For instance, it is common that an item cannot leave a block because the next block is still processing its item.

To track an item's cycle time, use either the Timing attribute feature (if the item is being tracked from its origin) or a Set or Equation(I) block with an Information block (if the item is being tracked from some place other than its origin).

These methods are discussed below. In each case, the Information block reads the attribute and calculates the difference between when the item started the cycle and when it ended. The dialog of the Information block displays the current, average, minimum, and maximum cycle time for all items with the specified attribute. Its output connectors report the count of items, the time between items, their cycle time, and the throughput rate.

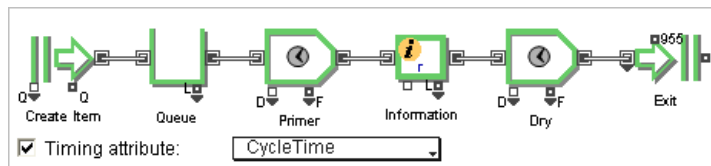


Cycle time portion of Information dialog

**Using the Timing attribute feature**

If you are tracking the item from its origin, use the *Timing attribute* feature in the Create block's Options tab to create a new value attribute and assign it to all items that are generated by the Create block. Then place the Information block at the end of the section you want to observe and select the name of the attribute as the *Timing attribute* in its dialog.

In the example shown at right, a value attribute named CycleTime has been created in the Create block, and the block's Options tab is set to *Timing: CycleTime*.



Cycle Time 1

The Information block is placed after the Primer activity and is set to *Calculate TBI and Cycle Time statistics*, and its *Timing attribute* is also set to *CycleTime*. For this model, the Information block calculates the time

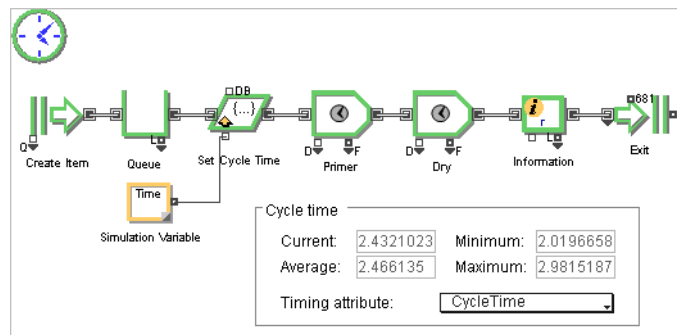
Discrete Event

from when items were first created to when they finish being primed. This includes the time items wait in the Queue.

### Using a Set or Equation(I) and Information blocks

If you are tracking the item from some place other than its creation point, put a Set or Equation(I) block at the beginning of the section you want to observe, create a new value attribute in that block, and set the attribute to the current time. Then place an Information block at the end of the section and enter the name of that attribute as the *Timing attribute* in its dialog.

In the example at right, an attribute named CycleTime2 has been created in the Set block. The Simulation Variable block (Value library) outputs current time and is attached to the Set block's value input connector. For this model, the Information block calculates just the time that the items take to go through the priming and drying processes.



Cycle Time 2

Discrete Event

## Item library blocks

### Executive block

The Executive block controls and performs event scheduling for discrete event and discrete rate models. An Executive block must be placed to the left of all other blocks in a discrete event or discrete rate model. Using it in a model changes the timing from continuous to discrete, and simulation time advances when events occur, rather than periodically.

This block can be used to:

- Manually control when the simulation stops. The default is that a simulation stops at the end time set in the Run > Simulation Setup dialog. You can also choose to stop the simulation when the number at the Executive's *count* input reaches a specified value.
- Allocate item availability. To conserve memory, this number should be something close to and less than the maximum number of items that you expect to see in the simulation. The default is that 10,000 items are initially available and additional items are made available in batches of 10,000.
- Declare string values for string attributes. A table in the Executive's Attributes tab is used to enter a descriptive text label (string) for each potential attribute value for a selected string attribute. For more information, see "Creating a string attribute" on page 106 and "Attribute types" on page 116.
- Manage all attributes. Tables in the Attributes tab allow you to select an attribute for renaming or deleting, and display blocks that use the selected attribute.
- Manage flow units and select global and advanced options for discrete rate models. For more information, see "Global and advanced options in the Executive" on page 364.

- Set information for the LP solver used in discrete rate models. For more information, see “LP technology” on page 376.

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

### Block types

As discussed on “Item connector messages” on page 262, Item library blocks pass messages through item connectors. There are three types of item-based blocks that determine how the item connector messages are handled:

- Residence-type blocks are able to contain or hold items for some duration of simulation time. Some residence blocks post events and some do not.
- Passing-type blocks pass item through without holding them for any length of simulation time. These blocks implement modeling operations that are not time-based; they usually do not post future events.
- Decision-type blocks route the items through the model. These blocks choose a route based on an item property, a random value, a sequence, or an input from a connector. Depending on what options are selected in the block, a decision-type block may or may not be able to hold onto items.

Discrete Event

### Why block types matter

Knowing these categories of blocks and how they relate to the processing of items will help you to build better models. For example an item will not enter a passing block before it has been determined that there is space in the next downstream residence block. And when you debug models it is useful to understand where the items can reside for any amount of time, as well as the time required for an item to move from one residence block in the model to another. In addition, some of the options in the blocks refer to specific block types. An example of this is the Transport block where you can specify that the distance to the next block is from the Transport block to the next non-passing (residence or decision) type block.

### Table of block types

Not all of the blocks in the Item library fit neatly into these categories, but it is helpful to use the categories as a framework for thinking about the messaging architecture. Following is a table of the blocks in the Item library and their associated type.

Block	Type	Block	Type
Activity	Residence	Queue Matching	Residence
Batch	Residence	Read(I)	Passing
Catch Item	Passing	Resource Item	Residence
Convey Item	Residence	Resource Pool	N/A
Cost By Item	Passing	Resource Pool Release	Passing
Cost Stats	N/A	Select Item In	Decision
Create	Residence	Select Item Out	Decision/Residence*
Equation(I)	Passing	Set	Passing



Block	Type	Block	Type
Executive	N/A	Shift	N/A
Exit	Residence	Shutdown	Residence
Gate	Decision	Throw Item	Passing
Get	Passing	Transport	Residence
History	Passing	Unbatch	Residence
Information	Passing	Workstation	Residence
Queue	Residence	Write(I)	Passing
Queue Equation	Residence		

\* If an item is allowed into the Select Item Out block before the decision is made (see the dialog check box), then it is a residence-type block. If the decision is made before the item enters the block, then it is a decision-type block.

### Common connectors on discrete event blocks

Many blocks use abbreviations or acronyms to indicate a connector's purpose. Some of these represent more than one purpose and are context sensitive. The following connector labels appear on many Item library blocks:

Connector	Meaning
$\Delta$	Delta
#	Count
A	Average cost (Cost By Item)
AD	Accumulated demand (Gate)
AS	Activity status
BT	Blocked time
C	Capacity (Activity, Workstation) Current cost (Cost By Item)
CI	Confidence interval
CT	Cycle time
D	Delay (Activity)
DB	ExtendSim database
DT	Down time (Activity)
DV	Down value (Shutdown)
F	Full (Activity, Queue) Field of a database table (Read Item, Write Item)
I	Interval between items
L	Length of waiting line (Queue, Activity, Workstation) Length of line (Information - will be 0 or 1)

Connector	Meaning
LO	Length out (Queue Matching)
MG	Match group (Queue Matching)
NB	Number blocked
P	Priority
PE	Preempt
PT	Process time
Q	Quantity
R	Renege time (Queue) Row (Cost By Item) Record (Read Item, Write Item when ExtendSim database is selected)
RS	Reset
SD	Shut down
T	Total cost (Cost By Item)
TBF	Time between failures (Shutdown)
TP	Throughput rate (Information)
TTR	Time to repair (Shutdown)
U	Utilization
UV	Up value (Shutdown)
W	Wait time for items leaving the queue

### Event scheduling

ExtendSim moves items in a discrete event model only when an event happens. Events are controlled by the Executive block 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.

Discrete event and discrete rate simulations use the same method for updating the simulation clock. Simulation models of this type are driven forward by event and the state of the model changes only at event times.

At each event, blocks that have posted an event to the event calendar for the current time receive a message notifying them that the time has arrived. Once all of the blocks have received their messages, the time for the next event is determined. Through this event scheduling mechanism the simulation clock jumps from one event to the next.

### Event calendars

ExtendSim utilizes a two-stage event calendaring method – the Executive block maintains a list of all events for the model and time-delay blocks maintain their own event calendars.

 This two-stage event calendar is very efficient and flexible. Unlike single stage event calendars, relatively little time is spent by the Executive in maintaining and searching the event list.

#### The Executive

The Executive block maintains a list of all event times for the model in its event calendar. At the beginning of each simulation event, the Executive locates the next future event and sends a message to each of the blocks in sequence that posted an event for that time. Once a block has completed processing its event, it will post its next event time to the Executive. If the block does not have a future event time, it will post a very large value as its next event time, effectively removing it from the list of pending events.

Blocks may have two or more entries on the Executive's event calendar. This is because they have different types of events that need to be processed. For example the Convey Item block has an event that occurs when an item is able to enter the block and an event for when the item leaves the block.

For more information about the Executive block, see page 255.

#### Internal event calendars

Each block that has a time delay associated with it (for example the Create, Activity, Pulse, and Shutdown blocks in the Item library) maintains its own, independent next event time.

Blocks such as the Activity, Convey, or Shutdown block can have multiple future events (one event for each item in the block) ongoing simultaneously. In this case, the blocks maintain their own internal event calendar, posting only the earliest of these events to the Executive's event calendar.

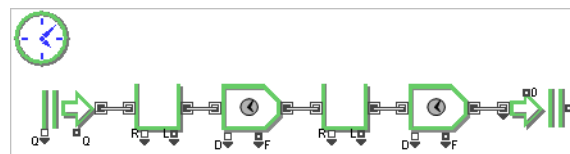
#### Zero time events

As the simulation progresses, there are many times when it is useful to generate a zero time event. This is done to allow an item to complete the process of moving into a block before the block attempts to perform additional actions on the item. For this purpose, the Executive contains a current events list. This is a short list of the blocks in the model that need to receive a message before the simulation clock advances.

A prime example of this is the Queue block. When an item arrives to a Queue, a zero time event is posted so that the Queue can return control to the upstream block that sent the item. The Queue receives another message before the clock advances so that an attempt can be made to send the item to the next downstream block. This feature enhances the efficiency and predictability of discrete event models.

#### Event Scheduling model

A discrete event model is helpful in understanding how event scheduling works. For this example: items arrive, wait at the first queue, are processed at the first activity, wait at the second queue, are processed at the second activity, and leave through an exit.



Event Scheduling model

In this model, there are three blocks that post events:

- The Create block posts an event for the creation of each item. The time between item arrivals is 0.6.
- The Activity 1 block posts an event for the earliest completion time of an item in the block. The duration of this activity is 1.0.
- The Activity 2 block posts an event for the earliest completion time of an item in the block. The duration of this activity is 0.5

As the simulation progresses through time, the event calendar in the Executive might look like this:

Time	Create posts next event time	Activity 1 posts next event time	Activity 2 posts next event time	Events
0.0	0.0	Infinity	Infinity	Item #1 is created
0.0	0.6	1.0	Infinity	Item #1 begins service at Activity 1
0.6	1.2	1.0	Infinity	Item #2 is created
1.0	1.2	2.0	1.5	Item #1 completes service at Activity 1 Item #1 begins service at Activity 2 Item #2 begins service at Activity 1
1.2	1.8	2.0	1.5	Item #3 is created
1.5	1.8	2.0	Infinity	Item #1 completes service at Activity 2
1.8	2.4	2.0	Infinity	Item #4 is created
2.0	2.4	3.0	2.5	Item #2 completes service at Activity 1 Item #2 begins service at Activity 2 Item #3 begins service at Activity 1
2.4	3.0	3.0	2.5	Item #5 is created

Discrete Event

Notice how the next event time is always the lowest of all of the event times for all the blocks; this is how a discrete event simulation works. Also, the table illustrates the concept of event scheduling but does not show all of the detail of what is happening as the items move through the blocks. For example, the Queue schedules a zero time current event as it moves the item through, but this is not shown in the table.

### Messaging in discrete event models

As discussed in “How ExtendSim passes messages in models” on page 533, the ExtendSim architecture allows application messages to be sent from ExtendSim to a model’s blocks and block messages to be passed between blocks.

Discrete event models use the same application messages as do continuous and discrete rate models. The block messages sent between Item library blocks are discussed below.

#### Block messages


Discrete event blocks have a sophisticated messaging structure for communicating with each other and with blocks in the Value and Rate libraries. These messages can be categorized as:

- Event
- Value connector
- Item connector
- Block-to-block

### Event messages

Event messages communicate between the Executive block and Item library blocks in a model. In a discrete event model the simulation clock advances from one event to another. Each time the clock advances, the Executive block sends event messages to the Item library blocks that have associated themselves with that event. There are two types of events: future and current.

- A *future* event message occurs when the simulation clock reaches a time posted by a block. For example, when an item enters an activity, the activity will post a future event to the Executive corresponding to the item's "finished time". Once the simulation clock has advanced to this future event, the Executive sends an event message to the activity, alerting it that the item has finished processing.
- A *current* event message occurs when a block wants to be activated before the simulation clock advances, but after it has completed its response to another message. For example, a queue will post a current event message to the Executive as it is pulling in items. After all the items have arrived to the queue, the Executive sends a current event message to the queue. This signals the queue to try and push all the items out of the block.

 The only blocks in the Value library that post future events are the ones that provide values or perform actions at specific times. Examples are the Clear Statistics block that resets the simulation statistics at a scheduled time and the Lookup Table block that provides values at scheduled times. Other Value library blocks lie dormant during a discrete event simulation unless they receive an activating message (either directly or indirectly through another block) from an Item library block.

### Value input and output connector messages

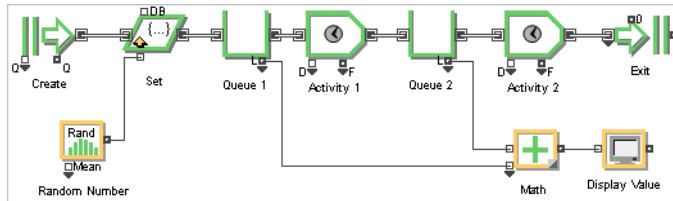
Blocks in a discrete event model send value connector messages either because a new number is needed by an input connector or because the value of an output connector has changed. These messages either request updated information for the input connectors or notify connected blocks that the output value has changed.

- When a message is sent from an input value connector, the sending block requests an updated connector value from the receiving blocks. Messages sent out the input connectors go only to the outputs of the directly connected blocks.
- Whenever an output connector changes, messages are sent to all of the inputs of the directly connected blocks. In this case, the sending block alerts the receiving blocks of a connector value change. Blocks that receive messages at their input connectors may, if appropriate, propagate messages:
  - Out other input connectors to make sure that all input values are current
  - To their output connectors to notify other blocks of the change in value

Through this mechanism, a single value change may cause any number of connected blocks to recalculate, ensuring that any system dependencies are automatically evaluated. For example, if the value of an input connector on an equation-type block changes, messages are first sent out the other input value connectors if they are connected (ensuring the equation will have up-to-date inputs prior to calculation.) Then, with updated inputs, the block recalculates its equation and posts the new results on its output value connectors. Once the new results have been posted, messages are sent out the output connectors, alerting any connected blocks that the results have changed.

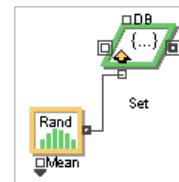
*Example of value connector messaging*

The sample model shown below illustrates how value connector messages work. In this example, items arrive, an attribute is set to a random number, and the items are then processed at two work areas (Queue and Activity blocks) in series. Three Value library blocks (Random Number, Math, and Display Value) provide a random number for the attribute value, add the two queue lengths, and display the sum of the lengths, respectively.



Example of value connector messages

When an item arrives to the Set block a message is sent out its value input connector. The Random Number block responds by providing a new random number each time it receives a message. This simple messaging example is shown at the right.

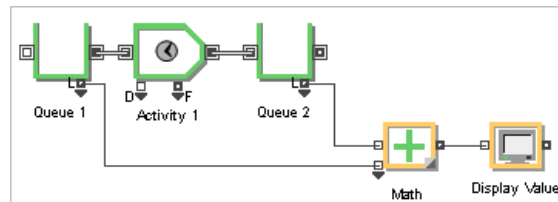


One message sent



In a discrete event or discrete rate model, the only time most Value library blocks (such as the Random Number block) are alerted to do something is when they receive a message on one of their value connectors. For instance, do not expect the Random Number block to continuously output a stream of random numbers in discrete event or discrete rate models.

In the more complex messaging case that is shown to the right, when an item arrives at the first queue its length will increase by 1. Since this changes the block's L (length) output connector, Queue 1 sends a message to all inputs connected to L (in this case the Math block). When the Math block receives this message, it sends a message to the L connector at Queue 2, ensuring that both inputs are up-to-date before any calculation is made. The values of the two length connectors are then added together and a third message is sent to the Display Value block, which then updates its animation and dialog.



Three messages sent

**Item connector messages**

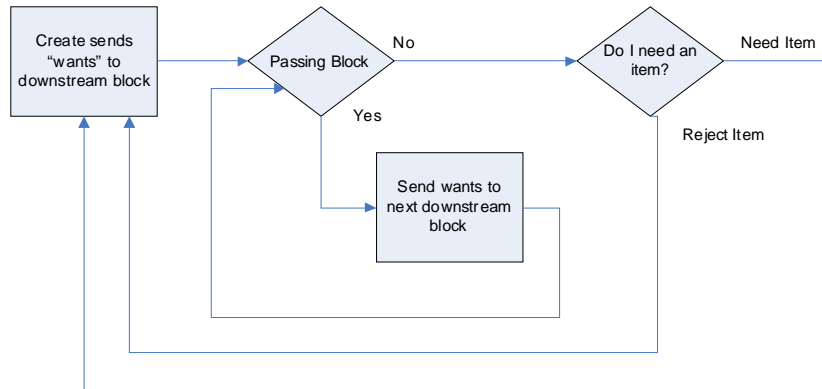
Item connector messages (primarily *wants*, *needs*, and *rejects*) propel items through the model. These messages use a conversation of messages to move items from one block to another. This mechanism allows for items to be both pushed and pulled from one block to the next. How these messages are handled depends on whether the block is a passing, residence, or decision block (see "Cycle timing" on page 254.)

*Pushing items*

In the case of pushing, the upstream block first sends a *wants* message.

- If the downstream block is a passing block, it forwards the message to the next downstream block through its output connector.

- If the downstream block is a residence block, it responds with either a *needs* message (if it can accept an item) or a *rejects* message (if it is unable to accept an item based on its status).
- If the downstream block is a decision block, it determines the status of the decision and any downstream blocks. It often does this by sending additional item or value messages and then responding with a needs or rejects.

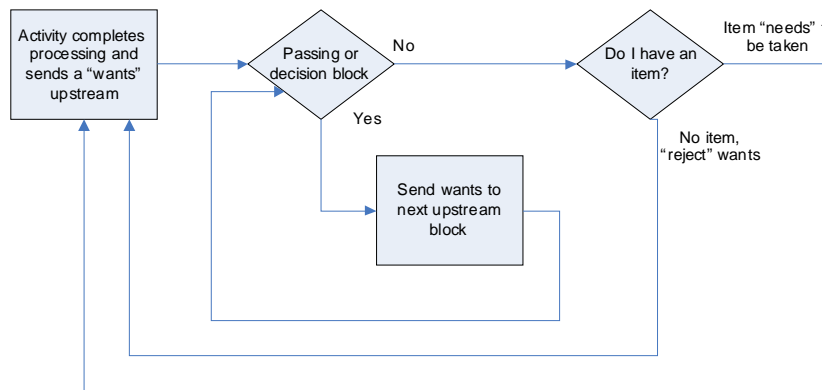


Flowchart for how items are pushed

*Pulling items*

To pull an item, a residence block sends the wants message upstream. This wants message is passed through the passing and decision blocks until it reaches a residence block. If the residence block has an item that is ready to leave, then a needs message is returned. If no item is available, then the residence block rejects the wants message.

Following is an example of pulling an item from an upstream block to a downstream block:



Flowchart for how items are pulled

This is only the first step in the process of moving an item. A number of messages follow that propel the item through the network of blocks. More details about those messages can be found in the Developer Reference.

***Block-to-block messages***

Block-to-block messages update the status of other blocks in the model. Sometimes a block needs to communicate with another block in the model, but there is no direct connection between them. For example, if a change in the shift status occurs, a Shift block needs to notify all of the blocks that reference that shift. These messages are sent 'through the air' to the blocks. In most cases, you will not even be aware that these messages are being passed back and forth. The actual operation and context of the message depends on the blocks involved in the conversation.



# Discrete Rate Modeling

## Introduction

Some things to know before you start  
modeling discrete rate systems

*“The question of doubt and uncertainty is what is necessary to begin;  
for if you already know the answer, there is no reason to gather any evidence about it.”*  
— Richard P. Feynman

Discrete rate modeling is based on rates of flow that change when events occur. In a discrete rate system, quantities of “flow” (material, product, data, etc.) are located in one or more parts of the model. During the simulation run, the flow moves from one location to another at a certain speed, called the effective rate. The movement between blocks that hold or route the flow follows paths, rules, and constraints that are set in the model.

As discussed in “Modeling methodologies” on page 43, the primary modeling approaches are continuous, discrete event, and discrete rate. In some situations (listed later in this chapter), simulating a system using discrete rate modeling is a more natural fit compared to using continuous or discrete event modeling. Processes that are event driven, rather than time driven, do not lend themselves to continuous simulation. Systems where there is no “item” that can be identified, or when there are so many items that identification is meaningless, can be more naturally represented using a rate-based approach rather than discrete event modeling. Furthermore, rate-based models run faster than discrete event models and are applicable to thinking in terms of flows, tanks, rates, and so forth.

☞ Any system or process that involves a quantity of something that is stored at one place, then moves to another place at a rate per time unit, can be simulated using discrete rate modeling.

Like continuous and discrete event modeling, rate-based modeling can help you perfect processes and products. It is useful for planning resource capacity by determining the rate at which products are being processed or sold. It is helpful for testing various schedules to maximize process efficiency. And it can be used to analyze the effect of processes on the internal and external environment.

☞ For information about discrete rate modeling in general, including how it differs from continuous and discrete event modeling, see “Modeling methodologies” on page 43.

### What this chapter covers

- Discrete rate application areas
- Simulating discrete rate systems
- Blocks for doing rate-based modeling
- An introduction to some important discrete rate concepts:
  - LP technology
  - Layout of a discrete rate model
  - The Executive block
  - Connectors and connections
  - Flow units and unit groups
  - Flow rates: constraining, effective, infinite, and potential
- How the Discrete Rate module is organized


### Discrete rate application areas

Discrete rate simulation is used in two diverse areas:

- To model commodities that would normally be considered “stuff” rather than “things”, for example powders or liquids, gases and other fluids in the following areas:
  - Petrochemical
  - Manufacturing

- Mining
- Water Treatment
- Pharmaceutical
- Metallurgy
- Electric power transmissions
- Any other industry that processes commodities in bulk or batches
- To model “things” that are so numerous that it would be inconvenient or overwhelming to model them individually:
  - Food and beverages (tea bags, cereals, soda cans, cheese)
  - Drugs, cosmetics, and biotech (pills, bottles of lotion)
  - Milling (carpet and paper)
  - Data storage and manipulation (samples, messages, packets)
  - Any other industry that mixes, fills, or packages products on high-volume or high-speed lines

Thus discrete rate models simulate flows that are either homogeneous (identical goods that are the same throughout and do not vary in essential characteristics) or heterogeneous (numerous items that are clearly distinct, but cannot be easily sorted or separated).

 As will be seen in the Discrete Rate Tutorial, it is common for discrete rate models to also include portions that are discrete event processes.

## Simulating discrete rate systems

Discrete rate modeling takes a very different approach compared to continuous or discrete event modeling.

### Comparison to discrete event and continuous modeling

The Value library blocks in continuous models and the Item library blocks in discrete event models act individually and independently to calculate values or move items. They may send messages and communicate with each other, but there is no overall global connection between the blocks in those types of models.

By comparison, Rate library blocks are dependent on each other and have an effect on one another. Discrete rate models are divided into areas where the included blocks are not independent but instead are part of a global system. The blocks within each area communicate through an internal linear program (LP) that provides the global oversight for that area. Each block in an LP area contributes a part of the LP equation for the area; the result of the LP calculation is the effective rate for that part of the model. This system is optimized such that, if a particular area does not need to recalculate, it won't.

Another major difference is how items move in a discrete event model compared to how flow moves in a discrete rate model.

- In a discrete event model, items move from one block to another instantaneously. An Item library block might hold an item for some simulated period of time, but there is no constraint on the movement of items between blocks and that movement is instantaneous.
- By contrast, the movement of flow in a discrete rate model must take some time. In the absence of any constraints, the rate of flow would approach infinity and the flow would move instanta-

neously from one point in the model to another; this is never correct. For this reason, flow movement must be constrained by rates and conditions that are built into the model, and a lack of appropriate constraints is a modeling error that will stop the simulation run.

### Discrete rate models

Discrete rate models are concerned with flows, constraints, rates, events, storage capacity, and routing.

- Flow is what is stored in and moves through a discrete rate system. Flow can be almost anything, as long as it is not important that specific properties of each part of it be directly identified. For instance, liquids, electronic transactions, and cereal can all flow in a model. Flow is expressed and measured in flow units – either generic units or defined units such as packets, gallons, transactions, boxes, etc.
- Flow moves through flow connections in one direction, from one block's outflow connector to another block's inflow connector. It moves at a rate that is expressed as a quantity of flow per time unit – the number of packets per second, gallons per minute, boxes per hour, and so forth.
- The discrete rate architecture maximizes the movement of flow. Unless limited in some manner, flow would approach infinity and overwhelm the system. Because the discrete rate architecture maximizes the movement of flow, every model must contain one or more constraints (typically a Valve) to limit the rate of flow to something less than infinity. Some examples of constraints include the presence or absence of flow in a Tank, the maximum flow rate defined by a Valve block, and the rule chosen to distribute flow in a Diverge block.
- While constraints determine the maximum rate that flow can move, the effective rate is the actual rate of movement. The effective rate for each section of the model is determined using linear programming (LP), given the set of constraints that has been defined by the model's structure. The model's set of effective rates define how fast flow actually moves from one section of the model to another. As the simulation clock advances from one event to the next, the quantity of flow which has moved is updated.
- The state of a discrete rate model changes only when an event occurs. An event might be a Tank that becomes empty or full, a maximum rate that changes during the run, a block that changes its output proportions, and so forth. Each time an event occurs, ExtendSim makes a calculation to determine, at that moment, what the effective rates are in each part of the model. Any portion of the model that can potentially be impacted by the new event has its effective rates recalculated. This takes into consideration the constraints put on the rates, the location of the flow, and storage capacity in the system.
- Each discrete rate model is conceptually divided into unit groups, rate sections, and LP areas. These divisions are handled automatically and internally, and are determined by the type of blocks used in a model, how the blocks are connected, the settings in the blocks, and so forth. Unit groups are introduced on page 271, rate sections are described on page 303, and LP areas are discussed on page 306.

A connection between two rate-based blocks can thus be viewed as an infinitely small pipe that is always full of something at a constant pressure – as soon as the effective rate is more than 0, the pipe's contents move at the highest rate possible based on all the constraints given by the system. When the effective rate is 0, the pipe is still full but the flow instantly stops.

While the ExtendSim discrete rate architecture preserves mass balance in the system, no explicit consideration is given to pressure, energy, momentum, or temperature, since these are beyond its scope.

## Blocks for building discrete rate models

The blocks in the Rate library are optimized for creating discrete rate models. In addition, you can build custom discrete rate blocks using the ExtendSim development environment.

### Rate library

The Rate library allows you to simulate a wide range of flow systems by connecting blocks together and entering parameters. The complexity of calculating the effective flow rate and the generation of events that dictate a new rate calculation are handled within the blocks, alleviating the need to do any programming in the ModL language.

The blocks in the Rate library can be categorized as follows:

- Some blocks hold and provide flow
- Other blocks impact the effective rate of the flow
- The remaining blocks are for routing flow

These blocks are optimized for modeling anything flowing through a system. They incorporate concepts like constraints, goals, flow prioritization, mixing, batching, unbatching, level indicators, and so forth. The blocks have been designed to meet most rate-based flow needs so you can quickly and easily perform complex high-volume/high-speed modeling tasks.

As mentioned in the Tutorial module, discrete rate models can use continuous blocks from the Value library for data management and model-specific tasks. Using Value blocks with Rate library blocks does not change the fundamental architecture of discrete rate models; they will still be event-based rather than use the time-based architecture of continuous models. Discrete event blocks from the Item library can also be used in discrete rate models; they are helpful for representing entities such as tankers, airplanes, people, and so forth that interface with flow.

See “Rate Library Blocks” on page 731, “Item Library Blocks” on page 723, and “Value Library Blocks” on page 715 for a listing and brief description of the blocks in those libraries.

### Creating custom discrete rate blocks

Because of the Rate library’s extensive capability, it is not likely that you would need to program your own discrete rate blocks. If you do want to do this, it is important to note that discrete rate blocks use different data structures and programming methods than continuous or discrete event blocks. It is suggested that you start with an existing discrete rate block as a base, using a *copy* of a Rate library block similar to the one you want to build. Read the Developer Reference before modifying discrete rate blocks so you have a better understanding of how those blocks work internally.

## Terminology and architecture

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

### LP technology

To provide global oversight to calculate the effective rates in a discrete rate model, ExtendSim uses linear programming (LP) technology. The purpose of the LP calculation is to determine the maxi-

Bias
Catch Flow
Change Units
Convey Flow
Diverge
Interchange
Merge
Sensor
Tank
Throw Flow
Valve

Rate library blocks

imum effective flow rates in the system given the constraints defined by block settings and the structure of the model. After all the rules for storage capacity and movement have been declared in the model, ExtendSim uses the LP calculation to cause as much flow as possible to move through the system. This calculation is handled automatically and internally. For more information, see the advanced topic “LP technology” on page 376.

### Layout of a discrete rate model

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

Other than the Executive block, you can place the blocks in a model anywhere you want, remembering that ExtendSim evaluates discrete rate blocks along the path of the connections.

- ☞ Since ExtendSim will always try to maximize the flow, causing the rate of flow to approach infinity in the absence of any constraints, it is important to place upper limits on the flow at strategic locations throughout the model. Otherwise, the flow would move instantaneously from one part of the model to another; this would be a modeling error.

### Executive block

The Executive block (Item library) does event scheduling and makes the LP calculation for rate-based models. It must be present in every discrete rate model and it must be placed to the left of all the other blocks in the model.



Executive

In addition to the information discussed on page 93, the Executive plays a special role in discrete rate simulations. The block’s Discrete Rate tab allows you to set global options for discrete rate models, manage quantity units, and select advanced options for specific Merge and Diverge modes. Its LP Solver tab has information about the linear program (LP) that provides global oversight for discrete rate models.

The settings in the Discrete Rate tab are explained fully on page 364.

- ☞ For most purposes you will not need to change the settings in the Executive block.

### Connectors and connections

The Rate library provides blocks for simulating rate-based flows. Most of the blocks in the Rate library have flow connectors and value connectors; the Interchange block also has item connectors.:

Connector type	Line type
Flow	
Value	
Item	

- In a discrete rate model, flow connectors report the effective rate of the flow at each event. The flow moves in one direction, from one block’s flow output (“outflow”) connector to another block’s flow input (“inflow”) connector.

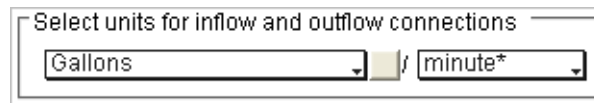
- Value connectors provide information about the quantity of flow and a block's capacity, as well as information about the effects that the flow has in the model.
- The item connectors on the Interchange block provide an interface between portions of the model that are discrete rate and portions that are discrete event.

When combining discrete rate blocks with blocks from other libraries, you will only be able to connect compatible connectors. To represent the flow from one block to another, an outflow connector has to be connected to an inflow connector. Each flow connector can have one and only one flow connection. However, it is possible to connect an outflow connector to both an inflow and a value input connector. In this case the value connector reads the effective rate from the connection. A flow connection cannot be made with an item connector. For more information, see "Connector types" on page 498.

### Units and unit groups

There are four types of units in the Rate library:

- *Flow units* indicate what is flowing from one flow connector to another. For instance, gallons, bottles, and transactions are all types of flow units. Each discrete rate block in a discrete rate model has a flow unit. The flow units are identical for all the blocks within a unit group (defined below).



Flow units

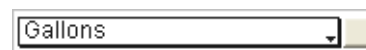
- *Time units* define how time is measured as the model run progresses. Like other ExtendSim blocks, blocks in the Rate library can use the default global time unit or a local time unit. Hours, minutes, and seconds are all examples of time units.



Calendar dates are not available if months or years have been selected as the specific global time unit for a discrete rate model. Furthermore, if Calendar dates has been selected, Rate library blocks will not be able to select Months or Years as their local time unit.

- *Length units* specify how long something is; they are usually entered as feet, meters, and so forth. The Convey Flow block has a length unit.
- *Block units* are an internal unit of volume specific to the Tank and Interchange blocks. If you select a block unit that differs from the flow units that come into and out of a block, you must enter a conversion factor. The conversion factor represents the ratio of the block unit to the flow unit.

A *unit group* is a collection of blocks connected together through flow connections and sharing one flow unit. To see the unit group, click the grey square to the right of a block's flow units popup menu. All the blocks in that block's unit group will be highlighted on the model worksheet.



Unit group selector to right of popup

For more information, see "Units and unit groups" on page 297.

### Rates

One of the most important aspects of a discrete rate model is the rate of flow. A rate is the ratio of the flow units to the model's time units. This is displayed in block dialogs as units/time, gallons/minute, transactions/second, boxes/hour, and so forth.

Several different types of rates are considered during the model building process:

- Maximum rate – the upper limit of the rate of flow, as described on page 303.
- Effective rate – the actual rate of flow. See page 303 for more information.
- Upstream supply and downstream demand – potential rates. See page 382.
- Infinite rate – any value equal to or greater than a large specified number. See page 304 for more information.

### How the Discrete Rate module is organized

The discrete rate portion of the User Guide shows how to build models to simulate rate-based flows moving through a system at a certain speed. It will show you how to design and document a rate-based model, run the simulation, test different scenarios, and analyze the results. The Discrete Rate module is divided into several chapters:

- Introduction
- Tutorial
- Chapters that discuss specific discrete rate modeling concepts and techniques:
  - Flow sources, storage, and units
  - Flow movement: rates and constraints
  - Routing flow directly and remotely
  - Delaying flow using goals, hysteresis, and the Convey Flow block
  - Mixing flow and items
  - Bias, animation, and other miscellaneous concepts and features
- Advanced topics such as LP technology, upstream supply and downstream demand, and messaging in discrete rate models.



It is important that you complete the chapters in the main ExtendSim Tutorial module that starts on page 14 before you proceed to the Discrete Rate Tutorial. If you will use any item-based blocks in your discrete rate models, it is also suggested that you complete the Discrete Event Tutorial that starts on page 100.



# **Discrete Rate Modeling**

## **Tutorial for Discrete Rate Systems**

How to build a discrete rate model

The key to discrete rate modeling is constructing a flow diagram using blocks from the Rate library to represent flows through the system. The Rate library is designed specifically for building discrete rate models. Blocks from other ExtendSim libraries, especially the Item, Plotter, and Value libraries, are often used with the Rate library to create discrete rate models.

The example in this chapter shows how to build a discrete rate model of a yogurt process; it will use many of the blocks from the Rate library. Starting with a simple model, then adding complexity and features, this chapter will show how to:

- Build a model of a simple rate-based process
- Add a maximum flow rate that varies with the time of day
- Add a second supply of product that is occasionally shut down for maintenance
- Mix the two supplies according to a proportion
- Create a filling operation that puts the liquid yogurt into containers
- Add a conveyor to simulate a cooling process
- Package the containers into cartons
- Create a palletization area where the cartons are stored
- Add a second palletization area in parallel to the first

While this example model simulates a mixing, filling, and packaging process, the Rate library is useful for simulating many diverse concepts and processes.



This tutorial assumes you have completed the chapters in the main Tutorial module that starts on page 14 and that you have read the Discrete Rate Introduction that starts on page 266. It is also suggested that you complete the Discrete Event Tutorial that starts on page 100.

### A basic discrete rate model

Rate-based models are mainly concerned with how quickly the flow moves in different sections of the model, and what the yields will be, given the constraints and configurations of the model. A common rate-based model involves a flow of product moving from one holding area to another, with a valve that determines how quickly the flow moves.


#### About the model

The Yogurt Production model represents a process that takes a supply of liquids, converts it into plain yogurt, then mixes the yogurt with fruit. The fruited yogurt mixture is poured into individual yogurt containers and cooled and the containers are then packaged into cartons. The final step is to place the cartons on pallets for storage.

The assumptions for the final model are:

- The supply of liquid to make the yogurt comes from one location and the fruit comes from another location. Both locations have an infinite supply.
- For most of the day, the liquid is processed into yogurt at a rate of 100 gallons/minute. Since fewer workers are available during lunch, the processing rate decreases to 60 gallons/minute for that hour.
- After processing, the yogurt is routed to the mixing area.
- The fruit is processed and delivered to a mixing area at a rate of 8 gallons per minute when equipment is not undergoing maintenance, and a rate of 2 gallons per minute when it is.
- Each 10 gallons of mix is composed of 1 gallon of fruit and 9 gallons of plain yogurt.

- The packaging process yields 12 containers of yogurt per gallon of mix.
- The cooling cycle occurs on a 100 foot long refrigeration unit. The yogurt must be cooled for at least 20 minutes before the containers can be packaged into cartons.
- Each carton holds 48 containers of yogurt
- There are two palletizing areas, one of which has a higher priority than the other.
- Pallets arrive every 2 minutes and can hold 24 cartons.
- Time units are in minutes and the simulation duration is 480 minutes.
- The blocks come from the Rate, Item, and Plotter libraries.

 The final Yogurt Production model, and the models that illustrate the steps described in this chapter, are located in the folder \Examples\Tutorials\Discrete Rate. To get the maximum benefit of this tutorial, it is recommended that you build the models yourself.

### Starting a model and setting simulation parameters

The following steps are typical when starting any discrete rate model:

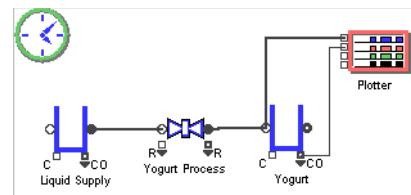
- ▶ Open a new model worksheet
- ▶ Give the command Run > Simulation Setup. In the Setup tab enter the simulation parameters:
  - ▶ **End time: 480**
  - ▶ **Global time units: minutes**
- ▶ If they aren't already open, open the Rate, Item, Plotter, and Value libraries
- ▶ Place an **Executive block** (Item library) on the top left corner of the model worksheet

As mentioned in the Introduction to this module, the Executive block does event scheduling and manages discrete rate and discrete event simulations. It must be present in every discrete rate and discrete event model.

### Start small

In building any simulation model, it is easiest to start with a simplified subset of the process and add detail until you arrive at a completed representation of the system that's being modeled. This allows you to test at various stages while making the model building process more manageable.





The first step is to model a single line of production, where one Tank holds product that moves to another Tank at a constant rate. When you have finished this portion of the tutorial, your model should look like the one shown above.




Basic yogurt production line

**Creating a model of the simple yogurt production process**

The following table lists the blocks that will be added to the worksheet and their use in the model. Except for the plotter from the Plotter library, the blocks in the table are from the Rate library.

Name (Label)	Block Function	Purpose in Model
Tank (Liquid Supply) 	Acts as a source, intermediate storage, or sink for the stuff of the model.	Contains an unlimited amount of liquids that can be processed into yogurt.
Valve (Yogurt Process) 	Acts as a constraint on flow. Controls, monitors, and transfers the flow at a specified rate.	Regulates the flow of liquid at 100 gallons per minute. (A constraint is required; otherwise, the flow would approach infinity!)
Tank (Yogurt) 	Acts as a source, intermediate storage, or sink for the stuff of the model.	Can hold an unlimited amount of processed yogurt.
Plotter, Discrete Event (Plotter) 	Displays information about the flow and about model values.	Reports how many gallons of yogurt per minute are processed (the effective rate) and the total amount of yogurt produced.

- ▶ Starting at the right of the Executive block, place the blocks on the model worksheet in a line from left to right, based on their order in the above table.
- ▶ Label the blocks as indicated in the table.

 An easy method for placing blocks on a model worksheet is to access an open library using the Navigator as discussed in “Library Window mode” on page 671.

**Making connections**

To indicate the flow of product, connect the blocks’ flow connectors as follows:

- ▶ Connect from the outflow connector on the first Tank (labeled Liquid Supply) to the Valve’s inflow connector.
- ▶ Connect from the Valve’s outflow connector to the second Tank (labeled Yogurt).


To gather information about the amount of yogurt processed:

- ▶ Connect from the Yogurt tank’s inflow connector to the top input on the plotter.
- ▶ Connect from the Yogurt tank’s LE (level of contents) value output connector to the plotter’s second input.

When you are finished, the model should look like the one shown on page 275.

**Entering dialog parameters and settings**

To reflect the basic assumptions for this model, the flow units and the constraints need to be defined.

 Each model must contain one or more blocks (typically a Valve with a non-blank maximum rate) to restrict flow. The ExtendSim discrete rate architecture attempts to move flow through the model as fast as possible. In the absence of any constraints, the flow rate would theoretically approach infinity and flow would move from one part of a model to another instantaneously. This condition would cause ExtendSim to stop the simulation and display an error message.

▶ For the Liquid Supply tank:

- ▶ In its Tank tab, check the ∞ (infinite) checkbox for the field labeled **Initial contents**. This places the word “infinite” in the initial contents field, as seen at right.

Initial contents:	infinite <input checked="" type="checkbox"/>	gallons
Capacity:	infinite <input checked="" type="checkbox"/>	gallons

Initial contents set to “infinite”

- ▶ In the block’s Options tab, select **New Unit** in the popup menu for **Flow group unit**: and enter **gallons** as the flow unit.
- ▶ In the dialog of the Valve block, enter **Maximum rate: 100 gallons/minute**.
- ▶ There are no entries to make for the Yogurt tank. Its default settings indicate that it has no initial contents and its maximum capacity is infinite, which is what you want.

In this model, the plotter will display the Valve’s maximum rate on its top input and the number of gallons of yogurt processed on the second input. Since the scaling for these numbers is so different, the plotter’s graph needs to be adjusted. To do this:

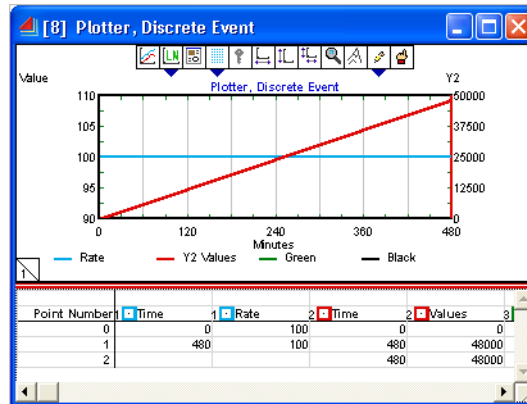
- ▶ In the plotter dialog’s toolbar, open the Trace Properties dialog.
  - ▶ Choose that the second input (Values) is plotted against the Y2 axis.
  - ▶ Change the style of the line for that second input to interpolated.

(For information on how to change the properties of lines, see “Trace properties tool” on page 590.)

- ▶ Save the model and run the simulation.

### Verifying results

This is a good opportunity to verify the results. There is never any change to the rate of flow, so there is no need for the model to recalculate the effective rate. This means that the simulation is finished in two events: the start event and the end event. The plotter indicates that the effective rate is 100 and a total of 48,000 gallons of product (shown on the Y2 axis) have been produced in the process. This makes sense because the assumptions were that 100 gallons of yogurt would be produced per minute and the simulation time is 480 minutes.



Plot of simulation

### Add a dynamic constraint

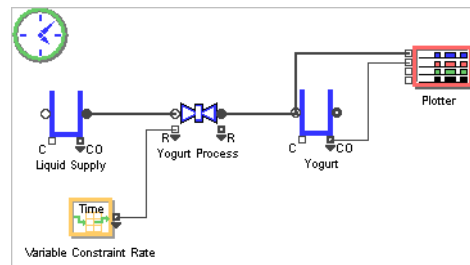
While the preceding model used a constant maximum rate, a more common situation is for a Valve's maximum rate to change with time. The model assumptions are that the liquid is processed into yogurt at a rate of 100 gallons per minute for most of the day, but that during the lunch period the rate is reduced to 60 gallons per minute. To show this in the model:

- ▶ Add a Lookup Table block (Value library) to the model.
- ▶ Connect the Lookup Table block's output to the **R** (maximum rate) input connector on the Valve.
- ▶ In the Lookup Table block's dialog, select **Lookup the: time**.
- ▶ In the block's Options tab, enter the column labels **Minute;Gallons/Minute**.
- ▶ In the block's Table tab, enter the values shown at the right for the Minute and Gallons/Minute columns. This will cause the Valve's maximum rate to be 100 gallons/minute for the entire model except for the period from time 240 to time 300, when it will be 60 gallons/minute.
- ▶ Label the block **Variable Constraint Rate**.

	Minute	Gallons/Minute
0	0	100
1	240	60
2	300	100
3	480	100

Values for Lookup Table's dialog

The model should now look like the screenshot at the right. When the simulation is run, the plot shows that the Valve's maximum rate (from the Lookup Table block) is 100 gallons/minute for most of the day, but changes to 60 gallons/minute for 60 minutes, as expected. It also shows that the yogurt output is reduced from the 48,000 gallons achieved in the previous model. This occurs because the Valve's maximum rate reduced from the constant 100 gallons/minute in the previous model.



Adding a variable constraint

The Lookup Table block will actively output values at each specified time, based on entries in the table in its dialog, and does not need to be prompted for output. This is discussed in “Polling constraints” on page 309.

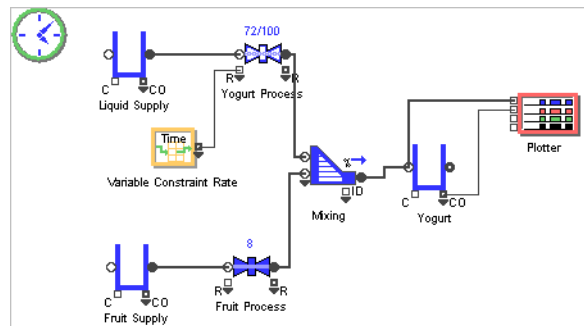
### Add a fruit processing line

So far you have created a yogurt processing line. The specification for the final model indicates that fruit is mixed with the yogurt, requiring a second processing line:

- ▶ Delete the connection from the Valve to the Yogurt tank and move the tank to the right.
- ▶ Below the Lookup Table block of the yogurt processing line, add another Tank block to the model.
  - ▶ In the block's Tank tab, check the checkbox in the field labeled **Initial contents**, causing the initial contents to be infinite.
  - ▶ In the block's Options tab, select **Flow group unit: gallons**.
  - ▶ Label the block **Fruit Supply**.
- ▶ Add a Valve to the right of the Fruit Supply tank.
  - ▶ Connect from the Fruit Supply tank's outflow connector to the Valve's inflow connector.
  - ▶ In the Valve tab, enter **Maximum rate: 8 gallons/minute**.
  - ▶ Label the block **Fruit Process**.

Notice that *gallons* have automatically been selected as the flow units for the Valve. This is because it is connected to the Tank. (If you had not first connected the Tank to the Valve, you could select gallons as the units in the Options tab.)

- ▶ To merge the two flows, add a Merge block to the model and place it to the right and between the Yogurt Process and the Fruit Process blocks.



Fruit process line added

- ▶ Connect from the outflow connector of the Yogurt Process Valve to the top inflow connector on the Merge.
- ▶ Connect from the outflow connector of the Fruit Process Valve to the second inflow connector on the Merge.
  - ▶ In the Merge tab of the block's dialog, select **Converge mode: proportional** from the popup menu.
  - ▶ In the Proportion column of the Merge block's dialog table, enter **9** for the Yogurt Process and **1** for the Fruit Process.
  - ▶ Label the Merge block **Mixing**.
- ▶ Connect the Merge block's outflow connector to the Yogurt tank's inflow connector.

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

When you run the model, the mixing process should output 80 gallons per minute and the entire process will yield about 37,600 gallons of yogurt. This is an interesting model to run with animation on. (Be sure to have animation set to the slowest speed.) When you do this, the rate displayed at the top of the Yogurt Process icon is sometimes displayed as the fraction  $72/100$ . This is the ratio of the effective rate to the Valve's maximum rate. In this model, there is sometimes not enough fruit and the entire process becomes constrained, so the effective rate can be less than the specified maximum rate.

In validating the model, notice that the effective rate for the yogurt part of the process can never be higher than 72. Since the maximum output of the fruit process is 8 gallons per minute, and the mixing process requires a ratio of 9 portions of plain yogurt to 1 portion of fruit, the maximum amount of plain yogurt that can be required is 72 ( $8 \times 9$ ) gallons per minute.

 A complete description of the animation information shown on the icons for Rate library blocks is given on "Animation" on page 370.

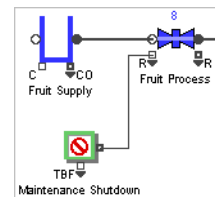
### Add maintenance

A common situation is for a process to have a slow production rate when some of the equipment is down for maintenance and a faster rate the rest of the time.

The fruit process has a slow rate of 2 gallons per minute during equipment maintenance and a normal rate of 8 gallons per minute. Maintenance occurs approximately every 60 minutes with a random duration of a minimum of 5 minutes, a maximum of 20 minutes, and a mostly likely time of 15 minutes. To reflect this:

▶ Add a Shutdown block (Item library) to the model. There are two ways to do this:


- ▶ Click the **Add Shutdown** button in the Valve's dialog. This automatically connects a Shutdown block to the Valve's R (maximum rate) input connector and opens the Shutdown's dialog.
- ▶ Place the Shutdown block on the worksheet from the Rate library. If you do this, connect the output of the Shutdown block to the Fruit Process Valve's R (maximum rate) value input connector.



Shutdown block added

▶ In the Shutdown block's dialog:

- ▶ For the shutdown configuration, enter **Down value: 2** and **Up Value: 8**
- ▶ For the time between failures (TBF) choose an **Exponential** distribution with a **Mean: 60**
- ▶ For the time to repair (TTR) select a **Triangular** distribution with a **Minimum: 5**, **Maximum: 20**, and **Most likely: 15**
- ▶ Label the block **Maintenance Shutdown**.

 Like the Lookup Table block, the Shutdown block outputs its information without being prompted. Thus the Valve does not need to ask it for data and it should not be set to poll constraints. Polling constraints is discussed more on page 309.

When you run the model with animation on, notice that the Valve is partially shut down for maintenance a random amount of time and that its maximum rate is reduced to 2 gallons/minute during maintenance. Also notice that fewer gallons of yogurt are produced during the process than before the Shutdown block was added.

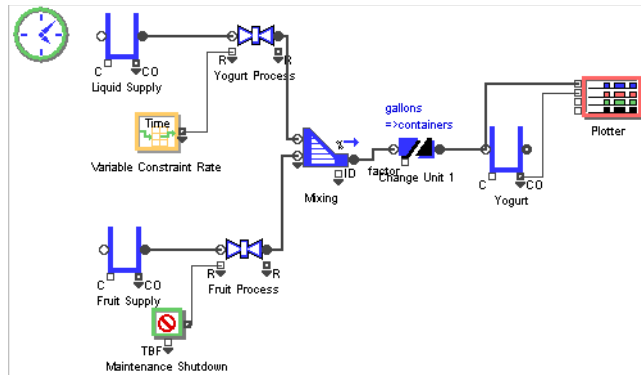


### Change the flow unit to containers for the filling process

It would be difficult to ship the processed yogurt without putting it into containers. The model assumptions state that the yogurt mix is packaged into containers at a ratio of 12 containers per gallon of liquid. To represent this:

- ▶ Delete the connection between the Merge block and the Yogurt tank.
- ▶ Add a Change Units block to the model and connect it between the Merge and the Yogurt tank. Label the block **Change Unit 1**.

The model should now look like the screenshot to the right.



Changing the flow unit

- ▶ In the dialog of the Change Units block:
  - ▶ Do not change the first setting (*Change units from: gallons*)
  - ▶ In the popup for the second unit setting (**to: gallons**), select **New Unit** and name the new flow unit **containers**.
  - ▶ So that each gallon will result in 12 containers, enter **Conversion factor: 12 containers/gallons**. (Be sure to select **containers/gallons** from the popup menu.)
  - ▶ If you check the box for **Show unit change on icon** in the Change Units dialog, the area above the icon will display the text *gallons=>containers*.

After the settings have been entered, the dialog of the Change Units block should look like the following:

Dialog of Change Units block with user entries

When the model is run, the number of yogurt containers produced each minute, and thus the total number of containers processed, will vary depending on the Valve's maximum rate and the flow restrictions caused when there is not enough fruit for the mix. Although it may vary from the example model, the plotter should indicate that approximately 370,000 containers of yogurt were produced. This makes sense because, as the Results tab in the Merge block shows, the process produced about 30,000 gallons of yogurt.

At the end of the simulation run, holding the cursor over each outflow connector will show the final rate at that connector. For example, when the Yogurt Process has an effective rate of 72 gallons per minute, the Fruit Process will have a rate of 8 gallons per minute, the Mixing block will

282 | **Tutorial for Discrete Rate Systems**  
A basic discrete rate model

indicate that 80 gallons of mix were produced that minute, and the Change Units block's output will show that 960 ( $80 * 12$ ) containers of yogurt were packaged.

☞ This model assumes that the process of pouring yogurt into containers occurs at the same rate as the process of mixing the plain yogurt with the fruit. In this case, the packaging process does not slow down the rest of the process. To model a packaging process that would have an impact on the rest of the process, connect a Valve between the Merge and Change Units blocks and enter the appropriate packaging rate.

### Cool the mixture

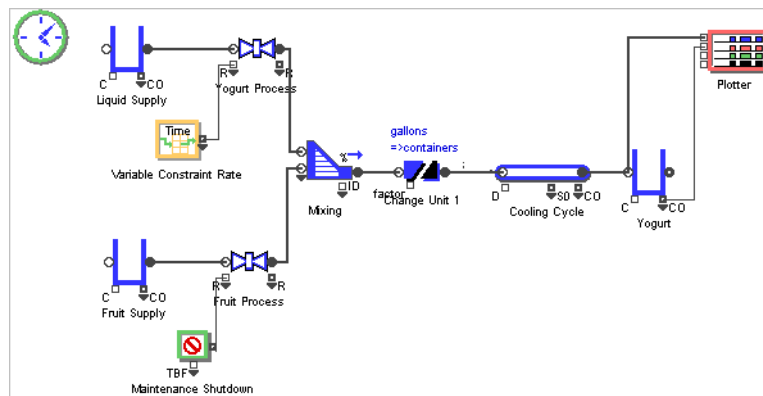
The yogurt process includes a 20 minute cooling phase on a 100-foot long refrigeration unit before the yogurt containers can be packaged into cartons. The Convey Flow block is designed to represent a delay in the movement of flow.

- ▶ Delete the flow connection from the Change Units block to the Yogurt tank.
- ▶ Add a Convey Flow block to the model and connect it between the Change Units block and the Yogurt tank.

Notice that if you make these connections first, the correct flow unit (containers) is automatically selected in the new block's Options tab.

- ▶ In the Options tab of the Convey Flow block:
  - ▶ Select **New Unit** from the **Length unit: length unit** popup menu. In the dialog that appears, enter **feet**.
- ▶ In the Convey tab of the Convey Flow block:
  - ▶ Notice that the block is already set to *Accumulating-maximum density* by default.
  - ▶ From the popup menu to the right of that popup, select the behavior **Delay determines travel time**.
  - ▶ Enter **Delay: 20 minute\***.
  - ▶ Enter **Maximum density: 500 containers/feet**.
  - ▶ Label the block **Cooling Cycle**.

The model should now look like the following:



Cooling phase represented by Convey Flow block

When you run this model, notice that there is no product flowing into the Yogurt tank for the first 20 minutes. This happens because it takes that long for the first containers to leave the refrigeration unit represented by the Convey Flow block. This causes *starving* in the downstream portion of the model that follows the Convey Flow block. Because it takes longer to get the finished product, fewer containers get produced than in the previous model.

- ☞ Although it doesn't happen in this model, if the Convey Flow block were full, it could slow down the upstream processes that feed into it. This is known as *blocking*.
- ☞ While the use of the Convey Flow block is appropriate for this model, be careful about placing too many Convey Flow blocks in a model as they are computationally intensive. The Convey Flow block should only be used if the system requires precise tracking of flow movement. A Valve and a Tank can often be used instead, with less impact on simulation speed. For more information, see "When to avoid using the Convey Flow block" on page 346.

### Package the containers

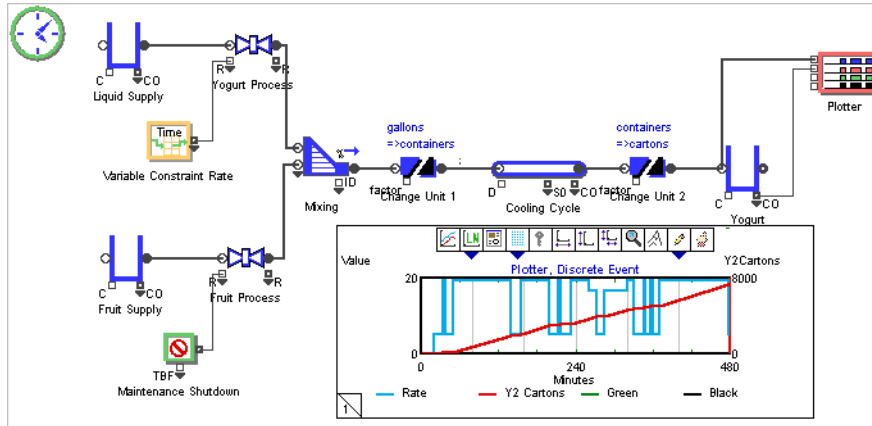
The next step of the process involves packaging the yogurt containers into cartons. This is represented using another Change Units block.

- ▶ Delete the flow connection between the Cooling Cycle and the Yogurt tank.
- ▶ Add a second Change Units block to the model;
  - ▶ Connect it between the block labeled Cooling Cycle and the block labeled Yogurt.

▶ In the block's dialog, do not change the first setting (*Change units from: containers*).

- ▶ In the popup Dialog of second Change Units block for the second unit setting (**to: containers**), select **New Unit** and name the new flow unit **cartons**.
- ▶ So that each carton will hold 48 containers, enter **Conversion factor: 48 containers/cartons**.
- ▶ If you check the box for **Show unit change on icon**, the area above the icon will display the text *containers=>cartons*.
- ▶ Label the new block **Change Unit 2**.

If you clone the plot pane onto the model worksheet, your model should be similar to the following:



Packaging containers into cartons

Although the amounts will vary depending on the yogurt process's constraint rate and the potential unavailability of fruit due to maintenance, when you run the model it should result in approximately 8,000 cartons. (Each carton holds 48 containers and the process should have produced about 390,000 containers of yogurt.)

### Add a palletizing area

The model's assumptions state that there is a palletizing area where cartons are stored for shipment. One empty pallet arrives every 2 minutes and each pallet can hold 24 cartons. If there is no empty pallet to replace it, the flow of yogurt stops when a full pallet leaves. As discussed below, this is easily represented using the Rate library's Interchange block and discrete event blocks from the Item library.

#### Interchange block

The Interchange block represents a tank, or holding area, where the flow of discrete rate blocks can interact with items from discrete event blocks. The block can only get one item at a time. In its default behavior (*Tank only exists while item is in it*), the block behaves similar to an on/off switch. When it has an item, it has a capacity for flow; in the absence of an item, the block has no flow capacity. (In the block's alternate behavior, the tank has capacity and items that come to it can contribute flow to the block and/or remove flow from it.)

In this model, empty item/pallets are generated randomly. The arrival of an item/pallet causes the Interchange block to have flow capacity; the maximum capacity of 24 cartons is entered in the block's dialog. Once the maximum capacity is reached, the full item/pallet leaves the block. The Interchange block then has no capacity until another empty pallet arrives.

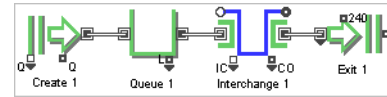
The Interchange block is discussed more fully in "Using the Interchange block to mix items with flow" on page 352.

#### Adding a palletizing area to the model

- ▶ Delete the Yogurt tank.

► Place the following blocks in the model and connect their item connectors as shown at right:

- Create (Item library)
- Queue (Item library)
- Interchange (Rate library)
- Exit (Item library)



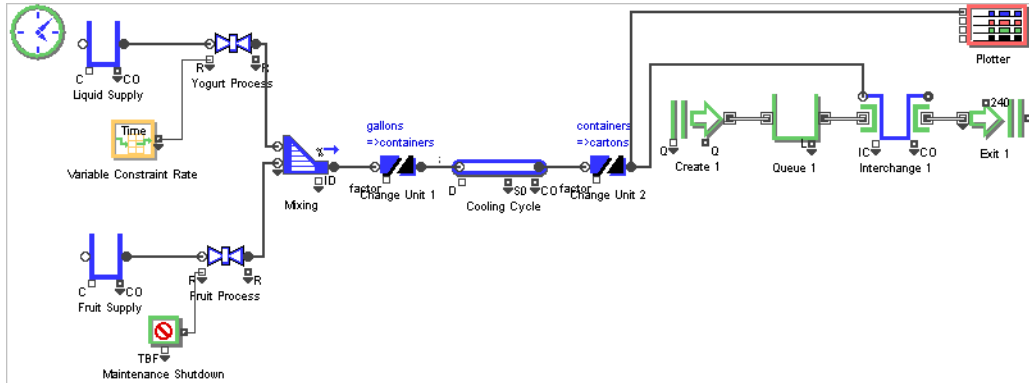
First palletizing area

- Label the 4 new blocks as indicated in the screen shot above.
- Connect from the outflow connector of Change Unit 2 to the plotter's top input connector.
- Connect from the outflow connector of Change Unit 2 to the inflow connector on the Interchange 1 block.
- In its dialog, set the behavior of the Create block to **Create items randomly**, choose the **Constant** distribution, and enter **constant: 2**. This causes one item/pallet to be available every 2 minutes.
- There are no changes required for the Queue block. It is already set to hold items in a first in, first out manner.
- In the Item/Flow tab of the Interchange block, define item behavior (Item is Tank) by making the following entries:
  - **On arrival, Item/Tank capacity is: a constant 24 cartons.** (Be sure to enter the number 24.)
  - **Release item: when tank contents  $\geq$  Level (load process).**
  - **Define Level: full.**

With these settings, the Interchange block has a capacity of 24 cartons each time it gets a new item, which represents an empty pallet. Once 24 cartons of yogurt have arrived through its flow connector, the block's Level will be full and the item (now representing a full pallet with 24 cartons) will be released. The Interchange block will then try to access another empty pallet; the yogurt process will stop until an empty pallet is available.

- There are no dialog changes required for the Exit block.

When you are finished with this section, the model should look like:



First palletizing area

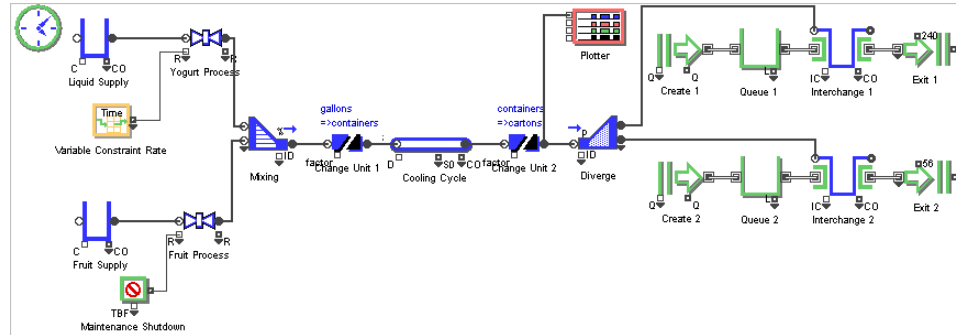
When you run this model, you should see an almost solid block of color on the plotter's plot pane. This is caused by the plot line being repeatedly redrawn as the effective rate goes from a high of about 52 cartons per minute to 0 and back again. To see this, stretch the plot wider until you can see some white areas between the colored areas. These white areas occur when the effective rate is 0. In this model, pallets aren't arriving quickly enough and the process is slowed from what it could be and frequently stops. One way to solve this would be to have two palletizing areas.

### Add a second palletizing area

The easiest way to add a second palletizing area is to duplicate the blocks in the first palletizing area to another part of the model.

- ▶ Select the 4 blocks in the palletizing area and give the Edit > Duplicate command. This creates a second set of blocks.
- ▶ Move the 4 blocks that comprise the second palletizing area below the first area.
- ▶ Label the new blocks **Create 2**, **Queue 2**, **Interchange 2**, and **Exit 2**.
- ▶ Add a Diverge block to the right of the Change Unit 2 block:
  - ▶ Connect from the Change Unit 2 block to the Diverge.
  - ▶ Connect from the Diverge block's top outflow connector to the inflow connector on the Interchange 1 block
  - ▶ Connect from the Diverge's second outflow connector to the inflow connector on the Interchange 2 block.
  - ▶ In the dialog of the Diverge, notice that by default the block is set to *Diverge mode: priority of outputs* and that a priority of 1 is assigned to the top Interchange block and a priority of 2 to the bottom Interchange block. Do not change these settings, since this is what you want.

The model should now look like:



Two palletizing areas

In this model, the top palletizing area has first priority for the cartons and the lower palletizing area only receives product if the top area is busy. Unlike the previous model, the plotter indicates that the effective rate of the process is hardly ever 0 cartons per minute.

The final model, named Yogurt Production, and the models for all the intermediate steps, are located in the folder Examples\Tutorial\Discrete Rate.

### Further exploration

Additional ways to enhance and explore this model include:

- Verify that the model is working as you expected by running it with animation on or by adding a Pause Sim block (Utilities library) to the model and pausing each step. (Animating a discrete rate model is described in “Animation” on page 370. The Pause Sim block is discussed in “Blocks that control or monitor simulation runs” on page 525.) The final Yogurt Production model located at Examples\Tutorials\Discrete Rate is animated and includes a Pause Sim block so you can step through each event.
- A Shift block could be added to the Convey Flow block, stopping the process at the end of the day and causing the Convey Flow block to be emptied of product.
- The palletization areas could include the time it takes to unload a full pallet and load an empty pallet. Adding this type of changeover is shown in the “Yogurt Changeover model” on page 355.
- The filling processes could be enhanced by adding delays for the filling, cleaning the equipment and other maintenance.





# **Discrete Rate Modeling**

## **Sources, Storage, and Units**

Providing and storing flow and the use of flow units

As discussed in “Simulating discrete rate systems” on page 267, quantities of flow are located in one or more parts of a discrete rate model. During the simulation run, the flow moves from one location to another at the effective rate. In order for the flow to move, one or more of the model’s blocks need to have the capacity to hold flow as time advances.

The Convey Flow, Interchange, and Tank blocks are residence type blocks – they have capacity and can hold defined amounts of flow. They can also be “pre-loaded” with an initial amount, serving as a source of flow for the system.

Flow units describe what is flowing from one Rate library block to another. Blocks that are connected together through flow connections and share the same flow unit are part of the same unit group. The Change Units block is used to create a new unit group. This causes the blocks downstream of the Change Units block to be in a unit group different from its upstream blocks.

This chapter discusses providing and storing flow and the use of flow units in a discrete rate model. It will cover:

- Defining a block’s flow capacity
- Setting an initial contents of flow
- Indicators that provide information about a block’s level of flow
- Defining and selecting time, flow, and length units
- Using the Change Units blocks to create a different flow unit group

This chapter focuses on setting capacity and initial contents for the Convey Flow, Interchange, and Tank blocks. Other aspects of those blocks are covered in different chapters:

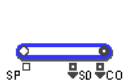
- The Convey Flow block is most often used for delaying flow and will be discussed more fully on page 342.
- The Interchange block is mainly used for interacting with items from discrete event portions of the model and will be discussed more completely starting on page 352.
- Setting maximum inflow and outflow rates for the Tank and Interchange blocks is described in page 310.

 The Tank Flow Unit model is located in the folder \Examples\Discrete Rate\Sources and Storage. The Yogurt Production model is located at \Examples\Tutorials\Discrete Rate.

## Blocks of interest

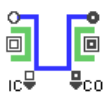
The following blocks from the Rate library will be the main focus of this chapter.

### Residence blocks for holding flow



#### **Convey Flow**

Delays the movement of flow from one point to another. Can accumulate flow to a maximum density, accumulate flow to fill empty sections, or act as a non-accumulating conveyor.



#### **Interchange**

Represents a Tank that can interact with discrete event items. The block has two behaviors: the Tank only exists while an item is in it; the Tank is separate from the item.



### Tank

Acts as a source, intermediate storage, or final storage (sink). The block has a capacity and can have an initial quantity of flow for the simulation.

### Changing the flow unit group



### Change Units

Changes the flow unit from one unit to another, resulting in a new flow unit group. The dialog has a field for entering the conversion factor and a popup menu for indicating the direction of the change.

## Capacity

The Tank, Interchange, and Convey Flow blocks are considered *residence* blocks. This means that they have capacity and can hold defined amounts of flow as time advances.

A residence block's maximum capacity can be a specific number or, in the case of the Tank or Interchange blocks, it can be set to infinite.

### Full and not-full

When a residence block's capacity is finite, its status can alternate between the full and not-full states. This change of state has an impact on the model's set of effective rates:

- If a residence block with finite capacity is not full, there is room for the flow level to rise. Consequently, the effective inflow rate can be greater than the effective outflow rate.
- If a residence block with finite capacity is full, the flow level is not permitted to rise; the effective inflow rate will be less than or equal to the effective outflow rate.

Any time a residence block with a finite capacity changes state between full and not-full, ExtendSim will calculate a new set of effective rates.

A residence block with infinite capacity can never be full during the simulation run. In this situation, it is similar to a residence block with finite capacity that is not full; its effective inflow rate can be greater than its effective outflow rate.

### Tank block's capacity

A Tank can be a source of flow, an intermediate storage for flow, or a final storage for flow (sink). A Tank's capacity can be infinite, a finite but non-zero number, or zero.

Capacity:   gallons

Default setting for Tank's capacity

- By default, the Tank has an infinite capacity to hold flow, as indicated by the *Maximum capacity: infinite*  $\infty$  setting in its dialog. In this state it will never be full.

A blank is the same as checking the infinite setting.

- A Tank's maximum capacity can be changed in the block's dialog by entering an amount in the *Maximum capacity* field (which unselects the  $\infty$  checkbox). It can also be changed dynamically through the block's *C* (capacity) value input connector. If the *C* connector is used, it overrides any entries made in the dialog. With a non-zero finite capacity, the Tank can be in either the full or not-full state at any point in time.

- If a Tank's capacity is set to zero, flow can still move through the block but the flow will not stay in the block for any length of time. In this case, the Tank is neither full nor not-full, and the effective inflow rate will equal the effective outflow rate.

☞ If a Tank has no outflow connection, by definition it is being used as a sink. If at some point the sink reaches the full state, its effective inflow rate will be set to zero for the remainder of the simulation run.

### Interchange block's capacity

The Interchange block represents a tank, or holding area, where flow can interact with items generated by discrete event blocks. Flow can enter the Interchange block not only through its inflow connector but also through the arrival of an item. Conversely, flow can exit the block through its outflow connector or through the exiting of an item.

The Interchange block has two options that affect how the block's initial contents and maximum capacity are set:

- *Tank only exists while item is in it.* This behavior is analogous to a truck (an item) that arrives at a loading dock (a tank) where the loading or unloading of product can take place at a certain rate. The truck arrives with a capacity and perhaps some quantity of product already in it. As long as the truck is in the dock, loading or unloading is possible and occurs at the specified rate. When the truck leaves, the dock's ability to load and unload product disappears (the inflow and outflow effective rates are set to zero).
- *Tank is separate from item.* This behavior is similar to a truck (an item) that brings product to a holding area (a tank) that may or may not contain product. The truck empties its load and perhaps takes some of the holding area's product with it when it leaves. This process occurs instantaneously. Whether the truck is at the holding area or not, both the truck and the holding area can have product. The holding area can receive or deliver flow to the system even if there is no truck (the inflow and outflow effective rates can be greater than zero).

Setting the maximum capacity with these options is described below.

#### **Tank only exists while item is in it**

This is the default behavior. With this setting the Interchange block has a capacity to hold flow only while an item resides in the block. With an item present, the block acts like a tank and therefore has a definable capacity, either finite or infinite. If the capacity is finite, as long as an item is present in the block, the block can alternate between the full and not-full states.

The block's capacity is fixed at the moment the item enters the block; it remains fixed until the item leaves. When the item leaves, the Interchange's capacity automatically goes back to zero. Therefore, at the time of item departure any flow currently in the block is loaded onto the item. The timing of when the item leaves the Interchange depends on logic set in the Interchange block. For more information, see "Item release conditions" on page 353.

To define the capacity for an Interchange block when it is set to this behavior, choose one of the options from the dialog's popup menu, shown at right.

Capacity (when item is present):	a constant
	infinite <input checked="" type="checkbox"/> units

Maximum capacity; default behavior

- *A constant.* Enter a number in the field; the default is infinite.
- *Value at IC.* The value at this input connector will control the block's capacity.

- *Value of attribute.* Select an attribute in the dialog. When an item arrives, the tank's capacity will equal the item's attribute value.

In the Yogurt Production model of the Discrete Rate Tutorial, the Interchange blocks are set to *Tank only exists when item is in it* and their capacity is set to the constant value 24.

### Tank is separate from item

With this option, the Interchange block's behavior is similar to a Tank block – it receives flow from its inflow connector, it holds flow, and it releases flow from its outflow connector. The difference is that an item's arrival can contribute flow to the existing contents and an item's departure can remove flow from the existing contents. The item's impact on the block's contents is entered in the *Define Item behavior* section of the block's Item/Flow tab.

To set the capacity for the block when this behavior has been selected, enter a number in the *Maximum capacity* field or leave it set to the default value of infinite.

Capacity:   units

Maximum capacity; alternate behavior

### Convey Flow block's capacity


The maximum capacity for a Convey Flow block is a combination of two factors:

- 1) The block's length and maximum density determine the *maximized* capacity. By default, ExtendSim calculates a maximized capacity for the Convey Flow block by multiplying its length by its maximum density. This is indicated in the Capacity field by the "maximized" capacity checkbox shown above. In this case, the block's maximum capacity will equal its maximized capacity.

Capacity:   units  
Maximized capacity:  units

Maximized capacity

- 2) A number in the Capacity field in the Options tab can reduce the capacity below the maximized amount. In some cases, it may be necessary to define a capacity smaller than the maximized capacity determined by the length\*density calculation. For instance, the Convey Flow block could have structural properties limiting how much weight it can safely support. To do this, uncheck the checkbox in the Capacity field of the Options tab and enter the desired number. In this case, the block's maximum capacity will be less than or equal to its maximized capacity.

 A Convey Flow block's *maximum* capacity can never exceed its *maximized* capacity, no matter what number is entered in the Capacity field.

For example, if the Convey Flow block's length is 100 and maximum density is 10, the block's maximized capacity will be 1,000. To reduce the capacity to something less than 1,000, enter a number (for instance 300), in the Capacity field. The block will then only be able to contain 300 units of flow, even though its calculated maximized capacity was 1,000.

### Setting an initial contents

All three residence blocks (Convey Flow, Interchange, and Tank) can be preloaded with starting amounts of flow. A residence block's initial contents can be a specific number or, in the case of the Tank and Interchange blocks, it can be set to infinite.

Once an initial contents is set, it cannot change during the simulation. The exception is the Interchange block when it is set to *Tank only exists while item is in it*. In this case, each arriving item can establish the initial contents.

- ☞ If the initial contents of a Tank or Interchange block is set to infinite, its capacity will automatically be set to infinite.

### Empty and not-empty

If a Tank or Interchange block's initial contents is finite, its status can alternate between the empty and not-empty states. This change of state has an impact on the effective rate calculations:

- If the Tank or Interchange block is not empty, its flow level can fall. Consequently, the effective outflow rate could be higher than the effective inflow rate.
- If the block is empty, it cannot provide more flow than what it concurrently receives. In this case, the effective outflow rate has to be less than or equal to the effective inflow rate.

When a Tank or Interchange block changes state between empty and not empty, ExtendSim will calculate a new set of effective rates.

- ☞ The Convey Flow block has a different mechanism for calculating a change of state between empty and not-empty. For more information, see the “Delaying Flow” chapter.

### Tank initialization

As seen to the right, a Tank's initial contents can be:

Default settings for initial contents

- 0 (the default)
- An entered number
- Infinite

To cause the Tank to have an infinite amount of initial contents, check the ∞ (infinite) checkbox in the *Initial contents* field. For example, in the Yogurt Production model from the Discrete Rate Tutorial, the Tank blocks that represented Liquid Supply and Fruit Supply had infinite initial contents, while the Tank that stored the Yogurt product had no initial contents.

- ☞ If a Tank has no inflow connections, by definition it is being used as a source. If at some point the source reaches the empty state, the effective outflow rate will remain at zero for the remainder of the simulation run.

### Interchange initialization

As discussed in “Interchange block's capacity” on page 292, the Interchange block has two options for behavior that affect how its initial contents are set. These are discussed below.

#### *Tank only exists while item is in it*

This choice allows the block to have an initial contents only when an item arrives. To set an initial contents for this block, choose one of the options from the dialog's popup menu, shown above:

Interchange initial contents, default behavior

- *A constant.* Enter a number in the field; the default is 0. To cause the initial contents to be infinite, check the field's  $\infty$  (infinite) checkbox or set the field to blank.
- *Value at ICO.* The value at this connector will control the initial contents.
- *Value of attribute.* Select an attribute in the dialog. When an item arrives, the initial contents will equal that attribute's value.

In the Yogurt Production model from the Discrete Rate tutorial, the Interchange blocks were set to *Tank only exists when item is in it* and *Initial contents (on item arrival): 0*.

### Tank is separate from item

To set the initial contents for the Interchange block when this behavior has been selected, leave the *Initial contents* field set to the default infinite amount or enter a number.

Initial contents:    $\infty$  units

Interchange initial contents, alternate behavior

### Convey Flow initialization

The initial contents for the Convey Flow block are set in its Initialize tab. The table allows you to customize a number of segments for the conveyor, each with its own initial contents. Each row in the table represents an individual segment of the conveyor possessing a uniform density that differs from the adjacent segments.

	High Limit	Low Limit	Density	Initial Quantity
0	100 length unit	75 length unit	10 units/length un	250 units
1	50 length unit	25 length unit	5 units/length unit	125 units

Example initial contents for Convey Flow block

The Initialize tab has a *Show Example* button that places example settings in the table. These are helpful for understanding how to make the entries you want; they can also be used as a starting point for entries. Shown above is the example setting for a 100 foot long accumulating-density conveyor that transports containers. The table indicates that the block would have an initial density of 10 containers per foot for the segment from 75 to 100 feet (a total of 250 containers) and 5 containers per foot for the segment from 25 to 50 feet (a total of 125 containers).

-  In this example, the sections between 0 and 25 feet and between 50 and 75 feet do not hold any product.

### Indicators

As the simulation runs, the level of flow in the residence blocks (Convey Flow, Interchange, and Tank) will vary over time.

You might want an indication when a block's flow level is within a certain range of values. This is common when monitoring a block to determine if its contents are approaching or have reached one or more important benchmarks. For instance, some emergency procedures might need to take place if a Tank's level reaches the "high" range; they can be discontinued when the contents return to a "normal" range.

For residence blocks, *indicators* are a method of reporting what category or range the current level of flow falls into. With this feature, each range is assigned a name, a lower limit, and an upper limit. When the level of flow reaches a value that falls within a different range, the block reports the change on its *I* (indicator) value output connector and alerts any connected blocks to the change in status.

-  While the Tank and Interchange blocks report information about the current level of flow from their *I* (indicator) connectors, the Convey Flow block reports how far (the *accumulation*

*length*) the accumulation point is from the end of the conveyor. (When the amount of product ready to leave exceeds the amount that can be received downstream, flow begins to accumulate from the end of the conveyor. For more information, see “Distribution of flow” on page 345.

### Setting indicators

The Indicators tabs on all three residence blocks have similar interfaces. Each Indicators tab has a table (shown on the right with example settings) for specifying an indicator name for each range of values, entering the low limits, and defining values (an ID number for each indicator) to output when the block's flow level falls within a particular range.

	Indicator Name	Low Limit	High Limit	Value to Output
0	Full	100%	100%	4
1	High	70%	100%	3
2	Medium	40%	70%	2
3	Low	10%	40%	1
4	Empty	0%	10%	0

Example indicators in Tank block

To create indicators, enter your own information or click the Show Example button to populate the table with some example indicator names and settings. In either case, ExtendSim will calculate the High Limit values based on the Low Limit entries.

- ☞ The top row has to have the highest range; the bottom row must have the lowest range.
- ☞ To add or delete table rows, use the +/- button in the table's lower right corner. For instance, to delete the example settings, change the number of rows to 0.

The screenshot above shows a Tank's names, limits, and ID values to output after the Show Example button has been clicked. Each indicator name corresponds to a range of flow contents defined by the Low and High Limits for that row. (The High Limit column is presented for clarity only, since those numbers are calculated using the values entered for the Low Limits.)

Unless the block has infinite capacity, the indicator limits can be expressed in absolute numbers (shown above) or as percentages.

See “Bucket Elevator 2 model” on page 357 for an example of how indicators are used in an Interchange block to control a Valve's effective rate.

- ☞ If the block has infinite capacity, the limits must be expressed as absolute numbers. If the block's initial contents are set to infinity, the indicators are disabled.

### Getting information about levels

There are two types of events that will cause a new indicator to be reported:

- When the level is increasing and the block's contents reach the next indicator's Low Limit.
- When the level is decreasing and the block's contents reach the next indicator's High Limit.

In each case, the new output ID is used to update the I value output connector, and any connected blocks are alerted to the change.

	Indicator Name	Low Limit	High Limit	Value to Output
0	Full	250000 pints		4
1	High	175000 pints	250000 pints	3
2	Medium	100000 pints	175000 pints	2
3	Low	25000 pints	100000 pints	1
4	Empty	0 pints	25000 pints	0

Example indicators in a Tank



Using the above table as an example, if the level in the Tank increases from 120,000 to 175,000 containers, the block will compare that level to the Low Limit and send the value 3 to its I (indicator) output connector. However, if the level of the Tank instead decreases from 230,000 to 175,000 containers, the block will compare that value to its High Limit and output the value 2.

- The value that is output at the I (indicator) connector depends on whether the level of flow is increasing or decreasing, and where in the range the new indicator level falls. The Tank block has a S (status direction) output connector that reports if the level is going up or down when the event occurs.

### Tank Flow Units model

The Yogurt tank in the Tank Flow Units model outputs values that indicate the level of flow in the Tank; those values are displayed on the third line on the plotter.

- For more information about the Convey Flow block, including the use of sensors, see the discrete rate chapter on “Delaying Flow”.

## Units and unit groups

This section focuses on defining, selecting, and changing flow units, discusses the effect of changing time units, and shows how to use the Change Units block to change the unit group.

### Definitions

The following sections discuss flow, block, time, and length units. A unit group is two or more blocks connected together through flow connections and sharing one flow unit.

Units and unit groups were introduced on page 271; they are described fully below.

### Flow units

The flow unit indicates what is flowing from one Rate library block to another. As is true for ExtendSim time units, flow units can be unspecified generic units (in which case the block dialog will just display the word “units”) or they can be specifically defined in the model. For instance, a defined flow unit could be a packet, gallon, transaction, box, liter, and so forth. Existing flow units can be selected, and new units can be defined, in the Options tabs of Rate library blocks. In addition, the Discrete Rate tab of an Executive block (Item library) has a section for managing flow units. This provides a central location where units can be added, deleted, or renamed.

Generic flow and time units in a Valve

- This chapter will be mostly concerned with flow units.

### Block units

The Tank and Inter-change blocks can have an internal *block unit* that is different than the flow unit. This is an internal representation of volume that is specific to the Tank or Inter-change block, and does

Flow and block units in the Tank

not affect the flow unit for the unit group. If you select a block unit that differs from the flow units that come into and out of a block, you must enter a conversion factor. The conversion factor represents the ratio of the block unit to the flow unit. Block units are discussed fully in “Defining block units” on page 299.

☞ Using block units is optional; the default block unit is the flow unit.

### Time units

Time units can be generic, in which case the block will just say “time” or can be specific. Each model can define specific time units, which become the default for the model. You can change a discrete rate block’s time unit from the model default time unit to any local time unit using a popup menu in the block’s Options tab.



Default time unit

If a local time unit is selected in a discrete rate block, that local time applies to the entire block but only to that block. Changing to a local time unit does not change the global time unit for the flow group or for any blocks in the rest of the model. For complete information, see “Time units” on page 526.

⚠ Calendar dates are not available if months or years have been selected as the specific global time unit for a discrete rate model. Furthermore, if Calendar dates has been selected, Rate library blocks will not be able to select Months or Years as their local time unit.

### Length units

For convenience, the Convey Flow block allows you to name a length unit. This is used internally by the block with other settings to determine the block’s speed. You can use the default generic unit “length unit” or declare a specific unit of length such as feet or meters.

### Unit groups

A *unit group* is two or more blocks connected together through flow connections and sharing one flow unit. Connecting the first block’s outflow connector to the second block’s inflow connector creates a unit group. Unless the unit group is explicitly changed, all the blocks that are connected through flow connectors use the same flow unit and are in the same unit group. If a flow unit is changed in one of the blocks in a unit group, the unit group does not change but all the other blocks in that group are updated automatically to the new flow unit.

To see the unit group, click the grey square to the right of a block’s flow units popup menu, shown above. All the blocks in that block’s unit group will be highlighted on the model worksheet.



Unit group selector to right of popup

You can define multiple unit groups, which use different flow units, in portions of a discrete rate model. For instance, one part of the model could be expressed in bottles and another could represent boxes of bottles. The Change Units block can create a different unit group.

### Declaring and selecting flow units

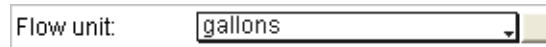
As mentioned above, flow units indicate what is moving from one flow connector to another, and a unit group is a collection of connected blocks that share the same flow unit. This section focuses on defining, selecting, and changing flow units.

⚠ To convert from one unit group to another in a model, use the Change Units block discussed on page 300.

### Where to declare a flow unit

Each block in the Rate library has the ability to select an existing flow unit or create a new one. This is done in the block's Options tab, which has a popup menu for either selecting an existing flow unit or creating a new one. Specifying a flow unit in one block causes that unit to be used by every block within the same unit group, and automatically sets that flow unit for any blocks that are subsequently added to that unit group.

To see the unit group, click the unit group selector button to the right of a block's flow units popup menu. All the blocks in that block's unit group will be highlighted on the model worksheet.



Unit group selector to the right of the popup menu

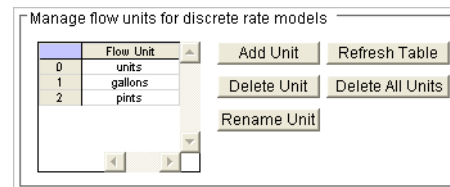
### Declaring a flow unit

To declare a flow unit, select an existing unit or create a new unit from the popup menu in the Options tab. The selected flow unit applies not only to the block but also to the entire flow unit group and it will automatically be set in new blocks that are connected within the unit group. For instance, page 277 of the discrete rate tutorial showed how to create a new flow unit named “gallons”. When the fruit processing section was added, the popup menu for the Fruit Supply Tank already included gallons selected as its flow unit.

### Managing flow units in the Executive block

The Executive block's Discrete Rate tab has a section for managing flow units in a model. The table displays all the units for a given model and provides buttons for adding, deleting, and renaming the units.

To use this feature, select the flow unit you want to change in the table, then click the appropriate button. ExtendSim will warn you and give options if the unit is being used in the model.



Flow units for Yogurt Production model

Discrete Rate

### Defining block units

As mentioned earlier, the Tank and Interchange blocks allow you to define a block unit that is different from flow units. This is an internal representation of volume only and is specific to the Tank or Interchange block. It does not change the flow units or the unit group for the flow that has entered or exited the block. If a block unit has been specified, a factor to convert the block unit into flow units must also be entered.

To define a block unit, in the Options tab of the Tank or Interchange block, select *Define a flow unit for the group and a block unit for the block*. In addition to providing fields for declaring a flow unit, this option displays a field for entering an internal block unit. It also has a field for entering the factor to convert between the flow unit and the block unit.

### Tank Flow Units model

The Tank Flow Units model is the same as the model described on page 280. However, the newer model has pints, rather than gallons, as the block units in the Yogurt Tank, causing the process's output to be displayed in the plotter as pints. Clicking the unit group selector button in its Options tab shows that this Tank is still part of the unit group that uses the flow unit "gallons".

Select units for the flow unit group and for this Tank

Define a flow unit for the group and a block unit for the block

Flow unit: gallons / minute\*

Block unit: pints / minute\*

Unit factor: 8 pints / gallons

Defining a block unit in the Tank

### Time units

A popup menu in the Options tab allows a block's time unit to be changed from the model default time unit to any local time unit.

If a local time unit is selected in a discrete rate block, that local time applies to the entire block but only to that block. Changing to a local time unit affects every parameter in the block, but it does not change the model's global time unit or the time units used in any other block in the model. For more information, see "Time units" on page 526.

## Changing the unit group

By default all the blocks connected to the same flow stream belong in the same unit group. However, the Change Units block has the ability to create a new unit group, causing connected blocks in a portion of the model to have a different flow unit.

### Change Units block

To change the flow units from one part of a model to another, use the Change Units block. While changing a flow unit in most blocks will change the flow unit for the entire group, adding a Change Units block to the model causes a new unit group to be created – the Change Units block's inflow connector will be part of one unit group comprised of those blocks upstream of the Change Units block; its outflow connector will be part of another unit group comprised of those blocks downstream of the Change Units block.

Change the flow unit, resulting in a new unit group

Change units from: gallons / minute\*

to: containers / minute\*

Conversion factor: 12 containers / gallons

Show flow unit change on icon

Dialog of Change Units block

The *Change units from:* popup menu defines the flow unit that is entering the block; the *to:* popup menu is for selecting or creating the new flow unit. The block has a field for entering the factor that converts the incoming flow unit into the outgoing unit, and a popup menu to select the direction the conversion should take. You can also change the time units for this block in its dialog.

### Yogurt Production model

An example of changing flow units is shown in the tutorial on page 281, where gallons of liquids were converted into containers of yogurt.

# **Discrete Rate Modeling**

## **Rates, Constraints, and Movement**

Limiting the movement of flow through rates and constraints

As discussed in the Introduction to this module, the movement of flow in a discrete rate model must take some time. It is a modeling error if the flow moves instantaneously throughout a model.

ExtendSim's discrete rate system attempts to move flow through the model as fast as possible. In the absence of any constraints, the effective rate of flow would approach infinity and the flow would move instantaneously throughout the model; this is never correct. For this reason, flow movement must be constrained by rates and conditions that are built into the model, and a lack of appropriate constraints is a modeling error that will stop the simulation run

In order to restrict flow rates:

- Discrete rate blocks are required to define their own sets of constraining flow rules
- Each area of a model must have one or more critical constraint mechanisms

Critical constraint flow rules, such as a block's maximum rate, place an upper bound on the rate of flow, limiting it to a number less than infinite. The blocks' aggregated set of flow rules ultimately defines how fast flow is permitted to move over time throughout the model.

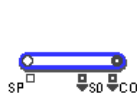
This chapter discusses rates, the blocks that constrain flow, and how model conditions impact the rate of flow. It will cover:

- Rates, rate sections, and the LP area
- Flow rules for defining how a block permits flow to move through it
- The blocks that specify critical constraints
- How to meet the constraint requirement
- A comprehensive example of constraints and rate sections

Most of the models illustrated in this chapter are located in the folder \Examples\Discrete Rate\Rates and Constraints. The tutorial models mentioned are located at \Examples\Tutorials\Discrete Rate.

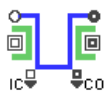
### Blocks of interest

The following blocks from the Rate library will be the main focus of this chapter.



#### **Convey Flow**

Delays the movement of flow from one point to another. Can accumulate flow to a maximum density, accumulate flow to fill empty sections, or act as a non-accumulating conveyor.



#### **Interchange**

Used to mix flow with items, this block can also limit its maximum rate of inflow and outflow.



#### **Tank**

The block most frequently used to store flow can also limit its maximum rate of inflow and outflow.



**Valve**

Controls and monitors the flow, limiting the rate of flow passing through. This block can also be used to set a goal for the duration or quantity of flow.

**Rates, rate sections, and the LP area**

One of the most important aspects of a discrete rate model is the rate of flow – the speed of flow movement. The flow rate is represented as the ratio of flow units to the model’s time units. This is displayed in block dialogs as units/time, gallons/minute, transactions/second, boxes/hour, and so forth.

Discrete rate models can be thought of as being divided into rate sections and LP areas. The flow connectors within each rate section have the same effective rate, which is the speed at which flow moves through those blocks. The LP area is composed of one or more rate sections whose effective rates might change during the simulation.

The types of rates considered during the model building process, rate sections, and the LP area, are discussed below.

**Types of rates**

The following rates are taken into consideration by ExtendSim and by a block’s flow rules. (Flow rules will be discussed on page 306.)

**Maximum rate**

ExtendSim’s discrete rate architecture attempts to move flow through the model as fast as possible. In the absence of any constraints, the flow rate would theoretically approach infinity and flow would move from one part of a model to another instantaneously; this would be a modeling error.

Maximum rate:   ∞ units / time

Default maximum rate for Valve

The *maximum rate* puts an upper limit on the movement of flow through a block. Six Rate library blocks have the ability to set a maximum rate. You can set an explicit maximum rate in the Interchange, Tank, and Valve blocks. The maximum rate for the Convey Flow block is mathematically derived. Maximum rates may also be implicitly specified under certain conditions in the Merge and Diverge blocks. In each case, the maximum rate is the highest rate of flow those blocks will allow, and hence the highest potential rate of flow for that part of the model.

An inflow connector for a Convey Flow, Interchange, or Tank block can have one maximum rate while the block’s outflow connector can have a different maximum rate. The maximum rate for the Convey Flow block’s inflow is derived from settings in its dialog; the maximum rate for its outflow is derived from dialog settings and model conditions. The maximum inflow and maximum outflow rates for the Interchange and Tank blocks can be entered directly in their dialogs.

In order to avoid an error condition, each area of a model must have some mechanism in place to restrict the rate of flow to a number that is less than infinity. If the required minimum set of constraints is not present, ExtendSim stops the simulation and displays an error message.

**Effective rate**

One of the most important reasons for creating a discrete rate model is to determine the actual rate of flow movement. The *effective rate* is the calculated actual rate of flow

Effective rate:  gallons / minute\*

Effective rate (Valve’s Results tab)

Discrete Rate

between Rate library blocks during the simulation run. It is the result of an internal calculation taking into account the maximum rates and all the constraints of the process. In some situations the effective rate is the same as the maximum rate; in others it is lower. One effective rate is associated with each *rate section* in a model, as discussed “Rate sections” on page 305.

In a rate section, the effective rate of flow cannot be higher than the lowest maximum rate for all the blocks in that section. In fact, it can be lower than the lowest maximum rate, and could even be zero (0), depending on model conditions.

While each rate section can have only one *effective* rate, a section can have more than one block that has a *maximum* rate. In fact, it is common to have several Valve blocks, each with their own maximum rate, in a rate section.

### Infinite rate

An *infinite rate* is a theoretical rate that would cause the flow to instantaneously move from one location in the model to another.

The Executive block’s Discrete Rate tab specifies that a rate equal to or greater than some number is considered infinite; the default is that a rate  $\geq 1e10$  is considered infinite, as shown above.

Any rate  $\geq$  1.0000000e+10 is considered infinite

From the Executive’s Discrete Rate tab

You can change the infinite number to be anything that you want. However, because of the 12 digit precision limitation of the effective rates, the number should be set as close as possible to the highest possible effective rate which would ever reach this limit. (Setting a correct infinite rate is more critical in the case of potential upstream supply and potential downstream demand calculations, an advanced topic discussed on page 382.)

### Infinite effective rate

Since instantaneous movement is not possible in the real world, the Rate library does not support an infinite effective rate. The infinity number specified in the Executive establishes an upper limit on the model’s allowable set of effective rates. If the simulation calculates an effective rate that equals or exceeds that number, ExtendSim will stop the simulation and generate an error message. This could happen, for instance, if a source tank is directly connected to a sink tank, without any intervening constraint.

### Infinite maximum rate

The Executive’s infinity number (by default,  $1e10$ ) can be used in a block’s rule set when conditions are such that it cannot in any way constrain the movement of flow. For example this would be accomplished by checking the  $\infty$  (infinite) checkbox for a Valve’s maximum rate. With an infinite maximum rate, the Valve will not limit the speed of flow passing through it.

If you set the Valve’s maximum rate field to blank or to a number  $\geq 1e10$ , the block would also not limit the speed of flow passing through it.

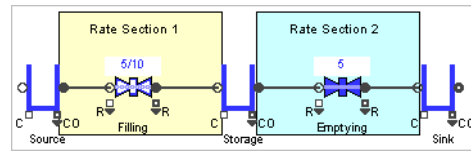
### Upstream supply/downstream demand

This potential rate is considered when using an advanced mode in the Diverge, Merge, and Sensor blocks. It is described fully in “Upstream supply and downstream demand” on page 382.



### Rate sections

A *rate section* is defined as a network of connected blocks, all possessing the same effective rate. Each rate section can include a succession of blocks and connections. A rate section always starts with an outflow connector and ends somewhere downstream with an inflow connector. Thus it will always contain at least two blocks sharing at least one flow connection.



Two rate sections in a model

While some blocks always define the boundary between two rate sections, other blocks never define a new rate section boundary:

- Because a residence block (Convey Item, Interchange, or Tank) can hold flow for some period of time, its effective inflow rate can be different from its effective outflow rate. Since the boundaries between rate sections never change, residence blocks always define the boundary between two different sections, even if their effective inflow rate is the same as their effective outflow rate. For example, a Tank's inflow connection ends one rate section and its outflow connection starts another section.
- Some passing (non-residence) blocks define a new rate section and others don't. While flow is not permitted to be held in a passing block for any length of simulation time, some passing blocks are capable of defining boundaries between sections. For example, a Valve is part of one, and only one, rate section. On the other hand, the Change Units block's effective inflow rate will always be different than its outflow rate because it performs a unit conversion. It therefore defines the boundaries between two rate sections.

All residence blocks, and certain passing blocks, always define the boundary between two different rate sections. These boundaries are established internally by ExtendSim at the beginning of the simulation run; they do not change.

The table below lists what role each block in the Discrete Rate library plays in defining the boundaries of a rate section:

Block	Always defines a new rate section?	Comments
Bias	No	The inflow effective rate is the same as the outflow effective rate. The block influences the effective rate associated with the section it is part of.
Catch Flow	No	The outflow effective rate is the same as the catch effective rate.
Change Units	Yes	The outflow effective rate is the inflow effective rate multiplied by the conversion factor.
Convey Flow	Yes	The outflow effective rate can be different than the outflow effective rate because it is a residence block.
Diverge	Yes	The effective rates can be different across all flow connectors – the input and all the outputs.

Block	Always defines a new rate section?	Comments
Interchange	Yes	See Convey Flow comment
Merge	Yes	See Diverge comment
Sensor	No	The inflow and outflow effective rates are the same.
Tank	Yes	See Convey Flow comment
Throw Flow	No	The inflow effective rate is the same as the throw rate.
Valve	No	The inflow and outflow effective rates are the same.

For an example of rate sections, see the “Comprehensive example” on page 315.

### Rate precision

The mathematical precision for effective rates is limited to 12 digits. This can become an issue if you separate any two effective rates in an area by more than 12 digits of precision. For more information, see “Precision” on page 360.

### LP area

While rate sections don’t change after the start of a run, the boundaries of the *LP area* change dynamically during the simulation. An LP area is composed of one or more rate sections linked together by the fact that their effective rates could change during the simulation run. When an event occurs that causes the effective rate for one rate section to be reevaluated, ExtendSim determines which other rate sections might be impacted. The affected rate sections constitute the LP area and become part of the LP calculation.

Since the LP area is computed internally, and because it is most important for the LP calculation, it is discussed fully in “The LP area” on page 377.

### Flow rules

*Flow rules* completely define how a block permits flow to move through during the simulation run. When calculating rates of flow, ExtendSim's discrete rate architecture tries to maximize throughput throughout the system, subject to a set of constraints. In order to restrict flow, discrete rate blocks are required to define their own sets of flow rules. The aggregated set of these rules ultimately defines how fast flow is permitted to move over time throughout the model.

A block’s particular set of flow rules is derived from four factors:

- The block’s fundamental behavior.
- How its dialog has been configured, such as setting a Tank’s maximum input rate or entering a conversion factor in the Change Units dialog.
- How its value connections have been connected. For instance, the Valve block’s R (maximum rate) input connector can be used to dynamically modify the block’s maximum flow rate.
- How its flow connections have been connected. A Tank is a source if only its outflow connector is connected; it is a sink if only its inflow connector is connected.

These flow rules completely describe the events or conditions under which a particular block may constrain the movement of flow through it. However, changes in a block’s constraints during a simulation cause its effective rates to be reevaluated and can cause a connected block’s effective

rates to be reevaluated, propagating calculations throughout an LP area. When recalculation is required, the Executive block (Item library) uses the aggregated set of flow rules from all the blocks in the LP area to calculate a new set of effective rates for the area. Thus a particular block's flow rules can be superseded by the global calculations of the Executive.

### Critical and relational constraints

There are two primary types of flow rules: critical constraints and relational constraints.

#### Critical constraints

While all flow rules cooperate to constrain the rate of flow, some blocks provide special rules called *critical constraints*.

If a rate section contains one or more critical constraints, they place an upper bound to the rate of flow for the blocks within that section. A critical constraint is unconditional – no matter what happens in the simulation, the effective rate of flow cannot be higher than the lowest critical constraint of any block in that rate section. For example, the *Maximum rate* field is a Valve's critical constraint; the entry in that field defines an upper limit on the rate of flow through the block. If that entry is the lowest critical constraint in the rate section, the effective rate for every block in that section cannot be higher.

Maximum rate:	<input type="text" value="1.0"/>	<input type="checkbox"/> ∞	units / time
---------------	----------------------------------	----------------------------	--------------

Critical constraint in a Valve

The blocks with the potential to set a critical constraint for flow are the Convey Flow, Diverge, Interchange, Merge, Tank, and Valve; of these, the Valve is most commonly used. As will be shown in “Meeting the critical constraint requirement” on page 312, these blocks must be placed at critical locations in order for the model to run properly.



ExtendSim's discrete rate system attempts to move flow through the model as fast as possible. Without any mechanism to impede its progress, the effective rate would theoretically approach infinity and the flow would move from one part of a model to another instantaneously. In order to avoid this error condition, each LP area of the model must contain one or more constraints (typically a Valve) to restrict the flow to a number that is below infinity. If the required minimum set of critical constraints is not present in a model, ExtendSim stops the simulation and displays an error message.

Discrete Rate

#### Relational constraints

*Relational constraints* define the way the effective rates of different sections are related to each other, creating dependencies between rate sections. For instance, the relational constraint between one rate section (effective rate  $x$ ) and another rate section (effective rate  $y$ ), could be defined as  $x \geq y$ ,  $x = y$ ,  $2x - 3 = y$ , or any other expression. Relational constraints get updated when the block reacts to new parameters or to changes in its state, but they don't affect a block's critical constraints.

An example of a relational constraint is the Change Units block, where the use of a conversion factor causes the outflow effective rate to be different than the inflow effective rate. The Change Units block defines the boundaries between one rate section and another; the conversion factor specifies the relationship of the two effective rates.

Inflow rate:	<input type="text" value="80"/>	gallons / minute*
Outflow rate:	<input type="text" value="960"/>	containers / minute*

Different inflow and outflow rates

For another example of a relational constraint, see “Comprehensive example” on page 315. For an advanced discussion of relational constraints, see “The relational constraint calculation” on page 381.



You don't enter relational constraints, they are determined by the behavior of the blocks.

### Comparison of constraints

Some blocks can set a critical constraint, some can set a relational constraint, and some can do both. Even for blocks that can set constraints, the block may in some situations place no constraint on the flow.

- The blocks that can set a critical constraint are the Convey Flow, Diverge, Interchange, Merge, Tank, and Valve.
- Relational constraints can be set by the Change Units, Convey Flow, Diverge, Interchange, Merge, and Tank blocks.

For example, a Tank where both the *Maximum inflow rate* and *Maximum outflow rate* are checked will set critical constraints for its inflow and outflow.

If neither *Maximum inflow rate* nor *Maximum outflow rate* is checked, the Tank will not have any critical constraints but could have relational constraints. If the Tank has a finite capacity but is neither full nor empty, it places no constraints on the flow. However, once the Tank reaches the full state, its inflow rate is required to be less than or equal to its outflow rate; this is a relational constraint.

The effective rate for a rate section cannot be any higher than the lowest critical constraint set for by any of the blocks in that section. Furthermore, because the aggregated set of flow rules also typically contains relational constraints, the effective rate for the section can vary anywhere between zero and the smallest critical constraint.

 For a table that lists the blocks and which constraints they can provide, see “Types of information provided to the Executive” on page 379.

### Defining a critical constraint

As mentioned earlier, a critical constraint defines the upper limit to the rate of flow through a rate section. While a particular rate section may or may not have a critical constraint, at least one of the rate sections within the LP area must have a critical constraint mechanism to limit the flow rate to a number that is less than infinity.

- You can explicitly set a critical constraint in the Valve, Tank, and Interchange blocks. You do this by entering a maximum rate in the block's dialog, obtaining a value for the maximum rate from the block's input connector, or linking the maximum rate field to the value of a cell in a global array or ExtendSim database.
- For the Convey Flow block, the critical constraint is derived from settings in its dialog and sometimes other model values, rather than being entered directly.
- A critical constraint may also be implicitly specified under certain conditions by the Merge and Diverge blocks.

The next sections describe how to set a critical constraint. See also “Meeting the critical constraint requirement” on page 312 for examples of how to apply the constraint requirement in your models.

## Valve

The Valve is the block most often used for explicitly setting a critical constraint. You can enter a value in the *maximum rate* field in the block's dialog, link the field to an ExtendSim database or global array, or connect the block's *R* (maximum rate) input connector to some value output.

Defining constraints for Valve

For an example of setting a fixed maximum rate for the Valve, see “Entering dialog parameters and settings” on page 277.

### Dynamically changing the maximum rate

There are two ways to change a Valve's maximum rate during a simulation run:

- Connect to the block's *R* (maximum rate) input connector
- Link the block's Maximum rate field to an ExtendSim database or global array

Connecting to the Valve's *R* input connector or linking its maximum rate field to a data source overrides any values directly entered in the maximum rate field. Instead, that field will display the current maximum rate as determined by the simulation run.

The checkboxes for “Initial maximum rate” and “Poll constraint every”, discussed below, are only used when the Valve's maximum rate is configured to change dynamically.

For an example of using the Valve's *R* input connector to cause the maximum rate to change dynamically, see “Add a dynamic constraint” on page 278.

As you saw in “Add maintenance” on page 280, the “Add Shutdown” button in the Valve's dialog automatically connects a Shutdown block (Item library) to the Valve's *R* input connector. This can be used to stop the flow, or reduce its rate of movement, for a period of time. See also “Shutting down” on page 179 for a description of how to use the Shutdown block.

#### Initializing the maximum rate

When the Valve's maximum rate is configured to change dynamically, the *Initial maximum rate* checkbox serves an important role. This is because the first effective rate calculations for a simulation occur just before simulation time starts moving forward. If the Valve's *R* connector is connected or if the maximum rate field has been dynamically linked, problems can arise at this stage because neither the block connected to the *R* connector nor the linked data source has yet had a chance to provide an initial value. The *Initial maximum rate* checkbox resolves this issue by initializing the maximum rate.

The initial value entered in the dialog will be used until the Valve gets a different value from its *R* input connector or from the linked data source.

For multiple runs, the *Initial maximum rate* checkbox prevents the Valve from using the last maximum rate from the current run as the initial maximum rate for the next run.

#### Polling constraints

The checkbox to *Poll constraint every...* can be used when a Valve's maximum rate is configured to change dynamically. This option directs the Valve to request a new maximum rate value at fixed intervals during the run. This causes the Valve to periodically query the block connected to its *R* (maximum rate) input connector or the cell linked to its maximum rate field for the new values. Any values received between the queries will be ignored.

This checkbox is optional when the maximum rate field is connected to a fixed number in a linked cell in a data source. It is required if the linked cell contains a random number or if the R input connector is connected to a passive block like the Random Number (Value library), since a passive block won't independently generate a new value for the maximum on its own. (The checkbox is not needed if the R connector is connected to block that actively generates values, such as the Lookup Table block set to output values at regular time intervals in the discrete rate tutorial.)

- ☞ Each time the maximum rate in a Valve changes, effective rates must be re-calculated across multiple sections. If you are using the poll constraint feature in several Valves, consider having them update at the same time. This will dramatically reduce the number of recalculations.
- ☞ While the polling feature can be handy during the early stages of the model building process, flow rates in real world systems rarely change at fixed intervals. Use this feature judiciously and with caution.

### **Controlling how and when the Valve applies its maximum rate**

The Valve's Control Flow tab has advanced options that allow you to manage how and when that block applies its maximum rate. By setting a goal or using hysteresis, you can explicitly control when the Valve's constraining rate will be observed, when it will be ignored, and for how long either of those will happen. These topics are discussed in the "Delaying Flow" chapter.

### **Tank and Interchange**

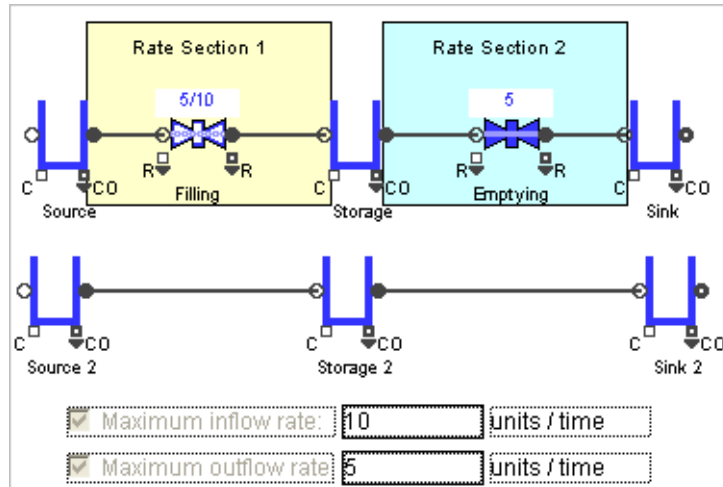
The Tank and Interchange blocks have dialog options for explicitly defining their maximum inflow and outflow rates. Unlike the maximum rate in a Valve, these constraints do not change dynamically during the simulation.

<input checked="" type="checkbox"/>	Maximum inflow rate:	10	units / time
<input checked="" type="checkbox"/>	Maximum outflow rate:	10	units / time

Default maximum rates for Tank

You can enter either a maximum inflow rate or a maximum outflow rate, or both of these.

The Tank Constraint example shows two flow streams with identical behavior. In the bottom flow stream, Tank 2 uses the options *Maximum inflow rate* and *Maximum outflow rate* to replace the filling and emptying valves found in the upper flow stream.



Tank Constraint model. Top stream with two Valves to constrain flow; bottom stream with maximum rates defined in Tank 2.

Instead of using a Valve block to constrain flow, setting maximum inflow and/or a maximum outflow rates in a Tank or Interchange block can be used to satisfy the model's requirements for a constraint.

### Convey Flow

The Convey Flow block calculates critical constraints for its inflow and outflow connectors separately. The critical constraints are derived from model conditions and settings in the dialog.

- The critical constraint for the Convey Flow block's inflow is calculated by multiplying the block's effective speed by its maximum density entry.

The effective speed can be less than or equal to the speed set in the dialog. If the block is non-accumulating, or if it is accumulating but cannot accumulate more, and the block's ability to deliver flow exceeds downstream demand, the effective speed will be lower than the entered speed.

- The critical constraint for the block's outflow is the result of the multiplication of the block's speed setting by the density of flow present at the outflow end of the block.

Setting the initial contents or capacity for a Convey Flow block is discussed in the chapter "Sources, Storage, and Units". The "Delaying Flow" chapter shows how to use the Convey Flow block to delay the movement of flow in a model.

### Merge and Diverge

The critical constraint for one or more of a Merge or Diverge block's branches can be implicitly specified under certain conditions. Most often, the result would be a rate of 0 (zero).

When a Merge or Diverge block is set to certain modes, flow can be blocked from moving through one or more of its branches. For example, if one branch of a Diverge block in Distributional mode has been assigned a blank value or a value  $\leq 0$ , flow through that branch is halted. Similarly, flow

through all but the selected branch is blocked when the Merge block is in Select mode. In both of these cases, the maximum rate would be 0 for the affected branches.

The “Mode table” on page 319 lists each mode for the Merge and Diverge blocks. The column labeled “Parameter values that always block the flow” indicates which conditions would always cause a branch to have an implied constraint of 0.

### Meeting the critical constraint requirement

As discussed earlier, while a particular rate section may or may not have a critical constraint, at least one of the rate sections within the LP area must have a critical constraint mechanism to limit the flow. Otherwise, the rate of flow would approach infinity.

By definition, residence blocks always delineate the boundary between two rate sections. A general rule is that there must be at least one critical constraint between every two residence blocks. (The critical constraints can be provided by the Convey Flow, Diverge, Interchange, Merge, Tank, and Valve blocks. The residence blocks are the Convey Flow, Interchange, or Tank.) The exceptions to the general rule include certain situations where a Merge or Diverge block is between two residence blocks.

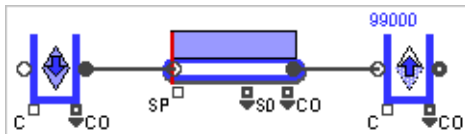
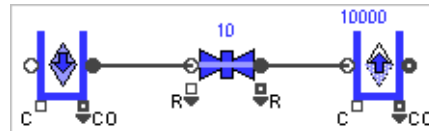
The following examples illustrate some ways the required critical constraint mechanism can be met in discrete rate models.

#### Valve or Convey Flow

The No Merge or Diverge model illustrates two typical ways to provide a critical constraint to the rate of flow between two residence blocks (in this case, Tanks) that don't have a Merge or Diverge block between them.

The example to the right uses a Valve to constrain the flow between two Tanks. This is the most straight forward and most common situation. In order for the Valve's maximum rate to provide the critical constraint it must be:

- Greater than or equal to 0 (zero)
- Less than 1e10 (the defined infinite rate)
- Not a blank



The example to the left uses a Convey Flow block to meet the requirement for a critical constraint. A Convey Flow block derives the critical constraint for its inflow from its dialog settings and the critical constraint for its outflow from its dialog settings and model conditions. Because it has critical constraints at both its inflow and outflow connectors, the Convey Flow block limits the rate of flow from the first Tank to the second to a number that is less than infinite.

#### Tank or Interchange

Instead of using a Valve to provide the critical constraint between two residence blocks, you can specify maximum inflow and maximum outflow rates for an intervening Tank or Interchange block. With these maximum rates, the Tank or Interchange will limit the rate of flow between the two residence blocks to a number less than infinite. This is shown in the Tank Constraint example discussed in “Tank and Interchange” on page 310.



### Merge or Diverge blocks

If a Merge or Diverge block is between two residence blocks, the inflow and outflow branches may or may not require a critical constraint mechanism.

- ☞ For any Merge/Diverge mode, if a critical constraint has been placed on a Merge block's outflow branch, no critical constraints are required on its inflow branches. Likewise, a critical constraint on a Diverge block's inflow branch means that no critical constraints are required on its outflow branches. If those constraints have not been placed, the critical constraint requirement depends on the block's mode.

The following table provides an overview of each mode's requirements for critical constraints when neither the Merge block's outflow branch nor the Diverge block's inflow branch has a critical constraint. (In this table, the word "variable branch" means an inflow branch for the Merge block or an outflow branch for a Diverge block.)

Mode	Critical constraint requirements if there is no critical constraint on the non-variable branch
Batch/Unbatch	Only on one of the variable branches
Distributional	Each variable branch
Neutral	Each variable branch
Priority	Each variable branch
Proportional	Only on one of the variable branches (See Note, below)
Select	Each variable branch
Sensing	Each variable branch

Note: For the Proportional mode, the variable branch with the critical constraint should not have a proportion  $\leq 0$ . Otherwise, that branch will be closed and the other variable branches will have potentially infinite effective rates. This is an error condition.

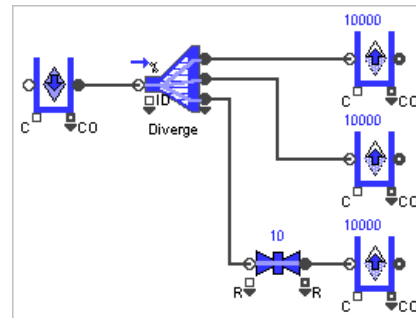
- ☞ Merge and Diverge blocks, including their modes, are described fully in the chapter "Merging, Diverging, and Routing Flow".

The two examples that follow use the Minimum Valve model to illustrate some of the table's concepts.

**Proportional mode**

The top section of the Minimum Value model indicates the critical constraint requirement when a Merge or Diverge block is in Proportional mode. If the block is located between two residence blocks, only one critical constraint is needed as long as the branch's proportion is neither 0 nor blank. (This lower requirement for constraints is an exception to the general rule described on page 312.) The effective rates for the other branches are deduced from the Valve's maximum rate.

In the Minimum Valve example shown on the right, a Valve is placed on a Diverge block's bottom outflow branch, and that branch does not have a 0 or blank proportion.

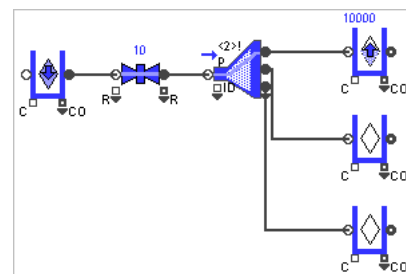


Only one constraint needed

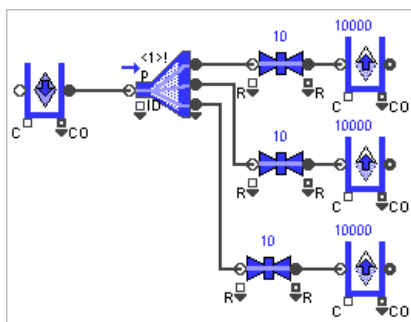
**Priority mode**

The lower section of the Minimum Value model indicates the critical constraint requirements when a Merge or Diverge block is in Priority mode. In this case, the number of critical constraints that must be placed on the branches between residence blocks depends on where those constraints are placed. These situations are shown in the Minimum Valve model.

If a critical constraint is placed between a residence block and a Diverge block's inflow branch, you do not need to place any other critical constraints on the Diverge block's outflows. Likewise, if you place a critical constraint on a Merge block's outflow branch, you do not need to place any critical constraints on its inflow branches. This is shown on the right, where a Valve with a maximum rate greater than or equal to 0 but less than 1e10 (the infinite rate) is on a Diverge block's inflow branch and there are no critical constraints required on its outflow branches.



No constraint on each outflow



Constraint on each outflow


If you don't place a critical constraint on a Diverge block's inflow branch, you must place at least one critical constraint on each outflow branch. Likewise, if you don't place a critical constraint on a Merge block's outflow branch, you must place at least one critical constraint on each inflow branch.

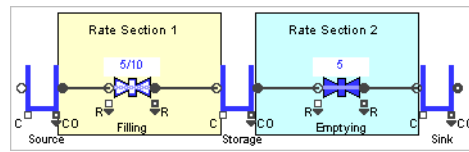
This is shown in the screenshot to the left, where there is no critical constraint on the Diverge block's inflow branch. This means each outflow branch must have a critical constraint, in this case a Valve with a maximum rate greater than 0 but less than 1e10 (the infinite rate).

Discrete Rate

## Comprehensive example

The following example illustrates many of the concepts from this chapter. The top line of the Tank Constraint model, shown on the right, has two rate sections, two critical constraints, and one relational constraint.

 The sections that follow use the abbreviation FPT to indicate “flow units per time unit”.



Tank Constraint model

### Rate sections

Rate sections are determined internally by a communication between Rate library blocks and the Executive (Item library). The boundaries between rate sections are established at the beginning of the simulation run; they do not change during the run even if the effective rates change.

At the beginning of the simulation run:

- The Filling valve has a maximum rate of 10, gets its inflow from an infinite Source, and sends its outflow to an empty Storage tank that has a capacity for 100 flow units. The system will thus calculate an effective inflow and outflow rate of 10 FPT for the Filling valve at the start of the simulation run. (This will change once the Tank fills.)
- The Storage tank has a capacity for 100 flow units, gets its inflow from a valve with a maximum rate of 10 FPT and sends its outflow to a valve with a maximum rate of 5 FPT. At the beginning of the simulation run, its effective inflow rate will thus be 10 FPT and its effective outflow rate will be 5 FPT.
- The Emptying valve has a maximum rate of 5 FPT and sends its outflow to an infinite Sink. Its effective inflow and outflow rate is 5 FPT.

At the start of the simulation run, the Storage tank's effective inflow rate is different from its effective outflow rate. Thus the first rate section for the Tank Constraint model starts at the Source block's outflow connector and ends at the inflow connector on the Storage tank. The second rate section starts at the Storage tank's outflow connector and ends at the Sink's inflow connector.

### Critical constraints

There are two critical constraints in the top line of the Tank Constraint model. The first critical constraint is the 10 FPT entered in the Filling valve's *maximum rate* field. The second is the 5 FPT entered in the Emptying valve's *maximum rate* field.

### Relational constraint

Relational constraints define the way the effective rates of different sections are related to each other. At the beginning of the simulation run there are no relational constraints – the effective inflow rate is independent of the effective outflow rate. When the Tank (which has a finite capacity of 100 flow units) becomes full, it applies one critical constraint: inflow rate must be less than or equal to outflow rate.

### Simulation's impact on the effective rates

Since it is empty at the start of the run, the Storage tank's initial set of flow rules will not include placing any restrictions on its inflow rate. Consequently, the initial effective rate of flow through Rate Section 1 is limited only by the Filling valve's critical constraint of 10 FPT.

However, this initial effective rate for the first rate section is only temporary. Since the Storage tank's capacity is finite and since Rate Section 2's effective rate is only 5 FPT, the Storage tank will eventually become full. Once this happens, the effective rate of 10 FPT in Rate Section 1 can no longer be maintained. Consequently, the Storage tank introduces a relational constraint that requires its inflow effective rate (Rate Section 1) to be less than or equal to its outflow effective rate (Rate Section 2). Once the Storage tank is full, its relational constraint causes the effective rate through Rate Section 1 to be reduced to 5 FPT.

# **Discrete Rate Modeling**

## **Merging, Diverging, and Routing Flow**

Using the Merge, Diverge, Throw Flow and Catch Flow blocks

When building models, you will frequently encounter situations where you want to route the streams of flow in a model. This is accomplished using the Catch Flow, Diverge, Merge, and Throw Flow blocks.

The Merge and Diverge blocks have similar interface and capabilities. These two blocks send and receive flow through a variable number of inflow and outflow connectors. Their dialogs provide rule-based options to merge or diverge flow in a discrete rate environment.

The Throw Flow and Catch Flow blocks also have similar interfaces. These blocks route flow remotely from point to point.

This chapter covers:

- Blocks for merging, diverging, and routing flow
- Merge and Diverge modes
- Additional features of the Merge and Diverge blocks
- Using the Throw Flow and Catch Flow blocks

 The models illustrated in this chapter are located in the folder \Examples\Discrete Rate\Merge and Diverge.

### Blocks of interest

The following blocks from the Rate library will be the main focus of this chapter.



#### Catch Flow

Receives flow sent from Throw Flow or Diverge blocks. Allows you to group blocks that can send the flow into sets, so that the list of possible connections can be filtered.



#### Diverge

Distributes flow from one inflow branch to one or more outflow branches at a time. The block has several modes for determining how the flow is distributed through the branches.



#### Merge

Merges flows from one or more inflow branches at a time into one outflow branch. The block has several modes for determining how the inflows should be received.



#### Throw Flow

Sends flow to Catch Flow or Merge blocks. Allows you to group the blocks that can receive the flow into sets, so that the list of possible connections can be filtered.

### Merging and diverging flow

The systems modeled using discrete rate technology typically have multiple flow streams that need to be merged into one stream or, conversely, one flow stream that needs to be diverged to multiple streams. The Merge and Diverge blocks have been designed specifically to model this type of routing behavior.

The Merge and Diverge blocks have seven different rule-based options that determine how they send and receive flow. These *modes* mostly behave as mirror images of each other in the two blocks. The list of modes, and their similarities and differences, are summarized in the “Mode table”, below. The examples that follow the table show how each of the modes can be applied.

☞ For the Merge block, each input connector is referred to as an *inflow branch*. For the Diverge block, each output connector is referred to as an *outflow branch*. Collectively they are known as the *variable branches*.

### Mode table

The following table lists the Merge and Diverge modes in alphabetical order and summarizes their main similarities and differences.

Mode	See page	Sum of inputs = sum of outputs?	Fixed rule?	Bias order required?	Parameter value at each branch that will <i>always</i> block the flow	Compatible with Sensing mode?
Batch/Unbatch	321	No	Yes	No	None	Yes
Distributional	324	Yes	No	Yes	Blank, <=0	Maybe
Neutral	326	Yes	No	No	None	Maybe
Priority	322	Yes	No	Yes	Blank	Maybe
Proportional	321	Yes	Yes	No	Blank, <=0	Yes
Select	319	Yes	Yes	No	None	Yes
Sensing	325	Yes	No	Yes	Blank, <=0	Yes

### Characteristics

Explanations for the mode characteristics are:

- Some modes use a fixed flow rule to obtain or distribute the flow – no matter what happens in the rest of the model, the fixed rule will be respected. For other modes, the flow rules express a preference and are only invoked in specific situations depending on model conditions.
- Competing requests for flow amongst Merge and Diverge blocks that have been set to the Distributional, Priority, and Sensing modes require the use of bias ordering. This is discussed in “Biasing flow” on page 360.
- While other parameter values may block the flow in certain circumstances (for instance, if a number is out of range), for some modes a Blank or zero (0) will always cause the flow to be stopped. Values that are out of range will cause an error message; a zero (0) or Blank will not generate an error message.
- As discussed in their respective sections, incompatibilities can arise if an area of the model has one or more blocks that use the Sensing mode and other blocks that use either the Distributional, Neutral, or Priority mode. These situations should be avoided whenever possible as they can give inaccurate results.

### Select mode

When the Merge or Diverge blocks are in Select mode, only one selected branch at a time is open. A table in the block’s dialog allows you to assign a unique ID number to each inflow branch (for the Merge block) or outflow branch (for the Diverge block). The ID connector on the block’s icon is then used to select which branch to open.

Options in the block’s dialog allow you to specify what happens if the value at the ID connector doesn’t match any of the branch IDs listed in the table:

- Choose top connection
- Choose bottom connection
- Stop flow
- Generate error

A blank value received at the ID connector always stops the flow until the connector receives a valid input.

The Select mode uses a fixed flow rule to obtain the set of effective rates for each branch and to determine which branch to route the flow to.

### Select Mode Diverge model

In the Select Mode Diverge model, a Create block is set to output a sequential value (1, 2, 3, or 4) every 20 time units. At its ID input connector, the Diverge block receives the value from the Create block, compares that value to entries in its dialog table, and selects the appropriate outflow connector. Three Tank blocks, each with an infinite capacity for flow units, are connected to the Diverge block. The Tank blocks are identified by the ID values entered in the Diverge block's dialog. For example, an Output ID of 1 indicates the Tank labeled "Infinite Sink 1".

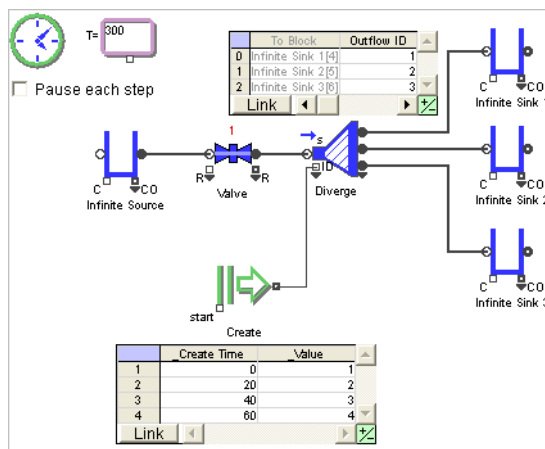
In this example the Create block is responsible for controlling the Diverge block. When the Create block sends a value of 2 to the Diverge block's ID connector, the flow is routed to Infinite Sink 2, and so forth.

Notice that in this model 4 is an invalid number and the Diverge block is set to *Invalid value at ID: stop flow*. When the Create block sends a value of 4, all flow through the Diverge block stops and a red bar appears on the block's right side. This pause in the flow could be used for a specific purpose, for instance to allow time to empty downstream Tanks.

Try running the model after checking *Pause each step* (in the upper left corner of the model). This will cause the simulation to pause so you can more easily see the effect of the Create block sending values to the Diverge block. Clicking the Pause/Resume button in the toolbar will continue execution to each succeeding event. (There can be more than one event without time advancing.) For more information, see "Stepping through a model" on page 522.

### Select Mode Merge model

This model is the mirror image of the Select Mode Diverge model discussed above. While the three source tanks provide an infinite supply of flow, it is the Merge block that controls which tank flow is drawn from. Since the Merge block is in Select mode, the Create block controls the routing of flow by providing different values (1, 2, 3, 4 sequentially every 20 minutes) at the Merge block's ID connector.



Select Mode Diverge model



In this model, the Merge block is set to *Invalid value at ID: choose top connection*. This means that when the ID connector gets a value of 4 from the Create block, it will select the flow from its top input connector.

### Batch/Unbatch mode

The Batch and Unbatch modes are used to cause a different total amount of outflow than what would be indicated by the total amount of inflow or to change the total amount of inflow into a different total amount of outflow.

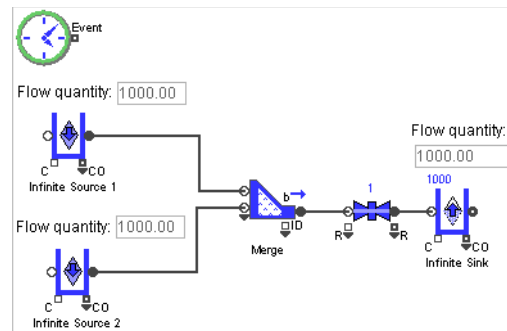
- When the Merge block is in Batch mode, each unit of flow from each inflow branch is combined into one outflow unit. The effective rates of each inflow branch and the outflow connector are thus required to be equal. In this mode, the Merge block's behavior is similar to that of the Batch block (Item library).
- When the Diverge block is in Unbatch mode, each unit of flow from its inflow branch is cloned into one unit of flow for each outflow branch. The effective rates for the inflow connector and each outflow branch are thus required to be equal. In this mode, the Diverge block's behavior is similar to that of the Unbatch block (Item library).

☞ The Batch/Unbatch modes are different from all the other modes because the amount of total inflow is *never* equal to the amount of total outflow.

#### Batch Mode Merge model

In the Batch Mode Merge example, the Merge block is set to *Merge mode: batch*. Each time unit the block takes one unit of flow from Infinite Source 1 and one unit of flow from Infinite Source 2. It then combines them to make one unit of output flow per time unit. Since the model runs for 1,000 time units, the Infinite Source 1 and Infinite Source 2 blocks each provide 1,000 units of flow.

Notice the amount of flow (1,000 units) that has entered the Infinite Sink is half the total amount of flow that has left the two source tanks. This is because the effective rate for the Merge block's outflow connector is required to be the same as the rate at each of its two inflow branches.



Batch Mode Merge model

Discrete Rate

#### Unbatch Mode Diverge model

In this example, one unit of flow per time unit from the Infinite Source is unbatched into two flow units per time unit – one for Infinite Sink 1 and the other for Infinite Sink 2. Notice the total quantity of flow (2,000) in the two sink tanks is double the amount of flow (1,000) that exited the source tank.

#### Proportional mode

With the Proportional mode, you define in a table what the proportion of flow through each branch will be. The proportion for each branch is defined in the table relative to each of the other branches. For instance, a value of 2 for the top outflow branch and 4 for the bottom outflow branch would indicate that the bottom branch should have twice the amount of flow as the top

branch. If a particular branch's proportion has been defined to be blank or  $\leq 0$ , the effective rate for that branch is set to 0 and the flow is stopped for that branch.

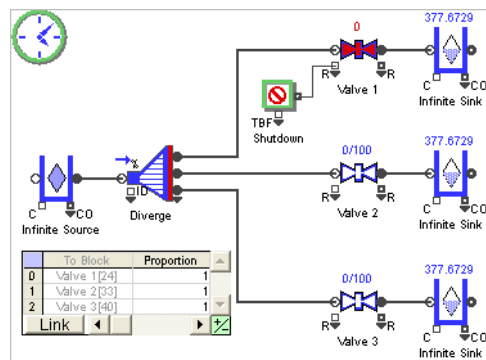
See “Merge blocks in Proportional mode” on page 367 for options when a Merge block is part of an empty loop.

**!** This mode uses a fixed flow rule where the effective rate at each branch is **required** to meet the proportion defined by the table. Consequently, if the flow through one or more of the branches is blocked or starved, the effective rates for all branches will be set to zero and all flow through the block is halted.

**Proportional Mode Diverge model**

In this example, flow coming from the Infinite Source is evenly distributed between the Diverge block's three outflow branches. This occurs because the proportions in the table in the Diverge block's dialog have been set to 1:1:1. With this proportion, the effective rate across all three branches is required to be the same – an identical amount of flow must pass through each branch.

The initial constraining rate for the three Valve blocks is set to 100. However, the Shutdown block forces Valve 1's constraining rate to alternate between 0 and 100 as the model runs. This has an impact on the effective rate for all three branches. When the constraining rate for Valve 1 switches to 0, the outflow from all three branches goes to 0 even though the constraining rate for Valves 2 and 3 is still equal to 100. This is because the Diverge block must enforce its ratio, which is 1:1:1 in this example.



Proportional Mode Diverge model

**Proportional Mode Merge model**

This model is the mirror image of the Proportional Mode Diverge model discussed above. While all three Valve blocks limit the supply of flow from the source tanks at an initial constraining rate of 100, the Shutdown block forces the constraining rate in Valve 1 to alternate between 0 and 100. As in the previous model, when the constraining rate in Valve 1 switches to 0, the effective rates for all three branches become 0 because the Merge block is in Proportional mode and must enforce the 1:1:1 equality it is set to.

**Priority mode**

The Priority mode allows you to attach priorities to the inflow branches of the Merge block and the outflow branches of the Diverge block. These priorities only impact the effective rates assigned to the branches when discrepancies arise between the upstream flow supply and the downstream flow demand; otherwise they are ignored.

- In the case of the Diverge block, when the upstream supply is greater than or equal to the downstream demand, the block passes as much flow through each branch as the downstream demand will allow and the priorities are ignored. However, when the cumulative downstream demand exceeds upstream supply, the priorities that have been assigned to each branch are used to calculate the appropriate effective rates for the outflow branches.

Discrete Rate

- In contrast, the Merge block passes as much flow as possible through each inflow branch when downstream demand exceeds upstream supply, ignoring the priorities. However, when the cumulative upstream supply exceeds downstream demand, the priorities assigned to each branch are used to calculate the appropriate effective rates for the inflow branches.

Special cases apply to the use of the Priority mode in a Merge or Diverge block:

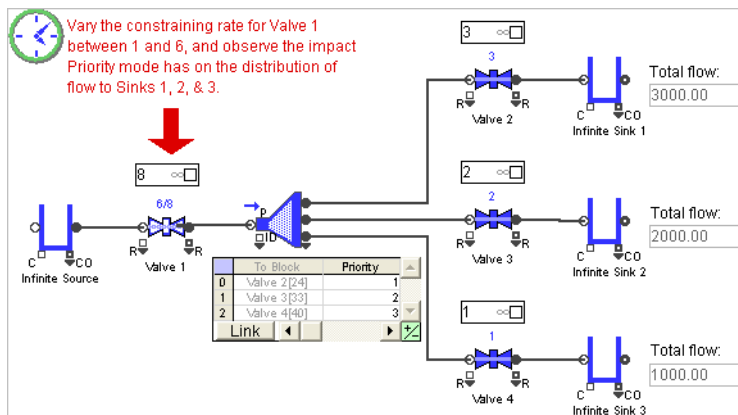
- If the priority for a particular branch has been set to blank, the effective rate for that branch will be zero and the flow will stop for that branch.
- If the priorities of two or more branches are equal, the flow will be divided among them in a “distributional” manner with equal proportions (see Distributional mode, below.)

The priority entries in a Diverge block’s dialog are not fixed rules but instead are situational; they are only used to resolve discrepancies when downstream demand exceeds upstream supply. For a Merge block, the entries are used to resolve discrepancies when upstream supply is greater than downstream demand.

⚠ Merge/Diverge blocks in Priority mode are not always compatible with Merge/Diverge blocks in Sensing mode. Consequently, an area of the model with some blocks in Sensing mode and others in Priority mode are prone to error. See “Cautions when using potential rates” on page 383 for more information.

### Priority Mode Diverge model

In this example, the constraining rates in the valves have been set such that the upstream supply of 8 flow units per time unit through Valve 1 exceeds the cumulative downstream demand of 6 set by Valves 2, 3, and 4. Because a large enough supply of flow exists to satisfy downstream demand, the priorities in this case are ignored and have no impact on the set of effective rates defined for each outflow branch.



Priority Mode Diverge model

However, if a “supply scarcity” is introduced by changing the constraining rate in Valve 1 from 8 to 4, the Diverge block will calculate a set of effective rates that distributes the now limited supply of flow according to the defined priorities. Since the priorities have been assigned in descending order (top outflow branch has highest priority), the Diverge block will do its best to satisfy the downstream demand that has been placed on the top outflow branch first. After that, if supply is still available, the Diverge block will attempt to service subsequent branches. This pattern is repeated until every branch has been satisfied or until the upstream supply of flow runs out, whichever comes first.

**Priority Mode Merge model**

When the blocks are in Priority mode, the difference between a Merge block and a Diverge block (illustrated above) is that the priorities defined in the Merge block’s table impact the effective rates for the inflow branches if there is a downstream “scarcity of demand”. The Priority Mode Merge model illustrates the use of Priority mode when 1) the downstream demand exceeds upstream supply, and 2) downstream demand is less than upstream supply.

**Distributional mode**

Similar to the Proportional mode described on page 321, the Distributional mode allows you to define a desired set of proportions for each branch. However, unlike the Proportional mode (but similar to the Priority mode discussed on page 322), these proportions serve as the decision rule for assigning effective rates to the branches *only* when discrepancies arise between the upstream flow supply and the downstream flow demand.

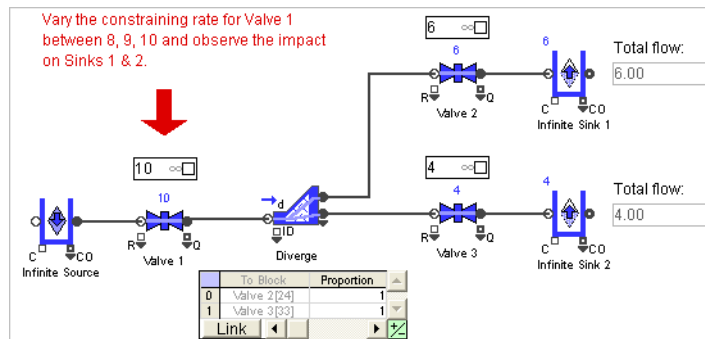
- In the case of the Diverge block, when the upstream supply is greater than or equal to the downstream demand, the block passes as much flow through each branch as the downstream demand will allow and the proportions are ignored. However, when downstream demand exceeds upstream supply, the proportions assigned to each branch are used as a guide to determine how the limited supply should be distributed across the outflow branches.
- In contrast, the Merge block passes as much flow as possible through each inflow branch when downstream demand exceeds upstream supply, ignoring the proportions entered in the dialog’s table. However, when upstream supply exceeds downstream demand, the proportions assigned to each branch are used as guides to determine how the limited demand should be distributed across the inflow branches.

The distributional proportions entered in a Merge or Diverge block’s table are significant only in certain situations; they are ignored otherwise. Proportions do not follow a fixed flow rule; they only impact the effective rates assigned to the branches when discrepancies arise between the upstream flow supply and the downstream flow demand.

⚠ Merge/Diverge blocks in Distributional mode are not always compatible with Merge/Diverge blocks in Sensing mode. Consequently, an area of the model with some blocks in Sensing mode and others in Distributional mode are prone to error. See “Cautions when using potential rates” on page 383 for more information.

**Distributional Mode Diverge model**

In this example, the proportions for the Diverge block’s two branches are set to 1:1. The constraining rates in the valves are defined such that the upstream supply of 10 flow units per time unit through Valve 1 equals the cumulative downstream demand of 10 set by Valves 2 and 3. Because a large enough supply of flow exists to satisfy



Distributional Mode Diverge model

downstream demand, the distributional proportions are ignored and have no impact on the set of effective rates defined for each outflow branch of the Diverge block.

Two examples highlight what happens when the Diverge block is set to Distributional model and there is a “supply scarcity” that causes the upstream supply to be less than the downstream demand:

- If the constraining rate in Valve 1 is changed from 10 to 8, the Diverge block will use the 1:1 proportions that have been defined in its dialog to allocate the now limited supply between the two downstream demanding branches. In this case, 4 units of flow per time unit will move through both the top and bottom branches.
- If the constraining rate in Valve 1 is set to 9 units of flow per unit of time, the situation is different. According to the 1:1 proportions that have been defined in its dialog, the Diverge block should allocate 4.5 units of flow to each of the two downstream demanding branches. However, the constraining rate for Valve 3 is 4 units of flow per time unit and that is all it can accept. The extra 0.5 units of flow will be routed through the top branch because the downstream demand for the bottom branch cannot keep up with the upstream supply (4.0 vs. 4.5) and the Distributional mode will always try to push as much flow as possible.

 The Diverge block's Proportional mode is used to resolve discrepancies when downstream demand is greater than upstream supply.

#### ***Distributional Mode Merge model***


When the blocks are in Distributional mode, the difference between a Merge block and a Diverge block (illustrated above) is that the proportions defined in the Merge block's table impacts the effective rates for the inflow branches only if there is a downstream “scarcity of demand”. The Distributional Mode Merge model illustrates the use of the Distributional mode when 1) the downstream demand exceeds upstream supply, and 2) downstream demand is less than upstream supply.

 The Merge block's Proportional mode is used to resolve discrepancies when upstream supply is greater than downstream demand.


#### **Sensing mode**

Similar to the Proportional mode discussed on page 321, the Sensing modes use proportions to calculate the effective rates for the branches. However, unlike the Proportional mode where you directly enter or control the proportions for each branch, the proportions for the Sensing modes are derived dynamically from the model as it runs.

- In the case of the Diverge block, Demand Sensing proportions for the outflow branches are calculated as a function of the potential downstream demand. For instance, the downstream demand placed on a particular outflow branch becomes the proportion for that branch.
- Similarly, the Merge block uses the potential upstream supply to define the Supply Sensing proportions for each inflow branch.

 Potential demand and supply rates are advanced concepts that are discussed in “Upstream supply and downstream demand” on page 382.

In the Sensing mode, the block's dialog has a table where you must define the maximum possible rate of flow through each branch. This upper bound is used as a way to limit throughput so that the proportions can be determined if the upstream supply or the downstream demand is infinite.

 The discussion on page 383 provides reasons why the Sensing mode should be used with extreme caution and some situations where it should be avoided altogether. Given the potential problems,

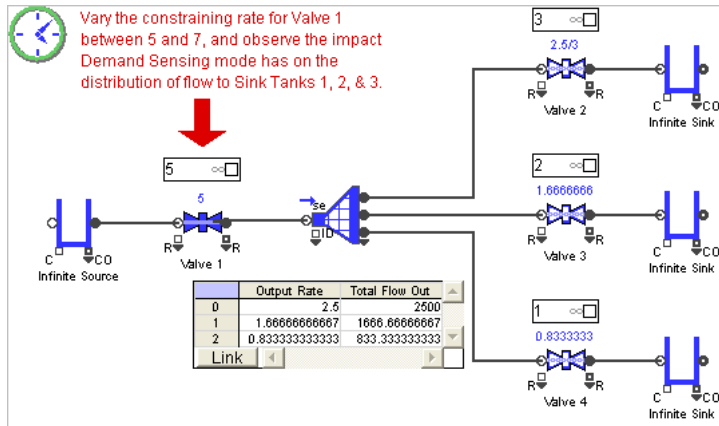
and because similar behavior can be achieved using the Distributional mode, the Sensing mode should be used only as a last resort.

### Demand Sensing Mode Diverge model

In this example, the constraining rates in Valves 2, 3, and 4 define the demand for flow downstream of the Diverge block. They therefore define the proportions used to distribute the flow across the Diverge block's outflow branches.

A maximum possible rate of 1,000 for each branch is entered in the Diverge block's table.

The block's Results tab (cloned onto the model worksheet) displays each branch's actual outflow rate and the amount of total outflow for the simulation run.



Demand Sensing Mode Diverge model

### Supply Sensing Mode Merge model

In the Supply Sensing Mode Merge example, the constraining rates in Valves 1, 2, and 3 define the supply upstream of the Merge block. They therefore define the proportions used to distribute flow across the inflow branches.

### Neutral mode

Unlike any of the modes discussed previously, the Neutral mode does not allow you to control the effective rates for the branches. This is a passive mode where no branch has a throughput advantage; the branch that gets chosen cannot be predicted. It is used when the system does not need to control how the flow is routed.

- In the case of a Diverge block, when the upstream supply is greater than or equal to the downstream demand, the block passes as much flow through each branch as downstream demand will allow. However, when downstream demand exceeds upstream supply, the distribution of flow across each branch cannot be predicted.
- In contrast, the Merge block passes as much flow as possible through each inflow branch when downstream demand exceeds upstream supply. However, when upstream supply exceeds downstream demand, the distribution of flow across each branch cannot be predicted.

The Neutral mode should be used carefully but can be handy in certain cases. As a general rule of thumb, if you don't care exactly which branch has priority, but you do want maximum flow, consider using the neutral mode. The Neutral mode can also be used to resolve conflicting decision rules. For example, using the Neutral mode in a downstream Merge block would allow an upstream Diverge block in Proportional mode to control the effective rates of the inflow branches in the Merge.

**!** Merge/Diverge blocks in Neutral mode are not always compatible with Merge/Diverge blocks in Sensing mode. Consequently, an area of the model with some blocks in Sensing mode and others in Neutral mode are prone to error. See “Cautions when using potential rates” on page 383 for more information.

### Features of the Merge and Diverge blocks

Some features available in the Merge and Diverge blocks of particular interest include:

- Bias order
- Dynamically changing parameters
- Internal Throw and Catch connections.

These features are described in the following sections.

#### Bias Order – resolving competing requests for flow

As models grow in complexity, it is common for the priorities or proportions defined in one Merge/Diverge block to compete or conflict with the priorities or proportions defined in other Merge/Diverge blocks. This problem of “competing requests for flow” is resolved by assigning a *bias order* to the competing blocks. This is accomplished through entries in either the Model Settings tab of the individual blocks or the Discrete Rate tab of the Executive block. The following example demonstrates one of the many ways competing requests can arise and be resolved.

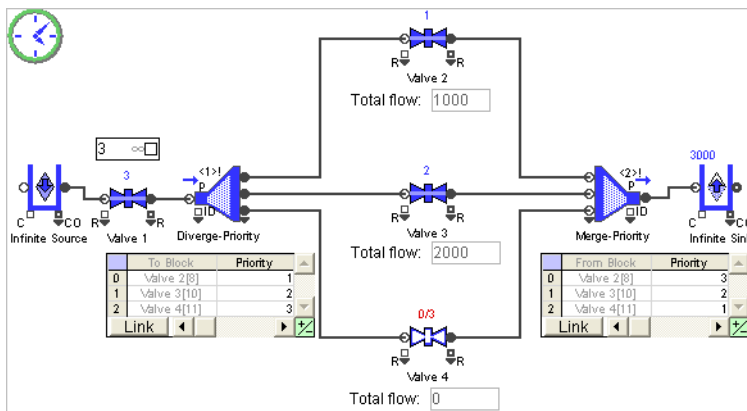
**I** Because certain modes allow flexibility in the way flow is distributed, Merge or Diverge blocks set to Distributional, Priority, or Sensing modes must specify a bias order to resolve conflicts between competing preferences for flow, as discussed below. For a complete description of the bias concept and bias order, see “Biasing flow” on page 360.

#### Competing Requests for Flow model

This model demonstrates how the priorities in two routing blocks compete against each other.

In this example:

- The Diverge block’s outflow branch priorities have been specified in descending order while the Merge blocks’ inflow branch priorities have been specified ascending order.



Competing Requests for Flow model

- The two blocks share common flow streams.

While the Diverge block in this model will try to satisfy its top outflow branch first, the Merge block will oppose that by trying to satisfy its bottom inflow branch. To resolve this conflict, the Diverge block’s priorities have been biased over the Merge block. This was accomplished by select-



ing *Each block defines its own bias order* in the Discrete Rate tab of the Executive block, then selecting the Diverge block in the Executive's table and entering a bias of 1.

Selecting the option "Show bias order on icon" in the Discrete Rate tab of the Executive block causes the bias value to be displayed near block icons as "<x>". In the above model, the bias order is indicated as <1> for the Diverge block and <2> for the Merge block, indicating that the Diverge block has precedence over the Merge block's requests.

 To see how the Bias block is used instead of Merge/Diverge blocks to resolve competing preferences for flow, see the "Prioritize With Bias Blocks" model located in the folder \Examples\Discrete Rate\Merge and Diverge and discussed on page 360.

### Internal throw and catch

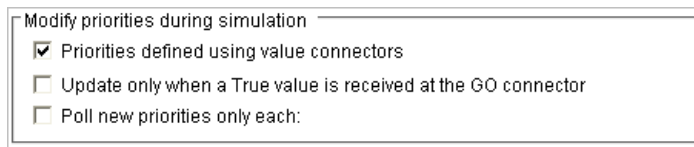
While the Rate library has two blocks, Throw Flow and Catch Flow, specifically designed to transport flow without the use of connection lines, the Diverge and Merge blocks have been given throw/catch abilities as well. See "Throwing flow and catching flow remotely" on page 329 for a full discussion.

### Changing decision rules dynamically

With the exception of the Batch/Unbatch and Neutral modes, decision rule parameters for the Merge/Diverge modes can be changed dynamically during the run. However, this is an advanced feature that requires some caution and extra insight into how ExtendSim works.

There are two ways to dynamically change a decision rule for a Merge/Diverge mode:

- Allow the rules to be controlled by other blocks. To do this, check the appropriate checkbox (such as *Priorities defined using value connectors*, as shown to the



Modifying decision rules dynamically (Priority mode)

right) on the Merge or Diverge block's Options tab. When this checkbox is selected, a set of value input connectors appear on the block's icon, with one connector for each branch.

- Link the parameter table to an ExtendSim database table or global array. Any changes made to the table or array while the model runs will have the same effect as using a block to dynamically change the values.


### Limiting the number of recalculations

While this advanced feature is useful for changing how effective rates are calculated on the fly, there are potential pitfalls. Since computations typically happen sequentially on a computer, new parameter values for each branch are changed one at a time. Ideally, the Merge/Diverge block will not recalculate the new set of effective rates for each branch until after all parameters have been updated. If this is not the case, however, the block will be forced to unnecessarily calculate an entirely new set of effective rates every time a parameter is updated. At the very least this will cause your runtimes to be longer than need be. At the very worst, redundant rate calculations could introduce bugs into your model when effective rates are temporarily calculated using one or more out-of-date parameter values.

There are three approaches that will help avoid this problem:



- 1) The issue can be bypassed if the inputs on one Merge/Diverge block are controlled by the outputs from one equation-based block, such as the Equation block (Value library). This is because the equation block will update all its outputs with the new results prior to alerting the Merge/Diverge block to the change. For an example of this, see the model “Change Priorities with Equation”.
- 2) By checking *Update only when a True value is received at the GO connector* in the block’s Options tab. This allows the calculation of a new set of effective rates to be controlled explicitly. In this case, changes to the parameters are ignored until a message is received at the *GO* input connector. This is shown in the example model “Change Proportions with Trigger” which requests a new set of proportions at the beginning of each goal. (A goal represents the production of 1000 units of flow and is repeated over and over until the end of the simulation.) The values at the inputs change every 10 time units, but because the chosen set of parameter values remains unchanged for the duration of the goal, effective rates are recalculated only at the beginning of each new goal.
- 3) By checking *Poll new parameters only each: x time units* in the block’s Options tab. This causes a new set of parameters to be updated at fixed intervals. In this case, changes to the inputs are ignored until the next interval in time arrives. In the example “Change Proportions Periodically”, the model picks a new set of proportions every time another 100 units of time is reached.

 The options *Update only when a True value is received at the Update connector* and *Poll new parameters only each: x time units* can be combined together.

### Throwing flow and catching flow remotely

The most common way to route flow from one block to another is by drawing a connection from an outflow connector to an inflow connector. This is a powerful mechanism for routing flow because it’s simple to implement and it provides a very clear picture of how the flow is being routed in a model. The use of connection lines between connectors, however, can prove to be cumbersome when many streams of flow need to be routed into or out of hierarchical blocks.

The ExtendSim throw/catch mechanism solves this issue by allowing flow to be moved without the use of connection lines. By creating a throw/catch connection via block dialogs, flow can be routed from a throwing block to any catching block in the model.

In the Rate library:

- Flow can be sent from the Throw Flow and Diverge blocks
- Flow can be received by the Catch Flow and Merge blocks.
- Any sending block can throw to any catching block regardless of location.

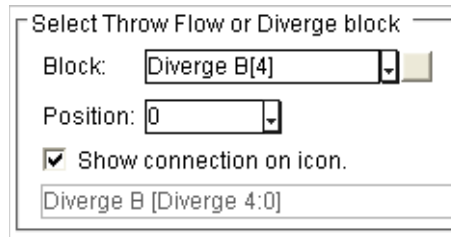
The rules restricting how normal flow connections can be drawn between outflow and inflow connectors also apply to throw/catch connections:

- The flow can go one way only – from throw to catch
- One throw can be connected with one and only one catch
- One catch can be connected with one and only one throw

The advantage of using a Diverge block to throw flow or a Merge block to catch flow, is that each outflow or inflow branch can throw or catch a separate stream of flow remotely. The Throw Flow and Catch flow blocks, on the other hand, are limited to one flow source or destination each.

### Creating a throw/catch connection

The creation of a throw/catch connection can be made from either the sending (Throw Flow or Diverge) or the receiving (Catch Flow or Merge) block. Connections are made by selecting the block to catch or throw the flow in a popup menu in a block's dialog. Each eligible block appears in the list with its block label and global block number. Once established, the connection information is automatically displayed in the dialogs of the sending and receiving blocks. In the screenshot above, the Catch Flow block will receive flow from a Diverge block labeled "Diverge B"; the Diverge block's global block number is 4.



Selecting a connection in a Catch Flow block

### Choosing the connector position for Merge and Diverge blocks

If a Diverge or Merge block is part of the throw/catch connection, after selecting the connecting block, you must also choose a Merge or Diverge connector position for flow to come from or go to. This is because the Merge and Diverge blocks have multiple inflow and outflow branches. Some of their inflows or outflows may not be used for throwing/catching and some throwing/catching blocks may get flow from or send flow to different branches on a single Merge or Diverge block.

The number that indicates a Merge or Diverge block's particular connector position is displayed in the leftmost column of the table in the Merge/Diverge block's Throw or Catch tab; the number of the topmost inflow or outflow branch is zero (0). You select the connector position from a popup menu to the right of the Position field in the corresponding block. The menu will list all the available connector positions for the named block. In the screenshot in the preceding section, the top outflow connector position (0) for the Diverge block labeled Diverge B is entered in the Position field of a Catch Flow block's dialog.

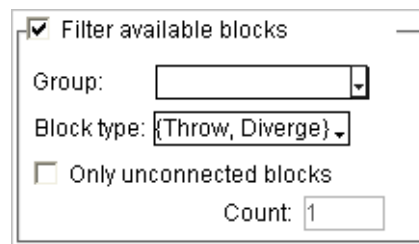
An asterisk to the right of a connector position number in the popup menu indicates that the connector is already being used by some other throw/catch block.

### Filter options to facilitate throw/catch connections

In large models, it is possible to have a great number of sending and receiving blocks from which a throw/catch connection can be made. To simplify the popup list of blocks eligible for connection, three types of filters can be applied:

- Group filter
- Block type filter
- Only unconnected blocks filter

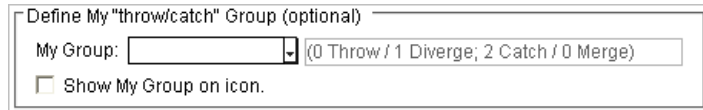
These filters can be used in combination with each other.



Filtering options

**Group filter**

Each block with throw or catch capabilities can be added to a throw/catch group. Groups can be created or selected through the group popup menu found in the sending and receiving blocks. When this popup is blank, the block does not belong to a group. When a block has been added to a group, its throw/catch options are limited to the blocks currently in that group.

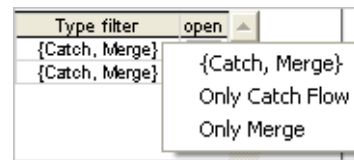


Defining a Group

**Block type filter**

By default, the block types a throwing block can connect to include both Catch Flow and Merge blocks and the block types a catching block can connect to include by Throw Flow and Diverge blocks.

- For throwing blocks, the list of blocks to select from can be narrowed to only Catch Flow blocks or only Merge blocks.
- For catching blocks, the list of blocks to select from can be limited to only Throw Flow blocks or only Diverge blocks.



Block type filters for a Diverge block

**Only unconnected blocks filter**

This filter narrows the selection of possible blocks to only those blocks without an established throw/catch connection.

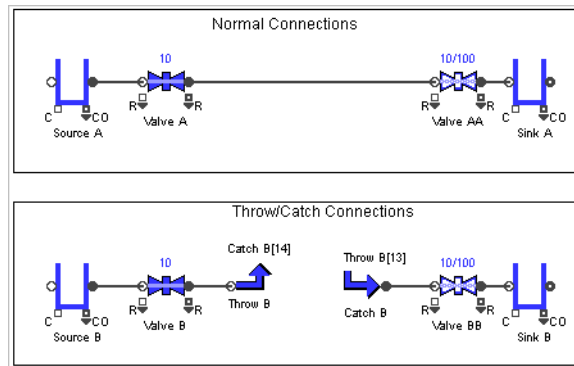
**Examples of throw and catch connections**

The two example models that follow show how to use catching and throwing in a model. The first model uses Throw Flow and Catch Flow blocks; the second model uses a Diverge block.

**Catch Flow and Throw Flow model**

The Catch Flow and Throw Flow model shows two lines, one above the other. They produce identical results even though the flow connections have been created differently:

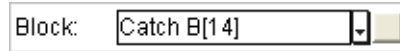
- The top line uses normal flow connection lines to connect Valve A to Valve AA.
- The bottom line uses a throw/catch connection to connect Valve B to Valve BB. In this line, the flow is sent remotely by a Throw Flow block and received by a Catch Flow block.



Catch Flow and Throw Flow model

332 | **Merging, Diverging, and Routing Flow**  
 Throwing flow and catching flow remotely

The particulars of the throw/catch connection can be viewed from the dialogs of either the Throw Flow or Catch Flow blocks. In the Throw Flow dialog you can select the Catch Flow block and see the throw/catch connection; in the Catch Flow dialog you can select the Throw Flow block and see the throw/catch connection.

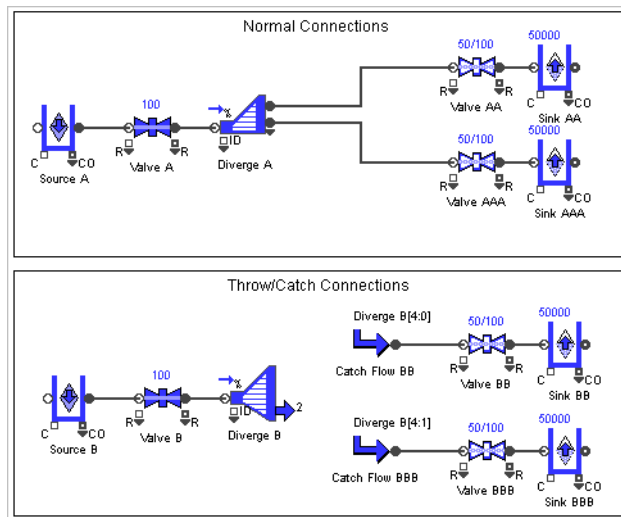


Portion of Throw Flow dialog

**Catch Flow and Diverge model**

Similar to the previous model, the top and bottom lines for the Catch Flow and Diverge model produce identical results even though the flow connections have been created differently:

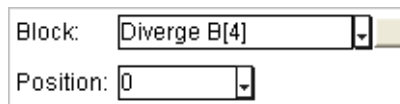
- The top line of the model uses regular connections to connect the two outflow branches of a Diverge block to Valve AA and Valve AAA.
- The bottom line uses throw/catch connections to connect the outflow branches of the Diverge block to Catch Flow BB and Catch Flow BBB.



Catch Flow and Diverge model

In the bottom line, both Catch Flow blocks indicate in their dialogs that the block labeled Diverge B is remotely sending the flow. Since the flow is being sent by a Diverge block which can have multiple outflow branches, and since each Catch Flow block must receive its flow from a separate source, the Catch Flow blocks must also specify which of the Diverge block's connector positions they will receive flow from.

In the screenshot at right, the Catch tab of Catch Flow BB indicates that it is receiving flow from the Diverge block labeled Diverge B. It is getting that flow from the Diverge block's top outflow branch (outflow connector position 0).



Portion of Catch Flow BB dialog

If the throw/catch connection has been properly defined for a particular branch of a Merge or Diverge block, the "Open" button for that branch will appear in the last column of the table in the block's Throw or Catch tab.

# **Discrete Rate Modeling**

## **Delaying Flow**

For a certain period of time, either maintaining flow at a certain speed or blocking it.

The “Rates, Constraints, and Movement” chapter discussed how to specify critical constraints (such as a Valve’s maximum rate) and the factors that determine the actual speed of flow moving through a model.

☞ Since the concepts of flow rules and critical constraints are central to the discussion of delaying flow, it is assumed that you have already read the “Rates, Constraints, and Movement” chapter.

This chapter describes how to use advanced methods to delay flow – for a specified period of time or until a specified condition has been met, either blocking the movement of flow or maintaining it at a certain speed. It illustrates several methods for delaying flow, including how to:

- Control how and when a Valve observes or ignores its maximum rate setting:
  - Setting a goal for a quantity of flow
  - Setting a goal for a duration
  - Using hysteresis to control when the block’s maximum rate will be observed
- Use a Shift block (Item library) with a Convey Flow, Interchange, Tank, or Valve to delay flow movement for a specified period of time
- Transport flow over a defined distance at a specified speed with a Convey Flow block

☞ Most of the models illustrated in this chapter are located in the folder \Examples\Discrete Rate\Delaying Flow. The tutorial models mentioned are located at \Examples\Tutorials\Discrete Rate.

### Blocks of interest

The following blocks from the Rate library will be the main focus of this chapter.



#### Convey Flow

Delays the movement of flow from one point to another. Can accumulate flow to a maximum density, accumulate flow to fill empty sections, or act as a non-accumulating conveyor.



#### Valve

Controls and monitors the flow, limiting the rate of flow passing through. This block can also be used to set a goal for the duration of flow movement or the amount of flow.

#### Controlling a Valve’s maximum rate

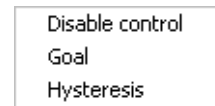
The section “Defining a critical constraint” on page 308 showed how to set a Valve’s maximum rate. Options on the Valve’s Flow Control tab provide advanced control over how and when the Valve applies its maximum rate. These settings determine when the Valve’s maximum rate is observed or ignored, and for how long.

The advanced control options include:

- Setting a goal for a certain quantity of flow to pass through the Valve.
- Setting a goal for how long flow can move through the Valve or for how long it should be stopped from moving.
- Determining what happens when the goal ends.
- Using hysteresis to delay the Valve’s response to system requirements.

### Using the Flow Control tab

By default, the settings in a Valve's Flow Control tab are disabled. Use the tab's popup menu, shown at right, to choose either the Goal or Hysteresis option.



Flow Control options

If the Goal option is selected, an additional popup menu appears to the right. Use this second menu to choose:

- Goal as a quantity
- Goal as a duration

### Observing the maximum rate for a goal

Whether you select a quantity or a duration goal, a Valve's maximum rate is observed while a goal is On. While the goal is off, you have the option to choose whether the maximum rate is observed or not and whether the flow is stopped. Thus your purpose in using a goal could be to block flow for a period of time and allow it to move through when the goal is off, allow flow for a period of time and block it at other times, accept a certain quantity of flow but stop flow when the goal is off, and so forth. In fact, you can choose to observe the maximum rate when the goal is off. This can be useful for a quantity goal, when you just want the Valve to report when a goal is finished.

- If its maximum rate is 0 (zero), the Valve will block flow while its goal is On.
- If its maximum rate is >0, blank, or infinite, the Valve will allow flow to pass through at that rate while its goal is On. (If the maximum rate is blank, the Valve uses an infinite rate.)

If a Valve's maximum rate is zero and a quantity goal is On, no flow will go through the block and the goal will never end.

### Options when goal is Off

Dialog options allow you to choose what will happen when the goal switches to Off:

- Stop the flow
- Ignore maximum rate (do not constrain flow)
- Observe maximum rate

### Setting a Valve's quantity goal

Using popup menus in a Valve's Control Flow tab, you can specify that the block has a goal to pass a certain quantity of flow from its inflow to its outflow. Additional options allow you to specify what the Valve should do once that quantity of flow has passed.

The quantity goal modulates a Valve's critical constraint (its maximum rate) by cycling between On and Off states.

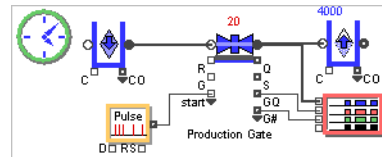
- While a quantity goal is On, the value in the Valve tab's *Maximum rate* field, or the value at the block's *R* (maximum rate) input connector, is observed. The goal remains On until either the amount of flow that has passed through the block reaches the target quantity, or the goal is interrupted. At that point the goal switches to the Off state.
- What happens when the goal switches to the Off state depends on the Off option selected in the block's dialog: stop the flow, do not constrain flow, or observe maximum rate.

To interrupt a goal, send a value to the Valve's *stop* input connector.

**Quantity Goal model**

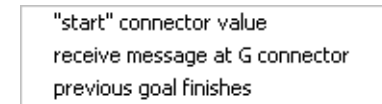
This model uses a Pulse block (Value library) to periodically start a new quantity goal. At the start of the simulation run and every 60 minutes afterwards, the Pulse block sends a True value (a number  $\geq 0.5$ ) to the Production Gate valve's *start* connector, starting a new goal.

The Valve's maximum rate is 20 gallons/minute and the simulation runs for 480 minutes. The control for the movement of flow through the block is provided by its Flow Control tab. The desired quantity of flow, 500 gallons, is entered in the dialog. While the goal is On, flow passes through the Valve at a maximum rate of 20 gallons/minute. After 500 gallons, the goal goes Off, the flow is stopped, and the effective rate goes to 0. As the simulation runs, a plotter displays the Valve's effective rate, the quantity of the goal and the times when it has been reached, and the number of each new goal.



Quantity Goal model

Settings in the Flow Control tab cause the Valve to *Start a new goal when "start" connector value = 1*. The goal-starting options (seen at right) can also be set to start a new goal when the block receives a message at its G (goal) input connector or when the previous goal finishes.



Options for starting a new goal

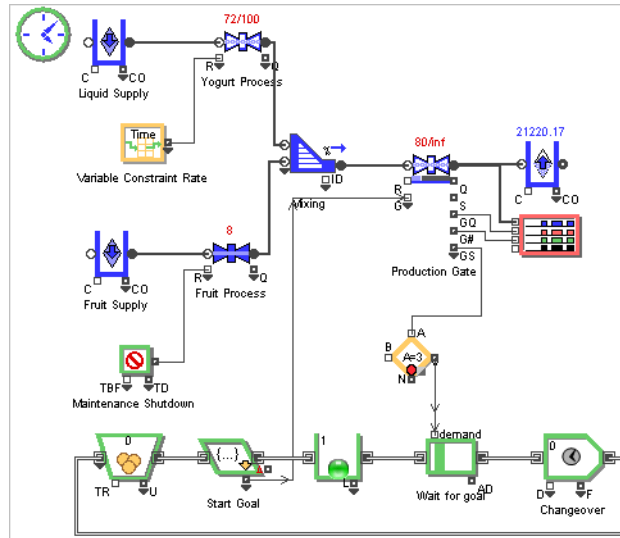
You could have avoided building the Quantity Goal model by instead mentally calculating the effect of the goal on the flow. The rate interactions and results determined by the next model would not be so easy to compute.

**Changeover Quantity Goal model**

The Changeover Quantity Goal model involves a more complex system than the Quantity Goal model from above. This model, based on the stage of the Yogurt Production model that is discussed in "Add maintenance" on page 280, shows how a quantity goal in a Valve can be used to control a sequence of production changeovers. The top part of the model is composed of Rate



library blocks and the bottom line is Item library blocks. A Decision block (Value library) transmits values from a Valve (Rate library) to a Gate (Item library).



Changeover Quantity Goal model

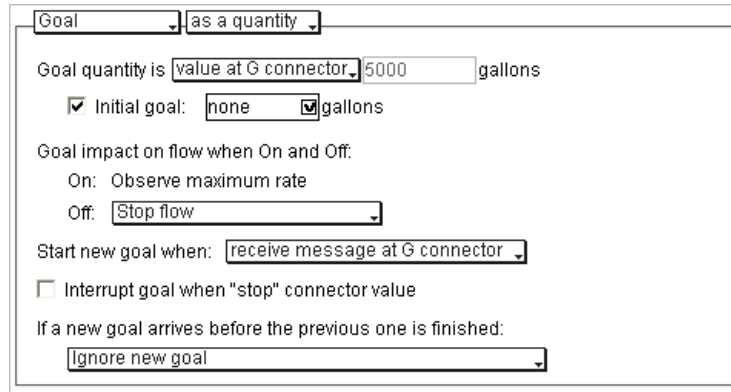
Since its initial goal has been set to “none”, the goal status is Off and flow through the Production Gate valve is stopped at the start of the simulation. However, when an item passes through the Get block (Item library) labeled Start Goal, a message is sent to the Production Gate valve’s G (goal) input connector. Once that happens, the valve’s goal switches from Off to On and the goal quantity is set to the value of the item’s Quantity attribute (5000 gallons).

In the bottom portion of the model, when the item leaves the Get block it moves into the Queue, where it is blocked from leaving by a Gate block. The Gate will remain closed until the Production Gate valve reaches its new goal of 5000 gallons. At that time the goal switches from On to Off, the flow stops (because that is the option set on the Control Flow tab), and the value at the GS (goal status) output is set to 3 (indicating that the goal has ended).

As a result, the Gate opens and the item moves into the Changeover Activity block where it is delayed for the amount of time required to perform a changeover. After the changeover has been completed, the item cycles back and initiates the next production cycle.

- By default, a Valve’s GS (goal status) output connector reports the following values:
- 0 when there is no goal
  - 1 when a goal is starting
  - 2 when a goal is in progress
  - 3 when a goal has ended
  - 4 when a goal is interrupted


The following screenshot shows how the Production Gate's quantity goal has been configured on its Flow Control tab:



Flow Control tab, goal is quantity

- The value for the Goal quantity is received through the G input connector. (In this model, the goal is 5000 gallons because each item has a Quantity attribute value of 5000.)
- The initial goal has been set to *none*. (This causes the goal to be in the Off state at the start of the simulation.)
- The Valve has been instructed to *Stop flow* when the goal is Off.
- A new goal is started each time the G input connector receives a message. (Getting a value at G causes two things to happen: 1) the goal switches from Off to On, and 2) the goal has a new quantity as defined by the value received at the G connector.)
- If a new goal is received before the previous one is finished, the new goal will be ignored. (In this model, a new goal cannot arrive before the previous one is finished.)

An interesting aspect of this model is that the Production Gate valve has been set to have an infinite maximum constraining rate. This means that it will, in and of itself, not limit the rate of the flow moving through it. However, the block's effective (actual) rate will be determined by the two upstream Valves. One of these Valves is connected to a Lookup Table block (Value library) that changes its maximum rate depending on the time of day. The other upstream Valve is connected to a Shutdown block (Item library) that causes the movement of flow to be stopped periodically for specified durations. This sequence causes some interesting effects in the model, and the Production Gate's effective rate ranges between 80 and 0 gallons/minute.

 This examples uses the G (goal) connector to control the “when” and “how much” aspects of the goal. Alternately, the Control Flow tab allows you to choose to start a new goal when the *start* connector receives a message or when the previous goal finishes.

### Setting a Valve's duration goal

Like the quantity goal, the duration goal is used to modulate a Valve's maximum rate. However, the duration goal cycles between the On and Off states as a function of time rather than the volume criteria used for the quantity goal.

A duration goal remains in the On state for some amount of simulation time before switching to the Off state:

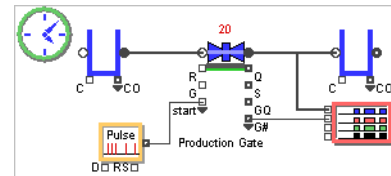
- While a duration goal is On, the value in the Valve tab's *Maximum rate* field, or the value at the block's *R* (maximum rate) input connector, is observed. The goal remains On until either the specified amount of time has passed, or the goal is interrupted. At that point the goal switches to the Off state.
- What happens when the goal switches to the Off state depends on the Off option selected in the block's dialog: stop the flow, do not constrain flow, or observe maximum rate.

 To interrupt a goal, send a value to the Valve's *stop* input connector.

**Duration Goal model**

The Duration Goal model is similar to the Quantity Goal model from above, except the goal allows the flow to move through the Valve for a certain period of time.

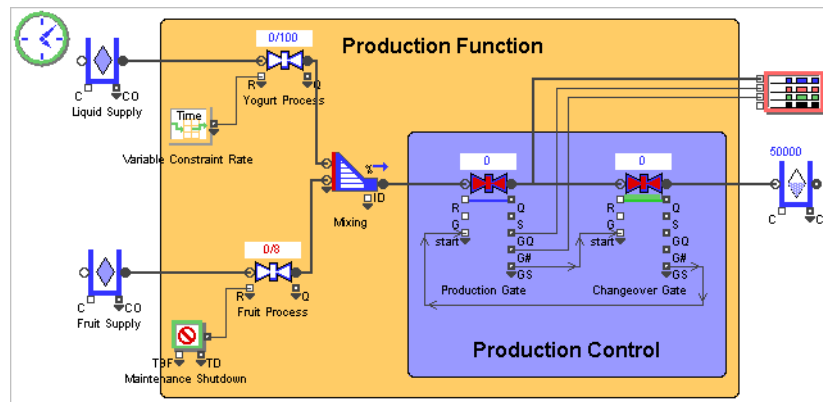
In this model, the Valve's Flow Control tab is set to observe its maximum rate of 20 gallons/minute for 45 minutes. When the goal is On, the block passes through whatever amount of flow it can, at the maximum rate of 20 gallons/minute. When the 45 minutes passes, the goal is set to Off and all flow through the block stops. The Pulse block (Value library) restarts the goal every 60 minutes. As the model runs, the block's maximum rate cycles from 20 gallons/minute to 0, depending on whether the goal is On or Off. As a result, the block's effective rate stays at 0 for the period of time (15 minutes) from when the previous goal ends until when a new goal starts.



Duration Goal model

**Changeover With Only Goals model**

The Changeover With Only Goals model is equivalent to the Changeover Quantity Goal model (discussed earlier) in terms of behavior. However, while the Production Gate valve still has a quantity goal in this model, the changeover is controlled using a second Valve with a duration goal, rather than by an item.



Changeover With Only Goals model

Since its Flow Control tab specifies that it has an initial quantity goal of 5000 gallons, the Production Gate valve starts the simulation by observing its maximum rate setting – infinity. However, as

Discrete Rate

## 340 | Delaying Flow

### Controlling a Valve's maximum rate

was true for the Changeover Quantity Goal model, the block's actual effective rate will be impacted by the Liquid Supply and Fruit Supply valves upstream.

The Production Gate's goal state is communicated to the Changeover Gate through the connection between the first block's *GS* (goal status) output connector and the second block's *start* input connector. Because of this connection and because of how the duration goal has been specified in the Changeover Gate, the duration goal starts in the Off state and will switch to On only after the Production Gate's quantity goal is completed. After 20 minutes, the Duration Goal is finished and the Changeover Gate sends a message to the Production Gate to start a new quantity goal.

Unlike the Duration Goal model that allows flow to pass through for a specified amount of time, the duration goal in this model causes flow to be blocked for a certain period. The Changeover Gate's maximum rate is set to 0 and the goal's duration is set to 20 minutes. While the goal is On, the block's maximum rate (0) is observed and no flow passes through, allowing for the changeover.

The following screenshot shows how the Changeover Gate's duration goal has been configured on its Flow Control tab:

Goal

Goal duration is    minute\*

Initial goal:

Goal impact on flow when On and Off:

On: Observe maximum rate

Off:

Start new goal when:  =

Interrupt goal when "stop" connector value

If a new goal arrives before the previous one is finished:

Flow Control tab, goal is duration

- The goal duration is a constant 20 minutes. The duration could be made variable by instead choosing *Goal durations is: value at G connector*.
- The 20 minute blocking of flow allows the changeover to occur. The maximum rate on the Changeover Gate's Valve tab is 0. This would cause the flow to be blocked in the absence of any goal being set for this block. Consequently, if the block has a duration goal and it is On, that maximum rate of 0 will be observed and flow will be blocked from entering. When the goal turns Off after 20 minutes, the Changeover Gate doesn't apply any constraining rate on the flow because it is set to *Off: ignore maximum rate*.
- A new duration goal is started only when the *start* input connector receives a value of 3. Consequently, a new duration goal begins only when the upstream Production Gate's quantity goal has finished.
- When the changeover has completed and the duration goal switches from On to Off, a signal is sent from the Changeover Gate to the Production Gate. This results in a new quantity goal starting in the Production Gate.

### Setting hysteresis in a Valve

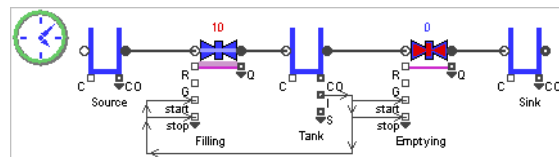
Hysteresis is a property of systems that causes them to not react instantly to a change. The purpose of adding hysteresis in a model is to introduce a delay in the time it takes some part of the system to switch from one state to another.

Hysteresis allows you to insert a lag or delay in a Valve's response to system requirements. It is used to avoid oscillations and to achieve better control over flow movement. This is accomplished by using model conditions to explicitly control both when a Valve's maximum rate is observed and when it is ignored.

Unlike the quantity and duration goals discussed earlier, where the conditions for applying the Valve's maximum rate were entered in its dialog, hysteresis must always get its control information from outside the block. The hysteresis option always relies on the Valve's *start* input connector to control when the Valve's maximum rate will be observed and its *stop* input connector to control when the maximum rate will be ignored. When the maximum rate is ignored, the Valve's dialog provides a popup menu for choosing if the flow stops or if the Valve does not constrain the flow.

#### Hysteresis model

In this model, the Filling valve opens when the Storage tank is empty and closes when the tank is full. Conversely, the Emptying valve opens when the Storage tank is full and closes when the tank is empty. As a result, the model repeatedly cycles through the following stages:



Hysteresis model

- Emptying valve closes and filling valve opens
- Storage tank starts accumulating flow
- Storage tank reaches the full state
- Emptying valve opens and filling valve closes
- Storage tank starts emptying
- Storage tank reaches the empty state

Based on settings in its Indicators tab, the Storage tank's I (indicator) connector outputs a value of 0 when empty and 2 when it is full. (Tank indicators are discussed on page 295.)

The Filling valve's Hysteresis settings are shown at the right. When this valve's *start* connector gets a 0 from the Storage tank's I output connector, the valve observes its maximum rate. When the Filling valve's *stop* input connector gets a 2, the block shuts down.

Observe Maximum rate when "start" connector	=	0
Ignore Maximum rate when "stop" connector	=	2
When ignoring:		Stop flow

Hysteresis settings: Filling valve

The Emptying valve's hysteresis settings are the opposite of the Filling valve, as shown at the right. When this valve's start connector gets a 2 from the Storage tank's I output connector, the valve observes its maximum rate. When its stop input connector gets a 0, the valve shuts down.

Observe Maximum rate when "start" connector	=	2
Ignore Maximum rate when "stop" connector	=	0
When ignoring:		Stop flow

Hysteresis setting for Emptying valve

### Delaying flow with the Shift block

The Shift block (Item library) is discussed fully in “The Shift block” on page 218. It is used to schedule capacity in certain blocks found in the Item and Rate libraries. Shifts come in two types: On/Off and Numeric.

Rate library blocks do not support a Number type of shift. The following Rate library blocks can be controlled using On/Off Shifts:

- **Convey Flow.** When the shift is Off, the effective inflow rate is set to 0, blocking any new flow from entering. Depending on which option has been chosen in the Convey Flow's dialog, the block's speed will either also be set to 0 or it will remain unchanged from what is set in the dialog. If the *Empty when shift is off* checkbox is checked, any flow already on the conveyor at the Off shift time will continue progress towards exiting the block.
- **Tank.** The effective inflow and outflow rates are set to 0 when the shift is Off, effectively shutting the block down.
- **Interchange.** Same logic as the Tank.
- **Valve.** Same logic as the Tank.

### Adding a Shift to a model

The easiest way to add a Shift block to a discrete rate model is to click the *Add Shift* button found on the Options tab of a Convey Flow, Tank, Interchange, or Valve block. This automatically does the following:

- Places a Shift block on the model worksheet below the originating block
- Enters the Shift name in the originating block's *Use Shift* field
- Opens the Shift's dialog so settings can be entered

To use the Shift, enter the required information in the block's dialog. (The Shift controls the originating block remotely; it does not need to be connected in the model.) Each Shift block starts with a default name for its shift. If you subsequently change the name of the shift, the new name will be reflected in the block that uses that shift.

For more information about using the Shift block, see “The Shift block” on page 218.


### Convey Flow block

Setting an initial contents and the capacity of a Convey Flow block is discussed in the chapter “Sources, Storage, and Units”. The current chapter focuses on the behavior of the Convey Flow block and how to set parameters that affect how flow is delayed through the block.

Flow entering the Convey Flow block is available to leave only after a specified delay that has been defined as a function of length and speed. The flow that enters is required to move some distance at a certain rate of speed before arriving at the block's exit point. The Convey Flow, then, is a resi-

dence block with flow distributed across its length at varying densities. (Density is the amount of flow that has accumulated at any one point on the conveyor.)

This block is useful for representing a conveyor, industrial oven, refrigeration system, or other similar piece of equipment with a length component where the position of flow must be taken into consideration.

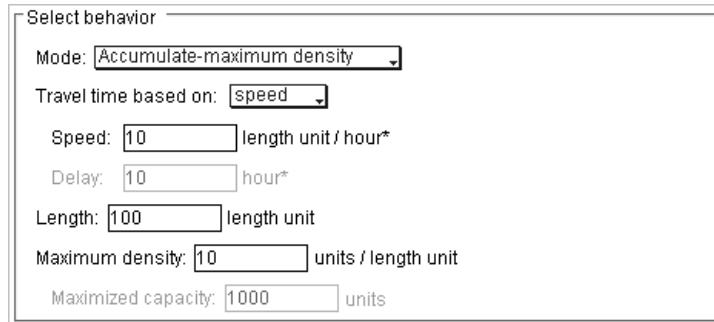
 A Convey Flow block is computationally intensive, so it should be used only if the system you are modeling requires very precise tracking of flow movement and position. For instances when the block should not be used, see page 346.

### Dialog settings

Movement of flow across the Convey Flow block is influenced by the dialog settings and parameters.

#### Determining speed and distance

The Convey Flow block offers two options: *Speed determines travel time* or *Delay determines travel time*. Depending on which of these is selected, the following entries can be made:



Dialog parameters for Convey Flow

- **Speed.** Specifies the maximum potential speed at which the conveyor can transport flow. However, when the potential supply of flow from the conveyor exceeds the downstream demand, the observed speed of the flow can become something less than the speed parameter.
- **Delay.** Represents the amount of time flow will spend in the block if there is no downstream blocking. If the block's dialog is set to *Speed determines travel time*, the delay is calculated by dividing Length by Speed. If the dialog is set to *Delay determines travel time*, the flow will take the entered Delay time to travel the stated Length.
- **Length.** Represents the distance flow must travel before reaching the block's exit point. The length of a Convey Flow block has to be greater than 0.
- **Maximum density.** Density is the amount of flow that has accumulated at any one point on the conveyor. The observed density of flow on a conveyor is a function of the upstream supply rate, the conveyor's speed, the downstream demand rate, and what settings have been chosen in the block's dialog. The Maximum density setting limits how high the pile of flow can be at any one point along the conveyor. For example, if the upstream supply rate is greater than or equal to the conveyor's maximum inflow rate (speed\*maximum density), then the amount of flow entering the conveyor will be equal to the maximum density. In this case, it is the conveyor's capacity to receive flow that limits its effective inflow rate.

- ☞ The parameters for speed or delay can vary dynamically during the simulation; the length and maximum density parameters remain fixed.

### **Convey Flow behavior**

The Convey Flow block is divided into segments, where the boundaries of each segment are defined by a change in the density of flow. Depending on the options chosen in the dialog, flow could accumulate or “pile up” along the length of the block any time the amount of flow ready to exit the block exceeds downstream demand.

The Convey Flow block has three options controlling how or if flow is allowed to accumulate:

- **Accumulate-maximum density.** Allows flow to accumulate up to its maximum density setting. If the conveyor's ability to deliver flow exceeds downstream demand, any flow delayed from exiting will begin piling up at the outflow end of the conveyor up to the maximum density level.
- **Accumulate- fill empty segments.** Allows flow to fill in any empty segments along the conveyor when the block's ability to deliver flow exceeds downstream demand. (An empty segment is an area along the block's length that has a density of 0.) This differs from the first option in that one section of flow is not allowed to pile onto another section of flow.
- **Non-accumulating.** This option does not allow flow to accumulate. Therefore, the block's speed slows when its ability to deliver flow exceeds downstream demand.

- ☞ The Compare Convey Flow model compares the behavior of three Convey Flow blocks, each set to one of these behaviors, under different emptying rates.

### **Constraining rates**

Critical constraints define an unconditional maximum upper bound to the rate of flow. As discussed in the chapter “Rates, Constraints, and Movement”, the Convey Flow block calculates critical constraints for its inflow and outflow connectors separately. The critical constraints are derived from model conditions and settings in the dialog.

- The critical constraint for the Convey Flow block's inflow is calculated by multiplying the block's effective speed by its maximum density entry.

- ☞ The effective speed can be less than or equal to the speed set in the dialog. If the block is non-accumulating, or if it is accumulating but cannot accumulate more, and the block's ability to deliver flow exceeds downstream demand, the effective speed will be lower than the entered speed.

- The critical constraint for the block's outflow is the result of the multiplication of the block's speed setting by the density of flow present at the outflow end of the block.

### **Convey Flow information**

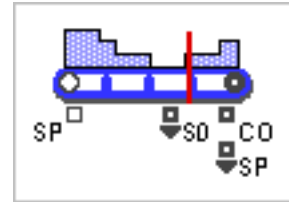
The Convey Flow block provides different types of information concerning the distribution of flow along the length of its conveyor. Some of the information is provided by default while other mechanisms for reporting data, like sensors and indicators, must be customized through the block's dialog.



**Distribution of flow**

If you select Run > Show 2D Animation before you run the simulation, the distribution of flow in the Convey Flow block will be displayed across the top of its icon, as shown on the right.

On the Convey Flow block's Animation tab, selecting *Show block's flow distribution in table during simulation* causes a table to appear. As the simulation runs, the table displays information about the current distribution of the flow in each segment of its length:



Flow and accumulation point

Show block's flow distribution in table during simulation:

	High Limit	Low Limit	Density	Quantity
0	10 feet	9 feet	4 tons/feet	4 tons
1	9 feet	7 feet	2 tons/feet	4 tons
2	7 feet	5 feet	0 tons/feet	0 tons
3	5 feet	3 feet	2 tons/feet	4 tons
4	3 feet	2 feet	4 tons/feet	4 tons
5	2 feet	0 feet	6 tons/feet	12 tons

Link

Table showing example distribution of flow

The length of a Convey Flow block is divided into segments, where the boundaries of each segment are defined by a change in the density of flow. The table above indicates that this Convey Flow block currently has 8 segments along its length.

**Accumulation point**

If the block is set to be accumulating, any accumulation will start at its outflow and go toward its inflow. The point beyond which no more flow can accumulate is known as the *accumulation point*. This point will probably move between the inflow and the outflow as the simulation runs.

The Results tab reports information about the accumulated flow: the distance from the outflow where the accumulation point is located, the indicator, and the accumulated quantity of flow.

If the command Run > Show 2D Animation is checked while the model runs, and there is accumulation, a vertical red bar will move on the block's icon during the simulation; this indicates the location of the accumulation point. The "Flow and accumulation point" screenshot above shows the red line of the accumulation point.

The accumulation point is located somewhere along the length of a Convey Flow block that has been set to accumulate flow.

**Sensors**

The Convey Flow block has a Sensors tab for specifying the locations and trigger points of sensors along the length of the conveyor. Each sensor reads and communicates the density of flow over time at a particular point on the conveyor. This information is displayed in a table in the block's dialog and reported by the block's S (sensor) output connectors – one S output for each sensor.

You must specify in the table not only the number of sensors desired but also the location of each one. (Use the +/- button in the table's lower right corner to specify the number of sensors.) The example

Locations are: absolute numbers Sensor result is: the density

	Location	Density Trigger	Result
0	10 feet	0 tons/feet	6 tons/feet
1	7.5 feet	0 tons/feet	2 tons/feet
2	2.5 feet	0 tons/feet	4 tons/feet
3	0 feet	0 tons/feet	6 tons/feet

Link Show Example

Sensor tab table

table shown here indicates that four sensors have been placed along a ten foot section of the conveyor.

- ☞ The S connectors should be used judiciously. Extra events are used to update them at the proper time and the calculation is computationally intensive.

### **Indicators**

As discussed in “Accumulation point” on page 345, the accumulation point indicates the point beyond which no more flow can accumulate. You might want an indication when the accumulation point is within a particular segment of the Convey Flow block.

The Indicators tab on a Convey Flow block is used to define segments to indicate where the accumulation point is along its length. Each segment is assigned a name, a defined range, and an ID number. The ID number is used to update the block’s I (indicator) output connector as the accumulation point moves from one segment to the next.

See “Indicators” on page 295 for complete information about creating and using indicators.

- ☞ The I connectors should be used judiciously. Extra events are used to update them at the proper time and the calculation is computationally intensive.

### **When to avoid using the Convey Flow block**

A Convey Flow block should be used only if the system you are modeling requires very precise tracking of flow movement and position. While the block is very precise, it is also very time consuming, so use it with caution. Following are some usage guidelines:

- The travel time has to be long enough to justify using the Convey Flow block. If the impact on the result of the simulation is small, it is a good strategy to ignore any travel time delays. For instance, the amount of time required for product to traverse a pipe separating two tanks is often insignificant and should usually be ignored.
- The distribution of flow along the Convey Flow block should impact the rest of the model in some significant way. If the effective inflow rate varies significantly during the simulation and/or if the speed changes, the product may be unevenly distributed across the length of the conveyor. This discontinuity impacts the rate at which flow is able to exit the block over time. The greater this variation in availability, the greater the potential impact on the rest of the model and the greater the block’s use can be justified.

One alternative would be to use a combination of Tank and Valve blocks to mimic a Convey Flow block. This is far less computationally intensive than the Convey Flow block by itself. While some configurations may not be as precise, you can use a combination of one Tank followed by one Valve block in such a way that the behavior is identical or almost identical to the results when using a Convey Flow block.

Another alternative is shown in the Tank and Valve to Convey model. In this example, the bottom flow stream behaves almost identically to the upper stream without using a Convey Flow block. Because Valves A1 and B1 vary their constraining rate, the Convey Flow block (Convey A) will get and create a lot of messages, slowing down the simulation. But the Convey B1 tank won’t create the extra events. This will be slightly less precise but much more efficient.

- ☞ Never use a Convey Flow block if the system’s behavior can be modeled using a Tank and Valve.

# **Discrete Rate Modeling**

## **Mixing Flow and Items**

Mingling discrete rate flows with discrete event items.

It is common for systems to exhibit a mixture of behaviors, where items from the discrete event arena intermingle with discrete rate processes. In these “mixed mode” cases, a proper understanding of both the Rate and Item libraries and how they can interact with each other is required.

There are two general techniques for integrating discrete event blocks from the Item library with discrete rate blocks from the Rate library:

- 1) Sending signals and sharing information via value connections. Value connections can be very useful for triggering some type of action as the system moves from one state to another. For example:
  - Value connections can trigger the generation of an item when the level of flow in a Tank reaches a certain level.
  - The value of an attribute on an item passing through some part of a discrete rate model could trigger a change in the constraining rate in a Valve block.
- 2) Mixing items with flow using the Interchange block, as you saw in the Discrete Rate tutorial on page 284. The Interchange block (Rate library) provides the ability for items and flow to interface with each other. For example, an empty tanker truck (an item) might arrive at a refinery and fill with gas (flow) at a continuous rate using an Interchange block. Once full, the truck might be routed through a series of Item library blocks until it reached its unloading destination. At that point the item would again interface with an Interchange block to discharge its load.

This chapter focuses on the techniques used when the need for “mixed mode models” arises. It will show how to:

- Control flow using blocks from the Item library
- Control items with blocks from the Rate library
- Mix flow with items using the Interchange block

 The models for this chapter are located in the folder \Examples\Discrete Rate\Flow and Items.

### **Controlling flow with items and items with flow**

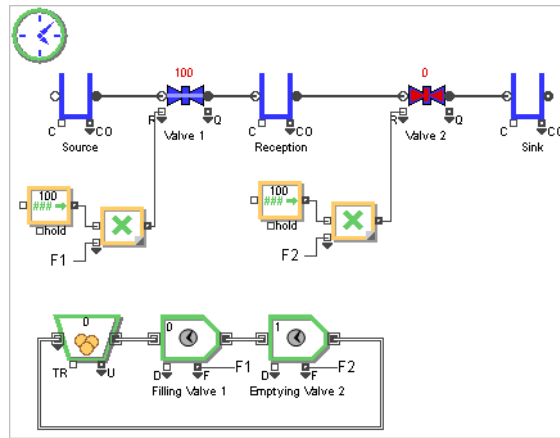
Rate library blocks can control the movement of items and Item library blocks can control the movement of flow. In either case, value connectors communicate state changes to the controlling blocks.

#### **Items controlling flow**

The Item Controls Flow model shows one of many ways blocks from the Item library can be used to control the movement of flow in a model. It uses one item to open a valve, allowing a tank to fill, then uses the same item to trigger the emptying of that tank.

**Item Controls Flow model**

Unless the presence of an item in an Activity block triggers them to open, Valves 1 and 2 are both closed during the simulation run. While an item is in the Activity block (Item library) labeled Filling Valve 1, it causes Valve 1 to open; this allows the Reception tank to fill. When the item moves to the Activity block labeled Emptying Valve 2, it causes Valve 1 to close and Valve 2 to open. This allows the Reception tank to empty into the Sink tank. The closing and opening of the valves is accomplished by detecting which Activity block has the item and for how long it is held there.



Item Controls Flow model

In this model, constant blocks (Value library) provide a potential constraining rate of 100 for the valves. Multiplying the constant value by an Activity's F (Full) output (which will be 1 if the Activity has the item in it and 0 if it does not) gives the valves their actual constraining rate (100 or 0). This causes the valves to open and close depending on where the item is in the model. Both Activity blocks are set to delay the item for a random amount of time, which represents how long the Reception tank can fill and empty.

Notice that the Reception tank can be full even while the item is still in the Filling Valve 1 activity. Since the tank can't accept any more flow when it is full, and can't start emptying until the item moves to the Emptying Valve 2 block, the flow stops before the Filling process has been completed. The next model shows how the Reception tank can let the item know that it is full.

**Flow controlling items**

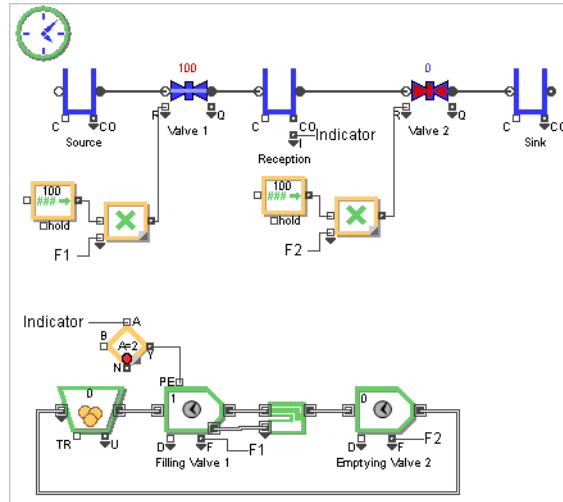
The example below illustrates one of ways blocks from the Rate library can be used to control the movement of items in a model.

**Flow Controls Item model**

This model extends the one above, adding that the state of the Reception tank alters the movement of the item and interrupts the filling process.

As in the prior model, the Reception tank fills and empties based on the movement of an item and delays set in the two Activity blocks.

In this model, whenever the Reception tank is full, it sends an indicator signal to the Decision block (Value library). This block in turn notifies the Activity block labeled Filling Valve 1, preempting the item whose presence opened Valve 1 and allowed the tank to fill. When the item is preempted, it moves to the Activity block labeled Emptying Valve 2. This causes Valve 1 to close, Valve 2 to open, and the Reception tank to empty into the Sink for the duration specified in the second Activity block.



Flow Controls Item model

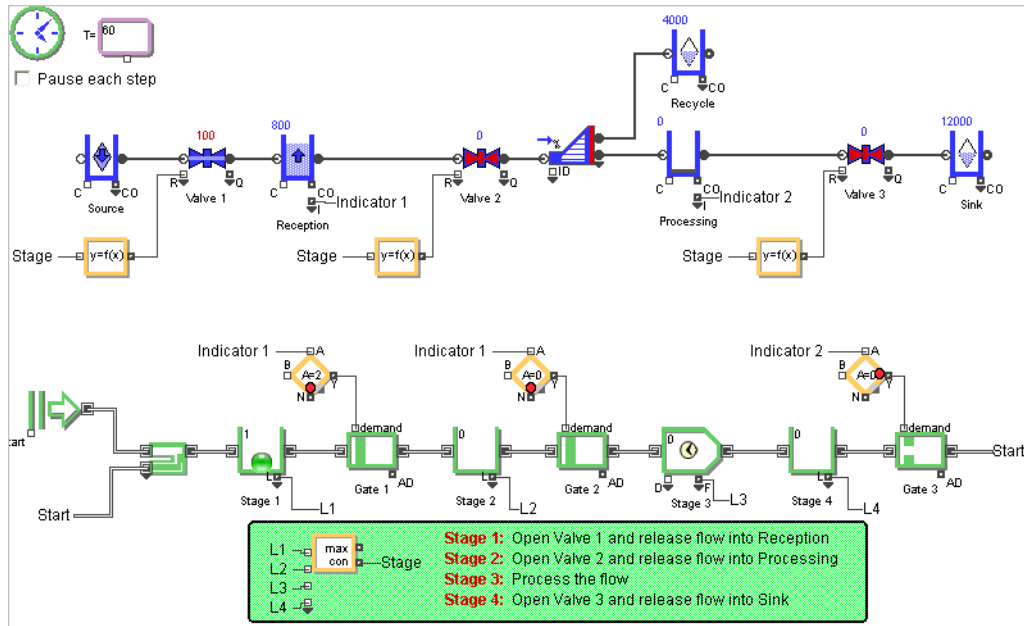
**Flow controlling items and items controlling flow**

The following model combines both of the previous concepts – flow controlling items and items controlling flow – into an even more complex system.

**Step The Flow Process model**

In this model the location of a “cycling item” triggers the opening and shutting of valves for the flow, while tank states control the opening and shutting of the gates for the item. The control logic is circular in nature: the item’s location defines the current stage; the current stage controls the

opening and shutting of valve blocks; valves impact the level of flow; and the level of flow controls the item's location.



Step The Flow Process model

**Flow controlling the item**

The top section of the model is where flow controls the item. The I (Indicator) connectors on both the Reception and Processing tanks (Rate library) control the three Gate blocks (Item library) used in the item stream at the bottom part of the model. Gate 1 is open only while the Reception tank is in the *full* state. Conversely, Gate 2 is open only while the Reception tank is in the *empty* state. Finally, Gate 3 is open only while the Processing tank is empty.

**Item controlling the flow**

The bottom section of the model is where the item controls the flow. In this case, the three Valve blocks (Rate library) in the top part of the model are controlled by the lengths of the three Queue blocks (Item library) in the item stream in the bottom portion. The item stream has only one item which cycles through the item-based blocks in a loop. Consequently, the queue lengths for each Queue in this loop will alternate between 1 and 0. For instance, Valve 1 is open only when the length in the Stage 1 queue equals 1. Similar logic applies to Valve 2 and Valve 3.

The activity in the model is divided into four stages: open Valve 1 and release flow into Reception; open Valve 2 and release flow into Processing; process the flow; open Valve 3 and release flow into Sink

**Stage 1: Open Valve 1 and release flow to Reception**

The I (Indicator) connector on the Reception tank is controlling Gate 1. At the start of the simulation, Gate 1 is closed because the Reception tank is not full (it's actually empty at this point), and the cycling item is blocked from leaving the Stage 1 queue. Notice the highlighted area where the Max & Min block (Value library) is used to translate the cycling item's location into a stage

number. Since the cycling item remains in the Stage 1 queue, the model is now in a Stage 1 holding pattern. Consequently, Valve 1 is opened (so the Reception tank starts receiving flow) and Valves 2 and 3 are closed.

#### **Stage 2: Open Valve 2 and release flow to Processing**

Once the Reception Tank reaches the full state, Gate 1 is opened and Gate 2 is closed. This allows the item to move on to the Stage 2 queue. The result is that Valve 2 opens, Valves 1 and 3 close, and flow starts moving from the Reception tank into Recycling and Processing. Gate 2 remains closed while the Reception tank empties.

#### **Stage 3: Process the flow**

Once the Reception tank is completely empty, Gate 2 is opened, Gates 1 and 3 are closed, and the item enters an Activity block labeled “Stage 3”. While the item remains in Stage 3, all three Valve blocks remain closed. The Activity block, which has a delay of 2 minutes, is used to keep the flow in the Processing tank for some period of time so it can be processed.

#### **Stage 4: Open Valve 3 and release the flow**

Once processing is completed, the item leaves Stage 3 and moves into the Stage 4 queue. Since Gate 3 is currently closed (because the Processing tank is not empty), the model is now in a Stage 4 holding pattern and Valve 3 opens. Once the Processing tank is empty, Gate 3 opens and the item cycles back to the State 1 queue where the whole processes starts all over again.

☞ While this model is useful for demonstrating how mixed mode models can control item and flow movement, the same behavior can be created without items using the Valve block’s Hysteresis option. For more information, see “Setting hysteresis in a Valve” on page 341.

### **Using the Interchange block to mix items with flow**

The Rate library’s Interchange block is unique in that it allows items and flow to interface directly with each other. Flow can enter the Interchange block not only through its inflow connector but also through the arrival of an item. Conversely, flow can exit the block through its outflow connector or through the exiting of an item.

The use of the Interchange block was introduced on page 284 of the Discrete Rate Tutorial, and the block’s capacity to hold flow is discussed starting on page 292 of the Flow Sources, Storage, and Units chapter. This chapter will describe how the Interchange block interfaces flow with items.

☞ The Interchange block is where an item can be filled with flow or emptied of flow.

There are a number of occasions where it can be useful to provide items with the ability to store, transport, and empty flow as they move from one section of a discrete rate model to another. For example, the attribute capabilities used to distinguish one item from another can also be used to distinguish one block of flow carried by one item from another block of flow carried by a different item. This can be an especially useful modeling construct since flow units by themselves are indistinguishable from each other.

#### **Behavioral rules**

The Interchange block has two very different modes (“Tank only exists while item is in it” and “Tank is separate from item”) that affect how the block behaves. (These modes were introduced on page 292 and will be discussed more fully on page 354.) Even so, the Interchange block always follows a fundamental set of rules:

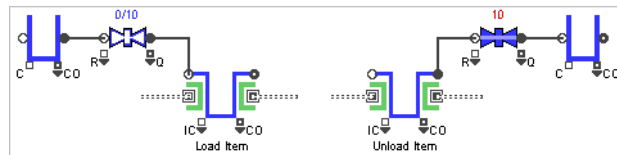
- The item input and item output connectors on the Interchange block must both always be connected.



- At least one of the Interchange block's inflow/outflow connectors must be connected (both may be connected as well).
- The Interchange block's capacity for holding items is permanently fixed at one item.
- The Interchange block loads and unloads the item and its flow instantaneously. (Flow present in the item when it enters the block is instantaneously available to the block; flow leaving the block with an item is instantaneously removed from the block.)

**The flow connector configuration**

The behavior of the Interchange block is dramatically affected by how the inflow and outflow connectors have been connected. As mentioned above, only one of the two flow connectors needs to be connected. If only the inflow connector is connected, arriving items can only be filled with flow, and if only the outflow connector is connected, arriving items can only unload flow.



Arriving item is always filled (left) or is always emptied (right)

**Item release conditions**

The release conditions determine when an item is scheduled to leave the Interchange block; they are the same for both block modes. Releases can also be accomplished at any time using the Preempt connector.

**Scheduled releases**

There are 5 options for defining when the item should be released:

- When contents  $\geq$  Target. This option requires the item to be filled with a certain amount of flow prior to release. The Target amount is entered in the dialog.
- When contents  $\leq$  Target. Requires the item's flow level to empty to a certain point prior to release. The Target amount is entered in the dialog.
- As soon as possible. Releases the item whenever there is downstream item capacity for the item, irrespective of the current flow level.
- Only with preempt connector message. Releases the item when a true value is received at the block's PE (preempt) value input connector.
- When level reaches indicator. Requires the flow contents to reach a certain level prior to release. Indicators (segments that indicate the level of contents) must first be entered on the block's Indicator tab for this option to be used.

when contents  $\geq$  Target (load process)  
 when contents  $\leq$  Target (unload process)  
 as soon as possible  
 only with preempt connector message  
 when level reaches indicator:

Release options for Interchange block

**Preemption**

The scheduled release conditions can be superseded at any time by using the preempt connector. Whenever the pre-

Release item when "Preempt" connector value

Preemption options for Interchange block

empt connector receives a value that corresponds to the preemption options selected in the Item/Flow dialog, seen above, it will trigger a preempt.

- To immediately dispose of an item after it releases flow, connect an Exit block (Item library) to the Interchange block's item output connector. To instead have an item present all the time, connect a Create block set to *Create items infinitely* to the block's item input connector.

### Interchange modes

The Interchange block has two modes:

- Tank only exists while item is in it
- Tank is separate from item


These are illustrated below.

#### Tank only exists while item is in it

In this mode, the Interchange's capacity to handle flow is completely dependent upon the presence of an item. The arriving item can be thought of as a "tank" with a capacity to hold flow. This item/tank can move through the item-based blocks just like any other item would. However, once it enters an Interchange block, the item/tank can release flow directly into the block's outflow connection and/or accept flow directly from its inflow connection. In the absence of the item/tank, the Interchange block has no flow capacity.

Two very important behaviors result from an item exiting the block when it is set to this mode:

- Once the conditions for item release have been met, the exiting item will always take with it any flow currently residing in the block.
- Until a successor item arrives, the Interchange's inflow and outflow will be blocked.

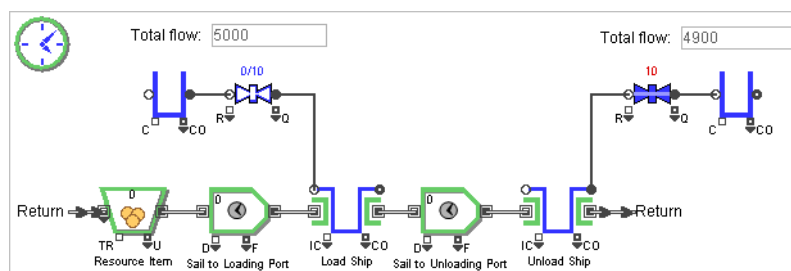
-  In this mode, the absence of an item eliminates not only the Interchange's capacity to hold flow but also its capacity to pass flow from an upstream source to a downstream sink. (This differs from the behavior of a Tank block, which passes flow through even if its capacity has been set to zero.)

#### Shipping model

A typical example of how the Interchange block could be used in this mode is illustrated by the Shipping model. In this example, an empty ship (an item) arriving at a loading port (an Interchange block) where it is filled with cargo according to a filling rate.

Once full, the ship sails for a period of time (represented by an Activity block) until reaching the new destination port (another Interchange block). At this point the ship's cargo is unloaded according to the unloading rate.

To simulate the loading process, flow is piped from the Interchange block's inflow connector into the item/ship. Once filled, the item/ship exits the block, taking the flow with it. Conversely, once

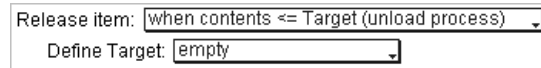


Shipping model

Discrete Rate

the item/ship arrives at the second Interchange block, flow is piped from it into the second Interchange block's outflow connector.

The item/ship travels with a *Quantity* attribute that has been set to a value of 1000. This attribute sets the ship's capacity. For the filling process, the Interchange releases the ship when it is full, that is, after 1,000 units of flow have been piped in. The second Interchange block releases the ship when it is empty, that is, after 1,000 units of flow have been piped out. At the end of the simulation run, the ship has not yet been released from the unloading dock because it still contains 100 units of flow.

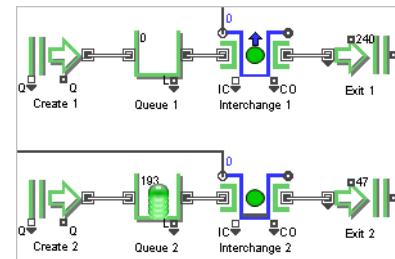


Release options for unloading

*Yogurt Production model*

The Yogurt Production model, located in the folder \Examples\Tutorial\Discrete Rate and discussed in the Discrete Rate Tutorial that starts on page 274, is an example of using an Interchange block set to *Tank only exists while item is in it*.

In this model, empty item/pallets are generated randomly by a Create block (Item library). The arrival of an item/pallet causes the Interchange block to have flow capacity; the maximum capacity of 24 cartons is entered in the block's dialog. Once the maximum capacity is reached, the full pallet leaves the block. The Interchange block then has no capacity until another pallet arrives.

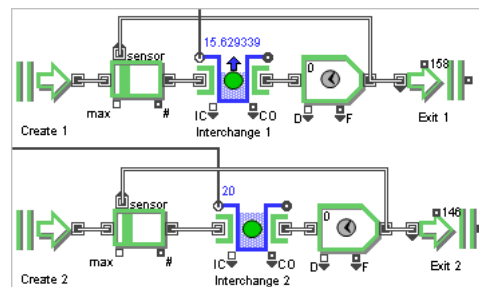


Yogurt Production palletization

*Yogurt Changeover model*

This model is based on the Yogurt Production model from above. The Yogurt Changeover model, however, provides a period of time (the changeover) for an operator to remove each full pallet and replace it with an empty pallet, and there is an infinite supply of empty pallets.

In this model, each Create block (Item library) can generate an infinite number of items so that empty pallets are available as required. Activity blocks (Item library) represent the two minute changeover time. Once a pallet has been released from an Activity, it notifies a Gate block (Item library) to open, pulling in a pallet from the Create block. This process causes the Interchange to not have a new pallet/item to fill until the changeover from the previous pallet is finished.



Yogurt Changeover palletization

The Create block's ability to *Create items infinitely* is specifically intended for this type of situation. It provides an infinite supply of items that are available **on demand**, creating items whenever there is a capacity for flow.

**Tank is separate from item**

In this mode, the Interchange block's ability to handle flow is identical to the Tank block irrespective of the presence of an item. That is, the Interchange can be set to have an initial amount of flow, its capacity can be set through the dialog or through a connector, and maximum inflow/outflow rates can be defined.

The only difference lies in the Interchange block's ability to pipe flow into and out of items. When the Interchange block is set to *Tank is separate from item*, the block can receive and hold flow that has been provided by its inflow connector or the arrival of an item and it can release flow through its outflow connector or through the exiting of an item.

- When the Interchange block is in this mode, an item carrying flow releases its entire load instantaneously upon arrival. However, depending on other settings in the block, the item can also take flow with it upon exiting. If this is the case, the Interchange block's flow level is decremented instantaneously when the item leaves. Furthermore, an item whose load of flow exceeds the block's capacity will be blocked from entering the Interchange.

*Bucket Elevator 1 model*

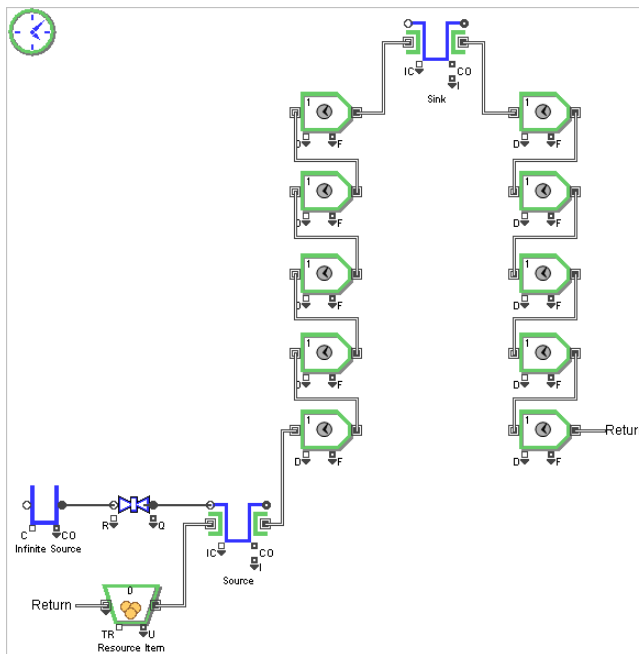
The Bucket Elevator 1 model simulates a series of buckets (items) pulling flow out of a source (an Interchange block) located at a low elevation, transporting the water in a series of steps to an infinite sink located at a higher elevation (another Interchange block), and then returning empty through a series of steps to the source. Both Interchange blocks are set to *Tank is separate from item*. With this setting, both the source and the sink have the capacity to hold water even in the absence of an item.

This model is similar to a continuous loop of buckets drawing water from a well, emptying into a catch basin, and returning to the well. There are ten bucket/items and each bucket has a capacity of 100 gallons; this is defined in the Resource

Item block by the attribute Capacity. There are also ten slots in this “pseudo-conveyor” – each represented by an Activity block that can hold one item/bucket. The delay at each slot is 1/10 the sum of all the delays, as determined by the items' Speed attribute.

Even if the source has less than 100 gallons, the buckets keep moving and grab as much water as possible. As each bucket reaches the top, its contents are released instantaneously into the sink, and the journey back down to the sink immediately starts. Running the simulation with animation on shows the buckets as they cycle from the well, up to the catch basin, and then back down again.

Discrete Rate

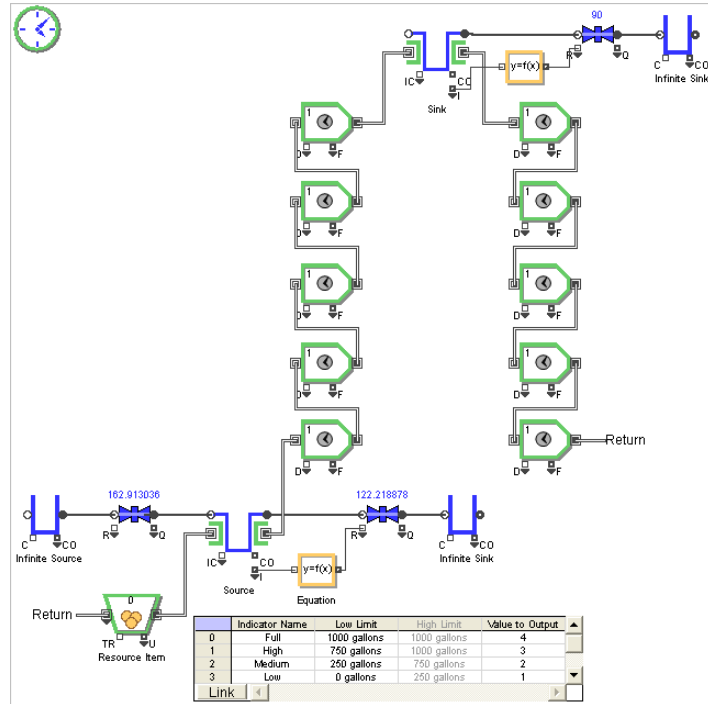


Bucket Elevator 1 model

*Bucket Elevator 2 model*

The Bucket Elevator 2 model is similar to the Bucket Elevator 1 model, except there is some added complexity. In this model, the flow blocks at the bottom and top of the model represent streams of water. In the bottom stream, some of the water is removed by the buckets, but the rest continues on at a varying rate of flow. Likewise, after the buckets have unloaded their contents into the sink, water flows from the top stream at a varying rate.

Both Interchange blocks have a capacity of 1,000. Based on settings in the Interchange block's Indicators tab, the level of water affects the constraining rate of the Valve that follows each of the blocks.



Bucket Elevator 2 model

Indicators are a method of reporting what category or range the current level of flow falls into; the table from the Source block's Indicators tab is shown to the right. An Equation block (Value library) looks at the Interchange block's I (indicator) output to determine the range the water level in the tank falls into. The block then adjusts the Valve's maximum rate depending on that information and an equation.

Indicator Name	Low Limit	High Limit	Value to Output
0 Full	1000 gallons	1000 gallons	4
1 High	750 gallons	1000 gallons	3
2 Medium	250 gallons	750 gallons	2
3 Low	0 gallons	250 gallons	1

Indicators for the source

The upper and lower streams in this model have different equations; the upper stream bases the Valve's maximum rate on a constant while the lower stream bases it on a random distribution. In the lower stream for instance, if the Source tank's level is equal to or less than the Low range, the maximum rate is a Triangular distribution that is most likely 30 units. If the level falls within or above the High range, the most likely maximum rate for the Valve is 150; otherwise, the most likely rate is 100.

For more information, see "Setting indicators" on page 296.

358 | **Mixing Flow and Items**  
Using the Interchange block to mix items with flow

Discrete Rate

# **Discrete Rate Modeling**

## **Miscellaneous**

Concepts that don't easily fit into other chapters

This chapter covers:

- The precision of calculations
- Using bias to give preference to some component or portion of a model and how that affects effective rates
- Global and advanced options in the Executive
- Value connector abbreviations and meanings
- The different types of information that animation displays

## Precision

An LP area is made up of one or more rate sections; it encompasses all the rate sections for which the Executive block has been notified that effective rates might change. The LP area has a linear program (LP) that is responsible for calculating an effective rate for each section contained within that area. (The LP area and LP calculations are discussed fully in “LP technology” on page 376.)

The maximum mathematical precision for an LP area is 12 digits. Because one LP can be responsible for calculating multiple effective rates for its rate sections, and because LP precision is limited to 12 digits, precision can become an issue not only for the individual effective rates but also for the effective rates calculated for the entire LP area. For example, if an LP area contains two rate sections where the first rate section's effective rate was 1,000,000 flow units per time unit (FPT), the effective rate for the second section could be no smaller than 0.0001 FPT.

 To preserve adequate precision for all rate sections, don't separate any two effective rates within an LP area by more than 12 digits of precision.

## Biasing flow

The discrete rate architecture includes a feature called *bias* – a method for stating a preference that flow travel one route rather than another.

 Bias is only relevant when the flow is merged or diverged.

The advantages of the bias concept are that it:

- 1) Gives you a way to specify preferences for how flow circulates in one part of the model compared to other parts.
- 2) Provides flexibility in resolving conflicts for how flow should be distributed among competing branches.

Bias is present in a model whenever you use one or more of the following blocks:

- A Bias block in a model that contains Merge or Diverge blocks set to a non-fixed mode (discussed in “Merge and Diverge blocks” on page 362).
- A Merge or Diverge block (Distributional, Priority, or Sensing modes only)

Because bias can skew the way flow is distributed, it is taken into consideration by the global LP calculation and can thus have an effect on effective rates. (For specific information on how bias is used in the calculation of effective rates, see the advanced topic “LP technology” on page 376.)


 We suggest that you read the chapters “Rates, Constraints, and Movement” and “Merging, Diverging, and Routing Flow” before the Bias section.



### Bias order


If a block in a model has bias, it has a *bias order* that indicates its ranking compared to all the other blocks with bias. Each biasing block is listed in order from the top (strongest) bias order to the lowest (weakest) bias order. The block at the top of the ranking list has a bias order greater than 0. Bias orders lower than the top have numbers higher than the top number; bias numbers that are Blank or less than or equal to 0 are ignored.

Since the bias order is used during the LP calculation of effective rates, changing the bias order often results in a different set of effective rates. It is therefore important to understand the concept of bias order and the influence it has on how effective rates are calculated.

 The bias of a Bias block is by definition stronger than the biasing effect of any Merge or Diverge block. So Bias blocks will always have higher bias orders than Merge and Diverge blocks.

### Bias block

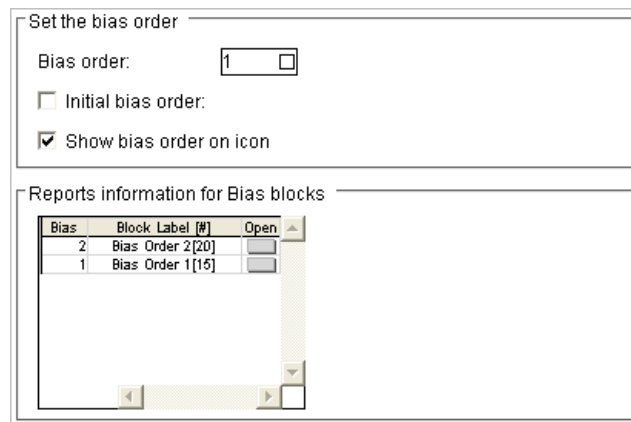
The Bias block allows you to specify a preference for where the flow should be directed. Wherever the Bias block is located in the model, it pulls in as much flow as possible. If a model has multiple Bias blocks, each has its own bias order.

 If the model's Merge and Diverge blocks use non-fixed rules to obtain or distribute flow, there is some leeway in how flow can be biased and the Bias block is useful. If the model's Merge and Diverge blocks all use fixed rules, there is no possibility of biasing the flow with the Bias block. (Fixed and non-fixed rules are discussed in "Merge and Diverge blocks" on page 362.)

### Dialog settings and bias order

The block's bias order can be set directly in its dialog or it can be modified dynamically through the B (bias) input connector or by linking the *Bias order* dialog parameter to an ExtendSim database. You can enter an initial bias order that has effect until the block gets a bias order value dynamically; you can also show the block's bias order on its icon.

The Bias dialog shown at the right is for one of two Bias blocks in a model. The block shown has the highest preference for flow, as indicated by the setting *Bias order: 1*.



Bias dialog

The dialog table reports information about each Bias block in the model, its bias order, block label or name, and block number. You can use the table's *Bias* column to change the bias order for any of the listed blocks.

### Calculation of the effective rate

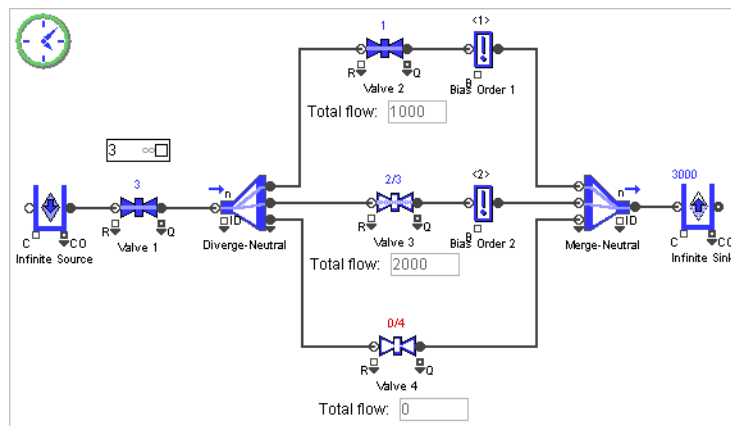
The preferences for flow defined by Bias blocks has an effect on the calculation of the effective rate. If there is more than one Bias block in the model, each block's preference is expressed in turn based on its bias order. As each Bias block takes its turn, it calculates the maximum effective rate which could circulate at its location, without taking into consideration the preferences expressed by

blocks with a lower bias order. When its maximum effective rate has been determined, that rate will be fixed for the succeeding calculations involving blocks with lower rankings.

- The bias order of Bias blocks can change dynamically during the simulation
- If multiple Bias blocks have identical bias orders, the way the flow is distributed between the effective rates cannot be predicted. The model will use one of the possible solutions.
- If a Bias block has a bias order that is Blank or is less than or equal to 0, the block does not express any preference.

### Prioritize With Bias Blocks model

This model illustrates how Bias blocks can be used to indicate preferences for flow. It is similar to the Competing Requests for Flow model discussed on page 327. However, this model uses Bias blocks to indicate the preferred route for flow, rather than bias order settings in Merge and Diverge blocks set to Priority mode.



Prioritize With Bias Blocks model

In the Prioritize With Bias Blocks model, the Merge and Diverge blocks are set to Neutral mode and the Bias blocks indicate flow preferences. The results are identical to the Competing Requests for Flow model.

This model is located in the folder \Examples\Discrete Rate\Merge and Diverge.

### Merge and Diverge blocks

As shown in the “Mode table” on page 319, some Merge/Diverge modes use a fixed rule to obtain or distribute the flow. For other Merge/Diverge modes, flow rules are only invoked in specific situations depending on model conditions.

Because it influences the way flow is distributed, the bias concept only applies to Merge or Diverge blocks set to non-fixed rule modes: Distributional, Priority, or Sensing. To avoid confusion when Merge or Diverge blocks have competing requests for flow, blocks with these modes must specify a bias order.

#### Fixed rule modes

The Batch/Unbatch, Proportional, and Select modes all use a fixed flow rule to obtain or distribute flow. For example, the way flow is distributed between the branches for a Merge/Diverge in Proportional mode will follow the proportions set in the block’s dialog. No matter what happens in the rest of the model, the proportions will be respected.

The bias setting options in these block’s Model Settings tabs will be disabled.

If the model contains Merge and Diverge blocks that are **only** set to the Batch/Unbatch, Proportional, or Select modes, bias has no impact on the effective rates.

**Non-fixed rule modes**

The Distributional, Neutral, Priority, and Sensing modes do not use a fixed rule to obtain or distribute flow. Instead, they provide a certain degree of freedom about where the flow can be directed.

A Merge or Diverge block in Priority mode, for example, impacts the flow as follows:

- Taking into consideration how much flow it can get, the block will do its best to direct as much flow as possible to its top priority branches.
- However, the block just expresses a “preference” for where to send the flow; model conditions determine how well those preferences can be achieved and the top priority branches may not actually get the most flow.

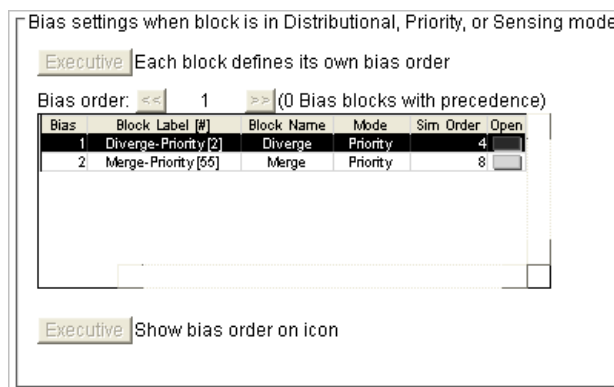
If the model contains **any** Merge or Diverge blocks set to the Distributional, Priority, or Sensing modes, the bias order must be specified. (There is no bias order required for the Neutral mode.)

**Setting a Merge or Diverge block’s bias order**

When set to the Distributional, Priority, or Sensing modes, the Merge and Diverge blocks must express a bias order. This is accomplished as follows:

- 1) By default, the Executive is set to *Bias order: defined by Simulation Order*. With this setting, the bias order for each Merge and Diverge block is automatically determined based on Simulation Order. Simulation Order is set in the command Run > Simulation Setup > Continuous tab; the default is Flow order. It would be unusual to change the simulation order from the default Flow order. (For further information on Flow order, see “Simulation order” on page 86.)
- 2) You can also directly enter a bias order for a biasing Merge or Diverge block. To do this, first change the Executive’s default setting from *Bias order: defined by Simulation Order* to *Bias order: each block defines its own*. Then do one of the following:

- In the block’s Model Settings tab, use the two array buttons (“<<” and “>>”) to change the block’s bias order. This also changes the block’s position in the tab’s bias order table.
- Or, in the Executive’s Discrete Rate tab, select the row that contains the desired block, then use the << and >> arrows to change its position in the table. The bias order changes when the position of the block in the table changes.



Model Settings tab

**Bias order table**

The bias order table in a Merge or Diverge block’s Model Settings tabs displays each biasing Merge and Diverge block, its bias order, block label or name, mode, and Simulation Order.

- If the blocks are set to *Bias order: defined by Simulation Order*, the table will be inactivated since bias order is determined automatically.
- If the blocks are set to *Bias order: each block defines its own*, the table is active only in Merge or Diverge blocks in the Distributional, Priority, and Sensing modes. In blocks with those modes, it can be used to change the blocks’ bias order, as discussed above.

The Model Settings tab also reports the number of Bias blocks in the model; Bias blocks always have a higher bias order than any Merge or Diverge block.

**Competing preferences**

“Bias Order – resolving competing requests for flow” on page 327 discusses how competing preferences between Merge and Diverge blocks is resolved using Merge and Diverge blocks.

**Global and advanced options in the Executive**

The Executive block (Item library) oversees the global discrete rate system. It is responsible for calculating a model’s effective rates, centralizing and coordinating the information from Rate library blocks as discussed in the advanced topic “LP technology” on page 376.

The Executive’s Discrete Rate tab is used as a central location for setting options used throughout a discrete rate model. These options are divided into global and advanced options, as discussed below.

**Global options**

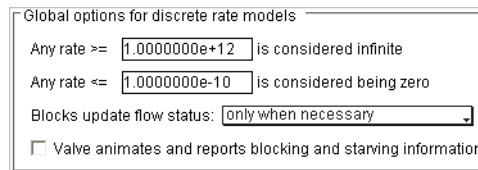
The Executive’s global options are:

- Defining the “infinite” rate
- Defining a “zero” effective rate
- Setting options for how often blocks should update their flow status
- Choosing that the Valve animate and report blocking and starving information
- Managing flow units

The first three options are listed in a global options frame at the top of the tab (shown below); the fourth option is located at the bottom of the tab. They are all discussed in the following sections.

**Infinite rate**

The Discrete Rate tab specifies that a rate equal to or greater than some number is considered infinite; the default setting is that a rate  $\geq 1e10$  is considered infinite. This information is important when setting critical constraints and for the determination of the effective rate. It is discussed fully in “Infinite rate” on page 304.



Global options in Executive

**Zero effective rate**

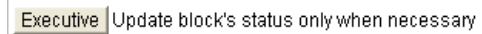
The Discrete Rate tab specifies that an effective rate less than or equal to some number is considered zero, resulting in no flow. The default setting for that number is 1e-10. It is strongly suggested that you do not change the default setting unless you have an excellent reason to do so.

**Update flow status**

In the Rate library, some values change continuously over time. The frequency of updating those values can be customized by setting options for the Convey Flow, Diverge, Interchange, Merge, Tank, and Valve blocks. They define how often the following information is updated:

- The level of flow in residence blocks (Convey Flow, Interchange, and Tank)
- The amount of flow passing through the Convey Flow, Diverge, Interchange, Merge, Tank, and Valve blocks

At a minimum, the discrete rate global system updates flow status only when it needs to. The updating options provide two additional opportunities to have calculations performed. The



Executive Update block's status only when necessary

Valve's default update setting

option that has been selected in the Executive is displayed in each block's Options tab (shown in the screenshot), along with any choices associated with that option. The choices in the popup menu are:


- Only when necessary. This default setting is computationally the most efficient option because flow status is updated only when needed by the system. With this setting, the information is updated:
  - When a block is determined to be part of an LP area
  - Whenever a block creates an internal event such as reaching a new indicator
  - When a block receives an active message at one of its value output connectors
- Each block defines how often. If this option is selected, each block's Options tab will give you the choice to check the option *Update animation and results at each event*. If that is checked, the calculation of the flow status will occur at each step for that block.
- Each block at each step. With this choice, every block will update at each step. This option has to be used cautiously because it is computationally demanding to update the information this frequently.

**Valve animates and reports blocking and starving information**

This option only affects the animation and reporting of Valve blocks. A Valve can be limiting, not-limiting, blocked, starved, or blocked and starved. By default, the Valve's Results tab and *S* (status) output connector only report whether the block is limiting (0) or not limiting (1). The Results tab also reports cumulative information regarding the percentage of time the block was limiting or not limiting.

When the global option is checked, the block's Results tab and its *S* output report all status information as a value:

- limiting (0)
- starved (1)
- blocked (2)
- starved and blocked (3)

 A Valve's complete status information is helpful during the early stages of model construction and for debugging purposes. However, it can slow the simulation, so by default the option is not checked.

The differentiations are animated on the Valve's icon as discussed in "Valve" on page 371.


**Manage flow units for discrete rate models**

This section of the Discrete Rate tab provides a central location where flow units can be renamed or added to or deleted from a model. To delete or rename a flow unit, select it in the table and click the appropriate button. Flow units are discussed on page 297.

**Advanced options**

The advanced options in the Discrete Rate tab only apply to specific situations:

- Merge or Diverge blocks in the Distributional, Priority, or Sensing modes
- Merge blocks in Proportional mode when there is an empty loop

 Merge and Diverge modes are discussed in the chapter "Merging, Diverging, and Routing Flow".

**Merge or Diverge blocks in Distributional, Priority, or Sensing modes**

This first set of options determines how bias order is set for certain Merge and Diverge blocks and whether the bias order is displayed on the block's icon.

*Bias order determination*

For a Merge and Diverge block in the Distributional, Priority, or Sensing mode, the choices are that the block's bias order number is defined by:

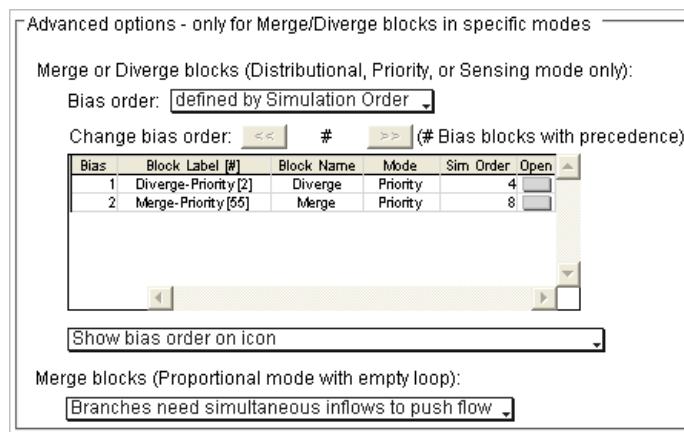
- Simulation Order. This is the default; it causes the bias order to be automatically calculated by the simulation order of the blocks in the model.
- Each block. This option allows the user to customize the bias order of each Merge and Diverge block (Distributional, Priority, or Sensing mode only) in the model.

For detailed information about the requirement for a bias order and how it is set, see "Merge and Diverge blocks" on page 362.

*Displaying the bias order*

For Merge and Diverge blocks in the Distributional, Priority, or Sensing mode, a popup menu provides choices for displaying their bias order:

- Show bias order on icon
- Don't show bias order on icon



Advanced options in Executive

- Each block decides whether to show bias order on icon

If the bias order is displayed on a block's icon, it will be in the format <#>, where # is the bias order number. If the third option is chosen, a checkbox will appear in each Merge or Diverge block's Model Settings tab. Check the *Show bias order on icon* checkbox if you want the block to show the bias order number on its icon.

**Merge blocks in Proportional mode**

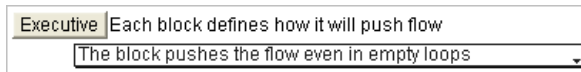
For Merge blocks in Proportional mode, a conflict can result if the block is part of an empty loop (can't get flow). If a Merge block's proportions are 1:2 for instance, what should happen if the top inflow branch is part of an empty loop and cannot get any flow from that loop?

The Executive provides three options for how empty loop situations should be resolved:

- Branches need simultaneous inflows to push flow
- Blocks push flow even in empty loops
- Each block defines how it will push flow

Executive options for empty loops

- Branches need simultaneous inflows to push flow. With this default setting, flow is stopped at all inflow branches if one or more of them are part of an empty loop.
- Blocks push flow even in empty loops. This choice allows the branches with flow to send it through. The result is that the branch that is part of the empty loop will then get some flow.
- Each block defines how it will push flow. This setting causes an additional popup menu to appear in the Model Settings tab of any Merge block in Proportional mode. The two choices in each Merge's dialog are:
  - The block pushes flow even in empty loops. This is the default setting; it allows branches with flow to send it through.
  - Each branch needs simultaneous inflows to push flow. With this choice, flow is stopped at all inflow branches if one or more of them cannot get any flow.



Model Settings tab of Merge block

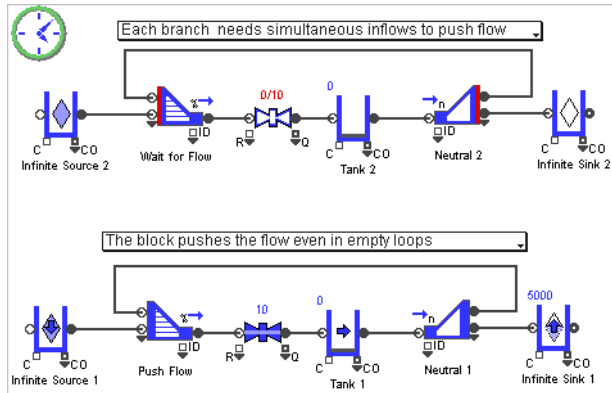
Which of the three empty loop options you choose depends on the behavior of the system you are simulating.

*Merge Proportion Setting model*

In this model the two flow streams have identical settings except for how the Merge blocks handle empty loops. So that each block can specify its own behavior, the Executive's setting for the section "Merge blocks (Proportional mode and empty loop)" is *Each block defines how it will push flow*.

The Merge block in the top stream (Wait for Flow) specifies that *Each branch needs simultaneous flow* while the Merge block in the bottom stream (Push Flow) is set to *Blocks push flow even in empty loops*. The flow is blocked at the Merge block in the top stream but flows through the Merge block in the bottom stream.

This model is located in the folder \Examples\Discrete Rate\Merge and Diverge.



Merge Proportion Setting model

### Common connectors on discrete rate blocks

Most Rate library blocks have inflow and outflow connectors and value input and output connectors. The Interchange block also has item input and output connectors.

Flow connectors pass information about the effective rate from one discrete rate block to another. Item connectors pass discrete items.

For value connectors, many discrete rate blocks use abbreviations or acronyms to indicate the connector's purpose. Some of these abbreviations represent more than one purpose and are context sensitive.

The following connector labels appear for value connectors on Rate library blocks:

Connector	In or Out?	Meaning
AL	Output	Accumulation length - for a Convey Flow block, the distance from its end to the accumulation point.
AQ	Output	Accumulated quantity - the amount of flow that has been accumulated along a Convey Flow block's accumulation length.
B	Input	Bias order number
C	Input	Capacity
CO	Output	Contents
D	Input	Delay
DR	Output	Potential downstream demand rate
factor	Input	Conversion factor in Change Units block
G	Input	Quantity of flow (quantity goal) or duration of time (duration goal) for a Valve. Can also be used to start a new goal, depending on Valve's dialog setting.
G#	Output	Goal number
GD	Output	Goal duration



Connector	In or Out?	Meaning
GS	Output	Goal status: 0 - no goal 1 - starting 2 - in progress 3 - ended 4 - interrupted
GQ	Output	Goal quantity
GO	Input	Activate a status update: 0 or 1 (Sensor and Merge/Diverge)
I	Output	Indicator (Convey Flow, Interchange, or Tank)
IC	Input	Item capacity
ICO	Input	Item contents
ID	Input	Inflow/outflow branch ID for Merge/Diverge in Select mode
IT	Input	Target value to release item
L	Output	Length of item line: 0 or 1
LE	Output	Level of contents
NB	Output	Number blocked: 0 or 1
PE	Input	Preempt item
PT	Output	Process time
Q	Output	Cumulative quantity
R	Input	Maximum rate (Note that the effective rate is reported by the flow input and output connectors.)
S (on Convey Flow)	Output	Status: 0 = empty 1 = intermediate 2 = full
S (on Interchange or Tank)	Output	Status direction (Interchange): -1 = down 0 = stable 1 = up
S (on Valve)	Output	Status [if Valve only reports limiting or non-limiting]: 0 = limiting 1 = non-limiting
S (on Sensor or Valve)	Output	Status [if Valve reports full status:] 0 = limiting (Valve) or unused (Sensor) 1 = starved 2 = blocked 3 = starved and blocked
S (0-n)	Output	Sensors - numbered from 0 to n (Convey Flow)
SP	Input	Speed parameter
SP	Output	Effective speed
SR	Output	Potential upstream supply rate

Connector	In or Out?	Meaning
start	Input	Start hysteresis
stop	Input	Stop hysteresis
TL	Output	Cumulative time Valve was limiting, effective rate > 0
TU	Output	Cumulative time Valve was not limiting, effective rate > 0
TLO	Output	Cumulative time Valve was limiting, effective rate = 0
TUO	Output	Cumulative time Valve was not limiting, effective rate = 0

### Animation

Rate library blocks can be animated during the simulation if Run > Show 2D Animation has been selected before a simulation run. For blocks with animation, the following information explains what each display means.

 The models illustrated in this section are located in the folder \Examples\Discrete Rate\Miscellaneous.










### Tank

Tanks animate information about their levels and information about the direction of flow within the Tank.

#### Level information

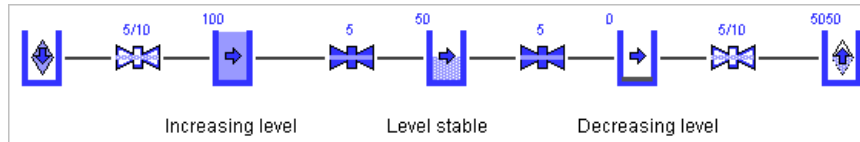
The table below shows animation of the Tank's level under different conditions.

Discrete Rate

Finite Capacity	Infinite Capacity	Specific Behaviors
<p>100</p>  <p>Full (contents 100, capacity 100)</p>	<p>Full</p>  <p>Full</p>	<p>1000</p>  <p>Overfull (contents 1,000, capacity 100)</p>
<p>50</p>  <p>Some flow (contents 50, capacity 100)</p>	<p>10</p>  <p>Some flow (10 units)</p>	<p>1000</p>  <p>Overfull (contents 1,000, no capacity)</p>
<p>0</p>  <p>Empty (contents 0, capacity 100)</p>	<p>Empty</p>  <p>Empty</p>	<p>0</p>  <p>No capacity and empty</p>

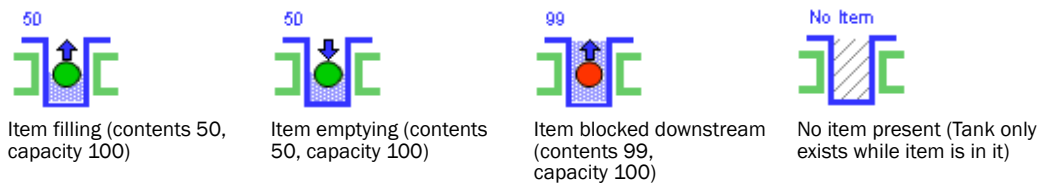
**Direction information**

An arrow is animated on the icon to indicate the direction of change in the Tank's level. When there is no arrow, it means that the Tank has neither an inflow nor an outflow rate.



**Interchange**

The Interchange block displays the same animation behavior as a Tank, shown above. In addition, a ball appears in the middle of the block's icon when an item is present in the block. The ball is red if the item is ready to leave the block but is blocked downstream. Otherwise, the ball is green. If the option *Tank only exist while Item is in it* is chosen and there is no item in the block, the tank icon animates as white with cross-hatching.

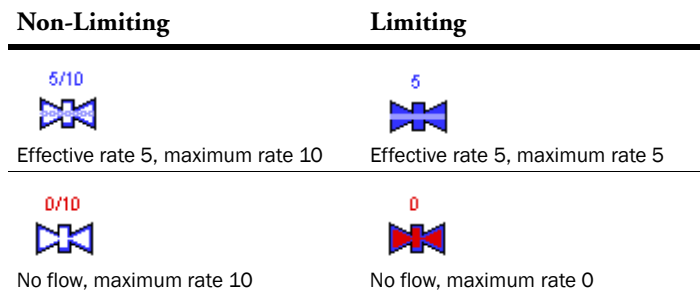


**Valve**

A Valve reports its maximum and effective rates when animation is turned on. If the option *Valve animates and reports blocking and starving information* is not selected in the Executive block's Discrete Rate tab (the default setting), a Valve will display only its limiting or non-limiting status. If that option is selected, it will also report its blocking and starving status. (See "Valve animates and reports blocking and starving information" on page 365 for full information.)

**Displaying limiting and non-limiting status**

By default, the Valve only animates information about its limiting or non-limiting status.

















*Limiting* is when the effective rate equals the maximum rate. In this case, the Valve is what limits the flow. When a Valve is *non-limiting*, the effective rate is less than the Valve's maximum rate. In that case, the Valve has no impact on the flow.

- When the effective rate is not the same as the constraining (maximum) rate, the two numbers are represented as *effective rate/constraining rate* above the icon. If the two rates are the same, only one number appears.
- If the Valve is limiting and the effective rate is greater than 0, the interior of the icon is plain blue. If the Valve is limiting to 0, the interior of the icon is plain red. If the Valve is not limiting, the interior of the icon is white.
- When the effective rate is greater than 0, a blue rectangle appears along the icon, and the rates are written in blue. When there is no flow, the blue rectangle does not appear on the icon and the rates are written in red.

**Also displaying blocking and starving status**

If the option *Valve animates and reports blocking and starving information* is selected in the Executive block's Discrete Rate tab, Valves will also animate their blocking and starving status information.

Limiting only	Blocked	Starved	Blocked and Starved
 5 Limiting	 5/10 Non-limiting, with flow	 5/10 Non-limiting, with flow	 5/10 Non-limiting, with flow
 0 Maximum rate 0	 0/10 Non-limiting, no flow	 0/10 Non-limiting, no flow	 0 Non-limiting, no flow
	 5 Limiting, with flow	 5 Limiting, with flow	 5 Limiting, blocked, and starved with flow
	 0 Limiting, no flow	 0 Limiting, no flow	 0 Limiting, blocked, and starved and no flow

If a Valve is blocked, it means that there are one or more blocks downstream which are limiting the flow through the Valve. If a Valve is starved, it means there are one or more upstream blocks that limit the flow to the Valve.

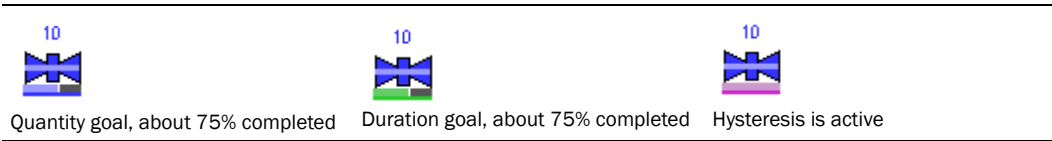
- If there is no flow, there will not be a horizontal line through the Valve. If there is flow, and the Valve is non-limiting, the horizontal line through the Valve is a patterned blue. The horizontal line is a plain blue color if there is flow.
- If a Valve is limiting, the central, vertical part of the Valve is plain blue (if the maximum rate is >0) or red (if the maximum rate = 0).

- If the right side of the Valve's icon is plain blue, the block is partially blocked by downstream blocks. If there is a red line on the right side of the icon, the flow is completely blocked downstream.
- If the left side of the Valve's icon is plain blue, the block is partially starved by upstream blocks. If there is a red line on the left side of the icon, the flow is completely starved upstream.

**Goal and hysteresis animation**

A Valve that uses a quantity goal to control its flow has a blue line below its icon. If the Valve controls its flow with a duration goal, the line is green. While the goal is On, a progression bar appears along the top of the blue or green goal line.

A Valve that uses hysteresis has a purple line at the bottom of its icon. While hysteresis is active, the purple line is thicker for all or part of its length.



**Sensor**

The shape and color of the Sensor block's icon indicates if the flow is being blocked, starved, or both., and if there is flow or not.

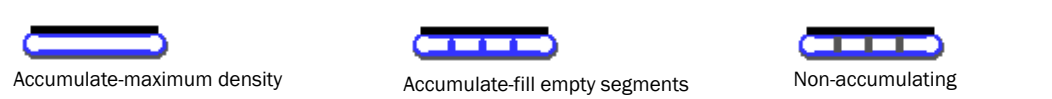
Flow?	Blocked	Starved	Blocked and Starved
Yes >>>			
No >>>			

Discrete Rate

**Convey Flow**

The animation of the Convey Flow block shows what mode it is in, the distribution of flow along its length, the position of the accumulation point, and other information.

**Mode animation**

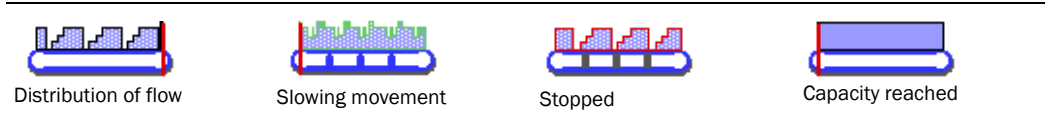


For further exploration, the Compare Convey Flow model compares the behavior of three Convey Flow blocks, each set to one of the possible modes, under different emptying rates. The model is located in the folder \Examples\Discrete Rate\Delaying Flow.

**Distribution of flow and other information**

A Convey Flow block animates the distribution of flow as follows:

- A blue shape along the top of its icon indicates flow distribution.
- A slowing of flow movement causes the blue shape to have a green border.
- The complete stopping of flow movement causes the blue shape to have a red border.
- If the block has reached its capacity, the shape is a solid blue rather than a dotted blue.
- If the block is empty, a black line will appear at the top of its icon, as seen above.
- The position of the accumulation point is indicated by a red vertical line that moves along the length of the icon. This is shown in the first, second, and fourth screenshots below and is discussed on page 345.




# **Discrete Rate Modeling**

## **Advanced Topics**

Some additional information  
for those of you who want to know more

The earlier chapters in the Discrete Rate module discussed important concepts you need to know to build discrete rate models. This chapter provides an overview of the way discrete rate calculations are made and describes the underlying functioning of the Rate library. This information is not necessary to build discrete rate models but will be of interest to advanced users.

 It is highly recommended that you read the previous discrete rate chapters before this one, particularly the chapter “Rates, Constraints, and Movement”.

### What this chapter covers

- The LP technology that has global oversight over a discrete rate model
  - How the LP area is determined
  - The sequence of events for the LP calculation
  - How bias affects the calculation
  - The types of information provided to the Executive
  - How a relational constraint is calculated
- The potential rates “upstream supply” and “downstream demand”
- Messaging in discrete rate models

### LP technology

Linear programming problems involve the optimization of an objective function subject to a set of constraints. The purpose of solving a linear programming problem is to maximize or minimize selected variables in the objective function.

*LP technology* is the method ExtendSim uses to provide global oversight to maximize the movement of flow throughout a discrete rate system. The discrete rate architecture employs an integrated LP Solver DLL to solve a series of equations to optimize effective rates at each point of the simulation run.

The purpose of the LP calculation is to determine the maximum effective flow rates in the system given the constraints defined by block settings and the structure of the model. After all the rules for storage capacity and movement have been declared in the model, ExtendSim uses the Executive block’s LP calculations to cause as much flow as possible to move through the system. This calculation is handled automatically and internally.

### Overview

In a discrete rate model, the Rate library blocks communicate with each other and with the Executive block. In turn, the Executive communicates with an integrated linear program (LP Solver). The Rate library blocks are dependent on each other, have an effect on one another, and are part of a global LP system that evaluates the entire model to calculate its effective flow rates.

- A *rate section* is a network of connected blocks, all possessing the same effective rate. Established at the beginning of the simulation run, rate sections do not change. (See “Rates, rate sections, and the LP area” on page 303 for more information.)
- An *LP area* is made up of one or more rate sections; the actual configuration can change during the simulation. A change in a block’s constraints during the simulation run initiates a propagation of messages through all the rate sections whose effective rates might change. This propagation defines the LP area at that point in time.



- Each rate section within the LP area contributes a part of an *LP equation* for a recalculation. The purpose of the recalculation is to determine the maximum effective flow rates in the system, given the constraints defined by block settings and the structure of the model. The result is the set of effective rates for each section in the LP area at that point in the simulation. The system is optimized such that only the rate sections in the LP area are recalculated; all the other effective rates in the system don't need to recalculate at that moment and won't.
- Among other things, the LP calculations take into consideration each block's:
  - Critical constraints, which place an upper bound on rate sections connected to that block.
  - Relational constraints, which define the way rate sections are related to each other.
  - Bias, which is a block's preference that flow travel one route rather than another.

### The LP area

The *LP area* is made up of one or more rate sections linked together by the fact that their effective rates could change at that point in the simulation – a change in the effective rates in one section might impact effective rates in the other sections. The rates, constraints, and biases within the LP area are used by the LP Solver to calculate the optimal set of effective rates for the rate sections contained within that area.

The effective rate of one section can affect the effective rate of another section through relational constraints. When an event occurs that causes a rate section's effective rate to be reevaluated, blocks propagate “rate block flow” messages (see “Block messages” on page 387) throughout the model to determine which other rate sections might be impacted by the new event. The affected rate sections constitute the LP area and rate sections outside of the LP area are not included in the recalculation, reducing redundant computations.

The boundaries of the LP area are determined through the propagation of messages between Rate library blocks. A change in a block's constraints during the simulation run causes the block to notify the Executive and send messages which propagate through all the rate sections whose effective rates might change as a result. Whether the block is the originator of the recalculation request or receives a propagation message:

- The block declares which of its connected rate sections are in the LP area. The propagation process then creates a global list of rate sections to include in the current LP area.
- If the block provides a relational constraint connection between two or more rate sections, the effective rates connected to that block are dependent on one another. The block then continues the message propagation to all the dependent block(s), who propagate the message to their dependent blocks, and so forth.

The change in the originating block will definitely cause a recalculation of the effective rates for all directly connected rate sections, and (depending on relational constraints between the sections) might cause a recalculation of the effective rates for other rate sections.


- ☞ The boundaries of the LP area will change dynamically during the simulation depending on which effective rates are involved in the recalculation and which relational constraint dependencies occur between rate sections.

### The sequence of events

A change in a block's constraints during the simulation run initiates a reevaluation of all the effective rates for the LP area at that point in time. The block's constraint change will definitely cause a recalculation of the effective rates for all directly connected rate sections, and (depending on rela-

tional constraints between the sections) might cause a recalculation of the effective rates for other rate sections. The sequence is:

- 1) A block's constraint changes.
  - If a block's status changes it can affect its effective inflow and/or outflow rates. For example, the effective inflow rate for a finite Tank block that is filling up is greater than its effective outflow rate. When the Tank becomes full, it creates an event because its effective inflow rate can no longer exceed its effective outflow rate.
  - When a block reacts to new parameters, the effective inflow and/or outflow rates must be reevaluated. For example, a Merge block in Select mode might choose its top inflow branch at the start of the simulation. If the block subsequently receives an order to select its bottom inflow branch, its effective rates have to be recalculated.
- 2) The block posts a zero-time event requesting a reevaluation of effective rates.
- 3) The LP area is determined based on the propagation of block messages.
  - Starting with the originating block, messages are propagated through the model to all the blocks that might be affected by the change. This propagation of messages defines the boundaries of the LP area.
  - The LP area encompasses all the rate sections with effective rates that might change during the calculation.
  - For a complete description of the LP area and how it is determined, see page 377.
- 4) As the LP area is determined, the blocks update their status.
  - Each block that is part of the LP area updates the amount of flow which has passed through it since the last update.
  - If the block is a residence block (Convey Flow, Interchange, or Tank), the amount of flow the block is holding is also updated.
  - Note: The frequency of status updates outside of an LP calculation is set in the Executive. See "Update flow status" on page 365 for information.
- 5) Blocks report to the Executive how they impact effective rates.
  - Each block in the LP area declares the flow rules (critical and relational constraints) that it applies to the rate section it is connected to.
  - The block's bias order is stored in a list. Each block declares its bias order (if any) and provides coefficient information to the Executive. The coefficients allow the Executive to build an objective function that considers the effect of the bias.
  - See "Types of information provided to the Executive" on page 379.
- 6) The Executive determines an objective function.
  - The objective is to maximize the flow rate for each rate section subject to the constraints defined by the blocks' flow rules.
  - Each decision variable in the objective function is the effective rate of a section in the LP area. For instance, the objective could be "Maximum effective rate = 1\*ER1+ 1\*ER2 + 1\*ER3...", where ER<sub>n</sub> is the effective rate for a rate section in the LP area.

- If there is no bias, or all the intermediate calculations related to bias order have been completed, the coefficient of each variable will be 1. If there is bias, the coefficient could be other than 1.
- 7) The Executive communicates with the LP Solver.
    - The Executive gives the LP Solver the objective function and the information from the blocks and rate sections within the LP area.
    - Critical constraints put an upper bound on some of the effective rates (the decision variables). Relational constraints provide a link between the effective rates of different rate sections and must be taken into consideration during the calculation. For instance, if effective rate X is less than or equal to effective rate Y, the objective function must conform to that information.
  - 8) The Solver performs a calculation.
    - The Solver calculates an optimized set of effective rates for the LP area.
    - The result is an intermediate LP calculation (if there is bias) or the final LP calculation (if there is no bias).
    - For full details, see “The LP calculation” on page 382.
  - 9) Steps 6-8 are repeated, if necessary.
    - If the blocks in the LP area have bias, the Executive must determine intermediate objective functions and the Solver must perform intermediate calculations. The number of recalculations is equal to the number of blocks with bias, plus 1.
    - For each recalculation, the list of critical and relational constraints is changed to include the constraints caused by the particular bias order being considered.
    - See “Bias information” on page 380.
  - 10) The rate sections are notified.
    - The Executive sends “Executive block flow” messages to the head of the rate sections to update to the new effective rates.
    - The blocks receiving this information update according to their dialogs and value connectors. They then post new events if necessary.
-  Since the purpose of the recalculation is to maximize the entire set of effective rates, some rates will change while others might not.

### Types of information provided to the Executive

When effective rates need to be recalculated, Rate library blocks provide critical and relational constraint and bias information to the Executive.

To learn which blocks contribute which types of information, see “Table summarizing constraint and bias information” on page 381.

### Flow rules

Flow rules consist of critical and relational constraints.

- If a rate section has a critical constraint, it places an upper limit to the effective rate within that section. Since effective rates are decision variables in the LP’s objective function calculation, critical constraints place an upper bound on some of the equation’s variables.

- Relational constraints describe the dependencies between different rate sections. In some blocks, relational constraints can vary depending on the state sensitivity of the block; in others they are permanently active.
  - An example of a *state sensitive* relational constraint can be found in a Tank block. (Tanks are always within two rate sections; the input side and the output side define the sections.) As long as a Tank is empty, its relational constraint is defined as “effective outflow rate is less than or equal to effective inflow rate”. Once the Tank becomes “not empty”, the relational constraint doesn’t apply.
  - An example of a *permanent* relational constraint can be found in a Change Units block where a conversion factor defines the relationship of the inflow effective rate to the outflow effective rate. If the factor varies over time, the relational constraint may also vary. But the dependency between the inflow and outflow effective rates is active for each calculation which includes the two rate sections.

### **Bias information**

When the LP area is created, the Executive ranks all Bias blocks and any Merge or Diverge blocks with a bias order in a list. The Executive considers the top bias from that list as part of the objective function and instructs the Solver to perform an intermediate LP calculation using that function and the current critical and relational constraints. Then the Executive takes the next bias order into consideration, and so forth.

Since bias affects critical and relational constraints, each succeeding bias order means a new objective function will be determined and a new set of critical and relational constraints will be added to the previous ones. The results from the previous LP are used as inputs to the next LP. This results in multiple intermediate LP calculations – one for each bias order in the list – until the final result.

Most blocks with bias order supply:

- Flow rules (critical and relational constraints) which apply to all the LP calculations. (This information is not supplied by a Bias block.)
- A set of coefficients that the Executive will use to build the objective function corresponding to this bias order. Depending on the bias information received from the blocks in the LP area, the intermediate objective function can include coefficients that are other than 1. (A coefficient of 0, for instance, indicates that a particular effective rate does not need to be maximized; it causes that effective rate to not be directly affected by the maximization.)
- Flow rules for that bias order which use the results of the intermediate calculation. (After the intermediate calculation, the results of the calculation are used to add new flow rules to the succeeding calculations.)

Some additional situations that enter into the calculation include:

- If a Bias block has a bias order that is Blank or is less than or equal to 0, the block does not express any preference.
- If multiple Bias blocks have identical bias orders, the effective rates for these blocks cannot be predicted. The model will use one of the possible solutions.
- Merge and Diverge blocks with bias order are always lower on the bias list than any Bias block.

**Table summarizing constraint and bias information**

The following table summarizes the types of bias and constraint information (if any) Rate library blocks supply to the Executive for the calculation of effective rates:

Block	Mode	Critical Constraint	Relational: Permanent	Relational: State Sensitive	Bias Order
Bias		No	Yes	No	Yes
Catch Flow		No	No	No	No
Change Units		No	Yes	No	No
Convey Flow		Yes	No	Yes	No
Diverge	Neutral, Proportional, Select, Unbatch	Depends	Yes	No	No
Diverge	Distributional, Priority, Supply Sensing	Depends	Yes	No	Yes
Interchange		Yes	No	Yes	No
Merge	Batch, Neutral, Proportional, Select	Depends	Yes	No	No
Merge	Distributional, Demand Sensing, Priority	Depends	Yes	No	Yes
Sensor		No	No	No	No
Tank		Yes	No	Yes	No
Throw Flow		No	No	No	No
Valve		Yes	No	No	No

 Diverge and Merge blocks can imply a critical constraint depending on the branch parameter and the mode.

**The relational constraint calculation**

The table that follows describes how the relational constraint is calculated for blocks that provide either permanent or state sensitive relational constraints.

For the purposes of this table:

- $X_{in}$  is the effective inflow rate of a block and  $X_{out}$  is its effective outflow rate.
- Index  $i$  describes one of the inflow branches (for a Merge) and one of the outflow branches (for a Diverge).
- The number of branches is  $n$ .

Block	Calculation
Change Units	The permanent boundary is $X_{in} = \text{factor} * X_{out}$
Convey Flow	If the block is in an accumulating situation (outflow rate is blocked or partially blocked downstream) and doesn't have any more capacity for accumulation, a state sensitive boundary applies: $X_{in} \leq X_{out}$

Block	Calculation
Diverge	Select. A permanent relational constraint applies between the inflow effective rate and the selected outflow effective rate: $X_{in}=X_{out}$ selected  Proportional. A set of permanent relational constraints applies to insure the proportions: $X_{out\_i} = factor\_i * X_{in}$ ( $i: 0=>n-1$ )  Batch/Unbatch. Permanent relational constraints: $X_{out\_i} = X_{in}$ ( $i: 0=>n-1$ )  Neutral, Priority, Distributional and Supply Sensing. Permanent relational constraint $X_{in} = X_{out\_1} + \dots + X_{out\_n}$  (Other calculations are beyond the scope of this document.)
Interchange	As long as the tank is full, $X_{in} \leq X_{out}$ . As long as the tank is empty, $X_{out} \leq X_{in}$ (State sensitive relational constraint)
Merge	Same as the Diverge with $X_{out}$ and $X_{in}$ reversed
Tank	See Interchange

### The LP calculation

The Executive block maximizes an objective function composed of the set of effective rates for the LP area. If there is no block with bias order involved in the LP area, the Executive block maximizes the sum of all the effective rates in the LP area – in this case, only one LP calculation is necessary.

If the blocks in the LP area have bias, the Executive must determine multiple intermediate objective functions and the Solver must perform multiple intermediate calculations. The number of recalculations is equal to the number of blocks with bias plus 1.

When the LP area involves blocks with bias order, the calculations are made in cascading order. For each bias order, an intermediate calculation is made with an objective function depending on the type of block that is being evaluated:

- Bias block. The function to maximize is the sum of the effective rates attached to the Bias blocks with an identical bias order. When the calculation is made, the function is used as a new rule for the succeeding intermediate calculations. The calculation is: sum effective rates within the bias order  $\geq$  result of the maximized function.
- Merge/Diverge in Priority mode. The function to be maximized contains the effective rates from the variable inflow and/or outflow branches of the block. The lower the priority of the branch, the higher the coefficient associated with the effective rate. The calculation is:  $\sum p * X_p$  (for  $p: 1=>n$  with 1 top priority and  $n$  lowest priority). When the calculation is finished, the objective function is used as a new rule for the next intermediate calculations ( $\sum p * X_p \geq$  result of the maximized function).
- Merge/Diverge in Batch/Unbatch, Distributional, Neutral, Proportional, Select, or Sensing mode. This calculation is beyond the scope of this document.

When all the intermediate LP calculations have been made, the last LP calculation maximizes the sum of all the effective rates for the LP area.

### Upstream supply and downstream demand

The effective rate is not the only result a rate-based model can provide. The *potential upstream supply rate* and the *potential downstream demand rate* can also be useful in special situations – the rates determine the branch proportions for Merge and Diverge blocks in Sensing mode (as dis-

cussed in “Sensing mode” on page 325) and the Sensor block can be used to report the potential rates for making model decisions.

- ☞ This is an advanced topic because the concept is complex and there is an elevated potential for error, as discussed in the section “Cautions when using potential rates”, below. Careful model verification and validation, and an advanced knowledge of the ExtendSim LP technology, are required to avoid unexpected results.

### Definition

The *potential upstream supply rate* is the theoretical rate at which an upstream source could provide flow to the beginning of a rate section if there weren't any downstream limitations on flow movement (downstream capacity is infinite). For instance, for a not-full Tank at the beginning of a rate section, the upstream supply rate would equal the Tank's effective inflow rate. A not-full Tank does not limit the inflow rate it can receive. In this case, the effective inflow rate is also the potential upstream supply rate.

The *potential downstream demand rate* is the theoretical rate at which a downstream section of the model could receive flow from the end of an upstream rate section if there were an unending upstream supply of flow (upstream source is infinite). For instance, for a not-empty Tank at the end of a rate section, the potential downstream demand rate would be equal to the Tank's effective outflow rate. A not-empty Tank does not limit the outflow rate it can provide. In this case, the effective outflow rate is also the potential downstream demand rate.

Upstream supply and downstream demand could potentially be infinite. For example, the upstream supply rate right after a Tank (if the tank doesn't declare any constraint on its outflow rate) is infinite.

- ☞ When flow movement is from left to right, the information to calculate the upstream supply rate is propagated from left to right while downstream demand information is propagated from right to left.

### Requirements for the supply/demand calculation

For a model's Executive block to calculate a potential supply or demand rate, at least one of the following blocks must be part of the LP area:

- Sensor block. The only purpose of the Sensor block is to display the potential rates wherever it is located in the model. The Sensor block only provides information; it has no direct impact on the calculation of effective rates.
- Diverge block in Demand Sensing mode. Proportions for the outflow branches are calculated as a function of the potential downstream demand. For example, the downstream demand placed on an outflow branch becomes the proportion for that outflow branch.
- Merge block in Supply Sensing mode. The block uses the available upstream supply rate to define the Supply Sensing proportions for each inflow branch.

### Cautions when using potential rates

We strongly recommend exercising caution when using the Merge/Diverge blocks in Sensing mode, or when relying on a potential rate reported by the Sensor block, because:

- Getting information about these rates slows down the simulation. The Executive has to make a lot more LP calculations to extract the potential rate information.

- In some cases the potential rates as reported by the Sensor block are not accurate. Depending on how flow is diverged and merged, the potential supply or demand rates could be aggregated even if they should not be. This is illustrated in the “Supply & Demand Warning model” described below.
- Merge/Diverge blocks in Distributional, Neutral, and Priority modes are not always compatible with Merge/Diverge blocks in Sensing mode. Consequently, models with blocks that mix Sensing mode with Distributional, Neutral, or Priority modes are prone to error.
- A Merge or Diverge block in Sensing mode has a bias order. Depending on the block’s location in the Executive’s list of bias orders:
  - The effective rates might be different.
  - The block might not follow the proportions specified by the Sensing mode.

For instance, the effective rates are different in the “Combine Priority Sensing model” and the “Combine Sensing Priority model”, and neither of those models follow the Sensing rule.

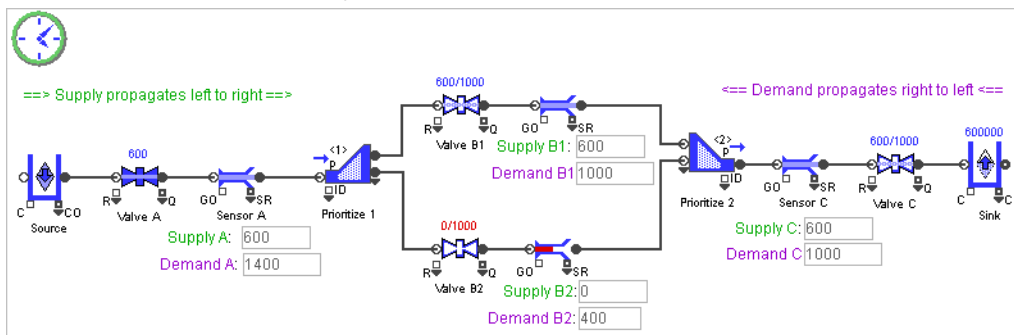
**Issues when relying on the Sensor to report potential rates**

Based on how a model is configured, and the mode selected in a Merge or Diverge block, a Sensor block could report erroneous information.

*Supply & Demand Warning model*

In this model, the Sensor blocks correctly report the upstream supply. But Sensor A reports an invalid downstream demand of 1400 gallons/minute. The actual downstream demand is 1000 gallons/minute, reported by Sensor C. This is determined by verifying the demand results, compared to what was expected, from right to left.

Discrete Rate



Supply & Demand Warning model

Evaluating the model from right to left (for the downstream demand rate):

- Sensor C reports a demand (Demand C) of 1000 gallons/minute because Valve C limits flow to 1000 gallons/minute. The Sink at the end of the line doesn't limit the inflow rate at all.
- Demand B1 is 1000 gallons/minute because the Prioritize 2 block has its top priority at branch B1. (Verification: If a Tank with flow is present at branch B1, Prioritize 2 and Valve C would accept 1000 gallons/minute coming from branch B1.)
- Demand B2 is 400 gallons/minute because the effective rate on branch B1 is 600 and Demand C is 1000: Demand B2=Demand C-effective rate B1. (Verification: If a Tank with flow is



present at branch B2, it could provide 400 gallons/minute because the total Prioritize 2 can accept is 1000 and the block already takes 600 from the top priority branch.)

- Sensor A reports a demand of 1400 gallons/minute because Demand B1 is 1000 and Demand B2 is 400. However, this is incorrect. (Verification: If a Tank with flow is placed right before Prioritize 1, the outflow effective rate would be 1000 gallons/minute and not 1400 gallons/minutes. The erroneous result comes from the fact that the association of “correct” local rules doesn’t guarantee a “correct” global result.

By understanding flow rules and how they can affect the global result, you can avoid this problem. The most important step is verifying this model from left to right for the supply and from right to left for the demand. Two solutions are 1) having the Prioritize 2 block be in Neutral mode, and 2) placing a Valve with a maximum rate of 1000 between Sensor A and Prioritize 1.

**Mixing Merge/Diverge block modes**

Situations where Merge or Diverge blocks in Sensing mode are mixed with Diverge or Merge blocks in Distributional, Neutral, or Priority modes within an LP area should be avoided as they are prone to give inaccurate results.

In these types of mixed-mode situations, the LP area is going to be recalculated multiple times to provide the effective rates and the upstream supply and downstream demand rates. All the blocks in the LP area must provide one set of constraints to the Executive so it can solve the effective rates and a second set of constraints so the Executive can solve the supply and demand rates. This causes repeated intermediate LP calculations, the results of which are affected by constraints which are applied for the supply and demand rates. This may yield inaccurate results.

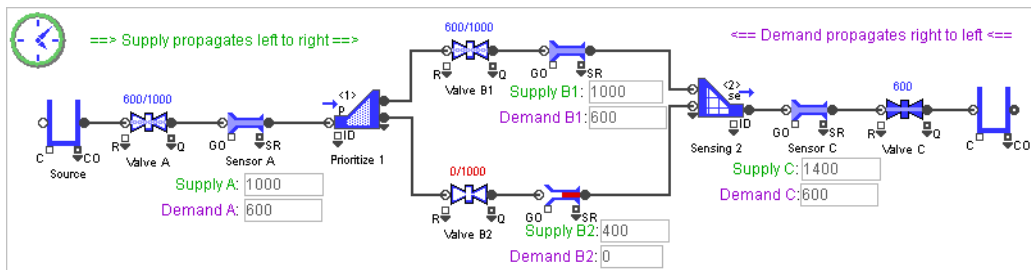
**Unexpected effects of bias order**

As discussed on page 362, the Distributional, Priority, and Sensing modes require bias order to be defined. This affects their effective rates. However, the only mode whose proportions are influenced by the bias ranking is the Sensing mode.

When effective rates within an LP area have to be recalculated, the Executive makes a list of blocks ranked from the top bias order to the bottom. Depending on a Merge or Diverge block’s ranking in the Executive’s list, the effective rates might be different. If the block is in Sensing mode, the proportions will also be influenced. The two models that follow illustrate these issues.

*Combine Priority Sensing model*

In this model, a Diverge block (Prioritize 1) is set to Priority mode and a Merge block (Sensing 2) is set to Supply Sensing mode. The blocks’ Model Settings tabs are set to *Each block defines its own bias order*. The table in the tabs indicates Prioritize 1 has the top bias order and Sensing 2 has the lower bias order.



Combine Priority Sensing model

Discrete Rate

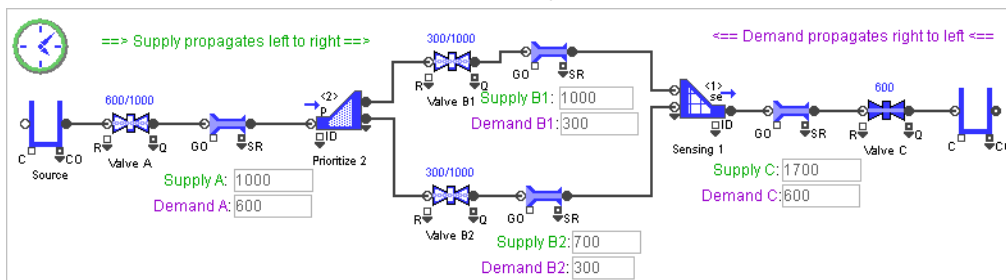
The supply in branch B1 is 1,000 while the supply in branch B2 is 400. However, the Sensing 2 block doesn't distribute flow following the expected supply proportion.

- Because the Prioritize 1 block has the top bias order, the Executive gives priority to that block in choosing how to distribute the flow between the branches B1 and B2.
- At the first intermediate result of the LP calculation, branch B1 gets an effective rate of 600 gallons/minute and branch B2 gets 0 gallons/minute.
- Because this decision has been made for branches B1 and B2, the Sensing 2 block with its lower bias order cannot control the proportion of flow it will get through branches B1 and B2. For example, if the supply from B1 is 1000 gallons/minute and the supply from B2 is 400 gallons/minute, the proportion between the effective rates for the Sensing 2 block's inflow is not 71% B1 (1000/1400) and 29% B2 (400/1400) but rather 100% B1 and 0% B2.

*Combine Sensing Priority model*

Like the preceding model, the Diverge block (Prioritize 2) in this model is set to Priority mode and the Merge block (Sensing 1) is set to Supply Sensing mode and the Model Settings tabs for the blocks are set to *Each block defines its own bias order*. However, in this model the table indicates that Prioritize 2 has a lower bias order than Sensing 1.

Discrete Rate



Combine Sensing Priority model

- The Sensing 1 block has the top bias order, so the Executive gives priority to that block to choose how the flow it receives should be distributed between branches B1 and B2.
- At this point of the LP calculation, the Prioritize 2 block gets a supply of 1000 gallons/minute and has not decided how to distribute the flow. With that limited information, the potential upstream rate is calculated as 1000 gallons/minute for Supply B1 and 1000 gallons/minute for Supply B2. Therefore, Sensing 1 decides to get 50% of the flow from inflow branch B1 and the other 50% of the flow from branch B2.
- Because the decision has been made for branches B1 and B2, Prioritize 2 with its lower bias order can no longer control the distribution of flow between branches B1 and B2.

**Messaging in discrete rate models**

As discussed in “How ExtendSim passes messages in models” on page 533, the ExtendSim architecture allows application messages to be sent from ExtendSim to a model's blocks and block messages to be passed between blocks.


Discrete rate models use the same application messages as do continuous and discrete event models. The block messages sent between Rate library blocks are discussed below.

### Block messages

Discrete rate blocks have a sophisticated and complex messaging structure for communicating with each other and with blocks from the Value and Item libraries. They can be categorized as:

- Event
- Value connector
- Item connector
- Flow connector
- Rate block flow
- Executive block flow


When a block initiates a recalculation of the set of effective rates, a succession of messages and calculations are also initiated. Understanding how block messages work can prevent redundant messages from being created. At the very least redundant messages will cause runtimes to be longer than need be. At the very worst, redundant rate calculations could introduce bugs into a model when effective rates are temporarily calculated using one or more out-of-date parameter values. For instance, see “Limiting the number of recalculations” on page 328.

 By limiting the number of times the set of rates have to be recalculated, the operational efficiency of the model is maximized. For example, see “Limiting the number of recalculations” on page 328.

### Event messages

Event messages communicate between the Executive block and Rate library blocks. In a discrete rate model, the simulation clock advances from one event to another. Each time the clock advances, the Executive block sends event messages to the blocks that have associated themselves with that event. There are two types of events: future and current.

- A *future* event message occurs when the simulation clock reaches a time posted by a block. For instance, when the level in a Tank increases, it posts a future event to the Executive corresponding to the Tank’s “full time”. Once the simulation clock has advanced to this future event, the Executive sends an event message to the Tank, alerting it that it is full.
- A *current* event message occurs when a block wants to be activated before the simulation clock advances, but after it has completed its response to another message. For instance, instead of a Valve immediately calculating a new effective rate when its constraining rate changes, it will post a current event to the Executive letting it know that it will have to recalculate at a certain time. This gives all the other blocks in the model the opportunity to update before the recalculation occurs.

 In discrete rate models, blocks from the Value library typically neither post events to the Executive nor receive event messages from the Executive. This has important ramifications on the behavior of continuous blocks in discrete rate models.

### Value connector messages

Blocks in a discrete rate model send value connector messages either because a new number is needed by an input connector or because the value of an output connector has changed. These messages request updated information for the input connectors or notify connected blocks that the output value has changed. For example, if a Valve block’s  $R$  value input connector is connected and the Valve receives a new value at  $R$ , the constraining rate on the flow changes. This will cause a recalculation of the set of parameters in the model.

These messages work the same in discrete rate models as in discrete event models. They are discussed fully at “Value input and output connector messages” on page 261.

**Item connector messages**

Discrete rate models often have portions that are item-based, using blocks from the Item library. The Rate library’s Interchange block also has item connectors; it provides a mechanism for interacting with item-based blocks in a discrete rate model.

Item connector messages (primarily *wants*, *needs*, and *rejects*) use a conversation of messages to propel items from one item-based block to another through the model. The item connector messages work the same in discrete rate models as in discrete event models. They are discussed fully at “Item connector messages” on page 262.

**Flow connector messages**

Flow connectors provide the value of the effective inflow and outflow rates. Flow connector messages cause the effective inflow/outflow rate to be updated for all connected blocks each time the LP calculation determines that the effective rates have changed.

**Rate block flow messages**

As soon as a block from the Rate library receives a message, if it determines that the effective inflow and outflow rates might change, it initiates a propagation of these messages. This propagation of messages is used to define the LP area – the area of the model which could be impacted by the originating block’s change.

**Executive block flow messages**

When a calculation of a set of effective rates for the LP area has been initiated, the Executive block calculates the new set of effective rates and sends messages to the blocks in order to propagate the results. These message update all the blocks in an affected area with a new effective rate.

# **3D Animation**

## **Introduction to E3D**

Some things to know before you run 3D animation

For communicating a concept, a 3D representation of the spatial location of objects and their movement over time can be an extremely powerful and effective tool. The ExtendSim Suite package includes a next generation 3D animation capability that is fully integrated with the ExtendSim simulation environment.

Simulation is concerned with building a logical model of the behavior or performance of a system. 3D animation, on the other hand, involves a model of a physical space. In ExtendSim you can build a logical model that is separate from the physical model. When 3D animation is desired, ExtendSim translates the logical model into a 3D representation.

This ability to separate the logical simulation model from the 3D environment is an important ExtendSim advantage because:

- A logical model cannot be directly built in 3D space. If the physical and logical aspects were not separated, every aspect of your system would have to be created as a physical representation of the real system. That would be both counter-intuitive and tedious.
- You may not need 3D for your purposes, so creating a 3D animation for every model would be a waste of time.
- You can ignore how the model will appear in 3D until the logical model is finished. This increases productivity since you can concentrate on model behavior and results without being concerned about layout.
- In ExtendSim, 3D animation is available if and when you need it.

The ExtendSim 3D (“E3D”) environment is designed to be used both for model presentation and for model comprehension and debugging. The E3D window provides a fully three-dimensional representation of the world of the model. The objects modeled in the E3D window maintain information about their positions in three dimensions as well as their other physical and behavioral properties. The E3D window is tightly integrated with the rest of the ExtendSim simulation engine and is open source, so you can control every motion of every object and every aspect of what is happening in the 3D world.

 E3D animation is only available in ExtendSim Suite.

## What this chapter covers

This chapter discusses:

- Blocks and objects for 3D animation
- Overview of 3D animation in ExtendSim
- Prerequisites for the E3D module

## Blocks and objects for 3D animation

There are two common ways to create 3D animation in ExtendSim: change dialog settings and/or add specialized 3D blocks to a discrete event model, or build custom blocks that use the 3D functions to perform animations. The first approach does not require any programming; the second method uses ModL functions to create custom animations that are outside of the discrete event arena and, if you want, don't even require a simulation to be run.

### Item library blocks

The Item library blocks are 3D-enabled. This means that these blocks and the items that pass through them can quickly segue from the 2D to the 3D world. Most of the Item library blocks have two tabs with customizable settings that affect 3D animation:

- An Item Animation tab for selecting a 3D object to represent the items that leave the block and, in some cases, for choosing customizable aspects of their appearance (known as “skins”).

- A Block Animation tab for choosing a 3D object to represent the block in the 3D world. You can also define its scale, rotation, and location in the 3D window, as well as other properties.

Furthermore, the Item library's Transport block is useful for representing the movement of an item from one point to another along a path, while the Convey Item block represents moving items along a conveyor.

The most common way to use the E3D environment is to build a discrete event model using the Item library blocks and run the simulation with the E3D window open. This displays the model and its items, events, and results in the 3D world.

 ExtendSim includes an extensive collection of 3D objects to represent items and blocks. You can also add your own 3D objects.

### Animation library

Blocks in the Animation library perform specific 3D functions such as enabling sunlight or placing scenery objects or text boxes in the E3D window. The Animate 3D block provides a non-programming method to perform a 3D action in response to an item's passage. For instance, you can create or delete an object, mount one object on another, and so forth.

### Custom 3D objects and blocks

Since the E3D environment is open-source, you can modify the animation capabilities of existing blocks, create entirely new 3D objects and 3D-enabled blocks, and modify object behaviors. ExtendSim has numerous ModL functions that allow you to define custom behavior for 3D objects and events. And GarageGames provides the Torque Script scripting environment for the – this provides additional control over the animation environment and object behaviors (see “Torque Game Engine” on page 392 for GarageGames information.)

ModL functions can even be used to perform 3D animations that do not require that the model be run. For instance, the 3D animation of the Boids model was created using custom-built blocks and does not require the model to be run. The Boids model is located in the folder \Examples\3D Animation.

## Overview

The ExtendSim 3D (E3D) window is where 3D animation occurs. This is a separate window from the worksheet where you build and run a simulation model. When you run a discrete event simulation with 3D animation on, 3D-enabled constructs from the model worksheet are represented as graphical objects within the E3D window. The E3D window contains an editor so you can modify the 3D representation of the model, add objects, and edit the terrain.

For instance, by default the blocks of a discrete event model are represented in the E3D window by 3D block objects, the items that move through the model are represented by 3D item objects, and the model's connections indicate the path of the 3D item objects.

### Features

Since it is common to use the Item library to create discrete event models that are then run with 3D animation, the following features are described in regards to the Item library. Keep in mind, though, that you can build custom blocks that are also 3D-enabled.

#### Animation modes

Each model has a saved *3D animation mode*: QuickView, Concurrent, or Buffered. The selected mode controls aspects of the interaction between the ExtendSim application, the E3D window, and the ModL block code.

### **Blocks appear as objects in the E3D window**

In a discrete event model, the blocks occupy a 2D model worksheet. 3D-enabled blocks from that model, such as those from the Item library, also automatically occupy a position within the E3D window. By default some of these blocks appear as rectangular objects with the same icon as they have in the 2D model, while others appear as specific objects. For instance, the default is that Select Item Out blocks appear as rectangular objects in the E3D window but Activity blocks appear as machines.

Unless you unlink their 2D and 3D positions, the location of the block objects within the E3D window (their “3D position”) depends on their current position in the 2D model.

☞ The appearance and location of block objects in the E3D window can be changed from the default. This will be shown in the tutorial chapters.

### **3D items appear as objects and travel on pathways**

By default, pathways in the E3D window are initially related to the connections between Item library blocks in the 2D model. As you will see in “Tutorial III”, you can modify the default pathways by unlinking a block’s 2D/3D position and moving the block in the E3D window.

In addition, entirely new pathways can be designated by using the E3D Editor to create a named path between two arbitrary points in the model. Then when an item takes a specific route in the model, it uses the corresponding pathway in the E3D environment.

### **3D objects have collision capabilities**

If item objects are specified as being collidable (the default), they will not occupy the same space. This can be useful to show production lines, traffic patterns, waiting lines, and other queueing situations.

### **3D objects can mount other objects**

In ExtendSim, two or more 3D objects can be temporarily or permanently joined together. This is known as *mounting*. For example, a person can get a document from an out-box, transport it to another part of the model, and place it into an in-box. The document is mounted with the person during the journey from the out-box and unmounted at the in-box. Objects can even be mounted on other objects hierarchically; that is, a person can hold a bin that is holding some documents.

The position at which an object can be mounted on another object is called a *mount point*. Most objects have only one mount point, but some objects, like people, have several mount points that are selectable depending on what object will be mounted.

### **E3D environment is modifiable**

Using the E3D Editor you can directly modify the E3D window – add scenery objects, modify the terrain, and change lighting, ceiling colors, and other environmental aspects to fit the model. These changes to the basic 3D view are saved in environment files.

### **Torque Game Engine**

The E3D environment is based on the Torque Game Engine (TGE) by GarageGames, Inc. This is a powerful 3D rendering architecture that GarageGames retails as a source code development environment. This provides several advantages, including:

- There is a large and growing community of animation enthusiasts who are familiar with the TGE. They can offer advice and provide consulting services.
- A large selection of free and inexpensive ready-made TGE components is available. If a particular component doesn’t exactly suit your needs, it can be easily adapted.
- Components developed using other standards can be converted to the DTS format used by the TGE.



Information about the TGE and additional 3D resources can be found on the GarageGames web site at [www.GarageGames.com](http://www.GarageGames.com).

### Controlling the E3D environment

Every motion of every object and every aspect of what appears in the E3D environment can be controlled. Several ExtendSim features contribute to providing this capability:

- 1) Animation tabs in Item library block dialogs give you a choice of 3D objects to represent items and blocks in the E3D window. In some cases you can even select particular properties of the 3D object, such as its color or scale. The animation tabs are described on page 474.
- 2) Specialized blocks that provide 3D features or support E3D behavior. For instance, the Transport block (Item library) discussed on page 482 is useful for representing the movement of items from one point to another along a path. And blocks in the Animation 2D-3D library, described on page 482, can be used to cause text and 3D objects to appear in the 3D animation area or to alter the appearance of the 3D world.
- 3) Settings in the Simulation Setup and Options dialogs affect both the contents of the E3D window and the relationship between ExtendSim, the model that controls the E3D window, and the E3D window itself. For example, these settings can enable 3D directional sound effects or cause the animation to run after the simulation rather than concurrent with it. These dialogs and commands are described in “E3D Editor menu commands” on page 484.
- 4) The environment files can also be customized using the E3D Editor to define the appearance of the animation area – any permanent pathways and 3D objects as well as specific information about the terrain, sky, sun, and so forth – without affecting the appearance of the 2D model.
- 5) And finally, ModL functions and message handlers allow you to develop customized 3D-enabled blocks and specialized 3D functionality. ModL functions can be accessed through equation blocks or by programming.

### Prerequisites

#### Software and hardware

- E3D animation is only available in the ExtendSim Suite product.
- A graphics/video card capable of supporting 3D rendering. At a minimum, you will need at least 64 MB of memory on the card and a 3D accelerator. A faster card with more memory will improve the performance of the E3D window.
- The minimum memory required for E3D animation is twice the minimum for ExtendSim.
- If your computer does not have QuickTime installed, the color of the 3D objects in the preview area of Item Animation and Block Animation tabs for Item library blocks may not display correctly. QuickTime is Apple Inc.’s technology for handling video, sound, animation, graphics, and so forth. It can be downloaded for free onto a Windows or Macintosh computer from the Apple Inc. web site.

#### Preparation

This module assumes that you have either read the following or have equivalent experience:

- Tutorial module, Chapters 1 and 2, starting on page 13.
- If you are building discrete event models, as opposed to creating custom 3D-enabled blocks, you should complete the Discrete Event module that starts on page 89.

## How the E3D module is organized

- Introduction
- Tutorial I – exploring the E3D window
- Tutorial II – animating an existing model in 3D
- Tutorial III – advanced features such as creating a path and mounting one item on another
- Environment Files and the E3D Editor
- Objects – creating, deleting, scaling, rotating, etc.
- Movement, Paths, and Terrains
- E3D tips and reference
  - Performance considerations
  - ExtendSim commands for 3D
  - Animation tabs in the Item library blocks
  - Animation 2D-3D library
  - E3D Editor menu commands

# **3D Animation**

## **Tutorial I**

Exploring the E3D window

This chapter focuses on the E3D window and the appearance of a simulation model in that window. The purpose of this chapter is for you to become familiar with the E3D terminology and environment. This chapter covers:

- Exploring the E3D window
  - Opening the window and learning about its options
  - Navigating within the window
- 3D animation modes
- Running a simulation and animating it in 3D


The Tutorial II chapter that follows this one will show you how to add 3D features to an existing 2D discrete event model.

## The E3D environment

The first part of this tutorial examines the E3D window and explores its features.

### Opening the E3D window

- ▶ Launch ExtendSim.
- ▶ Give the command File > New Model to open a blank model worksheet.

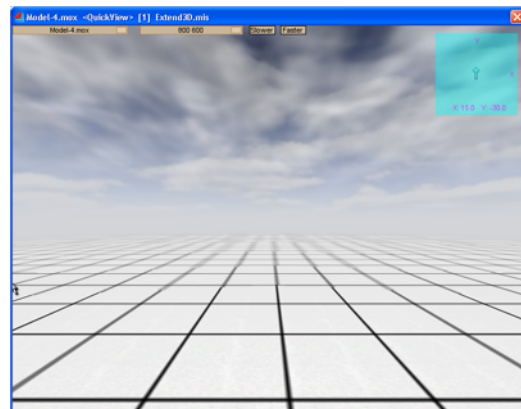
 The E3D window will not open unless a model worksheet is open. Even if a model is open, the E3D window will not open if you are not licensed for the ExtendSim Suite product.

- ▶ Give the command Window > E3D Window or use the toolbar's Open E3D Window button.


Depending on the speed of your processor, it may take a little while for the window to open.

### Exploring the E3D window

The E3D window opens as shown to the right. The window has four interface controls across the top of the window, a title bar, and a MiniMap. The rest of the window is the viewing area that displays the animation environment and a 3D rendering of 3D-enabled blocks from the simulation model. (Since the associated model worksheet is blank, there are no blocks in the E3D window at the right.)



E3D Window in QuickView mode - 2D model is empty


 You can open the E3D window for any model. However, if there are no 3D-enabled blocks in the model, the default E3D environment will not contain any 3D objects.

#### Interface controls

- The top left control is a popup menu that displays the name of the associated model. Only one E3D window can be open at a time, so if multiple models are open, this popup menu allows you to choose which model will be associated with the E3D window. The name of the selected model (in this case, Model-1) appears in the popup and also in the title bar at the top of the window.
- The next control is a popup menu for choosing the specified resolution, or size, of the window. Clicking this popup displays a list of possible sizes for the E3D window. Selecting a different size will cause a

redraw of the 3D window at the new resolution. Lower resolutions will make the E3D window smaller; higher resolutions will make the window larger. Resolution information is stored with the model.


- The third and fourth controls are for selecting the speed of the animation, either Slower or Faster. The selected speed factor (0.25 to 8) is displayed in the title bar in brackets; the default is a factor of 1. Information about the selected 3D speed setting is stored with the model.

 In QuickView mode the speed buttons in the E3D window do not affect the speed of the simulation, only the speed of the 3D animation. In Concurrent and Buffered modes, there is a direct correlation between 3D animation speed and simulation speed.

### Title bar

The title bar at the top of the E3D window lists the name of the model associated with the E3D window, the 3D animation mode enclosed in chevrons (in this case, “<QuickView>”), the 3D speed factor displayed in angle brackets (in this case, “[1]”), and the name of the model’s environment file (by default, new models use the Extend3D.mis environment file).

Modes are discussed in “3D animation modes” on page 400, the speed conversion factor is discussed on page 478, and environment files are described starting on page 432.

 The title bar also includes the ExtendSim icon. Clicking the icon (Windows only) reveals the Move, Editor, and Close commands. The Editor command enables the E3D Editor; editors are discussed starting on page 433.

### MiniMap and camera

The cyan square in the upper right hand corner of the E3D window is the *MiniMap*. This is a miniaturized display of the E3D window; it shows the position of a virtual camera and any local objects as if seen from directly overhead a location in the E3D window.

The *3D camera* is a virtual object in 3D space that represents your point of view of the E3D window – what you see in the window is as if you were looking through the current position of the camera.

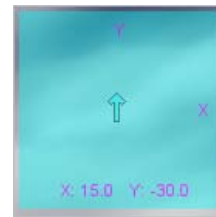
The camera can be manipulated programatically using ExtendSim’s ModL functions or by the use of keyboard controls, as discussed in the next topic. As you move the camera around in 3D space, the contents of the MiniMap will change to reflect the location and direction of the camera, as well as the presence of any other objects in the immediate area.

The arrow in the center of the MiniMap represents the location and direction of the camera. The X and Y labels at the right and top hand sides of the map, respectively, show the direction of positive X and Y values. The X and Y values at the bottom of the MiniMap report the location of the camera within the E3D window.

The MiniMap is displayed by default; you can hide the display by unchecking it in the Edit > Options > 3D tab.

### Animation area

The E3D window opens with the default environment for the animation area – an unbounded 3D world with a cloudy sky, a sun, and a flat gridded floor. When the E3D window is associated with an empty model worksheet, as you have done in this chapter, the 3D world will be empty. When the E3D window is associated with a model, it will contain 3D objects that represent the 3D-enabled blocks in the model.



MiniMap

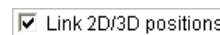


You can add scenery and fixtures to the E3D window and use 3D features and settings to modify the terrain, lighting, ceiling colors, and other environmental aspects to fit the model. And you can change the 3D objects and their location to enhance the 3D animation. This will be shown in other chapters.

### Selecting and moving

Clicking an object in the animation area of the E3D window selects it and causes it to display with a cyan frame around it. Clicking again unselects it. If the 3D object is the representation of an ExtendSim block, double-clicking the object will open the dialog box of the associated block.

By default, selecting and moving a block in the 2D model also moves its object in the E3D window. For instance, if you place an Activity block (Item library) in the model, you can click and drag it around the worksheet. When you release the mouse, the Activity's representative 3D object will correspondingly change location in the E3D window. This is because the 2D position of a block is linked by default to the 3D position of the object, as shown in the Block Animation tab and the screenshot to the right, above.

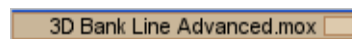


Linked positions

You need to use the E3D Editor to move a 3D object directly in the E3D window. If the object's position is linked to a block, and you move the object with the Editor, it will also move the block. The Editor is described in "The E3D Editor" on page 433.

### Changing the associated model

When the E3D window opens, it will be associated with the currently active model worksheet. If you make another model the active window, it will not change the E3D window's associated model. To have that model window be animated in the E3D window, you must select it from the popup menu at the top left of the E3D window. This will associate the new model with the E3D window.



Associated model popup menu

Each time you use the associated model popup menu, the 3D window will refresh and redraw with different contents based on the new model and whichever environment file it uses. Any 3D commands or function calls from models other than the currently associated model will be ignored.

### Navigating within the E3D window

The virtual 3D camera is used to navigate the E3D window. This can be done through the use of keyboard controls to pan and zoom. (You can also navigate using ModL functions in an equation block or block that you program.)

- ▶ Click the E3D window to make it active.
- ▶ To navigate within the E3D window using keyboard controls:
  - ▶ Use the arrow keys on your keyboard to move left, right, forward, and backward. (For left-handed movement, use the W (forward), S (backward), A (left), and D (right) keys.)
  - ▶ To pan the area, change the direction of the camera by right-clicking the mouse (Windows) while moving the mouse around the window. (On Macintosh, press the control key while mousing around the window.)
  - ▶ Use the mouse to change direction and the keys to move, both simultaneously, to navigate around the 3D world.

Note how the MiniMap changes as you navigate the 3D space.

- ☞ You can also use the *m* key as a “mouse-look toggle”. Clicking the *m* key once allows you to use the mouse to change the direction of the camera without having to right-click. To return the mouse to normal use, click the *m* key again.

## Manipulating the E3D window

### Opening the window

Under the following conditions, the E3D window automatically opens whenever the associated model opens:

- If the command Run > Show 3D Animation has been checked for that model
- Or, if *Show 3D animation during simulation run* has been checked for that model in the Run > Simulation Setup > 3D Animation tab

If either of these have been checked, and the E3D window has been closed, it will reopen when the simulation runs.

To manually cause the E3D window to open, do one of the following:

- Give the command Window > E3D Window
- Or, click the *Open E3D Window* tool in the toolbar

- ☞ A model must be open for the E3D window to open. Also, it may take some time for the E3D window to initialize and open.

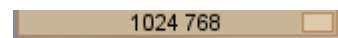
### Moving the window

By default, the E3D window behaves like a child window of the ExtendSim application. With this setting, the window can be moved within the ExtendSim application window just like any other ExtendSim window.

There is also an option (Windows only) to allow the E3D window to act like an independent application window. To do this, choose Edit > Options > 3D tab and check the box for *3D window outside application*. After restart, the E3D window will float outside the ExtendSim application window. This is especially useful if you have multiple monitors and you want the model displayed on one monitor and the E3D animation on the other. (This option is not needed for the Macintosh since this is the E3D window’s default Macintosh behavior.)

### Changing the window size

The resolution menu discussed in “Interface controls” on page 396 allows you to change the size of the E3D window to make it easier to view.



Resolution popup menu

### Closing the window

The E3D window can be closed:

- By clicking the close box in its title bar
- With the *esc* (escape) key
- By selecting *Close* from the menu that appears when you click the ExtendSim icon in the E3D window’s title bar (Windows only)

All three options leave the model window(s) open and the simulation running (if applicable). The E3D window automatically closes whenever you exit ExtendSim.

☞ The E3D window does not close when you close the associated model; it stays open so you can use the left-most popup menu to select another model to animate. Likewise, the model does not close, nor does the simulation stop, when the E3D window is closed.

### 3D animation modes

Each model has a saved *3D animation mode* which is selected in the 3D Animation tab of the Run > Simulation Setup dialog and displayed in the E3D window's title bar. The selected mode controls aspects of the interaction between the ExtendSim application, the E3D window, and the ModL block code.

#### Mode descriptions

There are three 3D animation modes:

- *QuickView* shows a default representation of the movement of items in the E3D window while the simulation is running. In this mode, only one item object moves at a time.
- *Concurrent* is a more realistic animation than QuickView and shows only the movement that requires simulation time. The 3D animation displays the movement of multiple item objects simultaneously and runs during the simulation run.
- *Buffered* is similar to Concurrent except it runs the 3D animation after the simulation has ended.

These modes are discussed in more detail on page 477.

#### QuickView versus Concurrent or Buffered

Each 3D animation mode has advantages and disadvantages. The QuickView mode is useful for getting an instantaneous 3D representation of a model. And with some easy changes to dialog settings, you can quickly run a 3D animation that has a more realistic appearance than the default. However, because of the way they handle object movement, the Concurrent and Buffered modes are more likely what you would use for presentation.

The primary difference between the QuickView and the Concurrent or Buffered modes has to do with time and how the display of items in the E3D window relates to the real world clock.

- In the QuickView mode, the amount of real-world time it takes a 3D object to move from one point to another is not representative of the simulation time it takes that object to move. When the block code executes a move of an item from one block to another, the code will instruct the E3D window to show a representation of that move. This will be done without regard to the ratio of simulation time to real time. This means that the length of time it will take that item to travel visually from one block to another will be just based on the model's animation speed.
- In the Concurrent and Buffered modes, the simulation time to real-world clock time ratio is carefully respected, so each simulation time unit will by default take one second to display in the E3D window.

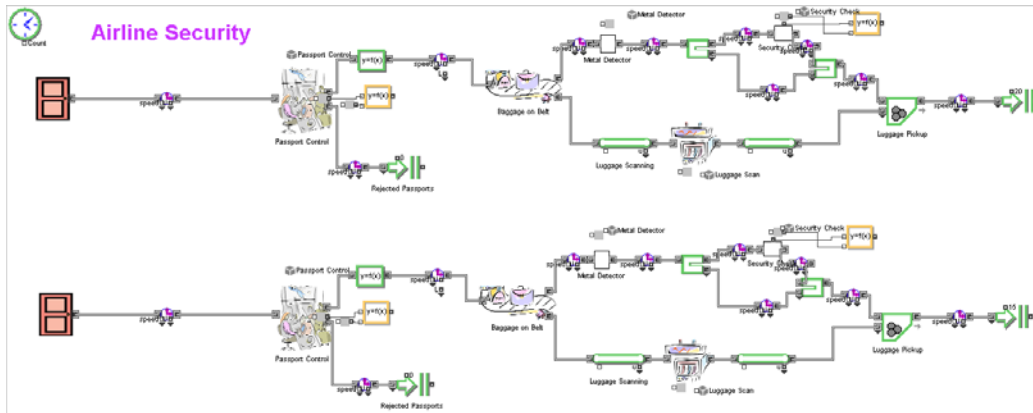
### Running a model with 3D animation

Now that you're familiar with the E3D window and terminology, the next section will show you how to open and run a model that has been adapted for 3D animation.



## Opening the model

- ▶ Open the Airline Security model located in the \Examples\E3D Animation folder.

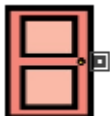


Airline Security model

### About the model

The Airline Security example is a discrete event model that represents passengers moving through security lines at an airport. The model includes:

- Waiting lines
- Carry-on baggage that must be associated with each passenger
- A passport examination desk before the security area
- Conveyors with x-ray machines for scanning the bags
- Walk-through metal detectors
- A special security check station where a random number of passengers are searched



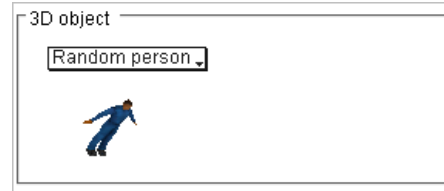
In the Airline Security model, people randomly enter the security area from each of the two doors on the left. The doors are hierarchical blocks that establish the frequency of passenger arrivals and specific information that will be used in the 3D world.

The passengers must first undergo a passport check; those who are rejected exit the airport. The remaining passengers continue to a conveyor where they deposit their luggage for screening by an x-ray machine. The passengers are scanned by a metal detector, and a random sample is selected for additional search procedures. After the screening process, passengers are united with their specific piece of luggage and exit the security area.

### Accommodations for the 3D world

The model has been specifically adapted to animate in the E3D window in Concurrent animation mode. The differences between this model and a standard discrete event model are:

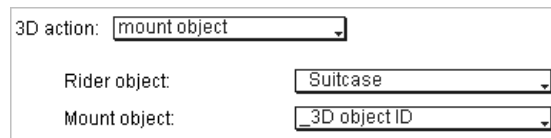
- 3D objects have been selected in the Item Animation and Block Animation tabs of Item library blocks. For instance, each passenger is a *Random person*, as shown in the Create block’s Item Animation tab at the right. (You won’t be able to see these selections unless you have the E3D window open.)



3D object selected in Item Animation tab

- Several Transport blocks have been added to animate real-time delays in item movement. These are placed between other Item library blocks so the E3D window will know how far the item travels from one block/object to another and how long it will take the item to get there. So that delays in movement are realistically represented, this information is used as part of the conversion of simulation time in the 2D model to animation time in the E3D window.

- Among other tasks, Animate 3D blocks (Animation 2D-3D library) within hierarchical blocks are used to create pieces of luggage that are then “mounted” to each passenger.



Animate 3D dialog - mounting luggage

- 3D Text blocks (Animation 2D-3D library) have been placed in the model at various locations to serve as labels in the E3D window. For instance, a 3D Text block places the label “Passport Control” at the appropriate location in the 3D window.
- 3D Scenery blocks (Animation 2D-3D library) add stationary figures for the various security stations.
- A custom environment file has been created to facilitate the movement of passengers along paths.

Notice that no programming was required to adapt this model for 3D animation. The only accommodations were the addition of some animation-specific blocks, the selection of 3D options in block Item Animation and Block Animation tabs, and modifications to the environment file.

### Running the model with 3D animation

To open the E3D window and animate the model:

- ▶ Give the command `Window > E3D Window` or click the *Open E3D Window* button in the toolbar
- ▶ Verify that the E3D window is associated with the Airline Security model
- ▶ Position the E3D window in a convenient location

If you have multiple models open, be sure the model associated with the E3D window is “Airline Security”. The associated model is listed in the E3D window’s popup menu, shown at right.



Associated model popup menu

- ▶ Give the command `Run > Run Simulation`, or click the Run Simulation button in the toolbar

As the animation runs, you will be able to see security personnel checking passengers' travel documents and passengers placing their bags on the x-ray conveyors, going through metal detectors, and then picking up their correct bags at the end of the screening process.

This model illustrates many concepts used in the 3D world such as material handling and the routing of passengers. Notice that arriving passengers are randomized (male, female, etc.) and are of differing heights. This indicates just some of the flexibility of the ExtendSim 3D modeling environment.

### Next step

The next step is to add animation-specific features to an existing model. You do this in the Tutorial II chapter that comes next.





# **3D Animation**

## **Tutorial II**


Adding 3D animation to a model

This chapter is an extension of, and assumes you have read, the preceding chapter. It will show you how to adapt a model so that it generates a concurrent 3D animation. It covers:

- Using Concurrent mode to have items move simultaneously
- Animating items as 3D objects
- Creating 3D objects to represent blocks
- Adding scenery and labels to the E3D window
- Using a 3D Controller block to clear items at the end of the simulation
- Causing the model to automatically open the E3D window

Enabling 3D animation for this model is fairly simple – just set options in block dialogs and add some specialized blocks to the model worksheet.

The chapter that follows this one will show how to perform more advanced tasks: using the Transport block to simulate travel time, determining the length of the path using block positions, mounting an object on an item object, and creating custom paths that are unlinked from the position of blocks in the 2D model.

 The models for this tutorial are located in the folder \Examples\Tutorials\E3D Animation\Production Line. The examples assume some familiarity with discrete event simulation.

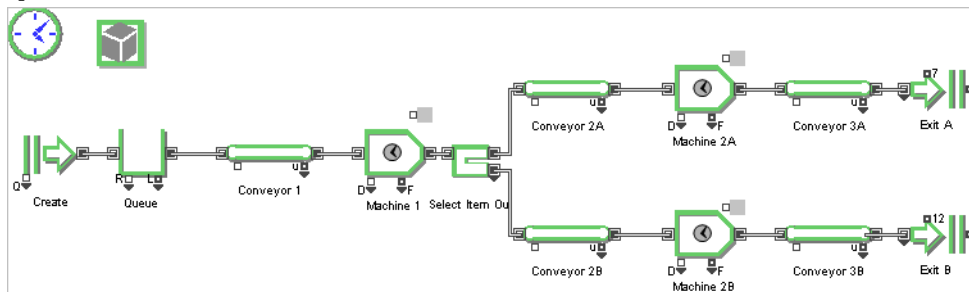
### Adding 3D behavior to an existing model

In this section you will modify a discrete event model so that it animates in the E3D environment.

#### The goal

Before starting, look at the finished model to get an idea of what you will accomplish.

- ▶ Open the Production Line Final model



Production Line Final model

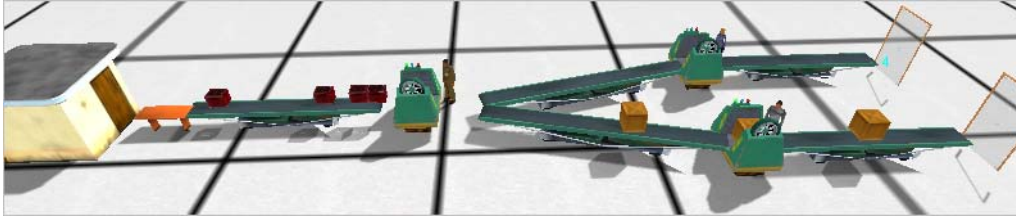
The command Run > Show 3D Animation has been checked for this model, so opening the model also opens the E3D window, and running the simulation also runs the 3D animation.

 If the E3D window is already opened when this model is opened, be sure the Production Line Final model is the one associated with the E3D window.

- ▶ Run the simulation



► Navigate around the E3D window



E3D window for Production Line Final model

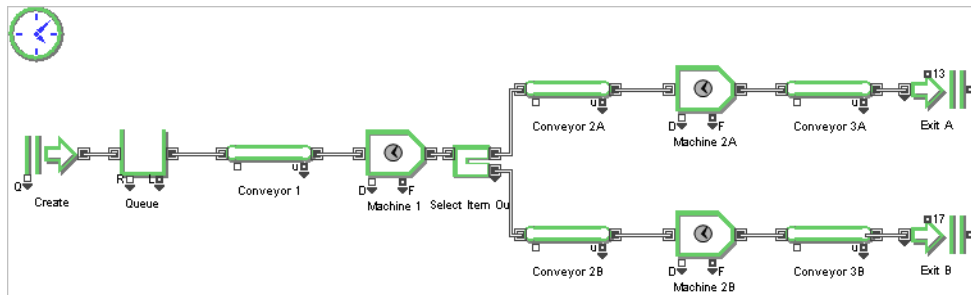
This is a model of a process where crates full of bottles move along conveyors and are processed by machines. A random number of crates are sent on one or another branches of the line.

Move the camera in the E3D window so you can see the entire model; zoom in and out to see the detail.

**Open the starter model**

Now that you have seen what the final model will look like, you can begin the process of animating a model in 3D:

- So that the E3D window is always associated with the correct model, close any open models
- Open the Production Line Start model



Production Line Start model

**Model particulars**

This model represents a production line consisting of five conveyors and three machines. After the first conveyor and machine, the line separates into two branches, each of which ends at an exit.

*About the model*

The production line processes crates full of bottles.

- Crates are generated by a Create block, approximately one every 1.1 seconds
- They are held in a Queue until pulled into the system
- Conveyors are 8 or 10 meters in length, run at 10 meters per second, and have a capacity of 8 or 10 items (depending on the length)
- The length of a crate is 1 meter
- Depending on the machine, the crates are processed for 1 or 2 seconds
- Attendants are posted near each machine
- Crates are sent randomly to one of the two branches by a Select Item Out block



- All the blocks are from the Item library
- The length unit is meters, the time unit is seconds, and the simulation ends after 180 seconds
- The 3D animation uses the default Extend3D.mis environment file

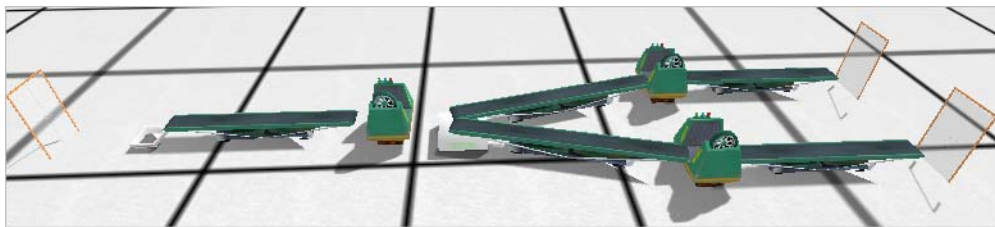
#### *Accommodations for 3D modeling*

If you are familiar with discrete event modeling, you may notice a difference between this model and a typical single queue/two server model. In this model, the location argument for the exponential distribution in the Create block is 0.1. In a logical model, the location argument is often left at zero for expediency. For this model, the location value ensures that the interarrival time can never be zero – there will always be some time between item arrivals.

Other than that, this discrete event model is the same as you would typically build without considering 3D animation.

#### **Running the 3D animation**

- ▶ Use the *Open E3D Window* button in the toolbar or give the command Window > E3D Window to open the E3D window.



E3D window for starter model

Notice that the window opens in the default QuickView mode.

- ▶ Place the E3D window in a convenient location and, if desired, change the window's resolution using the resolution interface control at the top of the window.
- ▶ Run the simulation. Since the E3D window is open, the 3D animation will also run.

As the simulation runs, you can see green balls moving in the E3D window just as they might in a 2D animation. A difference from the 2D model is that the blocks are represented in the E3D window by their default 3D objects – the Create and Exit blocks are displayed as doors, the Queue as a bin, the Convey Item blocks as conveyors, and the Activity blocks as machines. The conveyor belt moves, and when a green ball enters the machine the lights go on and the machine moves, indicating that a process is taking place.

- ☞ You can have 2D animation turned on simultaneously with 3D animation, but performance will be better if only one type of animation is on at a time.

#### **Save a model to explore**

So that you have a model to work with during the tutorial:

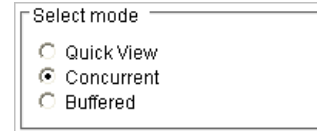
- ▶ If it is still running, stop the simulation
- ▶ Give the command File > Save Model As and save the model as “3D Production Line”.



**Cause objects to move simultaneously**

Since QuickView mode is just a visual representation of the 2D model, only one object moves at a time and items can stack on top of each other in waiting areas. So that the objects that represent items can move simultaneously and wait in a queue line, the 3D animation needs to be in Concurrent mode.

- ▶ If it is still running, stop the simulation
- ▶ Give the command Run > Simulation Setup
- ▶ In the dialog's 3D Animation tab, choose **Select mode: Concurrent**. Then close the Simulation Setup dialog.



Concurrent mode selected

If you run the simulation again, the green balls should be moving simultaneously and should order themselves in a line if there is more than one of them waiting to be processed.

**Create objects to represent items**

The specification for this animation is that crates are moving around rather than green balls. To remedy this, you need to create an object to represent the model's items. This involves selecting an object from a popup menu in a block's Item Animation tab, as shown below.

**An object for the Create block's items**

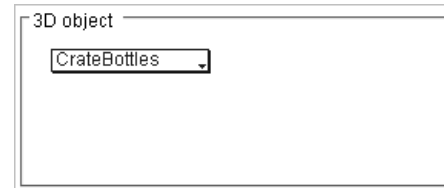
In the Create block's Item Animation tab, notice that the behavior is set to *Select item animation*. This allows quick customizing of the 2D and 3D animation from popup lists of objects.

- ▶ If it is closed, open the E3D window



The E3D window must be open in order to select a 3D object.

- ▶ Open the Create block's Item Animation tab
- ▶ In the tab's 3D object frame (shown at right), select the 3D object **CrateBottles**



Selecting 3D object for item animation

If you run the simulation again crates of bottles, rather than green balls, move from object to object.

**Objects for the items that go to the branches**

The Select Item Out block separates the production line into two branches. In the block's Item Animation tab, each row of the table represents one of the two outputs.

So that each branch will show a slightly different 3D object:

	Animation Option	2D Picture	3D Object	Skin 1	Skin 2
0	Change to		Crate	old	base
1	Change to		Crate	base	base

Selecting 3D objects in Select Item Out block

- ▶ Open the Select Item Out block's Item Animation tab
- ▶ In the **Animation Option** column, choose **Change to** for both rows
- ▶ Select:
  - ▶ **3D object: Crate** in the third column of the first row
  - ▶ **Skin 1: old** for the fourth column of the first row
  - ▶ **Skin 2: base** for the last column of the first row



The first row of the table should look like the screenshot, above.

- ▶ For the second row, select **3D object: Crate**, **Skin 1: base**, and **Skin 2: base**

**Save the model**

- ▶ Save your 3D Production Line model. (The tutorial model is named Production Line 2).

**Create objects to represent blocks**

For this model, most of the default block/object correlations (Activity blocks as machine objects, Convey Item blocks as conveyor objects, and so forth) are exactly what you want. Other objects need to be changed.

This section shows how to create objects and waypoints to represent blocks. Creating an object to represent a block is similar to creating an object to represent items, except the object is selected in the Block Animation tab.

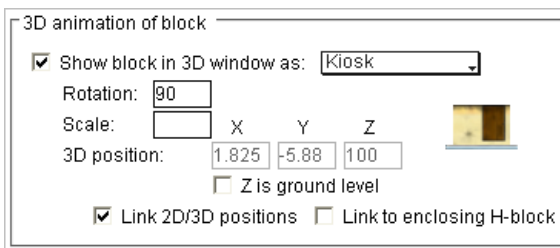
 Objects that represent blocks are created at a size that corresponds in scale to other objects.

**3D objects to represent blocks**

Some of the blocks should have their default object representations changed. For instance, instead of a door, the Create block could be represented as a supply room and the Queue block as a table.

In the Create block's Block Animation tab:

- ▶ If it is closed, open the E3D window
- ▶ Select **Show block in 3D window as: Kiosk**
- ▶ Enter **Rotation: 90** to turn the kiosk to the proper position



Selecting 3D object to represent Create block

In the Block Animation tab of the Queue block:

- ▶ Enter **Show block in 3D window as: Table**

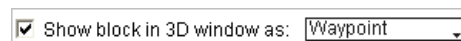
This will place a small building at the same location as the previous door object and replace the Queue's default bin object with a table.

**Waypoints**

Blocks that are superfluous to a 3D animation, such as the Select Item Out block in this model, should have their appearance in the E3D window minimized. Rather than being represented by a 3D object, these blocks can be represented by a marked position in the E3D environment. This minimization of the object is accomplished using *waypoints*.

In the Block Animation tab of the Select Item Out block:

- ▶ Enter **Show block in 3D window as: Waypoint**



Selecting Waypoint

When you do this, the object representation of the Select Item Out block (represented as a block shape in the E3D window) will disappear and be replaced by an invisible waypoint.

**Save and run the model**

- ▶ Save your 3D Production Line mode. (The tutorial model is named Production Line 3.)
- ▶ Run the model with the E3D window open.

You should see crates traveling from a storage area, being processed by machines that are oriented in the correct direction.



## Enhancing the model

So far you have adapted the 2D model to run in 3D. But there are some additional steps you can take to enhance the animation.


### Add scenery

The model has machines, but no workers. This is easily fixed by adding *scenery*. Any type of fixed object that is not already represented by a block in the 2D model can be represented as scenery – people who don't move, furniture, equipment that doesn't perform an activity, walls, trees, and so forth. There are two ways scenery can be added to a model:

- 1) Add a 3D Scenery block (Animation 2D/3D library) to the 2D model. This is the simplest method and is shown below.
- 2) Use the E3D Editor to add scenery to the E3D window. This has the advantage that it does not change the 2D model; it is discussed in “Using the E3D Editor to create scenery” on page 447.

### Adding workers

For this tutorial, the workers don't move but are instead part of the scenery.

- ▶ Place a 3D Scenery block (Animation 2D/3D library) close to each of the three Activity blocks labeled Machine. By default, the block appears with a minimized icon. 
- ▶ Be sure the E3D window is open so you can select the 3D object 3D Scenery
- ▶ In the dialog of each of the three 3D Scenery blocks:
  - ▶ For **Show block in 3D window as**. Then choose a Male or Female object and whichever Skin 1 and Skin 2 you want.
  - ▶ Enter **Rotation: 225**
  - ▶ Unselect the checkbox **Show 3D footprint in 2D model**. This option is sometimes helpful when placing blocks in a 2D model; you do not need it for this model.

The 3D Scenery blocks are linked to the objects. If the workers don't appear in the correct location in the E3D window, move the 3D Scenery blocks on the model worksheet closer to the Activity blocks. When you release the mouse, the objects in the E3D window will move accordingly. You can also play with the rotation to orient them as you want, or change their scale relative to the size of other objects.

### Add a 3D Controller block

At this point you may notice that when you stop the simulation the crates stay wherever they are.


- ▶ Add a 3D Controller block (Animation 2D-3D library) anywhere in the model.
- ▶ In its dialog, check **Clear items in 3D window when simulation ends**.

When you run the simulation, all the crate/items will disappear when the run ends but the scenery (the workers) and the objects that represent the 2D blocks (the kiosk, conveyors, machines, and doors) remain.



3D Controller

- ▶ Save your 3D Production Line model. (The tutorial model is named Production Line 4.)

 Since item objects will automatically be cleared from the E3D window at the start of the next simulation run, the 3D Controller block is not really necessary for clearing items in this model. Note, however, that the block has several uses for controlling the E3D window, such as turning on/off the clouds and changing the color of the ceiling/sky.

### Launch with the E3D window

So that you don't have to manually open the E3D window, you can choose to have the window open automatically each time the model is opened. To do this, do one of the following:

- 1) Either give the command Run > Show 3D Animation
- 2) Or, in the Run > Simulation Setup > 3D Animation tab, check *Show 3D animation during simulation run*

If the E3D window is subsequently closed, it will reopen when the simulation is run.

☞ The final model, with the E3D window opening when the model is launched, is labeled Production Line Final.

### Some things to notice

Now that you are familiar with this model, it is a good time to investigate some 3D features.

#### Internal animation

As you saw for the Production Line models, some 3D objects contain internal animation that shows the status of the object during the simulation. For instance, the Machine object supports four internal animation states (running, idle, blocked, and down) while the Conveyor animates a running belt.

If you don't want to see this behavior, unselect *Enable animation of 3D object* in the Block Animation tab of the activity-type blocks (Activity, Convey Item, Transport, and Workstation).

☞ The default object for a Transport block is a Waypoint so this option is unchecked by default.

#### Rotation of 3D objects

While you needed to rotate the Kiosk and scenery objects, most objects are oriented correctly for the flow of items in this model. This is because ExtendSim will try to place objects using the correct rotation for the model.

For example, the machine objects are oriented for a left to right flow of items. Furthermore, you did not need to change the rotation in the Convey Item blocks. When the E3D window opens, the rotation parameter for this block's object automatically adjusts to reflect how the block is connected in the model. For instance, Conveyors 2A and 2B are horizontal in the 2D model but are displayed at an angle in the E3D window.

#### Mounting objects

You may have noticed that the item objects move in the E3D window as you would expect them to – the crates move along the conveyor and the machines process the crates as they arrive. You will see how to explicitly mount objects in the next chapter. But for this model, there was no need to explicitly cause one object to be mounted with another object

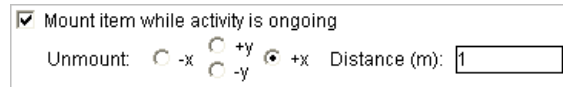
The movement of items in the Production Line model is due to two E3D features:

- Item objects automatically use the *from* and *to* location information in the Transport Animation tab of the Convey Item block. The default is that the *from* location is the previous non-passing block and the *to* location is the next non-passing block. This causes the item object to be displayed as moving along the path represented by the Convey Item block's object. (The options on the Transport Animation tab will be discussed further in the next chapter.)

From location is:	<input type="text" value="previous non-passing block"/>
To location is:	<input type="text" value="next non-passing block"/>



- By default the Item Animation tabs of Activity blocks are set to *Mount item while activity is ongoing*. This causes the item object (the crate) to stay with the Activity's object (the machine) until it has finished its processing. (As you will see in the next chapter, you may not always want that behavior.)



### Moving blocks linked to objects

For the Production Line models, each of the objects in the E3D window are linked to the blocks in the 2D model. This is seen in their Block Animation tabs, where the checkbox *Link 2D/3D positions* is selected by default.

If the 2D and 3D positions are linked, moving most types of blocks in the model will also directly move their objects in the E3D window. However, the Convey Item block has more complex behavior. If its Block Animation tab is set to *Stretch 3D object to conveyor's length* (the default setting), moving the block itself will have no effect on the position of its 3D object. That is because its displayed length is based on the starting and ending points of the conveyor in the model. If the *from* location is the *previous non-passing block*, and the *to* location is the *next non-passing block* (the default settings), the conveyor will be stretched between those two locations regardless of the position of the Convey Item block.

For example, moving Conveyor 1 between the Queue and Machine 1 will not change the location of its conveyor object in the E3D window. However, moving either the Queue or Machine 1 will result in a change in the conveyor's position.

You will not be able to see the change in position until the simulation is run.

### Conveyor

The Convey Item block (Item library) moves items along a conveyor, oven, cooling unit, moving walkway, or any other type of moving path. The items travel from the starting point to the end along the length of the conveyor.

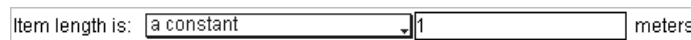
The block has three options for setting the travel time:

- Move time. The item is delayed for the specified time; speed and length are ignored.
- Speed and length. Speed and length are entered in the dialog.
- Speed and calculated length. Speed is entered in the dialog; length is determined based on dialog settings and block positions (the *from* and *to* locations).

These options are described fully on page 188. For this model, the conveyors' speeds and lengths are all known and have been entered in their dialogs. In the next chapter, you will see how to use the *speed and calculated length* option to cause the block to automatically determine its length.

### Item length

The Behavior tab of a Convey Item block has options for specifying the length of items. The item's length can be a constant, from an attribute, or based on length and capacity. In the Production Line models, item length is set at 1 meter for all the Convey Item blocks.



Item length setting



Whichever item length option is selected, the 2D picture or 3D item object does not visually change. However, the item length does affect calculations for accumulation and capacity, as well as the timing of when items are pulled onto and released from the block.

- ☞ If either *speed and length* or *speed and calculated length* has been selected as the travel time, the item's length is expressed as either feet or meters. However, if *move time* has been selected as the travel time, the item's length is expressed in time units. Thus for a move time of 10 time units, a constant item length would be x time units.

### Conveyor capacity

The capacity for a Convey Item block can be entered in the block's dialog. However, regardless of what is entered in the dialog, the block's actual capacity is the lesser of either the entered capacity or the length of the conveyor divided by the length of the item.

# **3D Animation**

## **Tutorial III**

Advanced 3D topics

This chapter is an extension of, and assumes you have read, the preceding two chapters. It will show how to:

- Use Transport blocks to animate people walking from one location to another
- Determine the distance the people need to walk based on block positions
- Mount a 3D object on an object that represents an item
- Use hierarchical blocks in 3D animation
- Unlink an object's position from the corresponding block's position
- Create custom paths that are unlinked from the position of blocks in the 2D model
- Create and save an environment file

☞ The models for this tutorial are located in the folder \Examples\Tutorials\E3D Animation\Bank Line. These bank line models assume some familiarity with discrete event modeling.

### Animating a bank line

This chapter illustrates how to use some of the more advanced ExtendSim 3D capabilities. So that you can focus on the features in this chapter, the example model already has some accommodations for 3D animation. Those 3D accommodations were covered in the preceding chapter and are listed on page 418.

#### The goal

Before starting, look at the finished model to get an idea of what you will accomplish.

- ▶ Open the 3D Bank Line Final model

This is a model of customers arriving, waiting in a line, being served by one of two tellers, and then exiting a bank. The command Run > Show 3D Animation has been checked for this model, so opening the model also opens the E3D window, and running the simulation also runs the 3D animation.

- ▶ Run the simulation
- ▶ Navigate around the E3D window

Move the camera in the E3D window so you can see the entire model; zoom in and out to see the detail. As the simulation runs, you will see customers entering the bank, forming a single line, getting their paperwork ready, walking to one of the tellers, being waited on, and leaving the bank.

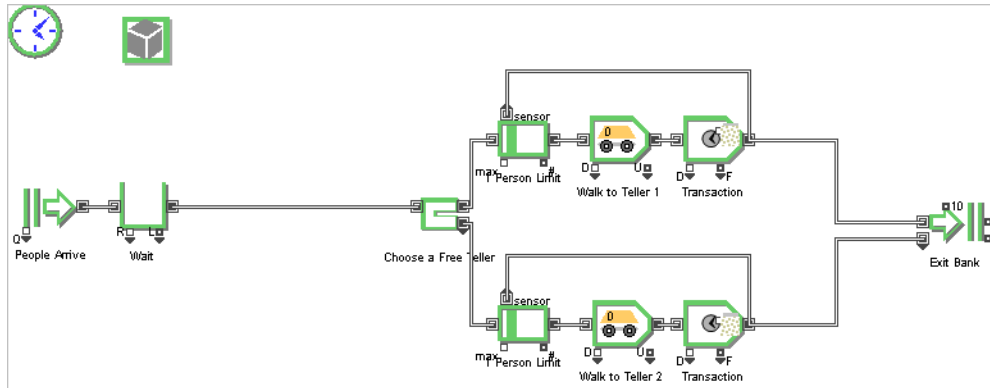
- ▶ Close the 3D Bank Line Final model

#### Open the starter model

Now that you've seen the goal, you can start adding 3D functionality to a starter model.



- ▶ Open the 3D Bank Line Start model. (Unlike the final model, this model is set to open without opening the E3D window.)



3D Bank Line Start model

### Animate the model in 2D

So that you can see some action in this model:

- ▶ Select the command Run > Show 2D Animation
- ▶ Run the model

### Model particulars

In this model, customers enter the bank randomly, approximately one every 5 seconds. Gate blocks restrict access to the tellers so that each teller can have only one customer in front of them at a time; all other customers must wait in the Queue block.

When a teller becomes available, the first customer in the queue walks to that teller. The walk takes 5 seconds and Transport blocks (labeled Walk to Teller) are used to delay the customer for that amount of time. Activity blocks (labeled Transaction) represent the time it takes for the transaction with the teller. The transaction time is specified by a triangular distribution with a minimum of 2, a maximum of 10, and a mostly likely time of 5. After finishing the transaction, the customer exits the bank.

### Differences from typical discrete event models

If you are familiar with discrete event modeling, you may notice some differences between this model and a typical single queue/two server model:

- 1) The location argument for the exponential distribution in the Create block (labeled People Arrive) is 1. In many discrete event models the location argument is left blank for expediency. For this model, a location value of 1 ensures that the interarrival time can never be too close to zero – there will always be at least one second between item arrivals. The real-life correlation is that people don't walk through a door at exactly the same time.
- 2) The time it takes a customer to walk from the waiting area to a teller is explicitly represented by a Transport block (labeled Walk to Teller). In many discrete event models, the travel time would not be considered significant or would be just added to the time it takes to perform the transaction. This model takes travel time into consideration, because for 3D animation you want to see the customers moving from the waiting area to the teller.

- 3) There is a large gap between the Queue (Wait) and the Select Item Out (Choose a Free Teller) blocks. This just provides space for inserting some blocks without having to move everything.

### Animate the model in 3D

In the 3D Bank Line Start model:

- ▶ Unselect the command Run > Show 2D Animation. (For maximum performance, you do not want 2D animation running concurrently with 3D animation.)
- ▶ Open the E3D window
- ▶ Verify that the E3D window is associated with this model
- ▶ Run the simulation. Since the E3D window is open, the simulation will animate in 3D.

🔍 As the model runs, you may notice some “interesting” behavior. This is discussed below in “What this model needs”.

### Accommodations for 3D animation

The Bank Line model has already been modified for 3D animation in the following ways:

- 3D animation has been set to run in the Concurrent mode.
- Objects to represent items and blocks in the E3D window have already been selected:
  - Items generated by the Create block (People Arrive) appear as random people
  - The Queue block (Wait) is represented by a door
  - The two Transport blocks (Walk to Teller) are, by default, waypoints
  - The two Activity blocks (Transaction) appear as desks
  - The Create, Select Item Out, and the two Gate blocks have been set as waypoint objects
- Two 3D Scenery blocks (Animation 2D-3D library) provide male and female tellers, with different skins, as scenery objects in the E3D window.
- A 3D Controller block (Animation 2D-3D library) clears items in the E3D window when the simulation ends.

Since these modifications were illustrated in the previous chapter, they are not repeated here. This chapter will instead focus on other ways to enhance the 3D animation.

### What this model needs

There are several odd things that you may have noticed about the customers in this model. For instance, they:

- Line up in the doorway, sometimes on top of each other
- Didn't bring any paperwork to the bank
- Stand on top of the teller's desk while completing their transactions
- Jump, rather than walk, into the bank and from the teller to the exit

The following sections show how to solve these problems.



So that the tutorial models aren't overwritten, you need to have your own model to work with during this tutorial.

**Save a model to explore**

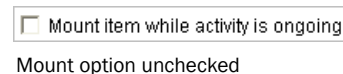
- ▶ Give the command File > Save Model As and save the model as “My 3D Bank Line”

**Unmount the Activity blocks**

By default, Activity blocks are set to have an item’s 3D object mount the block object while the activity is ongoing. While this was useful in the Production Line models of the previous chapter, where crates were being processed by machines, you don’t want customers standing on desks.

In the Item Animation tab of both Activity blocks (labeled Transaction):

- ▶ Uncheck the option to *Mount item while activity is ongoing*
- ▶ Run the simulation with the E3D window open



Now the customers will stand in front of the desks, rather than jump onto them.

- ▶ Save your model. (The equivalent example model is 3D Bank Line 2.)
- ▶ Either leave the E3D window open or close it if you prefer. (If you close it, you will need to open it to perform any 3D operation or to run the animation.)

**Add Transport blocks**

The Production Line model for the previous chapter used Convey Item blocks to represent a moving path from the start to the end of the conveyor. Likewise, the Transport block is useful for displaying the movement of an item from the start to the end of a path.

**Animating the travel time**

To show events as they occur in a model, 2D animation normally displays items moving from one block to another without a time delay. This is the type of 2D animation that is shown when the Run > Add Connection Line Animation command is enabled.

When a model is animated in 3D, the 3D item objects will likewise move from one 3D block object to the next without any time delay. This is seen in the Bank Line models, where customers move instantaneously from the Queue to the Activity blocks and from the Activity blocks to the Exit and can stack one above another. There are two reasons for this:

- Until a teller is available, the people have no place to go as they wait in line. If customers arrive at about the same time, or if the tellers can’t keep pace with the arrival of customers, the Queue block will show its contents (the customers) stacking on top of each other.
- The 2D model does not move people along a path at a specific velocity. The 3D world needs Transport blocks to not only represent how long the journey takes but to also show the people moving along a path during that time.

To have an animation where items don’t jump from one block to another or stack on top of each other as they wait in line, you need to show them moving along a path as they are delayed. This is accomplished by setting a travel time in a Transport block (Item library). This results in an animation delay that corresponds to the time that the item spends in transit from one location to another.

**What the model needs**

For this model, you need to add:

- A path for the waiting line, consisting of one Transport block between the Queue (Wait) and the Select Item Out (Choose a Free Teller) blocks



- Two paths to the exit, each represented by a Transport block between the Activity blocks (Transaction) and the Exit

The tutorial also shows how to minimize the icons of the Transport blocks already in the model.

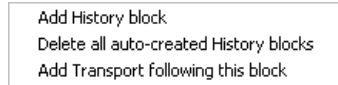
☞ For this part of the tutorial, it is assumed that customers take 5 seconds to reach any destination.

### Walking and waiting in a line

A Transport block can provide a mechanism to show people walking into the bank and lining up instead of stacking on top of each other.

In your My 3D Bank Line model:

- ▶ Right-click the Queue block's item output connector.
- ▶ Choose **Add Transport following this block**



Right-click the Queue's item output connector

This inserts a Transport block between the Queue and the Select Item Out blocks. By default, the block is inserted with its icon view set to "Left-to-right (small)".

- ▶ In the Transport's Behavior tab:
  - ▶ Choose **Travel time: move time** (the default)
  - ▶ Enter **Move time: 5 seconds**
  - ▶ Label the block **Walk in Line**

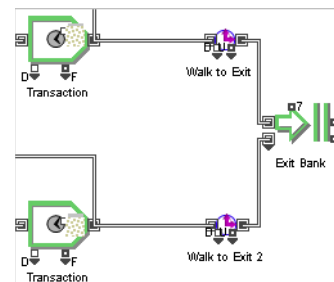
### Leaving the bank

In the current model, customers walk to the teller desk, finish their transactions, and then disappear. You could just leave the model like that, but it would be more fun to see customers walking to the exit. Transport blocks between the Activity blocks and the Exit block will show customers walking out of the bank.

⚠ When there is more than one connection line from the output of a block, you should create a Transport block directly in the model rather than right-clicking to insert it. Otherwise, the Transport block could get inserted onto the wrong connection.

To manually place the Transport blocks:

- ▶ Delete the connection between each Activity (Transaction) and the Exit.
- ▶ Insert a Transport block between each Activity and the Exit.
- ▶ Connect the Transport blocks between the Activity and Exit as shown here.
- ▶ In each Transport block's Behavior tab
  - ▶ Choose **Travel time: move time** (the default)
  - ▶ Enter **Move time: 5 seconds**
  - ▶ Label the Transport blocks **Walk To Exit 1** and **Walk To Exit 2**
  - ▶ At the bottom of the dialog, select **Left to right (small)**, rather than Left to right, for the icon view



Transport blocks for leaving the bank



### Minimizing the icons of the existing Transport blocks

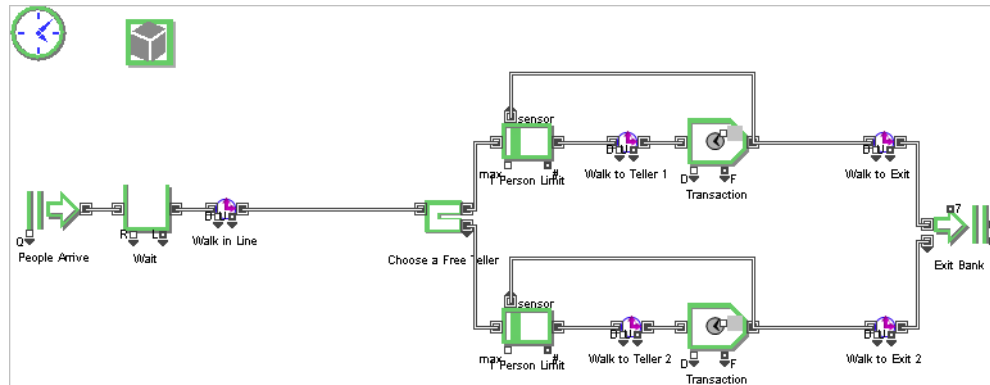
Since the Transport blocks that you have added use small icons, minimize the icons of the Transport blocks that were originally in the model.

- ▶ Either right-click the Transport block labeled “Walk to Teller 1” or select the icon view popup menu at the bottom of the block’s dialog.
- ▶ Select “Left to right (small)”, as shown at the right. This changes the block’s appearance in the model so that it uses a smaller icon.
- ▶ Do the same steps for the Transport block labeled “Walk to Teller 2”.
- ▶ If necessary, move the blocks so that the connection lines are straight.
- ▶ Save your My 3D Bank Line model. (The equivalent example model is 3D Bank Line 3.)



### The model so far

If you have followed this tutorial, your model should look similar to the following screenshot:



3D Bank Line 3 model

- ▶ Run the model again with the E3D window open.

The customers now line up after the entrance and walk, rather than jump, from the teller to the exit. However, they walk at different speeds. This is because the time to get to the end of a path has been set to a constant 5 seconds. But the lengths of the three paths (from the entrance to the head of the waiting line, from the waiting line to the teller, and from the teller to the exit) are all different.

Furthermore, unlike the Convey Item blocks in the Production Line models of the previous chapter, the length of the bank lines is not known in advance. In this case, settings in the Transport blocks can be used to determine the length of the paths. (The problem of customers walking through the desks will be solved later using custom paths.)

### Block positions to determine a path's length

As discussed on page 185, the Transport block has three options for expressing the item’s travel time:

- Move time. This acts just like a delay in an Activity block. As you can see in the current model, the item’s speed and the length of the path are ignored.
- Speed and distance. How fast the item is moving, and how far the item must travel to reach its destination, are entered in the block’s dialog. This is useful if you know the length of the path.



- Speed and calculated distance. The item's speed is entered in block's dialog. The distance is determined from information entered in the frame labeled "Select From and To locations for calculated distance". This is useful when you don't know the length of the path, but you know its starting and ending points. The third option, speed and calculated distance, will provide a more natural walking pace for customers in this model, since it considers not only their speed but how far they have to go.

### Setting the speed and determining the distance

Each of the Transport blocks in this model need to be changed to calculate the length of the path the customers will take.

☞ Rather than the constant move time of 5 seconds in the previous section, this part of the tutorial assumes a natural walking speed of 5 feet per second.

#### Walking to the front of the line

In the dialog of the leftmost Transport block (Walk in Line):

- ▶ Select **Travel time: speed and calculated distance**.
- ▶ Enter **Item speed: 5 feet/second**.
- ▶ In the frame that appears at the bottom of the block's dialog (seen at right), do not change the default settings.

		2D	3D			2D	3D
From X location:		126	6.3	To X location:		317	15.85
From Y location:		156	-7.8	To Y location:		156	-7.8
From location is:		previous non-passing block					
To location is:		next non-passing block					
Calculate distance:		in straight line					
Distance ratio:		use 3D distance ratio					

Frame for calculating the distance items travel

#### The calculated distance

Based on the frame's default settings, this path starts at the Queue (the previous non-passing block) and ends at the Select Item Out (the next non-passing block). The distance is calculated in a straight line and the default distance ratio is used to convert the metrics of the 2D model into coordinates in the E3D window. Since the default settings in the block's Behavior tab are exactly what you want, you do not need to make any changes in the frame that calculates the distance.

See "How the length is calculated" on page 189 for a more detailed explanation of the calculation.

#### Walking to the teller

In the dialog of the two middle Transport blocks (Walk to Teller):

- ▶ Select **Travel time: speed and calculated distance**
- ▶ Enter **Item speed: 5 feet/second**
- ▶ Do not change the default settings in the frame that appears at the bottom of the dialogs.

Notice that the Transport block automatically calculates not only the length of the path but how long (the move time) it will take the customer to reach the end. This information is reported in the block's Behavior tab.

#### Leaving the bank

In the dialog of the two Transport blocks (Walk to Exit) on the right side of the model:

- ▶ Select **Travel time: speed and calculated distance**
- ▶ Enter **Item speed: 5 feet/second**

- ▶ Do not change the default settings in the frame that appears at the bottom of the dialogs.

Notice that, although the customers' walking speed stays the same in every Transport block, each path is determined to be a different length.

**Run the animation and save the model**

- ▶ With the E3D window open, run the simulation

Now the customers should be walking at the same pace from one location to another.

- ▶ Save your model. (The equivalent example model is 3D Bank Line 4.)

**Mounting objects**

You may have noticed that the customers did not bring any paperwork with them. The ExtendSim mounting feature can be used to indicate that customers have some paperwork (for instance a check, withdrawal slip, or loan document) as they approach the teller.

In the previous chapter you saw how item objects (crates) are mounted on Activity block objects (machines) by default. In this chapter, the object must be mounted onto an item object – in other words, paperwork mounted onto the customers.

**Steps for mounting an object**

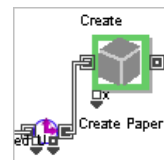
Since the paperwork is not already part of this model, you need to:

- 1) Create a paper object
- 2) Create an attribute to track the object
- 3) Mount the paper with the customers.
- 4) Enclose the blocks in a hierarchical block so that there is minimal disruption to the model

The Animate 3D block (Animation 2D-3D library) is an advanced method for executing an animation action in the E3D window as an item passes through the block. It is useful for creating and mounting objects.

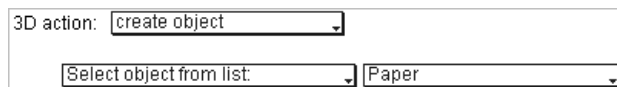
**Create the object**

- ▶ If it isn't already open, open the E3D window
- ▶ Delete the connection between the Transport (Walk in Line) and Select Item Out (Select a Free Teller) blocks
- ▶ Place an Animate 3D block (Animation 2D-3D library) above and to the right of the Transport block



- ▶ Connect the Transport block's output connector to the input connector of the Animate 3D block as shown here. (For now, do not connect the output of the Animate 3D block.)

- ▶ In the Animate 3D block's Item Animation tab:
  - ▶ Select **3D action: create object** (this is the default option)
  - ▶ Choose the option **Select object from list**



Creating a paper object in the Animate 3D block

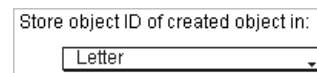
- ▶ In the popup menu that appears, select the **Paper** object
- ▶ Select **Skin 1: Random skin** (the default)
- ▶ Label the block **Create Paper**

### Create an attribute

The next step is to create an attribute so that the objectID of the paper object can be stored and used elsewhere in the model. (This is described in “Adding attributes to a model” on page 117.)

In the Item Animation tab of the Animate 3D block (Create Letter):

- ▶ Select **Store objectID of created object in: New value attribute**
- ▶ Name the new value attribute **Letter**
- ▶ Close the block’s dialog



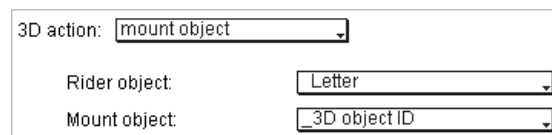
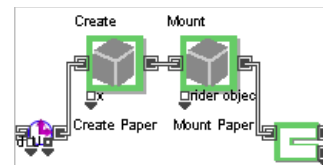
Creating an attribute

As discussed on page 463, every object has an objectID. This is a unique value that the E3D environment uses to identify the object.

### Mount the object on the item

The next step is to mount the paper object onto the items.

- ▶ Place another Animate 3D block to the right of the first one.
- ▶ Connect the blocks together so that the output of the first Animate 3D block is connected to the input of the second Animate 3D block, and its output is connected to the input of the Select Item In block.
- ▶ In the block’s Item Animation tab:
  - ▶ Select **3D action: mount object**



Mounting a paper object using the Animate 3D block

For objects that represent items, the value of their ObjectID is stored in their `_3D objectID` property. This allows you to perform actions on item objects by referencing that property.

- ▶ Save the model
- ▶ Run the animation

As customers reach the front of the line, they bring out their paperwork. However, there are two unnecessary block objects in the E3D window.

### Create a hierarchical block

The final step in this section is to enclose the Animate 3D blocks into a hierarchical block.

- ▶ Frame-select the two Animate 3D blocks
- ▶ Give the command Model > Make Selection Hierarchical
- ▶ Name the new hierarchical block anything you want and click OK to close that dialog
- ▶ In the dialog of the hierarchical block, label the block “Mount Paperwork”



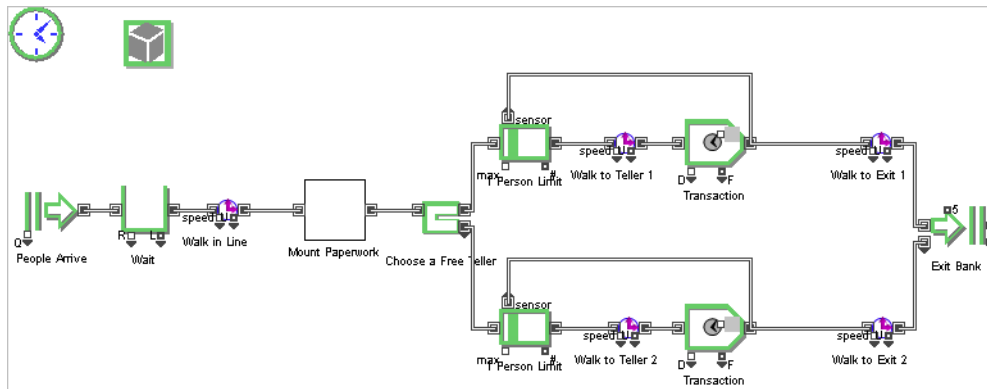


- ▶ Reposition the model's connection lines as desired
- ▶ Save the model. (The equivalent example model is 3D Bank Line 5.)

In the E3D window, notice that the Animate 3D blocks are no longer represented by objects.

☞ By default, Animate 3D blocks appear in the E3D window as a block object. However, hierarchical blocks and the blocks within them normally have no representation in the E3D window. See “Hierarchical blocks and 3D animation” on page 474 for how to change this.

Your model should now look similar to this:



3D Bank Line 5 model

⚠ The next two sections involve the positioning of objects in the E3D window. It is a good idea to not make changes to the layout of the 2D model after positioning objects using the E3D Editor. Otherwise, you may need to reposition those objects.

### Unlinking objects from blocks

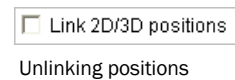
Customers now enter the bank, wait in line, and walk from the front of the line to the first available teller. However, the waiting line is pretty short and the walk to a teller is rather long. To lengthen the waiting line, you could do either of the following:

- 1) Move the Select Item Out block (Choose a Free Teller) closer to the Transport block (Walk to Teller). This would make a messy model.
- 2) Unlink the 2D position of the Select Item Out block from its 3D object, then just move the object. This will be shown below.

☞ Although mentioned in this chapter, the E3D Editor is discussed in detail in the next chapter.

### Unlinking positions

- ▶ In the Block Animation tab of the Select Item Out block, unselect the option **Link 2D/3D positions**, as shown at right.
- ▶ Make the E3D window the active window
  - ▶ Enable the World Editor by pressing F11



- ▶ In the animation area, select the waypoint object labeled “Select Item Out3”. When the object is selected, information about it is displayed in the panes to the right of the animation area. The object’s label is highlighted in the Tree pane at the top right of the window and its properties are listed in the bottom right Inspector pane, shown here.

Apply	Name: Select Item Out3
Transform	
position	20.55 -8.375 100.1
rotation	1 0 0 -7.01958e-15
scale	1 1 1

In the Editor, each object in the E3D window is labeled with a name ending with a number that corresponds to the block’s global block number. The global block number for the Select Item Out block is 3. To see a block’s global block number, open the block’s dialog or use the command Model > Show Block Numbers.

- ▶ Drag the object to the right so that it is located closer to the teller desks. You can select the object and drag it or use the Gizmo to move it. (The Gizmo will be discussed on page 435.) This should put its position on the X axis at about 20 (the screenshot above shows the position as 20.55). Be sure to only drag the object in the direction of the X axis!
- ▶ Press F11 again to return to the E3D window
- ▶ Save the model. (The equivalent tutorial model is 3D Bank Line 6.) Since the object represents a block in the 2D model, saving the model also saves the position of the unlinked 3D object.

### Creating custom pathways

Predefined custom paths allow you to mark each intermediate step along a predetermined route. They provide more flexibility than just choosing to have item objects move in a straight line or along connections and are a convenient way to have item objects move around block objects in the E3D window.

The E3D Editor provides control over every aspect of the E3D window. In its World Editor Creator (WEC) mode, it can be used to create custom pathways.

Although it is mentioned in this chapter, the E3D Editor is discussed in detail in the next chapter. Custom paths are described more fully in “Paths and markers” on page 467.

#### Use the correct Transport behavior

Before creating the paths, you need to change the setting in two of the Transport blocks. If *Travel time: speed and calculated distance* is selected for the Transport block’s Behavior tab (as it is in this model), the following factors are used to calculate the transportation time:

- Speed
- The starting and ending points *in the 2D model*
- The distance ratio
- Whether the Transport calculates distance along connection lines or in a straight line

If you instead select a custom path, the calculated travel time may not be appropriate. This is because the 2D distance is unlikely to be the same as the distance for the custom path.

When using a custom path, you should first change the setting in the Transport block’s Behavior tab to *Travel time: speed and distance*. (If you don’t do this, ExtendSim will give you a message and make the change for you.)

In the Behavior tab of the two Transport blocks labeled Walk to Teller 1 and Walk to Teller 2:

- ▶ Select Travel time: speed and distance

## Creating paths

In the next sections you will:


- 1) Create a new environment file
- 2) Create a path object
- 3) Create marker objects and positioning them on the path
- 4) Select the custom path in the Transport block

### Create a new environment file

An environment file holds information that is not saved with the model, such as static objects, paths, and terrain changes. The default environment file, `Extend3D.mis`, cannot be changed. When you modify the E3D environment by creating custom paths or changing the terrain, you must either create a new environment file or save the default environment file under a new name.

- ▶ If it isn't already open, open the E3D window
- ▶ Be sure your "My 3D Bank Line" model is the associated model (is displayed as the active model in the popup in the upper left hand corner of the E3D window)
- ▶ Press F11 to go to the E3D Editor
  - ▶ Give the command File > Save Environment As
  - ▶ Save the environment file as "My 3D Bank Line.mis" (the default option). It will be saved in the same directory or folder as your model is located.
- ▶ Press F11 to leave the E3D Editor and return to the E3D window
- ▶ Save your model.

"My 3D Bank Line.mis" will now be listed as the environment file in the Run > Simulation Setup > 3D Animation tab.

 When you create a new environment file, the model must also be saved. This ensures that the model will be associated with the proper environment file.

### Create a path object

In your "My 3D Bank Line" model:

- ▶ If it isn't already open, open the E3D window
- ▶ Press F11 to go to the E3D Editor
- ▶ Press F4 to select the World Editor Creator (WEC) mode
- ▶ In the WEC Creator Tree (the *lower* pane on the right):
  - ▶ Open the *Mission Objects* category
  - ▶ Open the *Mission* sub-category
  - ▶ Click the *Path* object
- ▶ In the dialog that appears, name the path "ExitTeller1" and click OK to close the dialog

The path named `ExitTeller1` will be listed at the bottom of the Tree pane (the upper pane on the right) within the `SimGroup-Mission Group` category.

### Create path markers

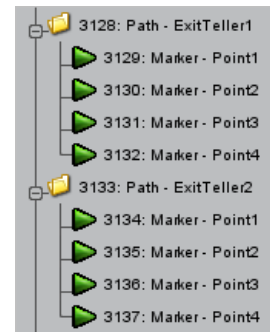
Each path will require four pathmarkers, each of which is created in exactly the same way. Pathmarkers are stored in the path's folder.

#### Create Point1

- ▶ In the Creator Tree pane (the *lower* pane on the right):
  - ▶ Open the *Mission Objects* category
  - ▶ Open the *Mission* sub-category
  - ▶ Click the *PathMarker* object
  - ▶ In the dialog that appears, name the marker "Point1" and click OK to close the dialog. A marker named "Point1" will appear in both the animation area and at the bottom of the Tree pane.

☞ The E3D engine does not allow spaces in the names of markers or objects. If you enter a space in the name, ExtendSim will warn you.

- ▶ Expand the Tree pane (the *upper* pane on the right) so that all the objects are listed
- ▶ At the bottom of the Tree pane:
  - ▶ Click and drag the marker named Point1 until it reaches and selects the path named ExitTeller1. The path's name will be highlighted when it is selected. This will both create a path folder named ExitTeller1 and store Point1 within that folder, as shown on the right.



Path folders with markers

#### Create the remaining markers

- ▶ Create three more markers by repeating the steps used to create Point 1
- ▶ Name the path markers Point2, Point3, and Point4
- ▶ Store the new markers in the ExitTeller1 path folder

#### Position the markers

- ▶ In the animation area of the E3D window, drag Point1 so that it is in front of the desk for the topmost teller. This is the start of the path.
- ▶ Position the other three markers in the E3D window so that they will cause customers to go around the desk and walk up to the exit. The marker labeled Point4 is the end of the path.

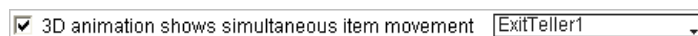
☞ For accurate placement, zoom the camera towards the desk so you can see both the front of the desk and the exit door.

- ▶ Give the command File > Save Environment As, to save the environment file
- ▶ Press F11 to exit the Editor
- ▶ Leave the E3D window open

#### Select the path

The next step is to use the ExitTeller1 path in the model.

- ▶ In the Transport Animation tab of the Transport block (Walk to Teller 1), select the path ExitTeller1, as shown here.



- ▶ Click the button *Get distance from 3D path length*. This places the length of the path in the Distance field of the block's Behavior tab and ensures that the model behavior will be the same whether or not the E3D window is active.
- ▶ Run the animation. Now when customers get to that teller, they will walk around the desk rather than through it.
- ▶ Save your model

### Repeat the process for another path

- ▶ Repeat the above steps to create an ExitTeller2 path with four markers (Point1, Point2, Point3, and Point4) positioned correctly so customers walk around the desk.
- ▶ Select the path ExitTeller2 in the Transport Animation tab of the Transport block labeled Walk to Teller 2.
- ▶ Click the button *Get distance from 3D path length*.
- ▶ Save the model.
- ▶ Run the animation to make sure both paths work correctly. If customers do not move along the route you want, go to the E3D Editor and move the markers to a better position.

The equivalent tutorial model is 3D Bank Line Final. Notice that the distance using the custom path is much longer than the distance that was calculated. This is because the customers now must walk around the desks to get to the exit, rather than walking through the desks.

 Paths are discussed in more detail in “Creating paths” on page 467.

### Enhancing the model

There are several other E3D features that could enhance this model. For instance, the 3D Bank Line Advanced model, located at \Examples\Tutorials\E3D Animation\Bank Line, shows how the customers at this bank could leave paperwork on the tellers's desks.



# **3D Animation**

## **Environment Files & E3D Editors**

The appearance of the E3D window and how it can be modified

As you saw in “Tutorial II” on page 405, you can add blocks to a 2D model to modify aspects of the E3D window’s animation area – insert scenery or text labels, turn the sun on or off, and so forth. And you can enable environmental aspects such as vehicle trails and shadows by going to the Edit > Options > 3D tab.

The appearance of the animation area can also be modified by using the ExtendSim E3D Editor to alter the model’s environment files, without altering the 2D model.

This chapter discusses environment files and the E3D Editor. It will focus on the E3D window’s animation area and the editors that can be used to add pathways, create or manipulate objects in the E3D window, or modify the terrain.

Additional information about the functioning of the E3D/World Editor can be found by reading the Torque World Editor documentation at [GarageGames.com](http://GarageGames.com).

## Environment files

The *environment file* defines the appearance of the animation area in an E3D window. It provides environmental information – lighting, terrain, and so forth – as well as the location of paths and 3D objects that are not shown in the model worksheet.

A model’s 3D environment is selected by choosing an environment file in the Run > Simulation Setup > 3D Animation tab. The default is the *Extend3D.mis* file that provides the environment you see on opening a default E3D window – a perfectly flat, gridded terrain with sun and clouds.

Environment file:

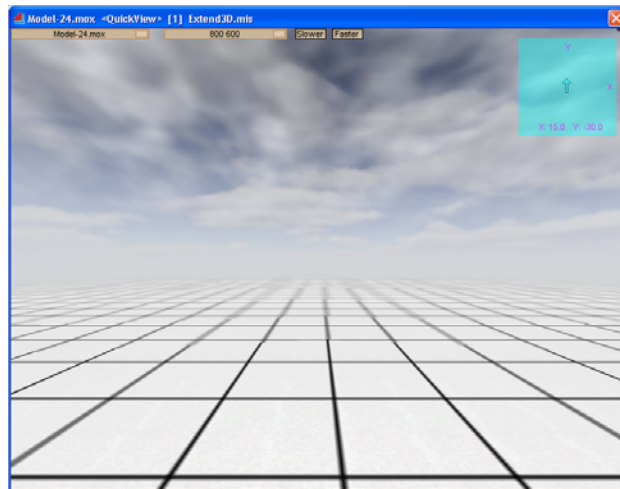
Default environment file in 3D Animation tab

Each 3D model has two animation files: a *.mis* (mission) file that contains information about the pathways and any 3D objects that exist by default in the 3D animation area, and a *.ter* (terrain) file that supplies information about the terrain texture and so forth. The two files will have the same name but a different extension; they are collectively known as the environment file. Thus when you select *Extend3D.mis* as the environment file in the 3D Animation tab of the Simulation Setup dialog, you are also selecting the associated *Extend3D.ter* file.

### Modifying the environment

Defining a custom environment allows you to specify the placement and location of fixed objects (such as desks, workstations, and interior walls) that you want to have appear by default in the E3D window, as well as a custom terrain and any custom paths and waypoints the model might use. This is accomplished by using the E3D Editor, discussed below, to cause changes to the E3D window that do not impact the 2D model. The information about these modifications is stored in the environment file for the model.


By default, environment files are saved at the same location as the model they are associated with. Whenever the E3D window is opened, the environment file for the associated model will be used to populate the E3D window.



Default E3D window’s environment for an empty model worksheet





 The default environment file, `Extend3D.mis`, can be accessed by every model. Only custom environment files need to be saved in the same location as the model they are associated with.

## The E3D Editor

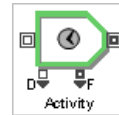
The E3D Editor is an interface to the E3D window that provides extensive and sophisticated control over the contents and appearance of the window. It allows you to alter aspects of the E3D animation area without affecting the appearance or structure of the 2D model. It provides greater flexibility than using blocks from ExtendSim libraries and it allows you to perform tasks you can't do using library blocks, such as:

- Defining custom pathways that are not linked to the position of the 2D model's blocks.
- Placing 3D objects directly in the E3D window, without having to add them as blocks in the 2D model.
- Creating specialized terrains, such as adding contours to the floor, removing its grid, or using a different floor pattern
- To give a very advanced animator complete control over the E3D environment.

### Learning about the E3D Editor

So that you can more clearly see how the E3D Editor works:

- ▶ Place an Activity block (Item library) anywhere on a new model worksheet, as shown on the right.
- ▶ Save the model as "EditorExploring".

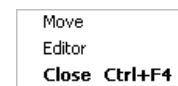


### Accessing the Editor

- ▶ With the EditorExploring model the active window, open the E3D window.

The appearance of this window is defined by the default environment file "Extend3D.mis".

- ▶ Activate the E3D Editor using one of the following methods:
  - ▶ Click the F11 key on your keyboard.
  - ▶ Or, click the ExtendSim application icon at the left side of the E3D window (Windows only). This causes a menu to appear so you can select the Editor.



 If the Editor does not open correctly on a Macintosh, be sure there is no default keyboard shortcut for F11 in the System Preferences > Dashboard & Expose dialog.

### Exiting and closing

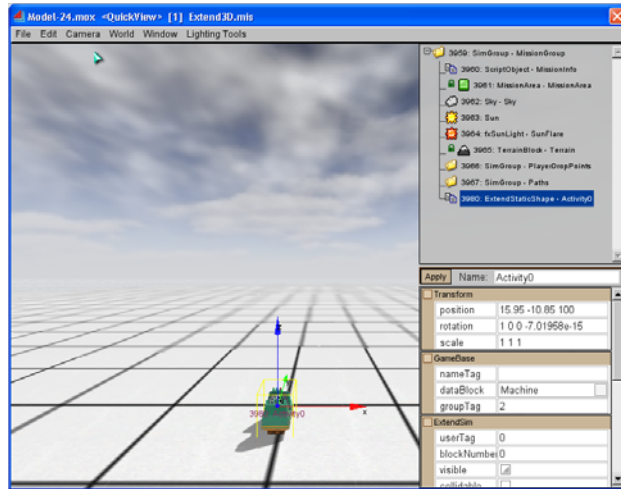
To exit the E3D Editor, use the close box at the top of the window. This also closes the E3D window. To toggle between the E3D Editor and the E3D window without closing either of them, use the Toggle E3D Editor command or the F11 key.



### Exploring the Editor

When you invoke the E3D Editor, it causes the E3D window to change and reveal:

- Commands in a menu bar across the top of the window. These are described starting on page 484.
- Two panes on the right side of the window. In the WEI mode, the top pane is the *Tree* that lists all the objects in the window and the components of the environment (such as sky and sun); the bottom pane is the *Inspector* that will display information about a selected object. (These panes are discussed in more detail in “WEI panes” on page 437.)
- A numbered object in the animation area. This machine is the default 3D object that represents the Activity block in your EditorExploring model. An object’s number corresponds to its listing in the Tree pane.



E3D Editor in WEI mode for EditorExploring model

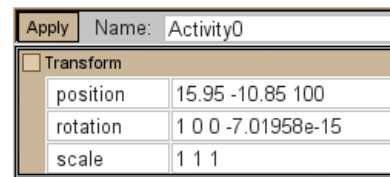
The E3D Editor appears by default in its World Editor Inspector (WEI) mode. Editor modes allow you to alter aspects of the E3D window without affecting the appearance or structure of the 2D model. They are discussed starting on page 435.

### Inspecting objects

The WEI mode is useful for inspecting objects and modifying their properties. It allows you to both see and set the position, rotation, and scale values of objects, as well as other object properties. The list of properties in the Inspector pane varies by object.

- ▶ To inspect an object, click it once in the window’s animation area or select it in the Tree pane.

When an object is selected, the contents of the Inspector pane change to show the object’s properties. The object’s name appears in the pane’s Name field; if the object is the representation of a block, the object’s name will be followed by the block number. Some properties of the Activity block’s 3D representation are shown in the screenshot to the right.



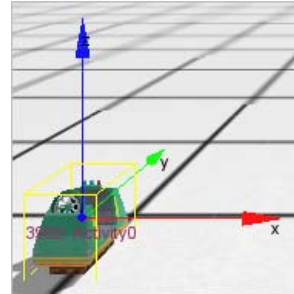
Activity block’s 3D object properties

### The Gizmo

Once you have selected an object in the window, it will appear with a three-axis (XYZ) device drawn through it. This device (called a *Gizmo*) is a tool for translation (movement), rotation, or scaling of the selected object.

Each axis arm has a different colored line: red (X), green (Y), and blue (Z). If you hover the mouse over one of the axis arms, it highlights in a yellow color. The object can then be moved in that direction.

Using the Gizmo to move and perform other tasks on objects is described in the chapter “3D Objects” that starts on page 441.



Gizmo for Activity default 3D object

- ☞ The axes of the Gizmo are expressed in terms of the object, not necessarily the E3D world. If an object is placed in the E3D window in its default orientation, the axes will be pointing in the same direction as the compass for the E3D window. However, if you rotate the object by 90 degrees, for example, the Gizmo’s axes will be 90 degrees out of sync with the E3D window’s compass.

### Modifying object properties

You can change a selected object’s properties by editing values the block’s dialog (if the object corresponds to a block in the model), changing property values in the Inspector pane, or by using ModL code. If you make a change to one of the properties in the Inspector pane, you must click the Apply button at the upper left corner of the pane to apply the change to the object. You can also alter some properties (an object’s position, rotation, or scale) by physically manipulating the object using the Gizmo. Changing object properties is described in the chapter “3D Objects” starting on page 441.

- ☞ If the 2D block position is linked to the 3D object position, changing the position of the object in the E3D window will change the position of the block in the 2D model.

## E3D Editor modes

The E3D Editor has several *editor modes* that allow you to alter different aspects of the E3D animation area. You can select a mode by either choosing it in the Editor’s Window menu (shown at right) or by pressing the equivalent function key.

If you select a different mode in the Window menu, the panes on the right of the E3D window change and sometimes disappear. These changes are discussed where the respective modes are described.

World Editor	F2
✓ World Editor Inspector	F3
World Editor Creator	F4
Terrain Editor	F6
Terrain Texture Painter	


Window menu: Editor modes



### Mode categories

There are three categories of modes for the E3D Editor:

- 1) World. The three World modes (World Editor, World Editor Inspector, and World Editor Creator) are used for inspecting and manipulating objects. They are discussed in “World modes” on page 436.
- 2) Terrain. The two Terrain modes (Terrain Editor and Terrain Texture Painter) are useful for modifying the contours and textures of the terrain. They described in “Terrain modes” on page 438.
- 3) Other. The Editor has some other modes (Mission Area Editor, Terrain Terraform Editor, Terrain Texture Editor, and GUI Editor) which, by default, are disabled and not shown in the E3D Editor’s Window menu. These advanced modes, along with some advanced menu commands (such as Import Terraform data) can be enabled by making a change to the EditorGUI.cs file located in the folder \Creator\Editor. These modes are beyond the scope of this documentation.

 The disabled modes and menu commands should be enabled only by very advanced animators and should be used with extreme caution.

### World modes

As described in detail below, there are three World modes:

- World Editor
- World Editor Inspector
- World Editor Creator

Each of the World modes allows you to manipulate an object's properties – its position, rotation, scale, and so forth. By default, the Editor appears in World Editor Inspector (WEI) mode, as seen earlier. The World Editor Creator mode is most often used to create an object in the E3D window so that it becomes a default part of a model's environment.

 Using the World modes to create an object or modify object properties is discussed in the “3D Objects” chapter that starts on page 441.

### World Editor

This base mode is intended for moving, scaling, or rotating objects. These operations can also be performed in the other World editor modes, but this mode gives you the maximum viewing area.

### World Editor Inspector

The World Editor Inspector (WEI) is the default mode you saw earlier. It allows you to inspect any objects in the E3D window and modify their properties.

#### *Accessing the World Editor Inspector mode*

To access the WEI:

- ▶ Open the E3D window
- ▶ Enable the E3D Editor (F11)
- ▶ Then either give the Window > World Editor Inspector command in the E3D Editor's menu or press the F3 key

*WEI panes*

The WEI has two panes that appear on the right hand side:

- The top pane contains the Tree. This is an explorer-type list of the 3D objects that are present in the E3D window, including environmental components such as the sky and sun. The location of an object's listing in the tree can be moved; that change will not have an effect on the object's position in the E3D window. An expanded Tree for a model that contains only an Activity block (Item library) is shown on the right.
- The lower pane contains the Inspector. This pane shows the values of properties of the selected object – its name and block number (if any), position, rotation, and scale, as well as other object properties. The particular information shown depends on the object selected. In the screenshot on the right, the Activity block has been selected in the animation area. It is also selected in the Tree's list and its name and properties are listed in the Inspector pane.

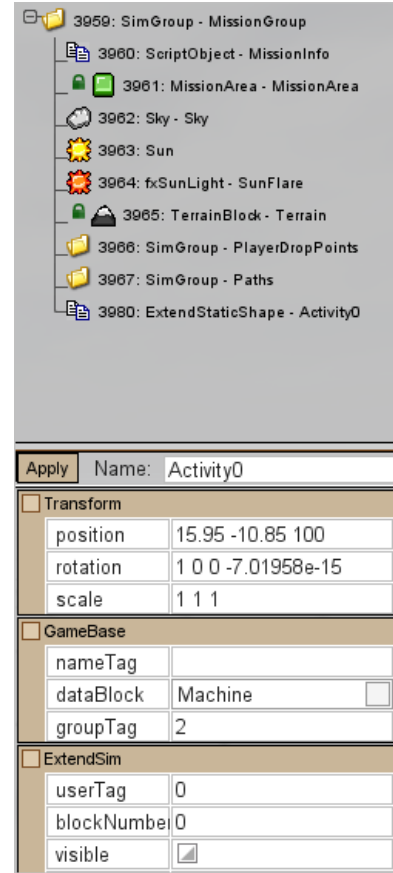
**World Editor Creator**

This mode is used for creating 3D objects in the E3D window and establishing custom pathways, among other tasks. (Causing an object to appear directly in the E3D window so that it becomes part of the environment is known as “creating” the object.)

*Accessing the World Editor Creator mode*

To access the WEC:

- ▶ Open the E3D window
- ▶ Enable the E3D Editor (F11)
- ▶ Either give the command Window > World Editor Creator in the E3D Editor's menu or press the F4 key



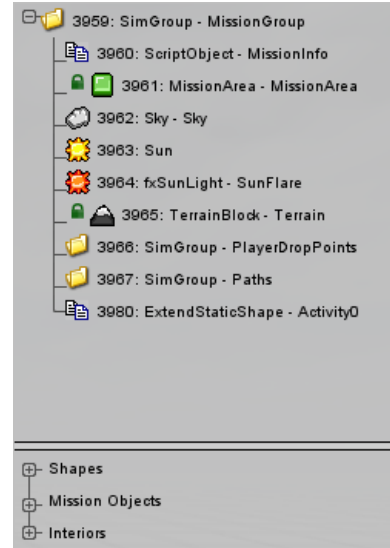
World Editor Inspector panes



### WEC panes

The WEC has two panes that appear on the right hand side:

- The top pane contains the Tree, as shown to the right. This is the same as the Tree in the WEI pane, described earlier.
- The lower pane contains the Creator Tree that contains a list of all the types of objects that are located in the ExtendSim 3D folders. The objects are divided into three categories:
  - Shapes. This category contains most of the functional objects you might want to add to the animation area: Vehicles, Scenery, Blocks, ExtendItems, and Person. Some shape objects have functionality (movement or behavior) and some are static.
  - Mission Objects. This category includes three sub-categories: Environment objects (clouds, skybox, sun, and so forth), Mission objects (Paths, PathMarkers, and so forth), and System objects (SimGroups). You will not usually need to add Environment objects since the default environment already contains the most-needed ones. Mission objects are quite important and useful – see Paths and Markers “Creating paths” on page 467 for additional information. You will not need System objects unless you script the E3D window.
  - Interiors. These are buildings and other types of structures.



World Editor Creator panes

### Terrain modes

As described in detail below, there are two Terrain modes:

- Terrain Editor
- Terrain Texture Painter

The Terrain modes are used to modify the contour and appearance (“texture”) of the floor or terrain of the 3D animation area.

☞ For E3D animation, the word *terrain* means the floor of the 3D animation area and the word *texture* means the appearance of the terrain – its color, pattern, gridding, and so forth.

The default environment file (Extend3D.mis) does not make much use of the E3D terrain capabilities. In fact, if you build models using the blocks in the Item library, you should not change the terrain from the default flat floor. This is because the ExtendSim library blocks assume that the terrain will be flat. For certain kinds of custom block models, however, the ability to manipulate the terrain in the E3D window can be quite useful. And you can always change the terrain’s texture without harmful effect.

☞ Using the terrain modes to modify the terrain is discussed in “Terrains” on page 470.

#### Terrain Editor

This mode is used for filling and excavating the terrain to add contours such as hills and valleys.

#### Accessing the Terrain Editor mode

To access the Terrain Editor:

- ▶ Open the E3D window

- ▶ Enable the E3D Editor (F11)
- ▶ Give the command Window > Terrain Editor in the E3D Editor's menu or press the F6 key

 The default terrain in the ExtendSim 3D environment file is at a height of 100 units. The ExtendSim library blocks assume objects will be at this height. Changing the contours of the terrain may cause issues with the standard ExtendSim blocks.

*The E3D window in Terrain Editor mode*

In this mode there are no panes but two new menus:

- Action. The Select and Adjust Selection options in this menu allow you to make a semi-permanent selection of the terrain. The Add Dirt, Excavate, Adjust Height, Flatten, Smooth, and Set Height commands determine what will happen when you click an area and move the mouse. The Paint Material command allows you to use a texture to change a part or all of the floor's appearance; it is used when in Terrain Texture Painter mode, discussed below.
- Brush. This command is for choosing the shape, consistency, and size of the brush when performing actions on terrains.



 See more information about these commands in “E3D Editor menu commands” on page 484.

Action menu

If you move the cursor around in the animation viewing area you will see a selection area appearing under the cursor. This is called the brush and is your primary tool for editing the terrain in the E3D window with the Action commands.

**Terrain Texture Painter**

This mode allows you to add terrain textures and change the E3D window's terrain so that it has a different appearance. When the Editor is in this mode, the Action menu defaults to selecting the Paint Material command, discussed above.

*Accessing the Terrain Texture Painter mode*

To access the Terrain Texture Painter:


- ▶ Open the E3D window
- ▶ Enable the E3D Editor (F11)
- ▶ Give the command Window > Terrain Editor in the E3D Editor's menu



*Texture pane*

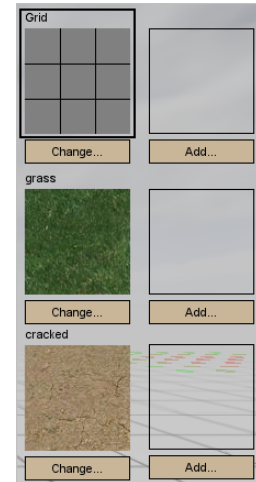
In addition to the Action and Brush menus discussed above, the Terrain Texture Painter has a Texture pane on the right side that has six slots where each slot can display a different terrain texture. Slots can be changed to a different texture (in the case of an existing texture) and different textures can be added (if the slot is blank). Textures are stored in the \Extend3D\data\terrains folder. They are accessed by clicking a texture slot's *Change* or *Add* button.

In the screenshot, the three slots on the left show pre-loaded textures that can be changed. Those on the right are empty slots where a texture can be added.

 No two slots can have the same texture. When changing or adding a texture, the new texture must be different from those already in any slot. Otherwise, the new selection will not have any effect. For instance, there will be no effect if you try to change the “Grid” texture to the “Grass” texture, and there is already a slot with “Grass” as its texture.

**Editor menus and commands**

The menus and commands for the various modes of the E3D Editor are described starting on page 484.



Slots in Texture pane



# **3D Animation**


## **3D Objects**

Creating and manipulating 3D objects

As mentioned in the “Controlling the E3D environment” on page 393, there are several ways to accomplish 3D tasks. With the exception of using an equation-type block or programming with ModL code (described in the Developer Reference), this chapter shows how to use those methods to:

- Create objects for E3D animation
- Use skins to affect the object’s appearance
- Change object properties such as position, rotation, scale, and visibility
- Mount one object on another object
- Use waypoints to mark the positions of invisible 3D objects and as movement destinations
- Access other object information: collision, gravity, ObjectID

This chapter is focused on objects. Paths, which are a specialized type of object, and the terrain, which causes the appearance of the “floor” of the E3D world, are discussed in the next chapter.

 This chapter assumes you have read all the Tutorial chapters for the E3D module and the “Environment Files & E3D Editors” chapter. Example models for this chapter are located in the folder \ExtendSim\Examples\3D Animation.

### 3D objects

A *3D object* is what you see (and sometimes what you don’t see but you see the effect of) in the E3D window. ExtendSim Suite includes a collection of pre-defined 3D objects; you can add new objects to that collection.

An object might appear in the E3D window as the representation of a component of the 2D model or it could exist only in the E3D window. It can represent a block or item from models that use the Item library; it can represent any stationary or moving component of a custom-block model. An object can also appear only in the E3D window, without having a correspondence to a block in the model. This is commonly done when the object is a piece of scenery, a marker for a path, or an environmental effect such as sunlight.



RobotArm object in E3D window

Objects can be created, manipulated, and deleted and they can be visible or hidden. They have properties and can have functionality (movement and behavior). Objects that represent items (or any custom entity that has movement) can be made to follow intricate paths.

Item library blocks, as well as the 3D blocks in the Animation 2D-3D library, have popup menus and checkboxes that allow you to create objects in the E3D window; you can also modify many of an object’s properties in those dialogs. Some environmental effects – shadows, sounds, and footprints – can be turned on or off in the Options dialog. The E3D Editor can be used to create objects such as scenery and paths directly in the E3D window and to manipulate object properties. You can also create custom blocks that have 3D object representations and custom behavior and properties.

### Types of objects

3D objects can be divided into the following categories:

- Interior objects – buildings and other interior structures.

- Shapes – vehicles, scenery, blocks, ExtendItems, and people. Some shapes have movement and behavior while others are static.
- Mission objects – environmental objects such as clouds and sun, as well as paths and markers.

A *waypoint* is a special type of 3D object that just marks a position, but is invisible, in the E3D window. Waypoints are useful for keeping non-essential blocks from appearing as objects in the E3D window or for placing text at specific positions in the window. They are discussed on page 459.

### Object properties

Each object has properties that vary depending on the type of object; all object properties can be changed. Typical properties include:

- Skins. Some objects have just one appearance in the E3D window, but many objects have multiple skin choices that affect how they look.
- Position, scale, and rotation. Objects appear in the E3D window in a default X/Y/Z location with a default size and orientation.
- Collidable. With this property, objects will not occupy the same space.
- Visible. Some objects, such as paths, are by default invisible and you only see their effects. Paths can be made visible and normally visible objects can be made invisible.

### Actions

The following table lists many of the object actions you might want to perform and describes which methods can be used to accomplish those tasks.


3D Action	Item Library	Animation 2D-3D Library	Options Command	Gizmo	Inspector Pane	Creator Tree Pane	Equation Block, ModL Code, or Torque Script	See Page
Create an object for a block or item	X	X					X	444
Create a non-block object						X	X	446
Create an environmental effect		X	X				X	448
Change skin	X	X					X	450
Move an item's object	X	X					X	483
Move a block's object	X	X		X			X	451
Move a non-block object				X	X		X	451
Rotate a block's object	X	X					X	455
Rotate a non-block object				X	X		X	456



3D Action	Item Library	Animation 2D-3D Library	Options Command	Gizmo	Inspector Pane	Creator Tree Pane	Equation Block, ModL Code, or Torque Script	See Page
Scale a block's object	X	X					X	457
Scale a non-block object				X	X		X	458
Keep objects from colliding	X	X			X		X	462
Mount/Unmount		X					X	460
Show/Hide	X	X			X		X	453

For the table:

- Non-block objects and non-item objects are those objects that do not have a corresponding block or item in the model.
- In the Item library, use the block's Item Animation and Block Animation tabs.
- The 3D-enabled blocks in the Animation 2D-3D library are:
  - 3D Controller – establishes and controls aspects of the E3D environment
  - 3D Scenery – creates scenery in the E3D window
  - 3D Text – displays text in the E3D window
  - Animate 3D – performs a 3D action as an item passes through
- You can use the Gizmo in any World mode
- The Inspector pane is enabled when the E3D Editor is in WEI mode
- The Creator Tree pane is enabled when the E3D Editor is in WEC mode
- Equation-type blocks (the Equation block in the Value library and the Equation(I) block in the Item library) can call ModL functions to perform 3D actions. Likewise, you can use ModL code to program custom blocks to perform 3D actions. Torque script is the scripting environment supported by the E3D window. ModL functions and Torque Script are described in the Developer Reference.

 If a 3D object has been created by the 2D model, and/or you modify any of that object's properties, the information about the object's creation and property change is saved when the model is saved. If the object has not been created by the 2D model, its creation and any modifications to its properties can only be saved by creating a new environment file or making and saving changes to an existing one. This is discussed in "Saving changes" on page 459.

## Creating objects

*Creating* an object causes it to be placed in the E3D window. Objects are created:

- To represent blocks from the 2D model
- To represent moveable entities, such as items, from the 2D model
- Directly in the E3D window as scenery, environmental effects, or paths for the 3D animation

Since paths are specialized, they are discussed in the next chapter starting on “Creating paths” on page 467. After an object has been created, you may want to change one or more of its properties, such as its appearance (skins), scale, or rotation. This is discussed in “Changing object properties” on page 449.

- ☞ Creating an object is not the same as defining an object. **Creating** a 3D object places a copy of the object in the E3D window. **Defining** a 3D object is the process of developing a new type of 3D object, as discussed in the Developer Reference.

### Create an object that represents a block

Most 3D-enabled blocks have popup menus that provide choices for indicating which 3D object will represent the block. Blocks that don't have these choices, such as the 3D Controller block (Animation 2D-3D library), don't have a representation in the E3D window.

Objects that represent blocks are created in the block. As discussed below, it is most common to use the Block Animation tab of Item library blocks or a 3D popup menu in the dialog of the 3D Scenery, 3D Text, or Animate 3D block (Animation 2D-3D library).

#### Using an Item library block

The following steps show how to create a 3D object to represent a Create block:

- ▶ Open a blank model worksheet.
- ▶ Save the model as “HowToE3D”.
- ▶ Open the E3D window. (Be sure the HowToE3D model is displayed as the associated model in the E3D window's top left popup menu. If it is not the associated model, select it.)
- ▶ Place a Create block (Item library) on the model worksheet. The block will automatically be represented in the E3D window as its default object, a door.
- ▶ In the Create block's Block Animation tab, use the popup menu (shown to the right) to create other objects to represent this block. As you make each selection, notice how the object changes in the Block Animation tab's preview area and in the E3D window.

Show block in 3D window as: Door

- ☞ If you do not have QuickTime installed, the color of the object in the tab's preview area may not be the same as the color of the object in the E3D window.

If a Skin popup menu appears to the right of the object popup menu, it can be used to change the object's appearance. This, along with other settings in the Block Animation tab, is discussed in “Changing object properties” on page 449.

Skin 1: taupe

So that it can be used for other sections in this chapter and for other chapters:

- ▶ Select the command Run > Show 3D Animation. This will cause the E3D window to open whenever the HowToE3D model opens.
- ▶ Save the model. Since the object has been created by the 2D model, saving the model saves the object information.

- ⚠ Your HowToE3D model will be used for other sections.

### Create an object that represents an item or other moveable entity

Items only occur in discrete event or (sometimes) discrete rate models, but you can create custom blocks to show other types of moving entities. To create an object to represent something that moves:

- Use a block from the Item library (this only creates objects that represent items).

- Use the Animate 3D block from the Animation 2D-3D. (This is an advanced method; see page 483.)
- Use an equation block or program with ModL functions to create an object that represents any moveable entity. This is discussed in the Developer Reference.

### Using an Item library block

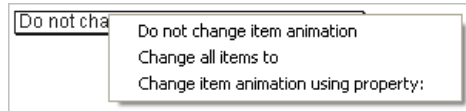
In addition to Block Animation tabs, Item library blocks also have Item Animation tabs. This is used for specifying which 3D object will represent the items that travel through the block.

The process for creating an object to represent items is similar to creating an object to represent a block. (See “Create an object that represents a block”, above.) However, the list of objects in the Item Animation tab’s popup menu is limited to those most appropriate to represent items. (The Animate 3D block in the Animation 2D-3D library provides a complete list of objects to represent items.)

Furthermore, rather than just choosing an object from a popup menu as was done for block animation, there are three ways the object can be created for item animation, as discussed below.

#### Item animation selection methods

Item Animation tabs have three options for creating a 3D object to represent the item (the Create block will have slightly different wording):



- Do not change item animation. With this default choice, the 3D object representation of the item passing through the block will not be changed from what was set in the preceding block. (The default 3D object created by a Create block is a green ball to correspond with the default green circle seen with 2D animation.)
- Change all items to. This choice provides a popup menu so you can choose an object to represent the items in the 3D world. Any item that arrives with some other object representation will be changed to the selected choice.
- Change item animation using property. The object will change depending on the value of the selected item property (attribute, priority, and so forth). This method is most common for creating different objects for the different types of items that pass through, or if you want each 2D animation picture to correspond to a particular 3D object.

For more details about these options, see “Selecting an animation picture” on page 552.

 The Create (Item library) and Animate 3D (Animate 2D-3D) blocks have a 3D object choice in their Item Animation tabs that other blocks don’t offer – *Random person*. If this is selected, the block will randomly generate male and female people objects with random clothing, face, and hands to populate the animation.

### Create a 3D object as scenery

In the previous topics, 3D objects were created to indicate the purpose of a model’s blocks or to represent items or other entities that move in the model. You might want to create an object as scenery instead.

*Scenery* is the term used to describe any object that is displayed in the E3D window but which is not a necessary part of the 2D model. You can select any object to be scenery, but some common choices are bed, chair, desk, door, filing cabinet, machine, shelf, table, tree, and wall. Text, which can be used to label objects or to stand on its own as a label or sign, is also considered scenery.

There are several ways to create objects for scenery:

- With blocks from the Animation 2D-3D library
- Using the E3D Editor to place scenery directly in the E3D environment



- Using an equation block, or program with ModL code as described in the Developer Reference.

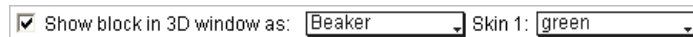
### Using an Animation 2D-3D library block

The 3D Text and 3D Scenery blocks (Animation 2D-3D library) are used to create text and other scenery objects in the E3D window. The following discussion concerns using these blocks to create objects; the blocks are discussed fully in “Animation 2D-3D blocks” on page 482.

- ☞ Using blocks from these libraries requires adding the block to the model but does not require creating a new environment file or making and saving changes to an existing one.

#### 3D Scenery block

A 3D Scenery block in a 2D model will cause scenery to be created in the E3D window. The block’s dialog provides a popup menu for selecting a representative 3D object, similar to the Create block discussed on page 445.



When initially placed in the 2D model, this block has a minimized icon. By default, the footprint of whichever 3D object you choose will be reflected in the model by the block’s icon. If you don’t want this effect, uncheck *Show 3D footprint in 2D model* in the block’s dialog.

#### 3D Text block

The 3D Text block works similar to the Create block discussed on page 445 in that you can select an object to represent it in the E3D window. However, this block also allows you to cause text to be written on the 3D object. If you select the Waypoint object, the text appears at a marked position in the E3D window like a free-standing sign.

To use this block:

- ▶ Select an object in the block’s object popup menu (the Waypoint choice will cause only the text to appear in the E3D window)
- ▶ Enter the text in the text field
- ▶ Select a color for the text
- ▶ Click the **Update** button to cause the text to appear in the E3D window

For instance, this block was used to label the Passport Control sections of the Airline Security model discussed on page 401.

### Using the E3D Editor to create scenery

Creating an object using the E3D Editor causes the object to be placed directly in the E3D environment without having to place a block on the model worksheet. The object could be a visible object (such as a piece of scenery) or an invisible object (such as a path or waypoint). Since the object has no corresponding block in the model, it is considered scenery.

While it is simpler to create scenery by using a 3D Scenery or 3D Text block, the advantage of using the Editor is that the scenery will be in the E3D window every time the environment file is opened. This is especially helpful for different models that require the same scenery.

This section assumes you have read the chapter “Environment Files & E3D Editors”.

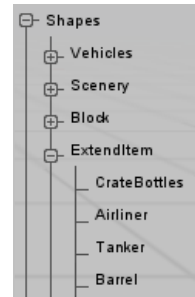
- ☞ Using the E3D Editor to create an object requires creating a new environment file or making and saving changes to an existing one. But it avoids having to place an extra block in the model.

*Accessing the E3D Editor*

- ▶ Open the HowToE3D model that you created on page 445.  
The E3D window for this model will automatically open.
- ▶ Access the E3D Editor by pressing F11
- ▶ Select the WEC mode by pressing F4. (Or choose the command Window > World Editor Creator within the E3D window.)

*Creating the scenery object*


- ▶ In the Creator Tree (the lower pane on the right):
  - ▶ Open the *Shapes* category.
  - ▶ Open the *ExtendItem* sub-category. (You can also select objects from other sub-categories, but most scenery is at this location.)
  - ▶ Click the *Barrel* object. Depending on the resolution, you may need to zoom back to see the object. You can also use the Gizmo to move the object, if it is not placed where you want.



Selecting an object

This will create a Barrel object in the E3D window's animation area without putting a block in the model worksheet.

When you create an object using the Editor, it will be selected by default. Clicking elsewhere in the E3D window unselects the object.

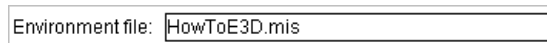
 The default drop setting in the E3D window is World > Drop at Screen Center. With this setting ExtendSim will attempt to place the new item at the center of the camera view, where the terrain is intersected. If a ray from the center of the camera view doesn't intersect the terrain, the object will default to being dropped right on the camera's position. Thus if the newly created object isn't immediately visible on your screen, or if it appears too large, back the camera up with the W key or the down arrow.

*To save the environment file and the model*

So that the barrel will be part of the 3D environment whenever the E3D window is opened for this model, you must create a new environment file or save the changes to an existing one:

- ▶ Save the environment file. With the E3D Editor as the active window, give the command File > Save Environment As and save the new environment. You can name the file any name you want, for example "HowToE3ED.mis". The environment file will be saved in the same directory or folder as you saved the model.
- ▶ Save the model. With the HowToE3D model window as the active window, save the model. This will also save the name of the new environment file with the model.

Now whenever you open the HowToE3D model, the E3D window will open using your environment file with a barrel object. The associated environment file is listed in the Run > Simulation Setup > 3D Animation tab, as shown here.



 For more information about saving changes, see "Saving changes" on page 459.

**Create an environmental effect**

Environment objects are things like clouds, the skybox, and the sun. While these objects are not obviously apparent in the E3D window, you can see their effects. To cause or disable an environmental effect:

- Select options in the Edit > Options > 3D tab



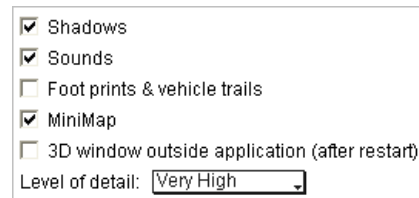


- Use a 3D Controller block from the Animation 2D-3D library
- Use an equation block, program with ModL code, or use Torque script and the World Editor Creator, as described in the Developer Reference

### Options dialog

In the Edit > Options > 3D tab, some environmental options can be turned on and off:

- Shadows
- Sounds
- Footprints and vehicle trails



Environmental choices in 3D tab

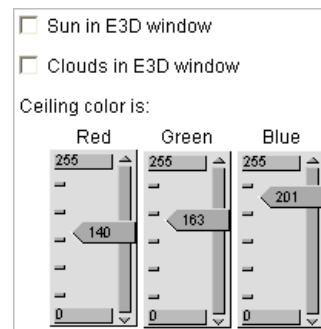
These settings are system-wide and affect the E3D windows for all models.

### 3D Controller block

This block has many uses, as fully described in “3D Controller block” on page 482.

It is also useful for turning off the sun or cloud effects in the E3D window or changing the color of the ceiling/sky. These options are shown at right.

When this block is placed in a model, unchecking the checkboxes disables the sky and/or clouds for the model’s environment file. If the option for the clouds object is not checked, you can modify the color of the ceiling using the Slider controls to achieve the desired effect.



Environmental choices in 3D Controller

These settings are saved with any model that includes the 3D Controller block.

## Deleting objects

The mechanism used to delete objects from the E3D window depends on how the objects were created.

- For objects that correspond to blocks in the model, delete the block from the 2D model. This will automatically delete the object that corresponds to that block. (If you instead delete the object using the WEC as described below, the object will reappear when the E3D window is reloaded and you may get error messages if you then try to delete the block from the model.)
- For an object that exists in the E3D environment but is not a direct representation of a 2D block, delete it in the WEC.

To do this, select the object in the WEC animation area and either:

- Press the backspace or delete key
- Or choose the command World > Delete Selection
- Or choose the command Edit > Cut

The objects that represent items are automatically deleted when the model is re-run or reloaded.


## Changing object properties

Once you have created a 3D object, you might want to change its properties. As was discussed in “Object properties” on page 443, each object has properties that vary based on the object. The following section

describes the most common object properties and tells when you can change them and how to do it. It covers changing an object's:

- Skin
- Position
- Visibility
- Rotation
- Scale

A list of all the potential object properties is beyond the scope of this document. However, the following sections describe using the most important object properties; the Developer Reference has information about others.

 If a 3D object has been created by the 2D model and you modify any of that object's properties, the information about the property change is saved when the model is saved. If the object has not been created by the 2D model, any modifications to its properties must be saved in the environment file. This is discussed in "Saving changes" on page 459.

### Changing skins

*Skins* is the 3D modeler's term for the texture files that show the surface details of objects in the 3D world. Each object has a skin property that affects the object's appearance in the E3D window. While some objects have just one appearance in the E3D window, many objects have a skin type with multiple choices. For those objects, you can change which skin is showing by:

- Selecting a skin from a popup menu in the dialog of the block that creates or changes the object.
- Using the Animate 3D block from the Animation 2D-3D library. (This is an advanced method; see page 483.)
- Using an equation block or programming with ModL, as discussed in the Developer Reference.

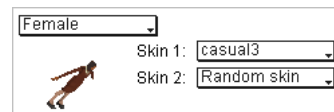
### Skin types


Many of the 3D objects have a single type of skin with multiple choices. For example, desk objects have one skin type that can be base, oak, or old.

 **Base** skin is what is seen in the preview area of a block's Item Animation or Block Animation tab.

The Male and Female objects have two skin types:

- Skin 1 is for the clothing
- Skin 2 is for the face and hands



 In a block's Item Animation tab, objects with a choice for skins also have a **Random skin** option. This causes a randomized display of all the possible skin choices for that object. For male and female objects, both skin 1 and skin 2 can be random.

### Using an Item library block

When you create an object using a block from the Item library, it is common to select the skin for the object at that time. For instance, the Create block you placed in the HowToE3D model on page 445 has a door with no skin options as its default 3D object. If you had instead selected a Crate object in the block's Block Animation tab, you could have selected either a base skin or an old skin. Skins for the objects that represent items are selected in the Item Animation tab of the block that creates or changes those objects.

### Move an object

While a simulation is running, the movement of objects that represent items will be taken care of by the block code and dialog settings. However, you may want to move objects that have been created by the 2D model (such as objects that represent block) or objects that have been created using the E3D Editor.

There are several ways a 3D object can be directly moved:

- If the object represents a block and 2D and 3D positions are linked:
  - Move the corresponding block in the 2D model
- If the object does not represent a block, or if the 2D and 3D positions are not linked:
  - Change X/Y/Z coordinates in the corresponding block's Block Animation tab
  - Move the object in the World editor
  - Change the value of the object's position property in the World editor
- Set the object's position using the Animate 3D block. (This is an advanced method; see page 483.)
- Use an equation block or program with ModL functions (discussed in the Developer Reference)

As you will see below, the method you can use depends on the circumstances.

#### Move a block in the 2D model

By default the Block Animation tabs of Item library blocks are set to *Link 2D/3D positions*. This means that when one of these blocks is moved in the model, the position of its representative object will change in the E3D window.

 Link 2D/3D positions

The object's position in the E3D window is displayed in the 3D position section of the Block Animation tab, shown at right. If 2D/3D linking is enabled, these coordinates are display only and cannot be edited.

	X	Y	Z
3D position:	7.125	-8.88	100

Moving a block is an easy way to manipulate 3D objects but requires that the locations of the blocks in the model be changed. Unlinking 2D/3D positions provides more freedom since you can then move 3D objects around without affecting the 2D model, as shown below.

The positions of the block and the object are both saved when the model is saved.

 This action only applies to objects that represent blocks with 2D/3D linking selected.

#### Change settings in the Block Animation tab

If *Link 2D/3D positions* is unchecked in a block's Block Animation tab, the object can be directly moved by changing the 3D position X/Y/Z coordinates.

 Link 2D/3D positions

For instance, if you change the X coordinate from 7.125 to 9.125 the object would move 2 animation meters (40 pixels) to the right in the E3D window. (The distance is based on the conversion ratios specified in the Run > Simulation Setup > 3D Animation tab, discussed on page 477.)

	X	Y	Z
3D position:	9.125	-8.88	100

This method has the advantage of moving the object's position in the E3D window without moving the block in the model. The position of the object in the E3D window is saved when the model is saved.

 This action only applies to objects that represent blocks with 2D/3D linking not checked.

#### Move the object in the World editor

Objects can also be moved to any position by dragging them using the World editor. After selecting an object, you can move it by:

- Selecting and dragging one of the arms of the Gizmo. This constrains the movement to one axis but does allow movement along the Z axis.
- Dragging it freely in the X and Y directions.

*To access the World editor*

- ▶ Open the E3D window
- ▶ Access the E3D Editor by pressing F11
- ▶ Select any World mode (for instance, select the WEI mode by choosing it in the Window menu or pressing F3)
- ▶ Select the object in the E3D window

*Using the Gizmo*

The Gizmo was introduced on page 435. Among other tasks, it can be used to move a 3D object along one axis.

- ▶ Access the World editor, as shown above.
- ▶ Select and drag the Gizmo's X, Y, or Z arm to move the object along that axis.

☞ If an arm on the Gizmo is selected, it turns yellow.

For example, in the HowToE3D model you saved on page 448, try using the Gizmo to move the barrel or door object.

*Moving the object freely*

Using the World editor, an object can be freely moved in the X and Y directions without using the Gizmo.

- ▶ Access the World editor, as shown above.
- ▶ Without selecting any of the arms of the Gizmo, drag the object to move it where you want. (If the object is very small, zoom in on the object.)

☞ If the object is the representation of a block, and the block's Block Animation tab has been set to Link 2D/3D positions, moving the object will also move the block. Unchecking the linking option allows the 3D object to be moved in the E3D window without affecting the block's position in the model.

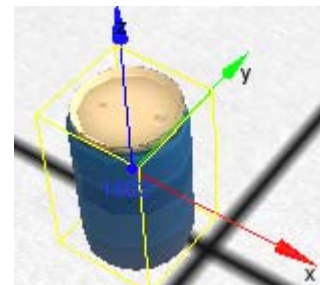
*Save the position*

How the position change is saved depends on the object:

- ▶ If the object has been created by the 2D model, the information about the object's position is saved when the model is saved. (For an object that represents a block, the position information is displayed in the block's Block Animation tab.)
- ▶ If the object has not been created by the 2D model, the environment file must be saved in order to save the object's position. See "Saving changes" on page 459.

**Change object property values**

While it is more common that you would change an object's position by one of the previous methods, a selected object's location can also be modified by changing the value of its position property.



Gizmo for Barrel object

*Access object properties*


Object properties can be changed in the E3D Editor when it is in Inspector mode. The properties of the selected object are listed in the Inspector pane, as shown at right.

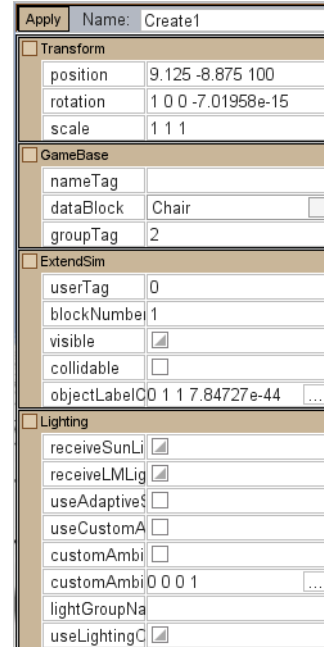
The steps to change an object's properties are:

- ▶ Open the E3D window
- ▶ Access the E3D Editor by pressing F11
- ▶ Select the WEI mode by pressing F3 (Or choose the command Window > World Editor Inspector within the E3D window.)
- ▶ Select the object in the animation area or in the Tree

*Move the object*

The position property is listed in X Y Z format, for instance as "9.125 - 8.875 100". Changing the first position number for that hypothetical object from 9.125 to 11.125 would move the object 200 pixels (40 pixels) to the right. (The distance is based on the conversion ratios specified in the Run > Simulation Setup > 3D Animation tab, discussed on page 477.)

- ▶ In the Inspector pane:
  - ▶ Change the values for the *position* property
  - ▶ Click the *Apply* button 



Try changing the position for the barrel or door object in the HowToE3D model from page 448.

*Save the position*

- ▶ If the object was created by the 2D model, saving the model saves the object's position.
- ▶ If the object was not created by the 2D model, save the environment file to save its position.


This is discussed in "Saving changes" on page 459.

**Show or hide objects**

By default most 3D objects are visible in the E3D window. An obvious exception are paths, which are invisible unless you make them visible. Like other object properties, visibility is an object property that can be manipulated. For instance, you may want to show or hide a piece of scenery in an environment file depending on the model. That would allow one environment file to be used by multiple models with different scenery needs.

When animating in 3D you might want to:

- Hide a normally visible object
- Show a normally invisible object
- Switch between hiding and showing an object based on some condition or situation
- Hide or show all objects of a particular type

 Objects that have been created by blocks should be hidden and shown using dialog options; the model should then be saved to save those changes. Objects that are not associated with blocks can be hidden and shown using the E3D Editor; this is a modification to the environment file and requires that the environment file be saved.

### Hiding an object

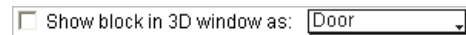
It is common that a block performs a function in the 2D model but you want its representative object hidden in the E3D window. Scenery objects created using the E3D Editor, as well as objects that represent items or other moveable constructs, can also be hidden.

Objects can be hidden by:

- Unchecking a dialog option
- Using a waypoint as a representation
- Using the E3D Editor
- With equation blocks or programming with ModL code, as discussed in the Developer Reference

#### Hide objects that represent blocks

You might want a particular block to not have a visible object representation. If the block is from the Item or Animation 2D-3D library, the easiest way to do this is to uncheck the field *Show block in 3D window as*. In the Item library this is located in the Block Animation tab, as shown above right. This causes the block to have no representation in the E3D window, no matter what object is selected in the object popup menu.



Making an object invisible

Another way to hide an object is to create a waypoint object to represent the block. A waypoint is a special type of 3D object that is invisible and just marks a position in the E3D window; it is discussed on page 459.

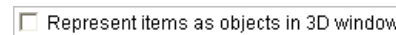


Marking an invisible object

Compared to making the object invisible, the advantage of using a waypoint is that the waypoint's location can be used for a particular purpose. For instance, one object could be made to move toward a waypoint, although the waypoint would be invisible in the E3D window.

#### Objects that represent items

In a discrete event model, items move from one Item library block to another. By default all items have a 3D object representation. By unchecking an option in the Create block's Item Animation tab, as shown here, you can choose to not create objects in the E3D window to represent items. Note that the model must be saved to save this change.

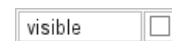


Making the items invisible

#### Scenery objects

A scenery object that has been created in the E3D window, but which does not correspond to a block in the model, can also be hidden for a particular model or for a certain purpose.

The process for doing this is the same as for changing a scenery object's position property, as described in "Change object property values" on page 452. If the scenery object's *visible* property is displayed in the Inspector pane, unselecting that property will cause the object to become invisible. Note that the environment file would need to be saved.



"Visible" property unselected

### Hiding a group of objects

You might want an entire group of blocks to not create objects in the E3D window. This is particularly true for passing or decision blocks like the Set or Select Item Out blocks in the Item library.



*Objects that represent groups of Item library blocks*

Rather than hide each individual object, as described earlier, you can use the 3D Controller block (Animation 2D-3D library) to select a group of objects to be represented as waypoints. This causes the objects to not appear in the E3D window but to instead just have their positions marked.

3D Controller: hiding objects

The 3D Controller block's 3D Options tab has several choices for which category of block should be represented as a waypoint. The categories are based on divisions for the Item library blocks – all blocks (residence, passing, and decision), passing blocks, decision blocks, or only Transport blocks.

Although these options apply mainly to Item library blocks, custom blocks can be designated with those categories and the 3D Controller block will hide the representative objects.

*Objects that represent 3D Scenery blocks*

In the Scenery tab of the 3D Controller (Animation 2D-3D library), you can choose to hide all objects that have been created using the 3D Scenery block (Animation 2D-3D library). This tab also has options for hiding the icon of the 3D Scenery block in the 2D model and/or the footprint of the 3D object.

3D Controller: hiding scenery objects

**Showing an object**

Since most objects are visible by default, the process for making them visible after they have been hidden is the reverse of hiding them (see above).

The path object is one of the few that is invisible by default. Causing the path object to be visible in the E3D window is discussed in “To modify path properties (optional)” on page 469.

**Conditionally showing and hiding**

Some common ways of showing and hiding an object in the E3D window are:

- With the 3D Text block (Animation 2D-3D library).
- Using the Animate 3D block from the Animation 2D-3D. (This is an advanced method; see page 483.)
- With equation blocks or programming with ModL code, as discussed in the Developer Reference.

*3D text block*

The input connector on a 3D Text block (Animation 2D-3D library) can create text in the E3D window if certain conditions occur. To cause the input connector to appear, select *3D object when input connector is true* in the block's dialog. Then connect to the connector to cause the sequence you want. During the simulation, the text will be shown in the E3D window if the value of the input connector is  $\geq 0.5$ ; it will be hidden otherwise.

Dialog option for 3D Text block

**Rotate an object**


Depending on the object, there are several ways a 3D object can be rotated:

- If the object was created by the 2D model:
  - Change rotation settings in the corresponding block's Block Animation tab or in the Create block's Item Animation tab.

Options in Block Animation tab



- Set the object's rotation using the Animate 3D block. (This is an advanced method; see page 483.)
- If the object hasn't been created using the 2D model:
  - Use the World Editor Inspector mode of the E3D Editor to rotate the object with the Gizmo or to change the value of its rotation property
  - Use an equation block or program with ModL functions (discussed in the Developer Reference)

 Scenery and other 3D objects created in the E3D window using the E3D Editor can be rotated using the Editor, but a 3D object created by the 2D model can only be *permanently* rotated by changing block dialog settings or by using the Animate 3D block.

#### **Use the Block Animation or Item Animation tab**

If an object has been created by the 2D model, its rotation can be adjusted in the block's dialog.


In the Create block's Item Animation tab, or in any block's Block Animation tab:

▶ Enter numbers in the rotation field:

▶ Close the block's dialog

The object's rotation value is saved when the model is saved

► For example, using the How To E3D model you created on page 448, try changing the rotation field in the Block Animation tab of the Create block. Then close the block's dialog to see the effect on the door object in the E3D window.

 The rotation value is based on a 360 degree range; blank is the same as 0 or 360. For objects that have a natural front, that front will by default be facing forward (towards the positive Y direction) for the 0 or default rotation setting.

#### **Rotate the object using the World editor**

If an object has been created using the E3D Editor, it can be rotated in the E3D window. To do this, either:

- Rotate the object with the Gizmo in any World mode
- Or, change the value of the object's rotation property in the World Editor Inspector

The object's rotation is saved when the environment file is saved.

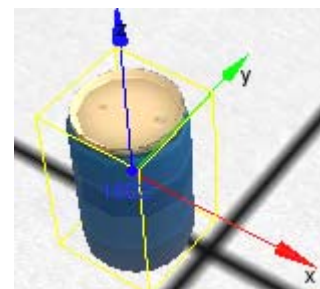
#### *To access the WEI*

- ▶ Open the E3D window
- ▶ Access the E3D Editor by pressing F11
- ▶ Select the WEI (World Editor Inspector) mode by choosing it in the Window menu or pressing F3

#### *Using the Gizmo*

The Gizmo was introduced on page 435. Among other tasks, it can be used to rotate objects.

- ▶ Access the WEI, as shown above.
- ▶ Select the object in the E3D window
- ▶ Press the ALT key (Windows) or the Command (Apple) key (Macintosh) while using the mouse to drag one of the Gizmo's X/Y/Z arms. The arm will turn yellow when it is selected.



Gizmo for Barrel object



- ▶ Save the environment file to save the changes, as discussed in “Saving changes” on page 459.

This rotates the object around the axis associated with the selected arm. Rotation around the Z axis is the standard kind of rotation used in the ExtendSim libraries.

#### Changing rotation property values

The selected object’s rotation can be changed by changing the value of its rotation property. The rotation property is expressed as four numbers – the first three are factors and the fourth is an angle in degrees. The three multipliers determine the amount by which the object is rotated around its axis based on the angle number.


Transform	
position	19.45 -5.9 100
rotation	0 0 -1 90
scale	1 1 1

Properties for the Barrel object

For example, a value of “0 0 1 90.0” means that the object is rotated 90 degrees around the Z axis. A value of “1 0 1 45.0” would mean that the object is rotated 45 degrees around both the X and the Z axes.

The most common form of rotation in the ExtendSim modeling world is around the Z axis, as this will rotate the front of the object on the flat plane.


To rotate the object using its rotation property:

- ▶ Access the WEI, as shown above
- ▶ Select the object in the E3D window
- ▶ In the Inspector pane:
  - ▶ Change the values for the *rotation* property
  - ▶ Click the *Apply* button 
- ▶ Save the environment file to save the changes, as discussed in “Saving changes” on page 459.

### Scale an object

Depending on the object, there are several ways a 3D object can be scaled:

- If the object was created by the 2D model:
  - Change the scale settings in the corresponding block’s Block Animation tab or, for items, in the Create block’s Item Animation tab.
  - Set the object’s scale using the Animate 3D block. (This is an advanced method; see page 483.)
- If the object wasn’t created using the 2D model:
  - Use the World Editor Inspector mode of the E3D Editor to scale the object with the Gizmo or to change the value of its scale property
  - Use an equation block or program with ModL functions (discussed in the Developer Reference)

 Scenery and other 3D objects created in the E3D window using the E3D Editor can be scaled using the Editor, but a 3D object created by the 2D model can only be *permanently* scaled by changing block dialog settings or by using the Animate 3D block.

#### Using the Block Animation or Item Animation tab

If an object has been created by the 2D model, its scale can be adjusted in the block’s dialog.

In any block’s Block Animation tab, or (for items) in the Create block’s Item Animation tab:

- ▶ Enter a number in the scale field:  Scale:



☞ The scale is relative to 1 and a blank is the same as 1. Thus a scale of 2 would cause the object to be twice the default size and a scale of 0.5 would cause it to be half the default size.

- ▶ Close the block's dialog

The object's scale value is saved when the model is saved

☞ For example, use the "How To E3D" model you created on page 448. Change the scale of the Activity block, then close the block's dialog to see the effect on the machine object in the E3D window.

### Scale the object using the World editor

If an object has been created using the E3D Editor, it can be scaled in the E3D window. To do this, either:

- Scale the object with the Gizmo in any World mode
- Or, change the value of the object's scale property in the World Editor Inspector

The object's scale is saved when the environment file is saved.

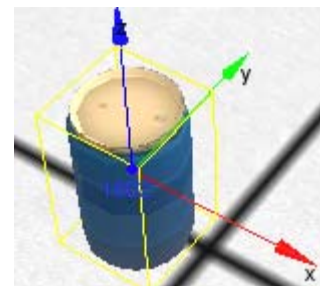
#### To access the WEI

- ▶ Open the E3D window
- ▶ Access the E3D Editor by pressing F11
- ▶ Select the WEI (World Editor Inspector) mode by choosing it in the Window menu or pressing F3

#### Using the Gizmo

The Gizmo was introduced on page 435. Among other tasks, it can be used to scale objects.

- ▶ Access the WEI, as shown above
- ▶ Select the object in the E3D window
- ▶ Press both the ALT key and the CTRL key (Windows) or the Command (Apple) key and Option key (Macintosh) while using the mouse to drag one of the arms of the Gizmo.
- ▶ Save the environment file to save the changes, as discussed in "Saving changes" on page 459.



Gizmo for Barrel object

With this method, the scaling will be in the direction of the selected axis and will distort the shape of the object. To scale an object in all three dimensions at the same time, change the values of the object's scale property in the World Editor Inspector, as described below.

#### Changing scale property values


An object's scale can be changed by changing the *scale* property. This property is displayed in the Inspector pane as three numbers, each representing the object's relative scale along an axis. Units of scale are relative to 1 and associated with each axis in X, Y, Z order. Thus a scale value of "1 1 1" indicates a scale of 1 for each axis.

Transform	
position	19.45 -5.9 100
rotation	0 0 -1 90
scale	1 1 1

Properties for Create block's object

Scaling an object from "1 1 1" to "2 2 2" would make it twice as big along each axis. (This is the same as entering a value of 2 for a block's scale field, as discussed in "Using the Block Animation or Item Animation tab" on page 457.) If you scale an object from "1 1 1" to "1 1 2" it will stretch along the Z axis but remain the same along the X and Y axis. This would cause it to look taller but also a bit distorted. A scale value of "1.1 0.5 1.0" would indicate that the item was scaled to 110% on the X axis, 50% on the Y axis, and 100% on the Z axis.


To access object properties:

- ▶ Access the WEI, as shown above
- ▶ In the Inspector pane:
  - ▶ Change the values for the *scale* property
  - ▶ Click the *Apply* button 
- ▶ Save the environment file to save the changes, as discussed in “Saving changes” on page 459.

## Saving changes

If you create objects or change their properties, you probably want to save those changes. There are two issues you should be aware of when saving changes:

- Objects that have been created by the 2D model, as well as any changes to the properties of those objects, are saved when the model is saved. For instance, if you move an object in the E3D window, and the object is associated with a block in the 2D model, that position change is saved when the model is saved. Furthermore, saving the model saves the name of the associated environment file; if the environment file has been renamed, the new name will be saved with the model.
- If you’ve used the E3D Editor to create paths or objects in the 3D window, or you have made changes to the properties of those objects, you will need to save the environment file to keep your changes. Either create a new environment file or save changes to an existing one, as described below.

 To save the changes made to object properties using the E3D Editor, you must first *Apply* the change, then save the environment file.


### Saving an environment file

With the E3D window as the active window:

- If the model’s current environment file is the default Extend3D.mis file, you will not be permitted to make changes to the environment file. Instead, with the E3D Editor as the active window, give the command File > Save Environment As and save the environment file under a new name.
- For any other environment file, with the E3D Editor as the active window, use the File > Save Environment command to save changes.

You can name the file any name you want. It will be saved in the same directory or folder as the model is located.

If the model uses a new environment file, or a renamed environment file, you must also save the model. This ensures that the model will be associated with the proper environment file.

 You will not be able to save changes to the default Extend3D.mis file; you must use the Save Environment As command instead. Furthermore, the Save commands in the E3D window or E3D Editor window do not affect any changes made to the 2D model.


## WayPoints

A *waypoint* is a special type of 3D object that marks a position but is invisible in the E3D window. Waypoints are useful for keeping non-essential blocks from being represented as visible objects in the E3D window. They are also helpful for marking positions as destinations for object movement in the E3D window. They can be given labels to act as signage for the animation.

### Creating a waypoint

There are several ways to create waypoints:

- Select Waypoint as the 3D object for any Item library block
- Use the 3D Scenery block (Animation 2D-3D library)
- Create a waypoint in the E3D Editor

 Waypoints can never be made visible in the E3D window but their labels will be visible.

### Create a waypoint object in an Item library block

A waypoint object is created the same way any other object is created, as is shown for the Create block on “Using an Item library block” on page 445.

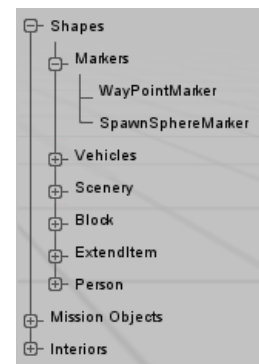
### In the E3D Editor

Accessing the E3D Editor’s World Editor Creator:

- ▶ Access the WEC as was done in “Accessing the E3D Editor” on page 448

### Creating the waypoint object

- ▶ In the Creator Tree (the lower pane on the right):
  - ▶ Open the *Shapes* category
  - ▶ Open the *Markers* sub-category
  - ▶ Click the *WayPointMarker* object



Selecting a waypoint

This will create a waypoint in the E3D window’s animation area. The location will be marked but no object will appear in the E3D window.

### Choosing a waypoint as a destination

If a block is represented by a waypoint in the E3D window, that waypoint will automatically be chosen as the destination for items traveling to the block. To specifically target a waypoint as a destination for item objects that don’t travel through the block, use the Set Waypoint option in the Animate 3D block (Animation 2D-3D library).

## Mounting objects

The process of attaching one 3D object (the “rider”) to a certain point on another 3D object (the “mount”) is known as *mounting*.

Each 3D object has one mounting node and at least one mount point.

- The *mounting node* is the rider object’s location that will be attached to the mount object. For instance, a Suitcase object’s handle is its mounting node.
- A *mount point* is the location on the mount object where the rider object can be attached. In the case of Male and Female mount objects, you would usually attach the handle of the Suitcase to their hands (mount point 1), rather than to their heads (mount point 0).

In many cases, mounting is performed automatically by the blocks. For example, the Activity block will, by default, mount the 3D object that represents the items being processed onto the block’s 3D object.

Furthermore, mounting causes some automatic actions. For instance, mounting a Box object onto a Male object automatically attaches the Box to the mount point on the Male object. The Male object then adjusts its animation to change its hand position, so that it looks as if it is holding the Box. If the Male is then given a destination, the Box will travel along with the Male object.

An object that represents an item can be mounted on an object that represents a block. Likewise, any 3D object can be mounted on the objects that represent items. And while it is not a requirement, an object that has been mounted on another object can also be unmounted.

### Item object on block object

An object that represents an item can be mounted on an object that represents a residence-type block.

- For Activity and Workstation blocks (Item library), the item's object is mounted by default on the block's object for the duration of the activity. This option can be turned off in the block's Item Animation tab. If the option is not checked, the item's object will move up to the block's object but will not mount it.
- Until they are released, objects that represent items automatically mount the object representations of Queue and Resource Item blocks (Item library). This is the blocks' default behavior and cannot be turned off.

In QuickView mode, item objects will mount on top of each other when they reach a residence-type block's object. In Concurrent or Buffered mode, they will either mount or line up and move along a path, depending on the block and various settings in its dialog.

An example of an item object being mounted onto a block object is in the Airline Security model, located at \Examples\3D Animation. In the hierarchical block labeled "Metal Detector", the objects that represent passengers are mounted to the Detector object for the length of time it takes for that part of the process. This is accomplished by choosing in the Activity block to *Mount item while activity is ongoing*.

Mount item while activity is ongoing

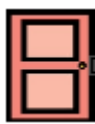
Item objects are not mounted on the Convey Item or Transport blocks. Instead, item objects use the *from* and *to* location information set in the block's dialog. This causes the item object to be displayed as moving along the path represented by the block's object.

### Object on item object

You may want to mount an object on an object that represents an item. For example, to mount a piece of paper on a customer in a bank line or to mount a suitcase on a passenger at an airport.

Mounting an object on an item object is usually accomplished using an Animate 3D or 3D Scenery block (Animation 2D-3D library) and is sometimes done within a hierarchical block to minimize the details at the top level.

An example of an object being mounted onto item objects is in the Airline Security model, located at \Examples\3D Animation.



In the hierarchical block that looks like a door, two Animate 3D blocks create a Suitcase object and then mount it onto the object that represents the person entering the airport. The mounting is accomplished by attaching a rider object (Suitcase)

3D action:	mount object
Rider object:	Suitcase
Mount object:	_3D object ID

onto the mount object, as seen above. The mount object is referenced by the `_3D` objectID property of the items that pass through this block; the value of the property is the value of the object's ObjectID (discussed below). Note that at this point the Suitcase object is not associated with an item or block in the 2D model.

### Scenery object on scenery object

An example of a scenery object being mounted onto another scenery object is in the Airline Security model, located at \Examples\3D Animation. The Equation block (Value library) just to the right of the "Passport control" section is used to mount the person representing the passport inspection officer to a chair object. The officer and the chair objects are both placed in the model through the use of 3D Scenery blocks (Animation 2D-3D library). The 3D Scenery blocks output the value of the `_3D` objectID property of their respective items to the input connectors on the Equation block. Since the

`E3DMountObject(inCon0, inCon1, 0);`

Mounting function with arguments

value of the `_3D objectID` property references the object, this information can be used to mount the passport officer onto the chair.

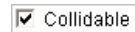
### Other object information

As mentioned earlier, objects have too many properties to fully discuss in this documentation. Some additional information is summarized below.

#### Collision

The `collidable` property defines whether an object that represents an item will respect the physical boundaries of other objects or if it will be allowed to pass through the other objects. If an object's `collidable` property is set to `True`, the object will stop before moving into the physical space of other objects in the E3D window. If `collidable` is set to `False`, the object will be allowed to move through other objects in the E3D window.

By default, the `collidable` property of item objects is set to `true` in the Item Animation tab of the Create block (Item library). The Block Animation tab in other Item library blocks has a checkbox for selecting whether the block object is collidable. The `collidable` property can also be accessed from the WEI.



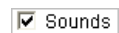
#### Gravity, friction, and momentum

Gravity is implemented in ExtendSim for certain types of objects and not for others. People and vehicles (for instance, the Shapes > Vehicles and Shapes > Person objects of the WEC) will respect gravity. If created at a height, or made to travel to a Z location higher than 100, these objects will drop back down to the ground. Other objects, such as the ExtendItem, Blocks, and Scenery objects in the Shapes category of the WEC, will not be effected by gravity. Instead, they will be able to travel along at a fixed height, or even travel to a different height, by simply setting a Z location higher than 100. In the Boids example, located at \Examples\3D Animation, Boid objects fly through the air by just setting 3D destination locations.

To maintain the constant speeds associated with making travel times match simulation times, the concepts of friction and momentum are not implemented in the objects that travel from location to location in ExtendSim.

#### Sound

The E3D window supports objects playing sounds. Sounds can either be associated with objects in the E3D window or they can be played by ModL function calls.



Use the Edit > Options > 3D tab to globally disable and/or enable sounds in the E3D window; they are disabled by default. If the object has ambient sound, and if *Enable animation of 3D object* has been selected in the block's Block Animation tab (it is enabled by default), the sound will be played when the simulation is run.

The machine object is an example of an ambient sound. To see this, open any model that has an Activity block represented in the E3D window by the machine object, such as the Production Line Final model located at \Examples\Tutorials\E3D Animation\Production Line. Select Sounds in the Edit > Options > 3D tab and run the simulation. While the machine is running, you will hear an active machine sound.

Sounds played in the E3D window are represented as 3 dimensional sounds. If you move the camera in the E3D window closer or further from the machine, or turn to the left or right, the volume and direction of the sound will change accordingly.

### Object ID

An ObjectID is a unique value associated with an object that the E3D environment uses to identify the object. Every object has an Object ID, including GUI objects such as the camera, and objects created by a model's blocks, by the E3D Editor, or by programming.

The value of the ObjectID is the main identifier that the E3D window uses to identify the objects it needs to support. For example the camera object, mentioned in “MiniMap and camera” on page 397, has an objectID. Knowing this allows you to call any of the other ModL functions that take an ObjectID value on the Camera. You could then use this information to dynamically set the position of the camera while an object is moving, so the camera follows along with the moving object.

For objects that represent items, the value of their ObjectID is stored in their *\_3D objectID* property. This allows you to perform actions on item objects by referencing that property.

### BlockNumber

The BlockNumber property is defined in the WEI, but is not available through the dialogs of blocks in the Item Library. This property is used to contain the block number of the block from the 2D model that created the 3D object. It is used internally by the Item library block code and is not something that you will ever want to change. However, viewing it can be useful for informational or debugging purposes.

### GroupTag and UserTag

Like the BlockNumber, the GroupTag and UserTag properties are used internally by the block code of the Item library blocks. They are described in the Developer Reference.





# **3D Animation**

## **Movement, Paths, and Terrains**

Creating paths and modifying terrains

This chapter shows how to:

- Use the Transport and Convey Item blocks to represent item movement and travel time
- Create custom pathways for items and other moveable entities to travel
- Change the terrain

☞ This chapter assumes you have read all the Tutorial chapters for the E3D module and the “Environment Files & E3D Editors” chapter. Example models for this chapter are located in the folder \ExtendSim\Examples\3D Animation.

## Traveling time

In a discrete event model, items move from block to block as dictated by the connections. These connections indicate the direction of movement, but they don't provide any delay for the items. If travel time is significant, it is common to either:

- Increase the delay time of destination blocks to compensate for the travel time
- Specify a minimum wait time in a Queue block's Options tab to simulate travel time

In a logical model, these approaches work well for simulating behavior and result in correct item travel times. When the model is animated in 2D or 3D, however, item/objects visually jump from block/object to block/object. This is rarely what you would want to see.

Instead of implying a travel time, it can be explicitly set in a Convey Item or Transport block (both from the Item library). When the model is then animated, multiple items will move simultaneously at a certain speed from one point to another, rather than jumping from object to object. The Transport and Convey Item blocks are the primary way to show multiple 3D objects moving at the same time in the E3D window.

☞ The Behavior and Options tabs of the Convey Item and Transport blocks are discussed in “Transportation and material handling” on page 185.

## Setting travel time in a Transport or Convey Flow block

### **Simultaneous item movement**

If the *Add connection line animation* command is enabled in the Run menu, 2D animation shows items moving from block to block. To show events as they occur, items do not move simultaneously and the movement is displayed without a time delay.

When a model is animated in 3D using the QuickView mode, the 2D animation is directly translated into 3D animation and 3D item objects will likewise move from one 3D block object to the next without any time delay.

To have an animation where items don't jump from one block to another, you need to show items moving as they are delayed. You do this by setting a travel time in either a Transport or Convey Item block. This results in an animation delay that corresponds to the time that the item spends in transit from one block to another.

There are two methods to add a travel time delay:

- Right-click a block's item output connector and choose “Add Transport following this block”. A Transport block will be inserted between the output connector of that block and the input connector of the next downstream block.
- Add a Convey Item or Transport block (Item library) to the model as you would any other block.

For 2D animation, to view the simultaneous movement of items in the model:

- ▶ Use a Transport or Convey Item block to represent the item's travel time.
- ▶ Check *2D animation shows simultaneous item movement* in the block's Item Animation tab.
 

2D animation shows simultaneous item movement in a straight line
- ▶ Unselect the command Run > Add Connection Line Animation. (This command is not compatible with simultaneous item movement.)

For 3D animation, to see the simultaneous movement of items in the E3D window:

- ▶ Use a Transport or Convey Item block to represent the item's travel time.
- Give the command Run > Simulation Setup > 3D Animating tab and select either the Concurrent or Buffered modes. These modes force the simulation model to maintain a constant ratio between simulation time and external clock time.

In 3D animation, the movement will be either along the connections or in a straight line, as specified in the Convey Item or Transport block's Options tab.

## Creating paths

The purpose of a custom path is to set a pre-defined route for objects to travel through the 3D world. In a 3D animation of a discrete event model, item objects move from one destination to another, so paths are not required. They are, however, frequently created for situations that involve intricate traffic patterns or models that use custom-built blocks.

### Paths and markers

A *path* is a route for an item or other entity that has movement. Each path is a collective object composed of multiple steps, or *markers*, laid out in a predefined order towards a destination. Paths are created using the E3D Editor in World Editor Creator mode and become a part of the E3D environment. They are never visible in the 2D model; by default they are also not visible in the E3D environment.


There are two ways to create paths:

- With the E3D Editor in World Editor Creator (WEC) mode, as shown in this section.
- Using an equation block or programming with ModL functions, as discussed in the Developer Reference.

### Markers

A marker is a point on a path, indicating a step towards the destination. A path can contain as many markers as you want; it can even contain just a single marker.

Markers have their locations specified in X/Y/Z dimensions. Certain objects in the ExtendItem category of objects (such as people and vehicles) will ignore the Z dimension by default, but other objects do not. Because objects in the ExtendItem category are most often used to represent items, and because objects that don't need the Z dimension will ignore it, the Z locations of markers should always be set to the correct value.

 The default height of the ground in the E3D window is 100 meters and the default height (Z dimension) of 3D objects is 100.1.

The steps to creating a path include:

- 1) Create a new environment file, if necessary
- 2) Create a path object in the WEC
- 3) Create marker objects for the path

- 4) Modify path properties in the WEI (optional)
- 5) Save the environment file
- 6) Store the path in the SimGroup - Paths folder (optional)
- 7) Set the movable object on the path
- 8) Save the model



If you create or modify a path, and you want that path available for the model, you must save a new environment file as discussed in “Saving changes” on page 459.

#### **To create an environment file**

Paths are created using the E3D Editor. They do not have a correspondence to the 2D model and must therefore be saved in the environment file.

If the model uses the default environment file, Extend3D.mis, you will need to create a new environment file or save the default file using the File > Save Environment File command. If instead the model uses a custom environment file, you just need to save it each time it is modified.

Full instructions for creating and saving an environment file start on page 459.

#### **To create a path object**

- ▶ Open the HowToE3D model that you modified and saved on page 448

The E3D window for this model will automatically open when the model opens.

- ▶ Access the E3D Editor by pressing F11
- ▶ Select the WEC mode by pressing F4
- ▶ In the WEC Creator Tree (the lower pane on the right):
  - ▶ Open the *Mission Objects* category
  - ▶ Open the *Mission* sub-category
  - ▶ Click the *Path* object
  - ▶ In the dialog that appears, name the path “Path 1” and click OK to close the dialog

#### **To create markers**

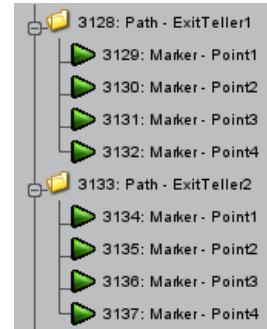
- ▶ In the WEC Creator Tree:
  - ▶ Open the *Mission Objects* category
  - ▶ Open the *Mission* sub-category
  - ▶ Click the *PathMarker* object
  - ▶ In the dialog that appears, name the marker “Marker A” and click OK to close the dialog. Marker A will appear in both the animation area and the Tree pane.
- ▶ Click and drag Marker A within the Tree pane until it reaches and selects the name of the desired path – in this case, Path 1. The path’s name will be highlighted when it is selected.

This will both create a folder named Path 1 and store Marker A within that folder.

- ▶ In the animation area, use the Gizmo to position the marker at the appropriate X/Y/Z location
- ▶ Create additional markers until you have as many as you need for this path, storing them in the Path 1 folder
- ▶ Save the environment file

**To modify path properties (optional)**

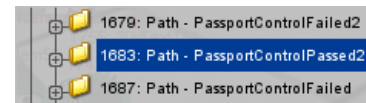
Like other E3D objects, paths have properties that can be modified using the E3D Editor or through ModL functions. A path's properties are most commonly changed to make it a looping path or to cause it to be visible in the E3D window.



Path folders with markers

To change a path's properties, use the WEI:

- Select the path's folder in the Tree pane, as shown at the right.
- In the Inspector pane, find and change the property – looping or visible, as discussed below.
- Click *Apply* to save the property change
- Save the environment file

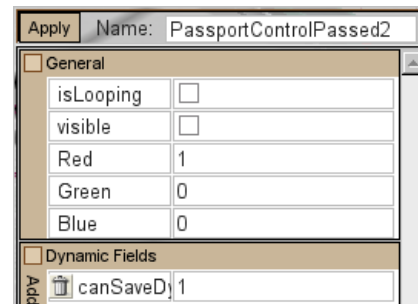


Path selected in Tree

*Looping*

Each path has a property that determines whether or not it is a looping path. If the path is looping, an object on the path will return to the first marker after it reaches the last marker. To cause a path to be looping:

- ▶ Select the path in the WEI Tree
- ▶ In the WEI Inspector pane:
  - ▶ Check the **IsLooping** property checkbox
  - ▶ Click **Apply**
  - ▶ Save the environment file



Path properties: looping and visible

*Visible and colored paths*


By default, paths are not visible in the E3D window (they are never visible in the 2D model). Setting a path to visible will display the path in the E3D window. A visible path can also have a color. To cause a path to be visible:

- ▶ Select the path in the WEI Tree
- ▶ In the WEI Inspector pane:
  - ▶ Check the **visible** property checkbox
  - ▶ Click **Apply**
  - ▶ Save the environment file

Changing the color of a path changes how it displays in the E3D window when it is set to be visible. To cause a visible path to have a color:



- ▶ Select the path in the WEI Tree
- ▶ In the WEI Inspector pane:
  - ▶ Enter a 1 (true) or a 0 (false) for the Red, Green, and/or Blue color properties
  - ▶ Click *Apply*
  - ▶ Save the environment file

 If you create or modify a path in the E3D Editor, and you want that path available for the model, you must save the environment file as discussed in “Saving changes” on page 459.

#### **To store the path (optional)**

Notice that, by default, a path does not have a representation in the animation area, but it is listed in the Tree pane at the top right of the window. To store all paths in one location:

- ▶ In the Tree pane, click and drag the path to within the SimGroup - Paths folder.
- ▶ Save the environment file.

 Paths do not have to be stored in the SimGroup - Paths folder, but it is a good collection point if you have a lot of paths.


#### **To set the item or moveable entity on the path**

There are several ways to set items on a path:

- Using a Transport or Convey Item block (Item library). An example of this, using a Transport block, is shown in “Select the path” on page 428.
- Animate 3D block (This is an advanced method; see page 483.)
- Using an equation block, or program with ModL code as described in the Developer Reference.

## **Terrains**

By default, the Extend3D.mis environment file has only one texture painted on the entire terrain floor – the Grid texture. Furthermore, the default terrain is a flat surface. However, terrains can be customized to have contours as well as multiple textures (color and patterns). As introduced on page 438, terrains are created and modified using the E3D Editor in Terrain mode.

 If there is no texture associate with the terrain, or if the texture cannot be located, the terrain will appear white.

The commands in the Action menu are used in conjunction with the commands in the Brush menu to modify or create custom terrains. As an example, if you select the Add Dirt command, move the cursor to the center of the viewing area, and click the left mouse button, you will be “adding dirt” to the terrain. This will have the effect of creating a small mound under the brushed area. How long you hold down the mouse button effects how much dirt is piled up. Each Action has a different behavior and between them all you have a lot of editing control over the terrain.

To do this, choose the Select command and click the terrain with the left mouse button, selecting the terrain under the brush. The selection area is indicated by squares (nodes) that are drawn at each point where the brush touches the terrain, turning from outlined to solid. Clicking again on other sections of the terrain will add them to the selection. Clicking while holding down the Ctrl key will remove the nodes under the brush from the selection. After selecting whatever portion of the terrain you wish, select the Adjust Selection menu command to raise and lower that selection. When this is selected, moving the mouse forward will raise the selected terrain, and moving the mouse backward will lower it.

 The Action and Brush menu commands are described starting on page 485.

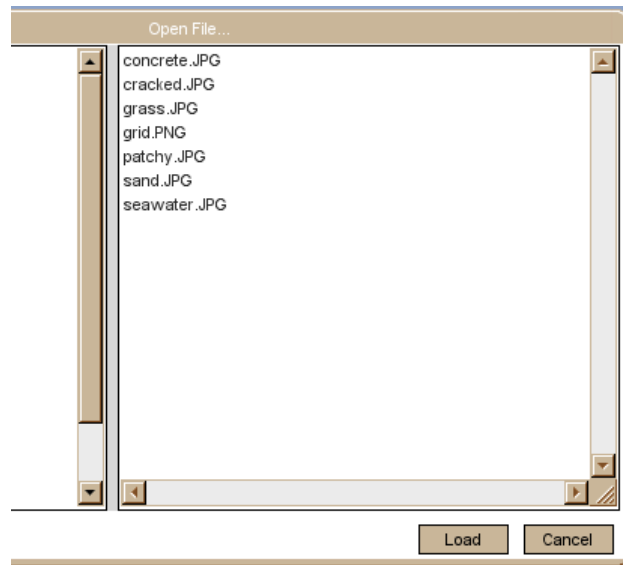
### Modifying the terrain

For example, to modify the terrain or “floor” of the E3D window:


- ▶ With the E3D window the active window, click F11 to access the E3D Editor
- ▶ In the E3D window, choose the command Window > Terrain Texture Painter


On the right side of the window, the Texture pane appears with six slots for selecting textures. (The Texture pane is shown on page 440.) Notice that, by default the Grid texture has been selected. This is the default texture for the default (Extend3D.mis) environment file.

- ▶ Click the **Change** button below the Grid texture pane
- ▶ In the Open File window that appears, expand the **Extend 3D** folder by clicking the + sign.
- ▶ Expand the **data** folder by clicking the + sign
- ▶ Within the data folder, select the **terrains** folder so that the textures appear to the right, as shown in the screenshot to the right.
- ▶ Select the **GridGreen** texture and click Load. The floor will change to a green colored grid.
- ▶ Press F11 again to leave the Editor
- ▶ To save the environment file, see “Saving changes” on page 459.



Textures for terrain

 As with custom paths, changing the terrain is a modification of the environment file. To save terrain changes, the environment file must be saved.

 No two slots can have the same texture. When changing or adding a texture, the new texture must be different from those already in any slot. Otherwise, the new selection will not have any effect. For instance, there will be no effect if you try to change the “Grid” texture to the “Grass” texture, and there is already a slot with “Grass” as its texture.







# **3D Animation**

## **Tips and Reference**

Tips, and explanations of dialogs and commands, for 3D animation

This chapter has some E3D tips and provides reference for the E3D environment. It covers:

- Tips when calling 3D functions from equation blocks
- Performance considerations
- ExtendSim commands and options for 3D animation
- The Animation tabs – Animate Item and Animate Block
- 3D enabled blocks in the Animation 2D-3D library
- Menus in the E3D Editor

 The four controls located along the top of the E3D window are described in “Interface controls” on page 396.

## Tips

### Using an Equation block to call E3D functions

E3D functions, such as E3DCreateObject or E3DPostCreateObject, can be called from an equation type block. This adds tremendous flexibility to your 3D modeling capabilities. However, the E3D animation mode that is currently enabled (QuickView, Concurrent, or Buffered) impacts the functions that you will want to call, especially when referencing 3D objects that represent items. (For a description of the three animation modes, see “3D Animation tab of Simulation Setup dialog” on page 477.)

Most E3D functions have a “post” and “non-post” version. The difference is that the post version of a function includes a time argument that allows the E3D engine to synchronize the real time to the 3D animation time. For example, the two functions E3DCreateObject and E3DPostCreateObject differ only in that E3DPostCreateObject has a time argument. The time argument is nearly always set to “CurrentTime”, the current simulation time.

#### 3D objects that represent items

- If the E3D mode is Concurrent or Buffered, then you should use the post type functions when dealing with 3D objects.
- In QuickView mode, where 3D actions happen immediately without a real-time synchronization, you should use the functions that operate on the 3D item object immediately.

#### 3D objects that represent blocks or scenery

When the 3D objects represent blocks or scenery and the E3D mode is Concurrent or Buffered, use the post functions if you want to change them at a specific point in time. If you are not concerned about time (for example you want to create an object, rotate, or mount an object when the model opens), you can use the non-post functions.

 In general, you should not use the post functions when the simulation is in QuickView mode.

#### Hierarchical blocks and 3D animation

By default, hierarchical blocks and the blocks within them have no representation in the E3D window. To cause a block within a hierarchical block to have an object representation, select an object in the block’s Block Animation tab and also select the option *Link to enclosing H-block*.

#### Items stack on top of each other

If multiple items arrive within the same simulation time unit, they will stack on top of each other. You may not want this behavior unless this is how your system behaves.

The exponential arrival distribution can result in arrival times that are extremely short, causing items to arrive at exactly the same time. To prevent this, enter some small number, such as 1.0, as the Location parameter.

This changes the minimum arrival time to one item per time unit, a more realistic minimum for item arrivals in 3D. You could experiment and make the Location parameter a little less, but if it is too small, item objects could collide and stack when they arrive.

### Performance Considerations

The E3D window is doing a complete textured rendering of a complex 3D environment. The performance of the animation is directly related to the number of objects being rendered and the capability and sophistication of your computer.

#### Suggestions for improving performance

There are several things that you can do to improve performance if the E3D window is feeling sluggish or if the 3D rendering is not able to keep up with the speed of the commands from the ModL code.

- 1) The first thing to consider is the capability of your computer's graphics card. It is critical to have a modern, powerful graphics card in your machine for any 3D graphics application. The 3D rendering done by the E3D window is comparable to the rendering done by 3D gaming software, so buying a card that is recommended for running a modern game is advised.
- 2) The number of objects being rendered in the E3D window at one time is one of the key factors that determine how much work the E3D window is doing. ExtendSim will not impose a limit on the number of objects you can try to put on the screen at one time, but displaying a large number of objects in motion can cause the 3D animation to slow down.
- 3) Shadows can be enabled or disabled from the Options dialog's 3D tab. They are cosmetic and add to the visual impact of the rendering of the 3D world, but are disabled by default since they will increase the load on the 3D graphics card and processor.
- 4) The 3D tab of the Options dialog also has a setting that allows you to choose the level of detail (LOD) of the 3D objects rendered in the 3D window. If you are not having performance issues, you should probably leave this popup menu on the Very High setting. If you are having performance issues you can try turning it down to High or Medium. The Low setting is not recommended unless you are running with very many objects, or on a machine that is not very capable. On the Lo' setting many objects will not look well defined.
- 5) The Item Animation or Block Animation tabs show moving previews of 3D objects. Leaving these tabs open while the animation is running will impact performance.

### E3D commands, options, and settings

ExtendSim has many application-level commands, settings, and options that affect the appearance of the E3D window and the behavior of the animation.

#### Opening the E3D window

Since the commands to display 3D animation are linked to each other, the E3D window automatically opens whenever the associated model opens:

- If the command Run > Show 3D Animation has been checked for that model



- Or, if *Show 3D animation during simulation run* has been checked for that model in the Run > Simulation Setup > 3D Animation tab

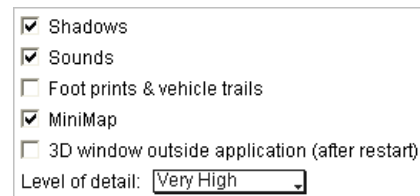
To manually cause the E3D window to open, do one of the following:

- Give the command Window > E3D Window
- Or, click the *Open E3D Window* tool in the toolbar
- Or, click Ctrl+3 (Windows) or command+3 (Macintosh)

☞ A model must be open for the E3D window to open. Also, it may take some time for the E3D window to initialize and open.

### 3D tab in Options dialog

Selecting the command Edit > Options > 3D tab displays the dialog on the right. This tab has several settings that control aspects of the E3D window.



3D tab in Options dialog

☞ These settings apply to every 3D animation, not just the one associated with the currently active model.

- **Shadows.** Checking this option causes objects in the E3D window to cast shadows. By default this option is not selected since calculating shadows is computationally intensive and can slow the animation.
- **Sounds.** Enables objects with sound capability to make a sound. Some of the 3D objects have ambient sounds and, if you program, there are ModL functions associated with playing sounds from the E3D window.
- **Footprints & vehicle trails.** These are cosmetic additions to the displaying of the 3D objects. The footprints are left after the motion of people objects and the vehicle trails after the motion of vehicles such as cars or forklifts.
- **MiniMap.** This option toggles the displaying of the MiniMap on the E3D window; it is checked by default. This map is quite useful for location information when building a model, but may not be necessary for presentation.
- **3D window outside application (Windows only).** By default, the E3D window behaves like a child window of the ExtendSim application. In the default behavior, the window resides within the ExtendSim application window just like any other ExtendSim window. If the option is checked, the E3D window will act like an independent application window and, after restart, will float outside the ExtendSim application window. (This option is not needed for the Macintosh because the E3D window is always outside of the application window.) This option is useful if you have multiple monitors.
- **Level of detail.** This popup controls the level of detail (LOD): Very High, High, Medium, or Low) with which 3D objects are displayed in the E3D window. The choice determines the number of polygons that are drawn for each 3D object; it does not affect the background of the E3D window. In most cases you should just leave this on Very High (the default). You might want to change this option is if there are many objects on the screen and the display of models in the E3D animation is slow.

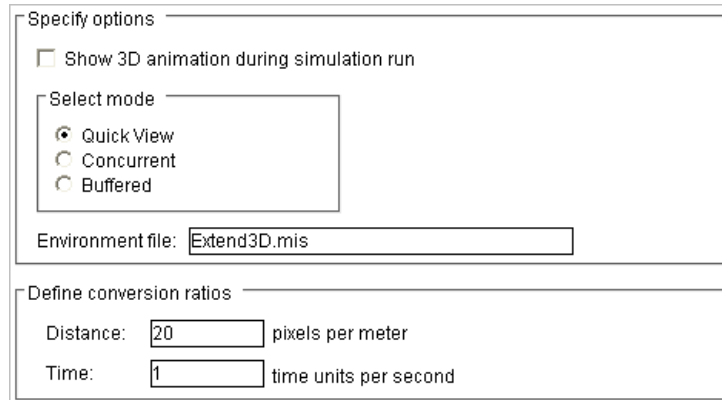
☞ If the LOD is set to one of the higher settings, the detail is automatically adjusted based on the camera's distance from the object. In that case, the object will be displayed with a low LOD if viewed from a far dis-

tance. If the LOD is set to Low, the object will be displayed at a low LOD whether the camera is close to it or not.

### 3D Animation tab of Simulation Setup dialog

Selecting the command Run > Simulation Setup > 3D Animation tab displays the dialog on the right. This tab has several settings that control the behavior of the E3D window.

 These settings apply only to the currently active model.



3D Animation tab in Simulation Setup dialog

- Show 3D animation during simulation run. This option causes the E3D window to open when the model opens. If the E3D window is subsequently closed, it will reopen during model initialization when the simulation is run. If this option is not checked, you will need to open the E3D window manually before a simulation run.

- Select mode. Defines how the E3D window and the simulation interact. There are three modes:
  - QuickView. This default mode allows a simulation model to display its behavior in the E3D window almost “as is”. Most of the model’s blocks will have a representation in the E3D window; a block’s location in the 2D model will determine the location of the corresponding 3D object. Similar to the 2D animation of the simulation model, only one object will move at a time in the E3D window. Furthermore, object movement in the E3D window is not an accurate reflection of timed item movement in the 2D model – there is no direct correlation between simulation time, animation time, and real time.
  - Concurrent. This mode is most commonly used for a model that has been designed to specifically support E3D functionality. In this mode, multiple objects can be moving in the E3D window at one time and the locations of the objects in the E3D window do not have to correspond to the location of the model’s blocks. In addition, the timing of 3D object movement is directly related to the travel time of items in the 2D model – animation time is based on the ratio of simulation time to real time.
  - Buffered. Buffered mode is a variation on concurrent mode where the information about what should happen in the E3D window is stored in an internal buffer and then replayed later. When the simulation model completes running, the buffered information will be complete as well and the 3D representation will be stored. At this point, a “start” button will appear in the E3D window. When the start button is clicked, the 3D visualization of the simulation will begin. You have to wait for the simulation to complete before the animation will begin. But because the model is not running while the E3D window is animating, the performance of the 3D animation might be better than for Concurrent mode.

The selected mode is shown in the title bar of the E3D window.



- Environment file. Selects a file that specifies the appearance and behavior of the background for the E3D window. The default is the “Extend3D.mis” file – an empty, unbounded 3D world with a cloudy sky and a gridded flat floor. This file will also be used if the field is left blank. You can also create custom environment files as described in greater detail in “Saving changes” on page 459.
- Define conversion ratios. These ratios specify the relationship for distance and time between the E3D window and the simulation model. In most cases the default values will not need to be changed.
  - The units of distance in the E3D environment are in meters while distance in the model worksheet can be stated in pixels. The distance ratio defines how many pixels in the 2D worksheet represent one meter in the E3D window. The distance ratio defaults to *Distance: 20 pixels per meter*. For models that use a direct relationship between the two windows, each 20 pixels of distance in the model worksheet will translate to one meter of distance in the E3D environment.
  - The time ratio controls the 3D animation’s display speed for the Concurrent and Buffered modes; the value of the ratio is reported in the E3D window’s title bar. (Since simulation time equals animation time for the QuickView mode, the time ratio is ignored for that mode.) For the Concurrent and Buffered modes, animation time in the E3D window is related to model simulation time by the time ratio. By default the ratio is defined to be *Time: 1 time units per second* (one simulation time unit to one second of real time). This means that if a simulation is set to take 60 time units and the 3D animation is being displayed in Concurrent mode, the display in the E3D window should take 1 minute. The time ratio value, and the animation speed, changes if you click the Faster or Slower buttons on the E3D window.

### Dialog tabs for animation

The main dialog tabs that affect 3D animation are:

- Item Animation
- Block Animation
- Transport Animation

Most of the blocks in the Item library, as well as the Tank and Interchange blocks (Rate library) and the Animate 3D block (Animation 2D-3D library), have Item Animation and Block Animation tabs. The Convey Item and Transport blocks (Item library) also have a Transport Animation tab.

#### Item Animation tab

This tab is for choosing objects to represent items in the E3D window. If a block has an Item Animation tab, the tab has a core set of options that are the same for each block. Some blocks have additional options that are explained below.

##### Core options

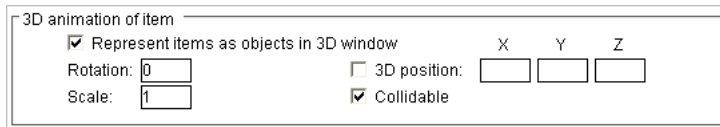
The item options create an object to represent the item as it leaves the block. The options are:

- Do not change item animation. The 3D object is the same for the item leaving the block as when the item entered the block.
- Change all items to. Allows you to select an object to represent each item that leaves the block.
- Change item animation using property. You can choose which object will represent which item based on the item’s properties (attributes, priority, and so forth).

These options are discussed in “Selecting an animation picture” on page 552.

### Create blocks

The Create block has choices for the Item Animation tab that are not present in other Item library blocks:



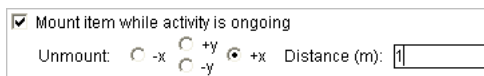
- Represent items as objects in 3D window. When this option is checked, a 3D object will be created whenever this block generates an item. If it is not checked, no object is created to represent the item. (The list of objects to represent items is shorter than the list of all ExtendSim objects; only objects with a group tag of items, players, and wheeled vehicles can be used to represent an item in this block.)
- Rotation. Sets the rotation in degrees of the 3D item object around the Z axis.
- Scale. Sets the scale in the X/Y/Z dimensions of the 3D item object relative to its natural size, which has a scale of 1.
- 3D Position. When checked, changes the initial position of the 3D object to the specified X, Y, and Z coordinates. Leaving a coordinate blank will use the location of the 3D block object as the coordinate for the 3D item object.
- Collidable. When enabled, the 3D objects will collide with each other and not occupy the same 3D area. This is useful for representing items waiting in a queue. If this is not checked, 3D objects can occupy the same 3D space. This option is only relevant to the concurrent and buffered modes.

### Batch blocks

Batch blocks provide two choices on their Item Animation tabs for what should happen to the 3D object when two or more items are batched together:

- *Create new 3D animation object.* A new 3D object will be created to represent the items that have arrived to this block. If *Do not change item animation* is selected in the Item Animation tab (the default setting), the 3D object will be determined by the `_animation` attribute option in the block's Properties tab. This can be overridden by choosing *Change all items to* or *Change item animation using property* to change the animation object in the Item Animation tab.
- *Mount objects (preserve uniqueness required; not available in QuickView).* The original 3D objects coming into the batch block are mounted onto a base object. The base object corresponds to the 3D object associated with the first item to arrive at the connector that has been selected by *The base object is the item from the connector.*

### Activity and Workstation blocks



The Item Animation tabs of Activity and Workstation blocks have these additional 3D options:

- Mount item while activity is ongoing. Mounts the 3D item object on the 3D block object while the activity is running. For instance, this could be used to show parts being processed by the Machine object.
- Unmount [-X, +Y, -Y, +X]. If *Mount item while activity is ongoing* is enabled, this specifies the direction that the mounted object will initially move to when it is unmounted.
- Distance (m). If *Mount item while activity is ongoing* is enabled, this specifies the distance that the mounted object will initially move when it is unmounted. The direction is determined by the *Unmount [-X, +Y, -Y, +X]* option.



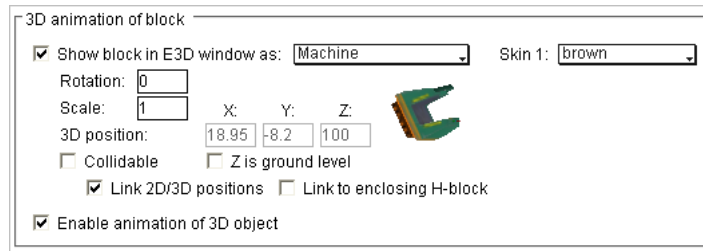
### Block Animation tab

This tab is used to create a 3D object to represent the block in the E3D window.

#### Core options

The following options are common to the Block Animation tabs of Item library blocks.

- **Show block in 3D window as.** When this option is checked, a 3D object will be created to represent the block. If it is not checked, no object is created. The popup menu provides a choice of objects to represent the block. Choosing a waypoint object causes an invisible marker, instead of a visible object, to be created in the E3D window.
- **Rotation.** Sets the rotation in degrees of the 3D item object around the Z axis.
- **Scale.** Sets the scale in the X/Y/Z dimensions of the 3D item object relative to its natural size, which has a scale of 1.
- **3D Position (X, Y, Z).** If Link 2D/3D is enabled, displays the position in the 3D window of the 3D object for this block. If Link 2D/3D if turned off, the fields can be used to set the position of the 3D object in the 3D window.
- **Collidable.** Turns on the collidable flag for the 3D object representing this block. When collidable is enabled, other 3D objects will be prevented from occupying the same 3D space as this object.
- **Z is ground level.** Sets the Z coordinate to whatever ground level has been set to. This is useful if ground level has been changed from the default of 100 meters.
- **Link 2D/3D positions.** Links the locations of the block in the 2D model worksheet with the 3D object representing the block in the 3D window. If the block is moved on the model worksheet, the 3D object will move correspondingly.
- **Link to enclosing H-block.** Links the position of the 3D block to the worksheet position of the hierarchical block containing this block. Generally, you would want to have this checked in only one of the blocks inside of a hierarchical block.



#### For activity type blocks

The Block Animation tabs of Activity, Convey Item, Transport, and Workstation blocks have an additional 3D option:

- **Enable animation of 3D object.** Some 3D objects contain internal animation that shows the status of the object. For instance, the Machine object supports four internal animation states (running, idle, blocked, and down) while the Conveyor supports running.

The Block Animation tab of a Convey Item block has this additional 3D option:

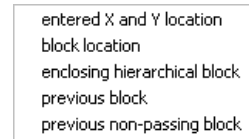
- **Stretch 3D object to conveyor's length.** This will cause the associated 3D object to stretch to the length specified in the block's Behavior tab.



### Transport Animation tab

In addition to their Item Animation and Block Animation tabs, the Convey Item and Transport blocks (Item library) have some 3D options in a tab labeled “Transport Animation”. This tab provides several choices for how the object that represents an item should move.

- 1) From and To locations. The choices for the *from* location are shown at the right; the choices for the *to* location are similar. Although in this tab they are only used to display movement, not to calculate distance, these options have the same meaning as those discussed in “Calculated distance” on page 186.



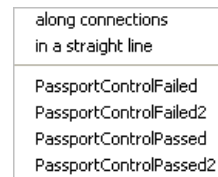
From location options

If either *move time* or *speed and distance* is selected as the travel time in the block’s Behavior tab, you can choose the starting and ending locations for item movement on the Transport Animation tab. If *speed and calculated distance* is selected, the *from* and *to* locations are determined by settings on the block’s Behavior tab; the location options on the Transport Animation tab will not be available.

- 2) 3D animation shows simultaneous item movement. Checking the box allows simultaneous item movement.

In Concurrent and Buffered modes, 3D animation can display multiple objects moving simultaneously. In QuickView mode, only one object moves at a time even if you choose simultaneous item movement in the Transport Animation tab.

- 3) The popup menu to the right of the simultaneous movement checkbox allows you to choose how the movement will be displayed:



Movement options

- Along connections. The objects will follow a path that is equivalent to following connections in the 2D model. The *from* and *to* locations are specified in the block’s Behavior or Transport Animation tabs, depending on the selected travel time option.
- In a straight line (the default) starting at the *from* location and ending at the *to* location.
- Along a pre-defined path. For example, the screenshot above shows four custom “PassportControl” paths. Paths are created in the E3D Editor, as discussed in “Creating paths” on page 467, and the E3D window must be open for the names of available paths to appear in the popup menu. (Paths are invisible by default; to view a custom path in the E3D window, access the E3D Editor.)

If either *move time* or *speed and distance* is selected as the travel time in the block’s Behavior tab, you can choose in the Transport Animation tab how the item should move. If instead *speed and calculated distance* has been selected, item movement *along connections* or *in a straight line* is determined by the settings in the block’s Behavior tab. In this case, you can only choose in the Transport Animation tab if simultaneous movement should be shown or if a custom path should be used.

- 4) Get distance from 3D path length. Calculates the length of the currently selected path and puts that length into the distance parameter field on the block’s Behavior tab.
- 5) Move 3D object immediately to start of movement position (Transport block only). Sometimes it is useful to move the 3D object to a starting point. This is common when the 3D item object is not close to the start of the path or if there is an obstruction between the current location of the 3D object and the *from* location. Note that if this option is used, the object will “jump” in zero time and then begin its timed movement. It will also jump if there is an obstacle (such as another 3D object) in its path.



- 6) Show path. Flashes the path used for a few seconds, when 3D animation shows item movement along connections or on a custom path.

The use of the Transport and Convey Item blocks is discussed in “Transportation and material handling” on page 185.

### Animation 2D-3D blocks

There are four 3D-enabled blocks in the Animation 2D-3D library:

- 3D Controller
- 3D Scenery
- 3D Text
- Animate 3D

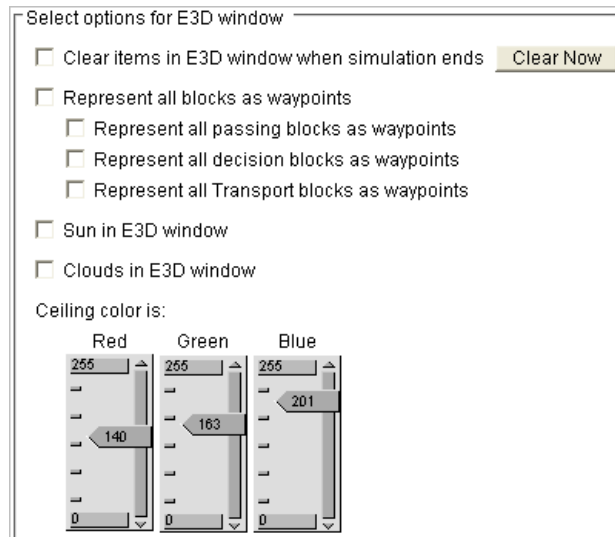
These blocks are discussed in detail in the following sections.

#### 3D Controller block

Use this block to set behavior for the E3D window or to select the 3D options for a range of blocks.

##### 3D Options tab

- Clear items in E3D window when simulation ends. This removes objects that represent items at the end of the run, so that the window looks as it did before the simulation started.
- Represent blocks as waypoints. You can choose to cause all 3D-enabled blocks to appear as waypoints or just a certain type of 3D-enabled block.
- Sun in 3D window, clouds in 3D window, ceiling color. Unchecking the checkboxes causes sun and/or clouds to disappear from the E3D window. If the option for the clouds object is not checked, you can modify the color of the ceiling using the Slider controls.



3D Controller: 3D Options dialog

##### Scenery tab

This tab allows you to:

- Hide all objects created by the 3D Scenery block
- Hide 3D Scenery blocks in the 2D model
- Hide the 3D footprint for all 3D Scenery blocks in the 2D model

The 3D Scenery block is discussed below.



### 3D Scenery block

This block represents scenery and other fixed 3D objects in the E3D window. When initially placed in the 2D model, this block has a minimized icon. By default, the footprint of whichever 3D object you choose will be reflected in the 2D model by the block's icon.

Except for the following additions, the options in this block's dialog are the same as options in Block Animation tabs of Item library blocks, discussed on page 480.

- Show 3D footprint in 2D window. This causes the footprint of the 3D object to appear as an area in the 2D model. This is useful for knowing where to place blocks in the model so that they are well positioned for the 3D window.
- Show 3D objectID. Enables an output connector so other blocks can access the 3D ObjectID. With the ObjectID, you can manipulate the properties of the object in an equation-type block or a block that you build.
- Show. The block's input connector can be used to dynamically hide and show the 3D object, or display the value of the input connector on the object, during the simulation. The choices are to show:
  - 3DObject when input connector is TRUE
  - Value of input connector on 3D object

### 3D Text block

Displays text in the E3D window. The text can be displayed on a visible object or on a waypoint object. If it is displayed on a waypoint, only the text appears in the E3D window.

This block has many of the same choices as for the Block Animation tabs of Item library blocks, discussed on page 480. An additional option is provided by the popup menu shown to the right:

True connector value shows text  
 Append connector value to text  
 Hide connector (text always visible)

3D Text options

- True connector value shows text. Causes the text to be shown or hidden depending on whether the input connector gets a true value.
- Append connector value to text. Causes the value received at the block's input connector to be appended to the text.
- Hide connector (text always visible). This is the default option and just shows the entered text.

### Animate 3D block

The Animate 3D block is an advanced method for executing an animation action in the E3D window as an item passes through the block. Use this block to augment the standard 3D options built into the Item library blocks.

In this block's Item Animation tab, select one of the 3D actions from the popup menu and enter the appropriate parameters. These actions correspond directly to functions that can be called from an equation block or from a custom 3D-enabled block:

- Create object
- Delete object
- Mount object
- Set destination
- Set position
- Set rotation



- Set scale
- Set target
- Set path
- Set skin
- Set waypoint
- Unmount object

The Animate 3D block can be used for almost any 3D action, even one that does not involve the item that passes through the block. For instance, it can be used to mount one static object onto another or move an item's object from one point to another.

☞ Most of the options will only be useful if the Concurrent or Buffered 3D mode is used. In QuickView mode, the 3D objects move without simulation time elapsing, so any action that has a simulation delay (or speed) associated with it will not be processed. Also, in the QuickView mode mounting one moving object on another moving object will not work properly, as only one object moves at a time and objects are deleted and re-created each time they move. Furthermore, because QuickView mode does not have a direct relationship between simulation time and real time, options such as *Set destination* and *Set target* will give unexpected results.

The individual actions are described in the block's Help and many of them are illustrated in the "Animate 3D Block" model located at \Examples\3D Animation\Tips. In that model, Animate 3D blocks are used to create barrels at a fixed location, set intermediate destinations to a random X/Y/Z position, set their final destination to a fixed location, and then delete the objects. In the model, Activity blocks (Item library) are used to delay the items for the amount of time the 3D objects will require to arrive at their destinations.

### E3D Editor menu commands

The E3D Editor, described starting on page 433, has its own set of menus and commands. Depending on the mode selected, the E3D Editor's menus will change somewhat. The first five menus listed below apply to either the World or Terrain modes; the World menu applies only to the World mode and the Action and Brush menus apply only to the Terrain mode.

To access the E3D Editor and either the World or Terrain mode:

- ▶ Open the E3D window
- ▶ Enable the E3D Editor (F11)
- ▶ To access the World editors, give the command Window > World Editor Creator (F4) in the E3D Editor's menu.
- ▶ To access the Terrain editors, give the command Window > Terrain Editor (F6) in the E3D Editor's menu.

☞ For more information about accessing and using the World modes, see "World modes" on page 436; for the Terrain modes see "Terrain modes" on page 438.

#### File

The first four commands in the File menu are associated with creating, opening, and saving environment files. These commands work as you would expect them to.

The Toggle E3D Editor command toggles between the E3D window and the E3D Editor. It is equivalent to pressing the F11 key.

☞ Because the Extend3D.mis file is protected, you cannot make changes to it. Instead, use the *Save Environment As* command to create a new environment file containing the changes you've made using the E3D Editor.

### Edit

This menu has Undo and Copy/Paste commands that work just as you would expect.

### Camera

The Camera menu provides control over the speed of the virtual camera, described in “MiniMap and camera” on page 397. These commands affect the forward, backward, left, and right motions of the camera when pressing the W/A/S/D keys or the keyboard's direction keys.

### Window

The Window menu contains the mode commands, discussed in “E3D Editor modes” on page 435.

### Lighting Tools

This menu has a Light Editor and lighting commands. The Full Relight command is especially useful if the lighting seems confused – it causes lighting sources and shadows to be recalculated.

### World

This menu is specific to the World editor modes. It has commands for hiding and showing the selection in the E3D window, deleting the object, and dropping the object at specific locations in the window.

### Action

Specific to the Terrain modes, this menu allows you to add dirt, excavate, and otherwise adjust the contour of the floor in the animation area of the E3D window. These commands are used in conjunction with the commands in the Brush menu.

- The Select and Adjust Selection commands are for making a semi-permanent selection of the terrain. They work together to allow you to choose a section of the terrain and apply the equivalent of the Adjust Height command to it.
- The Add Dirt, Excavate, Adjust Height, Flatten, Smooth, and Set Height commands determine what will happen when you left click an area and move the mouse to “brush” the area.
- The Paint Material command allows you to change the texture or color of the floor. It is the command used when changing the texture in the Terrain Texture editor.

### Brush

Specific to the Terrain modes, this menu is for choosing the shape, consistency, and size of the brush when performing actions (from the Action menu) on terrains.

There are three selection sections in this menu and each allows one choice.

- Shape. The Box Brush is square and will affect a square grid of nodes on the terrain. The Circle Brush shape is obviously circular. If you switch between these two shapes you should see the shape of the brush changing as soon as you move the cursor onto the terrain.
- Consistency. This can also be seen in the viewing area of the E3D window. Each of the terrain nodes that are going to be affected by the brush action will be drawn with a square on it at the point where the brush touches it. These squares, outlined in the case of the brush and solid in the case of the selection mode described in the Action menu, are drawn in different colors based on how hard the brush is. The hardness determines how much the Action will effect the terrain that the brush touches. If you select a Hard Brush,

all the nodes will be drawn in red and the Action will affect each node equally. If you select a Soft Brush, the central nodes will be drawn in red and the outer nodes will blend toward green. With a Soft Brush, the Action will affect the central nodes (red) more than the outer nodes (green).

- Size. The brush will affect an area the size of the selected command.

 Terrains are discussed on page 470.

# How To

## Libraries and Blocks

A description of ExtendSim libraries and blocks  
and the many ways to use them

*“The first thing to have in a library is a shelf.  
From time to time, this can be decorated with literature.  
But the shelf is the main thing.”  
— Finley Peter Dunne*

Working efficiently with libraries and blocks is critical to simulation modeling. This chapter discusses:

- The libraries available with ExtendSim products
- How to open and use libraries
- Creating and managing libraries for block developers
- Working with blocks, including using variable connectors
- Managing blocks in libraries
- Hierarchical blocks

### The ExtendSim libraries

ExtendSim libraries are repositories for the blocks that you use to build models. ExtendSim ships with several libraries, each one containing blocks that are used for modeling specific types of systems. You can also develop your own libraries of custom blocks, or create libraries to save and organize hierarchical blocks in a way that best works for your modeling needs.

- Libraries that are most often used, such as the Value and Item libraries, are stored at the top level of the ExtendSim7/Libraries folder.
- The Example Libraries folder within the Libraries folder contains libraries created for specific purposes, such as the Tutorial library for block developers
- The libraries in the Legacy folder are included for backwards compatibility, so models created with those older libraries can run. They are no longer supported and should not be used to create new models.

The following describes the libraries that ship with ExtendSim products. (Note that some libraries are only available in specific products.)

#### Animation 2D-3D library

The Animation 2D-3D library lets you add custom animation to models and hierarchical block icons. The Animate Item and Animate Value blocks in this library are used to add 2D animation to models and hierarchical blocks, as described in “Blocks for customized animation” on page 553. The 3D animation blocks are discussed at “Animation 2D-3D blocks” on page 482; they are used with the 3D animation feature of the ExtendSim Suite product.

#### Electronics library

The blocks in the Electronics library are used to simulate system level design of analog, digital, signal processing, and control systems. Electronics blocks are used in the model “Noisy FM system” on page 75.

#### Item library (not available in ExtendSim CP)

The blocks in the Item library are used primarily in discrete event modeling, although they can also be included in discrete rate models. If you add a single block from this library into a continuous model, the model automatically becomes a discrete event model. Discrete event models track individual quantities and entities that can have unique characteristics. The Appendix that starts on “Item Library Blocks” on page 723 has a complete list and brief description of the blocks in this library.



**Plotter library**

The Plotter library holds all the common types of Plotter blocks used to graph and output data for models. Some of these are specific to continuous or discrete event models, while others can be used with any type of model. In addition to the simple plotters you have already seen in the tutorials, ExtendSim plotters can show scatter plots, moving strip charts, histograms, and so on. All plotters are described in detail in “Plotters” on page 588.

**Rate library** (not available in ExtendSim CP or ExtendSim OR)

The Rate library is used for discrete rate simulations, where you model the flow of materials according to some rate-based calculations. Discrete rate models often use blocks from the Value and Item libraries. A complete list and brief description of the blocks in this library starts on page 732.

**Utilities library**

The Utilities library contains a collection of helpful blocks for performing various tasks such as counting the number of blocks in a model, fitting data to a curve, synchronizing the model to real time, timing the duration of a simulation, adding active buttons to models, and so on. A complete list and brief description of the blocks in this library starts on page 735.

**Value library**

The Value library contains blocks that are primarily used for continuous modeling, although they play a critical role in other types of models as well. You used some of these blocks when building the Reservoir model in the tutorial at the beginning of this User Guide. Continuous models represent a smooth flow of values that are recalculated at periodic time steps. A complete list and brief description of the blocks in this library starts on page 716.

**Example Libraries folder**

The following libraries are located in the ExtendSim7/Libraries/Example Libraries folder.

**Custom Blocks library**

These blocks have been created for very specific purposes, such as to illustrate a concept or hard-code certain behavior. For instance, the Planet block from this library is used in the model discussed at “Planet Dance” on page 79.

**ModL Tips library**

This library contains blocks that illustrate the techniques described in the ExtendSim Developer Reference.

**Item Templates library** (not available in ExtendSim CP)

Contains hierarchical discrete event blocks that serve as templates for specific behaviors, such as setting the arrival time based on the time of day.

**Tutorial library**


Contains blocks built in conjunction with tutorials in the Developer Reference.

**Legacy folder**

Several libraries are stored in the \Libraries\Legacy folder. The legacy libraries are furnished for backwards compatibility, so that you can run models built with previous releases of the software. Depending on which ExtendSim product you purchased, the Legacy folder will contain some or all of the following libraries:

- Animation
- BPR
- Discrete Event
- Flow
- Generic
- Items (DB)
- Mfg (Manufacturing)
- Quick Blocks
- SDI Tools

**The legacy libraries (listed above) have been replaced by the Animation 2D-3D, Item, Item Templates, Rate, and Value libraries.**

 The legacy libraries will not be included in future ExtendSim releases. They are no longer supported and are supplied without warranty of any type or for any purpose. Legacy libraries are included in this release so that you can run models built in previous releases; they should not be used to create new models.

## Using libraries

### Opening a library

Whenever you open a model, ExtendSim automatically opens the libraries that are used in the model. You can also manually open a library or instruct ExtendSim to automatically open some libraries when it starts.

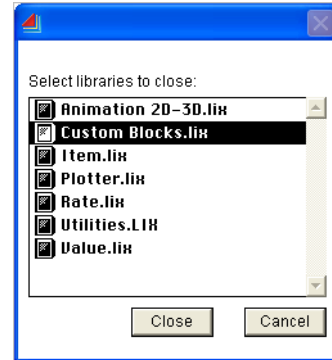
- To open a library manually, for example when you are starting a new model, choose Library > Open Library. You can also double-click the library file or drop-launch it. You open libraries one at a time with these methods.
- To specify that a specific library, or several libraries, open when ExtendSim launches, go to the Edit > Options > Libraries tab and enter the names of the libraries you want pre-loaded. See “Options” on page 688 for more information.

When a library opens, ExtendSim adds the library name, in alphabetical order, to the bottom of the Library menu. Select a library name to see a submenu of the different block categories for that library, which further expand to show the list of blocks for each category.

To view all the blocks in a library alphabetically rather than grouped by category, go to the Edit > Options > Libraries tab and uncheck *List blocks by category*.

### Closing a library

Unless you close ExtendSim, libraries stay open whether they are used in a model or not. You can close libraries that are not used in a model by choosing Library > Close Library and selecting the library you want to close. To close multiple libraries, Shift select the libraries (or use the Ctrl (Windows) or Command key (Mac OS)), then click Close. It is unlikely that you will need to close libraries often since open libraries do not take up much memory and it is usually convenient to leave all your commonly-used libraries open. Once you save the model, the closed but unneeded libraries will not automatically open when you open the model again.



Close Library dialog

ExtendSim will warn you and will not let you close a library that is being used by an open model.

### Searching for libraries and blocks

When you open a model, ExtendSim automatically searches for the blocks the model uses. It first searches for the libraries in which the blocks last resided.

#### Library searches

The search order ExtendSim uses to locate libraries is:

- The *Alternate path*, if any, specified in the Options dialog
- The Libraries subfolder within the ExtendSim folder
- The folder containing the model

You can manually stop this search process at any time by pressing Ctrl+period (Windows) or Command+period (Mac OS). You can also force a manual search process by unchecking *Automatic search* in the Edit > Options > Libraries tab.

If you have renamed, deleted, or moved a library from the expected folder since the model was last saved, manually stopped the search process, or un-checked *Automatic search* in the Options dialog, ExtendSim will not be able to find the library.

**!** Libraries should be kept in the Libraries folder, in the same folder as the model that uses them, or in a folder specified as the *Alternate path* in Edit > Options > Libraries tab. Otherwise, ExtendSim will always ask you for the location of the library.

If ExtendSim can't find a library when the model opens, it will put up a file selection dialog that asks "Where is the library xxx?". Your choices are:

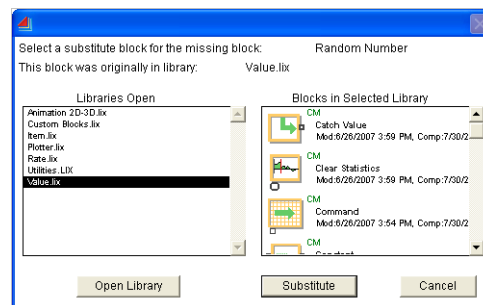
- Cancel the library search operation. ExtendSim will then try to find the missing blocks, as described in "Block searches" below.
- Find and open the library manually. Since ExtendSim only needs to know where the blocks for the model are, you could also use this choice to substitute a library that has a different name (but the same named blocks) as the search library. See "Substituting one library for another" on page 495 for more information.

When the model file is subsequently saved, any new library name and/or location will be saved as well, so searching will not be necessary the next time the model file is loaded. (Although if *Automatic search* is unchecked, ExtendSim will still ask you to locate the libraries.)

### Block searches

As described above, ExtendSim searches for the blocks a model uses by first searching in the libraries that contained those blocks. However, ExtendSim may not be able to find the blocks if you have renamed the blocks or removed them from the library. Once the library search is finished, whether successfully or in situations where the search has been cancelled, ExtendSim prompts you to locate any missing blocks. The dialog offers two choices:

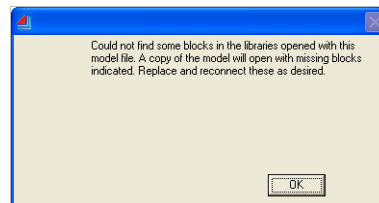
- Choose Open to locate and open another library where the block resides. If ExtendSim is unable to locate the block in this library, the Substitute Block dialog will open which will allow you to substitute an alternative block from any open library. When the model file is subsequently saved, any new block name and/or location will be saved as well, so searching will not be necessary the next time the model file is loaded.



Substitute Block dialog

Any substituted block must be substantially the same as the searched-for block, or ExtendSim will report error messages.

- Select Cancel if the block no longer exists or if you can't substitute another block. In this case, ExtendSim notifies you that it will put text in the place of the block.



Warning message

### Library windows

The first choice in the blocks list for each library listed in the Library menu is Open Library Window. This opens a window that shows all the blocks inside that library in alphabetical order. You can also open a library window using the Navigator, as discussed at “Library Window mode” on page 671.

- ☞ If you enter libraries in the Edit > Options > Libraries tab to be pre-loaded when ExtendSim launches, you can also choose **Open library window** so that the window for the selected library will also automatically open.

The top of the library window gives information about the library version, size, and date last modified. The blocks in the library are listed with pictures of their icons and, if the *Show library window dates* option is selected in the Options dialog, their last modified date.

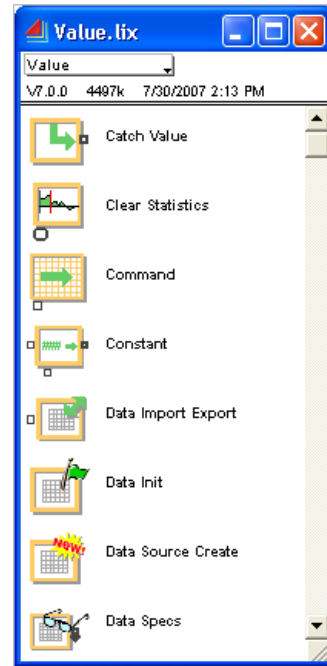
Library windows provide an additional method for adding blocks to models, as shown at “Library Window mode” on page 671. If you program blocks, you can also use library windows to manage blocks, as discussed in “Managing blocks” on page 502, and to access a block's structure as discussed in the Developer Reference.

Note that closing the library window only closes the window, not the library itself.

- ☞ If a block has been compiled with external source code, it will be listed in the library window with the designation *CM* (code management) on the right side of its icon. If a block has been compiled with debugging code, its name and any additional information will be listed in the library window in red and it will show in a model window with a red border around it. Compiling options are used by block developers and are discussed in the Developer Reference.

### Creating and maintaining libraries


You do not need to start a new library unless you build your own blocks or want to save hierarchical blocks in a library.

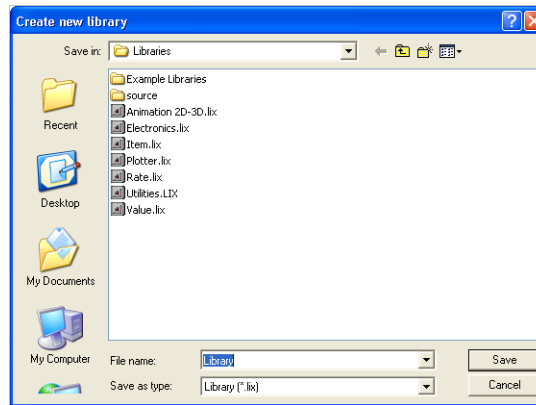


Library window for Value library

### Creating a new library

To start a new library, choose Library > New Library. A dialog appears where you can select a location for the new library and give it a name. Libraries should be saved in the ExtendSim7/Libraries folder so that ExtendSim can quickly find them.

 If you build your own blocks or modify ExtendSim's blocks, you should put them in your own library. Do not put them in the libraries that come with ExtendSim. Otherwise when you install a new version of ExtendSim, you could lose the blocks you built.




New Library dialog

### Saving and compiling libraries

Every time you make a change to a library, such as adding a new block or moving a block from one library to another, ExtendSim automatically saves the library. If you build a block and save it, both the block and the library are saved.


When you create blocks using ExtendSim's ModL programming language, the blocks need to be compiled to machine code so they can be used to build models. If you make a change to an Include file that is used by more than one block, it's usually simplest to recompile the whole library so the blocks incorporate the Include file.

 Unless you program your own blocks or move a library from one operating system to another (for example, if you move a library from Windows to Mac OS) you do not need to compile. The libraries and blocks that are packaged with ExtendSim are already compiled.

To compile a block or a library, use the compile commands in the Develop and Library menus:

- The Develop > Compile Block command compiles the block to machine code without saving or closing the block. This is useful for testing new code for syntax errors as you are building the block. This command is only enabled when the block's structure window is the active window.
- The Library > Tools > Compile Open Library Windows command compiles all libraries whose windows are open and saves the changes. This command is only enabled when at least one library window is the active window.
- The Library > Tools > Compile Selected Blocks command compiles all the blocks that are selected in a library window and saves changes. This command is only enabled when a library window is the active window and one or more blocks are selected in the library window.

Compiling a library or block with debugging information allows you to set breakpoints, watch points, and so forth. Compiling with external source code causes ExtendSim to generate an external file that contains the source code; this is useful for version control.

 Blocks with debugging code show in the library window with their names in red and show in a model window with a red border around their icon. Blocks with external source code will show in the library window with *CM* (code management) next to their icons.

For more information about saving and compiling libraries, see the Developer Reference.

### Substituting one library for another


If you create your own blocks, you might want a model to use the blocks from a library with a different name. For example, if you have developed a newer version of a library and you want the model to use those blocks. Normally ExtendSim opens the library too fast for you to stop it.

In order to bypass this process, you can choose to not have ExtendSim automatically find and open libraries when a model opens. You do this by deselecting the *Automatic search* option in the Edit > Options > Libraries tab. If you do this before opening a model, ExtendSim will stop and request the location of each library the model uses. You can then use the dialog to find and open the libraries you want the model to use. This will work even if the library name is different than the one the model originally used, because what is important to the model is the block name within the library, rather than just the library name.

Once all the required libraries are open, save the model. So that the model will again open the libraries automatically, go to the Edit > Options > Libraries tab and select *Automatic search* before reopening the model.


### Arranging blocks in libraries

If you build your own blocks, you could put all of the blocks you use (up to the 200 block limit) in one huge library. That way, you would never have to try to remember which block was where or remember where your libraries are on your hard drive. However, this arrangement could make it difficult to maintain your blocks.

 Simulations run neither faster nor slower when you group your blocks in a single or multiple libraries. The only performance consideration is that it initially takes more time to open multiple libraries than it does a single library.

There are typically three classes of libraries you might want to create: general-usage, subject-specific, and model-specific.

- General-usage libraries hold blocks that might be used in a wide variety of models.
- Subject-specific libraries are for blocks that are only relevant to one subject, such as paper making.
- Model-specific libraries hold blocks that are only used in a single model.

 Do not add blocks to the libraries that come with ExtendSim or move blocks from those libraries. If you add blocks to an ExtendSim library, your work will be lost when you update. If you move blocks from an ExtendSim library to a library you create, the blocks in your library will not be updated when the ExtendSim library is updated.

### Protecting the code of library blocks

If you build your own blocks, you may not want others to have access to your code. As discussed in the Developer Reference, ExtendSim blocks contain ModL code and can also reference Include files.

You can prevent others from accessing the code in your libraries by 1) not giving them the Include files and 2) removing the ModL code of the blocks. To remove ModL code from your blocks, use the Library > Tools > Protect Library command. This process creates a duplicate of the library, with all the source code removed. Keep the original library in a safe place, since there is no way to recover the ModL source code from the protected library.

☞ To protect a hierarchical block's structure and prevent a user from double-clicking the hierarchical block to see the underlying submodel, use the Model > Lock Model command discussed at “Locking the model” on page 677.

A protected library can be used in the same way as any other library except that the ModL code cannot be altered or viewed. This means that someone using the library has all the functionality of the blocks in that library but no ability to see how the blocks work.

☞ To keep a library from being used to build models, while still allowing it to be used to run simulations, see “Convert Library to RunTime Format” on page 697.

### Converting libraries to RunTime format

As discussed at “The ExtendSim LT-RunTime version” on page 678, the ExtendSim LT-RunTime version is useful for distributing models to those who do not have a full ExtendSim license. To have your custom libraries be available to run models in the LT-RunTime version, but not available to build models, convert them to RunTime format. Converting libraries to RunTime format also prevents them from being used in the full version of ExtendSim. The conversion process is described in “Convert Library to RunTime Format” on page 697.

## Working with blocks

ExtendSim blocks have a user interface (their icon, dialog, and help) and internal, integrated ModL code that determines how the blocks behave.

Blocks are stored in repositories called *libraries*. The entire definition for a block (its program, icon, dialog, and so on) is stored in the library. When you include a block in a model, the block itself is not copied to the model. Instead, a reference to the block is included in and stored with the model. Any data you enter in the block's dialog is also stored within the model.

### Customizing block icons

The blocks in the libraries that come with ExtendSim (such as the Value and Plotter libraries) are displayed on the screen as icons that depict their function. Once you place a copy of these blocks in your model, you might want to make its icon more closely indicate its role in your particular model. However, using programming to change a block's icon in one model will change it in every model using that block, since blocks and their internal descriptions reside in libraries, not in models. In addition, since some blocks are animated, it is not easy to change a block's icon without having to modify its ModL code. For these reasons, *you should not change the icon on the blocks in the libraries that come with ExtendSim.*

If you build your own blocks, you can give them any icon you want. However, there are two better ways to customize blocks in your model without having to directly modify icons:


- You can paste pictures on the worksheet to customize the model. Pictures automatically go behind blocks and text. For example, you can place a map of a region behind your model, or show the layout of a plant. This is described in more detail in “Working with pictures” on page 562.
- You can also add a picture to a block without affecting the original one in the library by making the block hierarchical. This technique is described in more detail in “Modifying hierarchical blocks” on page 548.

### Icon views

As you saw in “Dialogs” on page 16, some blocks have multiple icon views. If a block has icon views, the choice of view will determine how the block looks on the model worksheet.



For example, an icon view could provide:

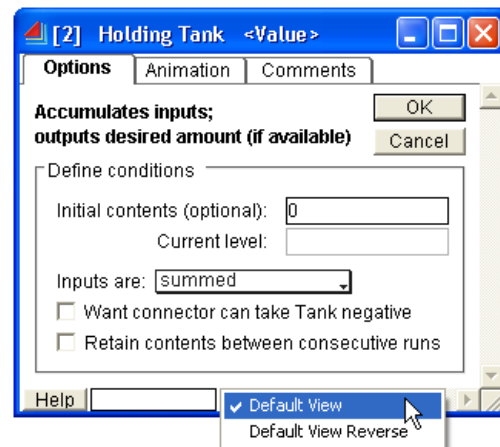
- ▶ Alternate connector positions, such as an input connector on the right side of the block. This helps avoid awkward connection line placements when building models.
- ▶ Choices of icon size, such as regular and reduced size. An example is the History block (Item library) where the *Status* view is smaller than the *Default* view. 
- ▶ Different icons depending on choices that have been made in the block's dialog. For example, a block that represents a flow could be pointed forward in the view labeled *Forward* and pointed backward in the view labeled *Backward*.
- ▶ Alternate icons depending on what the block represents in a model. This is often used for hierarchical blocks that represent a specific process in one part of the model but a different process, with the same functionality, in a different part of the model. For instance, in the Markov Chain Weather model discussed on page 50, the hierarchical block "Weather Forecast" uses views to indicate the current state for a state/action model.

Blocks without icon views will not have any choices in the Views popup menu at the bottom of the block's dialog. If additional views are present, they can be selected by right-clicking on a block or by using the Views popup menu. They can also be called by functions.

ExtendSim facilitates the creation of different icon views while developing or editing a block's structure. For more information see the Developer Reference.

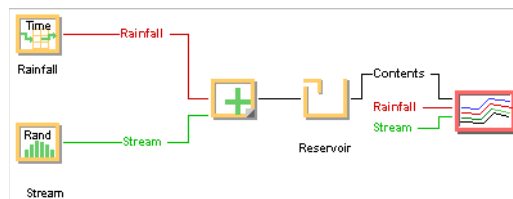
### Connectors

Connectors are used to input and output values, items, or flow for each block.



Icon views for Holding Tank block

The Model menu has commands that let you show or hide the connectors in the model workspace. Hiding connectors can improve the appearance of the model, especially for complex models with many blocks.















Reservoir model with connectors hidden

How To

### Connector types

Connectors on a block are visually different, depending on how they're used. This table describes the different types of connectors that can be seen on or (if you program) added to blocks in ExtendSim.

Type	Input	Output	How they're used
Value			Continuous blocks use value connectors to pass values from one block to another. Blocks in the Item and Rate libraries may also use value connectors for the same purpose.
Item			Discrete event blocks use item connectors to pass discrete items. Blocks in the Rate library may also use item connectors for the same purpose.
Flow			Discrete rate blocks use flow connectors to pass information about the effective rate from one block to another. You can also connect from a flow connector (input or output) to a value input connector.
Universal			Universal connectors are usually used as inputs. Value, Item, Flow, and User-Defined output connectors can connect to Universal inputs.
Array			Array connectors are for passing arrays of information from one block to another.
User Defined			User Defined connectors can be programmed to behave in any manner the developer wants.

Other than some specific instances, you cannot connect from one type of connector to another. For example, attempting to draw a connection from an Item output connector to a Value input will result in an error message. The two exceptions are that Value, Item, Flow, and User-Defined outputs can be connected to Universal inputs, and Value outputs can be connected to Flow inputs.

### Variable connectors

Regardless of type, each connector can be single or variable depending on how the block is constructed. Variable connectors act like a row of single connectors, where the row can be expanded or contracted to provide a required number of connectors. Some blocks have only single connectors, some have only variable connectors, and some have a mix. Variable connectors are usually designated by a black arrow, as shown below.

The Constant block has two single input connectors and one single output connector:



A Math block in Add mode has a variable input connector and a single output connector:



The Math block's variable connector expanded to provide three inputs:



Each block's functionality determines whether or not it has variable connectors, where they are located, and what each connector represents. For example, a variable connector could have connectors captioned *minimum* and *maximum* placed on the bottom of the block for downward expansion.

Variable connectors can be expanded or contracted; they can also be collapsed. You expand or contract a variable connector to provide the desired number of inputs or outputs. You collapse a variable connector to improve model appearance.

**Expanding or contracting a variable connector**


Wherever possible a block will anticipate its usage and provide the required number of connectors. For example, the Holding Tank block (Value library) has a variable connector that can be expanded to provide three inputs: the amount wanted from the tank, a trigger to reset the tank's settings, and the amount of initial contents. You might use one, two, or all three of those inputs, but the block won't allow more than three inputs.


For other blocks, the number of connectors you might need depends on choices you make in the dialog or how you use the block in a model. There are two primary methods for causing a variable connector to change the number of available connectors:

- Drag the variable connector (as discussed below) until the desired number of connectors is achieved. An example of this is the Reservoir model in “, Building a Model ,” where you expanded the variable connector of the Math block (Value library) to two inputs, one each for rainfall and stream. If instead you had six water sources you could have expanded the variable connector to accommodate all six.
- Change some setting in a dialog, causing an increase or decrease in the number of connectors. An example of this is the Random Number block (Value library), where you select a distribution from a popup. Depending on the distribution chosen, the variable connector will provide from two to four inputs for specifying the distribution's arguments. Since the number of arguments for each distribution is fixed, you will not be able to expand the number of connectors beyond


How To

the proper number. You can, however, contract the variable connector as discussed below. Note that some blocks with variable connectors, such as the Equation blocks, do not allow you to increase or decrease the number of connectors except through dialog settings.

 To increase the number of connectors, hover the cursor over the black arrow on the variable connector until it becomes a dragging cursor (shown at left). Then click and drag in the direction of the arrow. As you drag, more connectors appear and the number of connectors is displayed.

 You will not be able to expand a variable connector beyond what is reasonable for the block, given its usage and dialog settings. And some blocks, for example the Equation blocks, do not provide a black arrow for expanding the number of connectors; the number of connectors in those blocks is only controlled through the dialog.

To decrease the number of connectors, hover the cursor over the black arrow at the end of the last connector until it becomes a dragging cursor. Then click and drag back towards the variable connector's starting point.

 You will not be able to contract a variable connector below what is reasonable for the block, given its usage and dialog settings. The number and location of connections to the variable connector will also affect its ability to be contracted. If none of the connectors, or only the first connector, is connected, you can contract the variable connector back to its unexpanded position by clicking and dragging, as described above. However, if you have connected to more than one connector, you can only contract the variable connector to the point of the outermost connection. In this case, to reduce its size you need to collapse it, as discussed below.

#### **Collapsing a variable connector**

As discussed above, if you have made connections to a variable connector you might not be allowed to contract it back to its unexpanded position. This is a safety mechanism, so that it is clear what connections have been made. However, to simplify the appearance of the model, you can collapse a variable connector to an unexpanded state even if it has many connections to it.

To collapse a variable connector, place the cursor over the black arrow until it becomes a dragging cursor, then double-click. The connector now appears with a red + sign in place of the black arrow, as shown on the right.



To reverse the process, place the cursor over the red + sign until it becomes a dragging cursor, then double-click.

#### **Connecting to different connector types**

As you saw in earlier chapters, you can use continuous blocks (such as those from the Value library) in your discrete event and discrete rate models, which means you can have blocks in a single model that use different types of connectors (some blocks have multiple types of connectors on them as well).

You can connect value connectors together, item connectors together, and flow connectors together. You can also connect value and flow connectors to each other, but you cannot connect value and flow connectors to item connectors. This is because value and flow connectors only pass values while item connectors pass unique items.

 Any type of connector—value, flow, or item—can be connected to universal input connectors.

Also note that if you create blocks that have user defined connectors, those connectors can only be attached to other user defined connectors and to universal connectors.

## Dialogs

Most block dialogs let you change or view the settings before, during, and after a simulation run. Some of the things you can do using a dialog include:

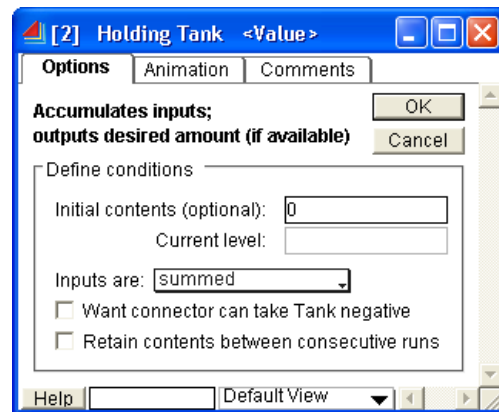
- Choose from multiple functions (e.g. the Math block)
- Enter initial values
- View simulation results
- Add comments to help document your models
- Set animation criteria
- Choose notification options

To open a block's dialog, double-click or right-click the block's icon. For example, if you double-click the Holding Tank icon, the dialog at right opens.

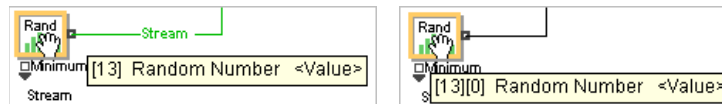
In the dialog's title bar is the block's global block number, its name, and, in braces, the library it resides in.

As each block is added to a model it is assigned a unique and sequential global block number from 0 to n-1. This sequence does not change. If a block is deleted, its number becomes an "unused slot" which is available when another block or text is added to the model window.

Each block within a hierarchical block has a second, local, block number that reflects its relationship to the other blocks in the hierarchical submodel.



Holding Tank dialog



Left: Random number block on worksheet, global number is 13

Right: Inside hierarchical block, global number is still 13, local number is 0

At the bottom of every dialog is a Help button that provides more information, such as the block's purpose and use, what each connector does, the meaning of each dialog item, and so on. Beside the button is a text box where you can enter a label for the block, up to 31 characters. The View popup box lets you change the view of the icon for the selected block, when available.

Some dialogs also calculate and display values that are generated as the model runs, so if you leave a dialog open during the simulation, you can watch the impact on different variables. You can even change some of the settings in a dialog as you run the simulation, such as choosing different buttons or typing new values.

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

### Animating blocks

ExtendSim provides built-in animation for many blocks and you can also create custom animation for any block. To learn more, see “Animation” on page 551.

### Hierarchical blocks

It's not unusual for models to have thousands of blocks in them, which can make it difficult to understand what is happening in the model. Hierarchy helps solve this problem by grouping several blocks together into one hierarchical block that represents a portion of the process being modeled. A hierarchical block can contain submodels, text, graphics and clones of dialog items and tables. Hierarchical blocks can even contain other hierarchical blocks, so there are multiple levels of hierarchy.

 The Navigator window in Model Exploring mode is a useful tool for quickly drilling down through the different levels of hierarchical blocks to find the one you're looking for.

Because hierarchy is mainly used to enhance or simplify model appearance, it is discussed in further detail in “Hierarchy” on page 540.

### Managing blocks

 The topics within this section are only for block developers and those who save hierarchical blocks in libraries.

#### Copying blocks

To make a copy of a block in the *same* library, select the block in the library window and choose Edit > Duplicate. The Duplicate command copies the block into the current library and renames it with the block name followed by the word *Copy*. This is common when you want to use the ModL code in one block as the template for a different block.

To copy a block from one library to another, open the library windows for each of the libraries. Then drag the block from its library window to the destination library window.

#### Changing a block's name

To change the name of a block, select the block in the library window and choose Develop > Rename Block. Then type in the new name. If you change the name of a block, models that use that block will not be able to find it because they are expecting the original name. ExtendSim will then ask you where the block is located and present a dialog box for searching, as discussed in “Block searches” on page 492.

#### Removing blocks

It is rare that you will want to remove a block from a library but, if you do, select the block in the library window and choose Edit > Clear or press the Delete or Backspace key. ExtendSim will not let you remove a block that is in use in an open model window. If you remove a block that is used by a model and later open that model, ExtendSim will present an error message and give you the opportunity to indicate the location of the missing block; if it can't find the block, it will put a placeholder in the model window.

#### Corrupted blocks

On rare occasions, you may get a message as you open a library that indicates that a block has been corrupted or is bad. The corrupted block will appear in the library window as *\*BAD\*Blockname*. To save the rest of the library, copy the uncorrupted blocks to a new library and discard the old library. Then copy a backup copy of the block (if you have one) into this new library. This is a good reason to always back up your work!

# How To

## Creating a Custom User Interface

Personalize the ExtendSim  
modeling environment

*“If you can dream it, you can do it.”*  
— *Walt Disney*

Every modeler's needs are different. ExtendSim offers a variety of methods to customize the application's interface so you can work in the most convenient and effective manner. This chapter discusses several methods you can use to create a user interface, including:

- Cloning dialog and plotter items to change model parameters and report results
- Centralizing data in a database
- Simplifying and organizing models with hierarchy
- Creating a dashboard interface
- Using the Notebook to document and manage your models
- Adding Controls to change and monitor critical parameters on the fly
- Interacting with the model user
- Using external applications as an interface
- Documenting models using text and graphics

## Cloning

In most programs, buttons and dialog parameters are found only in dialogs. The advantage of this is you always know where to find them. However, having all your choices in dialogs can be a disadvantage in large models. For instance, you may want easy access to parameters in blocks that are scattered in multiple hierarchical layers throughout the model. Or you might want to provide a more accessible interface for other users of the model. ExtendSim overcomes these problems by giving you freedom to *clone* dialog and plotter items and place them in a more convenient location, effectively creating a link with the original dialog item.


Cloning gives you easier access to dialog and plotter items when you want to change settings or monitor simulation results. Dialog parameter fields, tables, text, buttons, checkboxes, and radio buttons, as well as graphs and data tables from plotters can all be cloned. Cloned items can be placed in the model window, Notebook, or a hierarchical block's worksheet, and text labels can be used to make the cloned item easy to understand. You can clone multiple items from the same dialog or plotter or clone the same item to more than one location. Every clone acts exactly like the original: if you change the original or any clone, all instances are updated immediately.

Using cloned dialog items puts you in direct control of your models. Clone the dialog items to a centralized location so you can easily change parameters. Label the clones so they are well-documented and use them as the simulation runs, such as clicking buttons or entering values in text entry boxes. For example, clone a Constant block's **Constant value** field to change the value between simulations without having to open the block's dialog, making it easier to test different assumptions. Or have an area of the model or its Notebook that lets you monitor several numeric values at a time, rather than just using the Plotter's graph to watch the simulation results.

 Clones that display changes will cause the simulation to run slower.

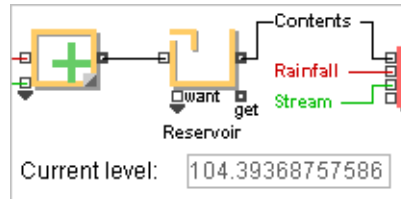
### How to clone a dialog item

The same technique is used to clone an item onto the model or hierarchical block's worksheet or into a Notebook.

- ▶ Open the Reservoir 1 model (located in the ExtendSim7\Examples\Tutorial folder)
- ▶ Double-click the Holding Tank block to open its dialog.
- ▶ Click the Clone layer button in the toolbar to select that tool. 



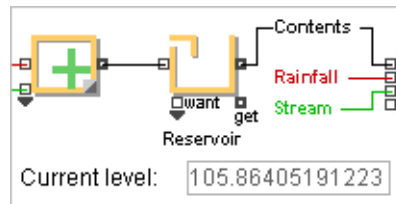
- ▶ Select both the **Current level** text and its parameter field and drag them onto the model window below the Holding Tank block.
- ☞ Hold down the Shift key to select multiple items or drag a frame around them. Clones can be moved, resized, and aligned after they are placed.
- ▶ Close the dialog.



Cloned Holding Tank dialog items

- ▶ Run the simulation.

The cloned item will display the changing level in the Reservoir as the simulation runs.



Cloned item after simulation run

- ☞ If the model runs too quickly to see the change, go to Run > Simulation Setup > Setup tab and change the end time to something much larger, such as 1000, then run the simulation again.

### Using cloned items

Once you have a cloned item, you can drag it almost anywhere you want. Many people prefer to have all the items together at one side of the model window, for example, or to clone all dialog items into the Notebook or into a hierarchical block.

- ⚠ You cannot move a cloned item from one hierarchical block to another unless the target hierarchical block contains the source hierarchical block.

To move a cloned item, choose the Clone tool from the toolbar, click on the item, and drag it. To align two or more clones, select them and use the Model > Align command.

To resize a cloned item, choose the Clone tool and click once near the center of the cloned item so that the resizing handles appear. Then click and drag a handle to change the size and shape of the clone.



Resizing cloned item

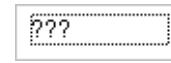
To remove cloned items from a model, simply select them (using the Clone tool) and press the Delete key or choose Edit > Clear.

- ☞ If you delete the block from which you cloned a dialog item, the cloned item is automatically deleted.

To find and open the dialog from which an item was cloned, choose the Clone tool and double-click the cloned item.

### Unlinked clones

If a block's structure is modified by deleting a dialog item or changing the tab order of the dialog items, clones associated with that block may become *unlinked*. In this case, the clone will be grayed out and "???" will appear in place of the cloned dialog item.



Unlinked clone

Choosing the clone tool and double-clicking the unlinked clone will cause an alert message to appear. Then the dialog of the block from which the clone originally came will open. You may then select a replacement item to clone, or simply delete the clone if it is no longer needed.

## Centralizing data in a database

Cloning is a quick and easy way to create a user interface. However, for large models it may be too tedious to clone all the necessary items for easy access. ExtendSim databases are useful for organizing data for complex models into a central location.

You can create databases to store parameters to be used in the model, model results, or a combination of the two. Use the linking technologies to dynamically link dialog parameters or tables with the database. Or use the Read and Write blocks to dynamically access database data. Since databases are primarily data management systems, see further discussion at "ExtendSim databases for internal data storage" on page 638.

## Hierarchy

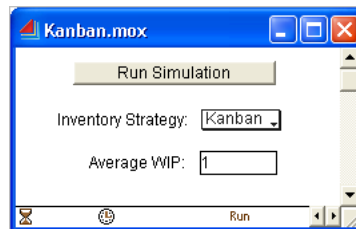
Hierarchy is useful for customizing a model's interface. It reduces the number of blocks visible on the model worksheet, so models are easier to understand and navigate. It is also used to organize the model into easily-recognized components, separating the model into process elements that are more closely tied to reality.

For instance, you could encapsulate portions of a model into hierarchical block then clone dialog items to the top of the hierarchical block's worksheet for easy access. Or add custom pictures as icons for groups of hierarchical blocks that all relate to a certain part of the process you're simulating. You could also create a hierarchical Help block for the model user, as discussed at "Help block" on page 514.

For more information about working with hierarchy and customizing and animating hierarchical blocks, see "Hierarchy" on page 540.

## Creating a dashboard interface

In addition to permitting cloned items in the Notebook or model worksheet for creating a user interface, ExtendSim lets you add customized buttons, popup menus, and on/off switches.



Model window showing a custom button, a custom popup menu, and a cloned dialog item

## Buttons



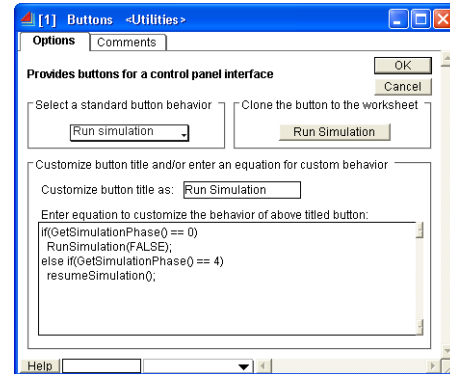
The Buttons block (Utilities library) makes it easy to create buttons that will activate frequently-used commands. A number of predefined buttons can be selected from the popup menu in the block's dialog. You can also create a custom button for a specific purpose by changing the label and/or equation.

The pre-defined buttons are:

- Animation On
- Animation Off
- Open Notebook
- Run Simulation
- Pause Simulation
- Save Model
- Open Block
- Open H-block
- Open Database Table
- Open Database

Select or create the button in the block's dialog, then clone the button from the dialog to the model worksheet, Notebook, or hierarchical block. The button will execute the command whenever it is clicked. Each button requires a separate Buttons block. To store the block, place it anywhere on the model worksheet or within a hierarchical block. A common place is to store the Buttons block behind its cloned button.

The Animation On and Animation Off buttons in the Buttons block only control 2D animation during the simulation run; they do not control 3D animation.



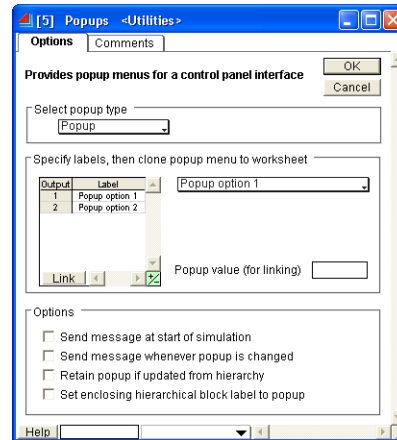
Buttons block dialog

### Popup menus



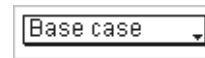
The Popups block (Utilities library) lets you create a popup menu of customized options that a user can select from to control model behavior. Connect the block's output to the input connector of any block that requires that value. Depending on the option selected, the block will output a value specified in its dialog table.

To create the popup menu, enter a label for each option in the second column of the table in the block's dialog. Then clone the menu item to a worksheet, hierarchical block, or Notebook and connect the Popups block's output to the block you want to control. When the user selects an option from the menu, the Popups block outputs the corresponding numeric value from the first column in the dialog's table. You can create a popup menu with as many options as you want.



Popups block dialog

For an example where the Popups block is used, see the Monte Carlo model, located in the folder \Examples\Continuous\Standard Block Models. The hierarchical block labeled “Scenario in the lower right corner contains a Popups block. That block is used to create the popups menu for the cases.



Popups menu

You can link the table in the Popups' dialog to an ExtendSim database table, creating a popup menu for selecting a database record.

### On/Off Switch



The Switch block (Utilities library) can be used like an On/Off switch to control some aspect of a model. The block's input connector is used to turn the Switch on and off; its output connector reports the status. A value of 0 (zero) indicates the Switch is off and a value of 1 (one) indicates it is on. This block is similar to the Switch control described in page 510, but it has some additional features: the Switch block's dialog has a Switch that can be cloned to the model worksheet, Notebook, and so forth; there is a dialog item that allows the Switch's status value (1 or 0) to be linked to a database, global array, or Excel spreadsheet; and in a discrete event model, the Switch block can send a message to a connected block to notify it of a status change.

### Additional blocks to control model execution

To give the user more customized control over model execution, the Run Model and Pause Sim blocks (Utilities library) provide additional settings that determine what happens when the model is run or paused. You can clone dialog items from the Run Model and Pause Sim blocks to the model worksheet, as part of creating a user interface. For more information, see “Blocks that control or monitor simulation runs” on page 525.

### Notebooks

As you saw in the tutorial in Chapter 1, a Notebook is a window you can customize to help you organize and manage the data in a model. Each model has its own Notebook which can contain clones of dialog and plotter items (see “Cloning” on page 504), as well as text, pictures, and draw-

ing objects (see “Text and graphics” on page 514. To see an example of cloning a plotter’s graph to a Notebook, see “Cloning” on page 38.

A typical use for Notebooks is for collecting all of the items you might want to watch in one place. Since you can leave a Notebook open as a simulation runs, use it as a central display for all of the important values in all the dialogs.

Another common use is as a control panel. Clone all the dialog items that you might want to change while the simulation is running to the Notebook so you do not need to open the dialogs from the worksheet in order to make a change.

☞ If a model’s Notebook already has data or other contents in it, it will say (has data) beside the Window > Notebook command.

When you save a model with the Notebook open, the Notebook will automatically open with the same configuration the next time the file is opened. A Notebook can be many pages long. With the Notebook as the active window, choose File > Show Page Breaks to show the number of pages and how they would print.

You can copy the contents of a Notebook as a picture which can be pasted into other applications such as a word processing document or a presentation. Simply select the items you want, then choose the Edit > Copy To Picture command to copy the picture into the Clipboard.

## Controls

ExtendSim has three special tools in the Model > Controls command that are used to control blocks and show values directly as the simulation runs.

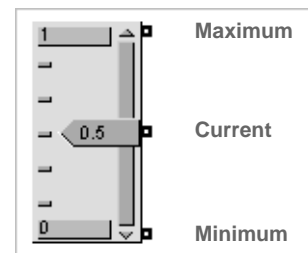
The controls are the *Slider*, *Switch*, and *Meter*. The Slider and the Switch are used to set values in your models. The Meter is used to see values as the model is running. The Meter also has a dialog that you can access by double-clicking it.

### Slider

A Slider control lets you slide an indicator along a scale to change the value of its output. You set the maximum and minimum values by selecting the numbers at the top and bottom of the Slider and typing in the desired value.

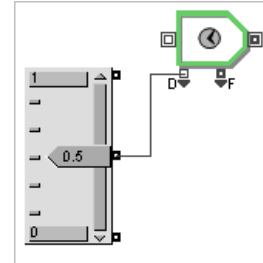
☞ If you enter a value at the bottom of the Slider which is higher than the maximum, or if you set the value at the top of the Slider to be less than the minimum, ExtendSim will warn you.

As you slide the indicator, the current value is displayed on the indicator and is output through the middle connector. There are also connectors next to the minimum and maximum values that output those values.



Slider

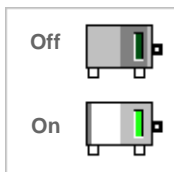
The Slider is useful to output a number in a range when the number does not need to be exact or to experiment with model parameters. For example, if you have an Activity block where you want to specify the delay as the simulation is running and a delay of 8 is slow but a delay of 2 is fast, you might put in a Slider with 8 as the maximum and 2 as a minimum. Then, as the simulation is running, simply drag the Slider's bar up and down to indicate slow and fast.



Using a Slider with an Activity block

### Switch

The Switch control looks like a standard LED switch that glows green when it is on. It has two inputs and one output.

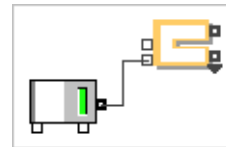


Switch showing off and on states

The Switch outputs either a 0 (zero) or a 1 (one) depending on the state of the LED. When you click on the left side, the Switch turns off, the LED turns dark gray and the Switch outputs a 0 (zero). Clicking on the right side (the side with the LED) causes the Switch to turn on, the LED to glow green, and the Switch to output a 1 (one).

The Switch control is valuable in controlling blocks that have true-false inputs. For example, you might attach a Switch to the **SelectIn** connector of a Select Value Out block, as shown below.

The input connectors at the bottom of the Switch are used to change the state of the Switch by setting their side of the Switch to true from within your model. For example, you might want to set the Switch to off when a model starts, then change it to on after some period of time. The next time you run the model, the Switch would be automatically set to off again when the model begins.



Using a Switch (on state) with a Select Value In block

When either of the Switch's inputs gets a true value (defined as 0.5 or greater), it selects that side of the Switch. If the other input later gets a true value, the Switch will shift to that side.

If an input receives a true value and that side of the Switch is already selected, no change is made.

### Meter

The Meter visually displays how values vary between a known maximum and minimum. Set the maximum and minimum values through the Meter's dialog or by connecting other blocks (such as Constant blocks) to the top and bottom connectors.



Meter

This is a good control to use if you want to see values while the simulation runs but you don't need to save them to a plotter.

## Interacting with the model user

Some ExtendSim blocks provide a convenient method for monitoring conditions in the model, requesting input from the user, and reporting changes. In addition, if you build your own blocks you can add customized alerts and prompts to display results and prompt for input data.

### Notify block



Set to play a sound

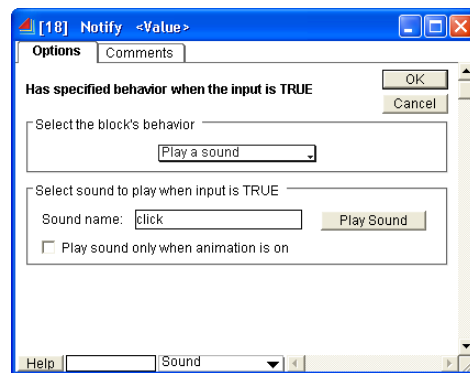
The Notify block (Value library) provides three options for sending messages to the user: **Play a sound**, **Prompt for output value**, and **Stop the simulation**. For all three options, the Notify block has an input connector that receives a True or False value that is used to determine whether or not to play a sound, stop the simulation, or issue a prompt. When the Prompt option is selected, the block also has an output connector to output the value the user enters after being prompted.

All three options define True as being  $\geq 0.5$ .

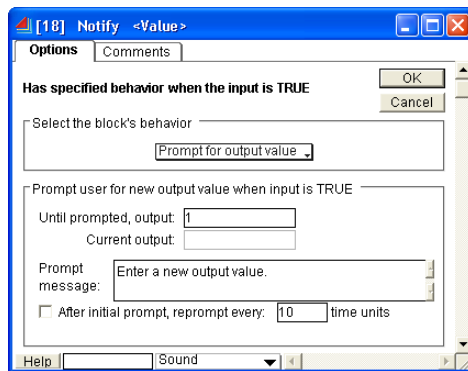
#### Play a sound

This option plays a sound when its input connector gets a True value (alternately, you can choose to have it play only when the input value is True and animation is on). To use a sound, choose the **Play a sound** option and enter the sound name in the Notify block's dialog. You can enter the name (such as **click** or **crack**) for any sound located in the ExtendSim Extensions folder. To hear that sound, click the Play Sound button.

Sound files you create and place in the Extensions folder must be in **.wav** format (Windows) or an **snd resource** (Mac OS). On Windows, you can also enter the name for any sound in Window's Sounds Control Panel.



Sound option selected in Notify block's dialog



Prompt option selected in Notify block

#### Prompt for output value

This option is mainly used to prompt the user to change the block's output value. Until the input connector gets a True value, the block outputs the value specified in the dialog. When the input is  $\geq 0.5$ , the block prompts the user to enter a different value to output.

You use this block to pause a simulation, request a value from the user, then continue the simulation using

the new value. The output value can be any number, including 0. If the user clicks the Cancel button in the prompt dialog, the simulation stops.

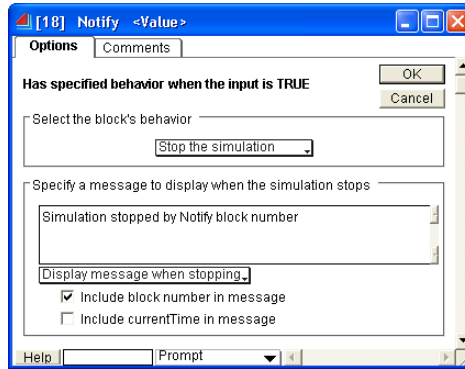
### Stop the simulation

This option is used to stop the simulation and alert the user if the monitored parameter gets a True value.

You can customize the block to display any warning message you want, up to 255 characters. You can also set the message to include the block's number and the time the event occurred.

### Equation blocks

The Equation block (Value library) and Equation(I) block (Item library) provide flexibility and control for creating user interface elements. Most ModL functions can be called from an Equation block, including functions specifically for interacting with the model user. The blocks provide similar functionality, but the Equation block calculates its equation when it gets a value and the Equation(I) block calculates its equation when an item arrives. Equation blocks are useful when you want a more complex set of rules for the interaction than the Notify block provides.



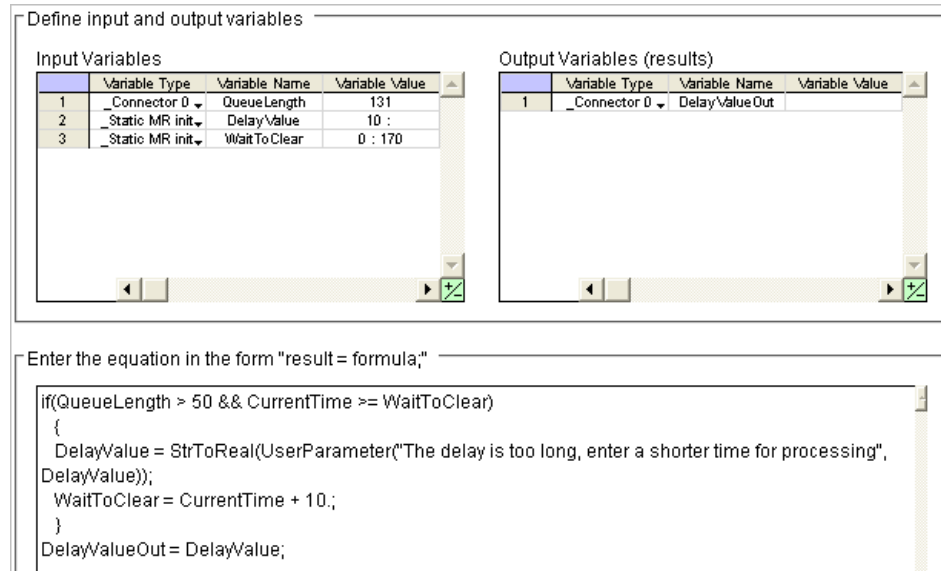
Stop option selected in Notify block



For example, you could use an Equation block to prompt for a value if a certain condition exists and then change the result based on input from the user, like the Prompt block. But then add another condition, such as a delay between when the user is prompted.



The equation in the following screenshot causes a message to appear to the model user, asking for a new processing time if the length of the queue exceeds 50 and at least 10 time units have elapsed since the last value was requested.



### Additional interactive features if you program

If you develop your own libraries of blocks or want to modify existing blocks, ExtendSim provides even more capability for delivering messages and interacting with users. Here are some ideas from the ExtendSim Developer Reference:

- Change what is shown in a dialog depending on what occurs in the model or what the user selects or enters in the dialog. For example, you can change the text that is displayed in a block's dialog depending on which button a user clicks.
- Some of the available functions include displaying a message, prompting the user to input a value, or making a sound. These functions can also be used for debugging ExtendSim's ModL code, although using the Source Code Debugger is the preferred approach.
- DLLs (Windows) and Shared Libraries (Mac OS) are especially handy where you want to add a feature or functionality that ExtendSim's language (ModL) does not support. For example, you could use a DLL or Shared Library to display a picture or graphic in a separate window when a user clicks a button or to create a customized sound resource based on numerical values from the model. DLLs and Shared Libraries are segments of code written in any language, such as Visual Basic or C++. ExtendSim's DLL and Shared Library functions allow you to call these code segment resources from within a block's ModL code and perform operations.
- Embedded objects can be used to place a document or display from another application, such as Excel, into an ExtendSim model. Use one of the methods discussed at "Embedding an object (Windows only)" on page 655 to transfer information between ExtendSim and the ActiveX (Windows) control.

- COM/ActiveX Automation allows ExtendSim to communicate with other applications either as a server or a client. As a client, ExtendSim uses the external application's COM object model to communicate with it. As a server, ExtendSim responds to COM/ActiveX messages. For more information, see "ActiveX/COM/OLE (Windows only)" on page 665.

### **External applications as an interface**

Other applications can serve as an interface to an ExtendSim model. This is accomplished by linking to ExtendSim models or blocks directly, using technologies such as ActiveX commands, or indirectly through text files.

Using an external application works much the same as using the ExtendSim database feature, since the application stores the parameters to be used in the model and the results of the simulation. For more information, see "Exchanging data with external applications" on page 657.

### **Documenting models**

Creating a user interface also involves adding information or graphics to clarify what is happening in a part of the model or to explain why a particular modeling approach was used. You do this by putting text, pictures, or draw objects on the worksheet or by creating a hierarchical "Help" block, as shown below.

#### **Text and graphics**

You can improve the organization of your model or Notebook by grouping elements together and using text and graphics to identify or highlight different sections. To learn more about adding text and graphics, including using colors and patterns to enhance them, see "Working with text" on page 538, "Graphic shapes, tools, and commands" on page 561, and "Patterns and colors" on page 562.

#### **Help block**

To create a Help block, choose Model > New Hierarchical Block and name the H-block "Help". Then add explanatory text, pictures and/or draw objects to the hierarchical block's worksheet, close and save the block, and place it where you want on the model.

For more information about working with hierarchy and customizing hierarchical block icons, see "Hierarchy" on page 540.

# How To

## Model Execution

Tips for running simulations

*“The future is something that everyone reaches  
at the rate of sixty minutes an hour.”  
— C. S. Lewis*

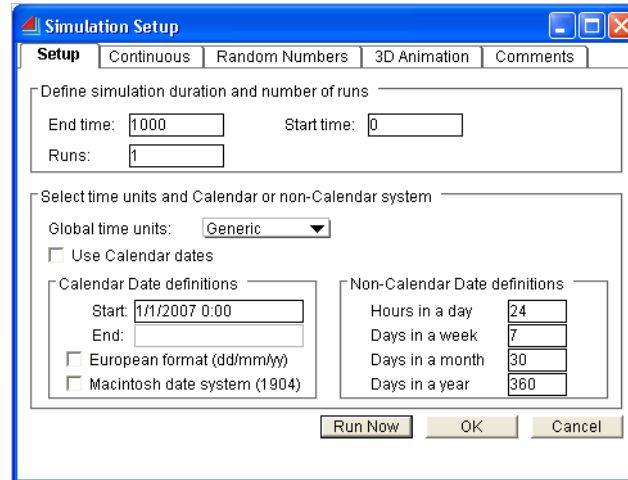
With ExtendSim, you have several options for controlling simulation runs, including:

- Using the Simulation Setup dialog to determine how models will run
- Important points when running a model
- Information provided in the status bar
- Using blocks to control or monitor simulation runs
- Saving intermediate results
- Setting the timing for your runs
- Determining an appropriate simulation order
- Choosing time and other units that are most relevant for your models
- Determining the length and number of runs
- Speeding up and slowing down simulations
- Working with multiple models
- How ExtendSim passes messages in models

### **Simulation setup**

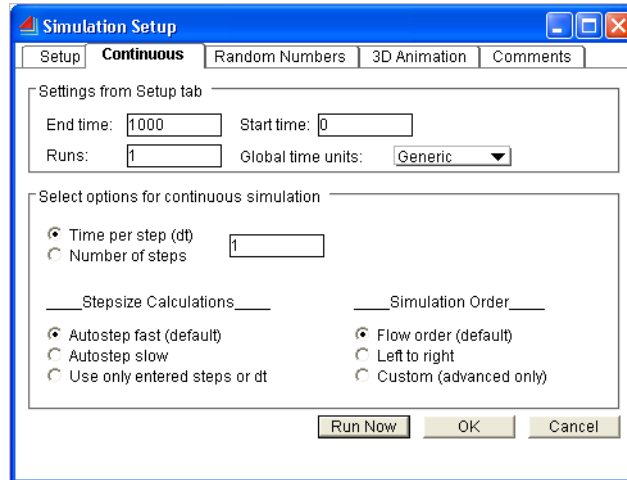
Before running a simulation, you need to specify how long it will run. Use the Simulation Setup dialog to set the simulation duration, choose a time unit for the model, have the model run multiple times, or select other simulation and animation options. To access the dialog, choose Run > Simulation Setup. The dialog has five tabs which are described in detail below.

Setup tab



Choice	Description
End time	The time that the simulation will end. See also “Simulation timing” on page 82 for continuous models and “Moving items through the simulation” on page 246 for discrete event and discrete rate models.
Start time	The current time at the start of the simulation. By default this is set to 0, since that is the most common starting point. Set it to a different value if the model uses the time value for some calculations. See also “Simulation timing” on page 82 for continuous models and “Moving items through the simulation” on page 246 for discrete event and discrete rate models.
Runs	The number of consecutive times to run this simulation. In the status bar, discussed on page 524, the numbering of simulation runs starts at 0.
Global time units	Time unit for the entire model. Local time units can be defined within the dialogs of blocks that contain time parameters. See “Time units” on page 526.
Calendar date definitions	Allows you to specify calendar-based timing if <i>Use Calendar dates</i> is selected. See “Calendar dates” on page 528.
Non-Calendar date definitions	Only available if <i>Use Calendar dates</i> is not selected. Used by ExtendSim to automatically convert local time units to the global time unit. See “Time unit conversions (non-Calendar dates)” on page 529.

### Continuous tab

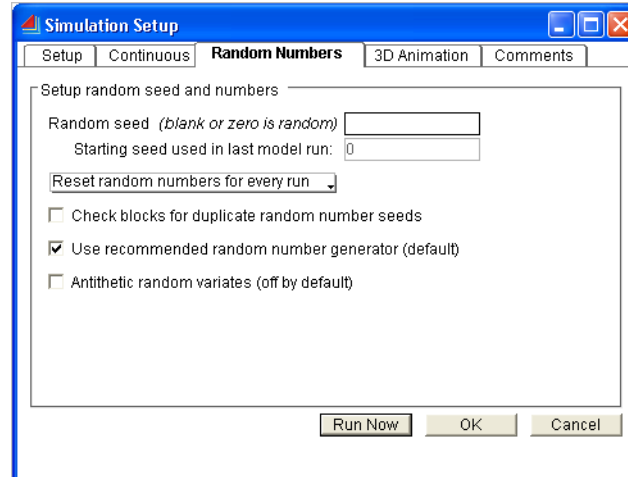


 The Continuous tab is only used for changing the time step, stepsize calculation, or simulation order for a continuous model.

Choice	Description
End time, Start time, Runs, Global time units	For the convenience of continuous modelers who want to change stepsize, these four settings from the Setup tab are repeated on the Continuous tab. Changing any of these settings on one tab will change them on the other. See their descriptions in “Setup tab” on page 517.
Time per step (dt)	Represents delta time (dt), or the length of time per step. This is the default choice for determining the granularity of the simulation run. For most purposes use the default setting of 1, meaning that each step will be one time unit long. A value for the number of steps is automatically calculated based on the dt entered. The value is computed as: $\text{floor}(((\text{EndTime}-\text{StartTime})/\text{DeltaTime}) + 1.5)$ . To see this, select the <i>Number of steps</i> option after changing the <i>Time per step (dt)</i> . See also “Specifying dt or the number of steps” on page 83.
Number of steps	Another method for determining the granularity of time for the simulation run. In most cases, this would be a number equal to the duration (length of the simulation run); the model calculates values once for each step and each step is one time unit long. A default value for delta time is automatically calculated based on the number of steps you enter. The value is computed as $(\text{EndTime}-\text{StartTime})/(\text{NumSteps} - 1)$ . To see this, select the <i>Time per step (dt)</i> option after changing the <i>Number of steps</i> . See also “Specifying dt or the number of steps” on page 83.
Stepsize calculations	These are only used in continuous simulations that change the DeltaTime system variable, such as electronics models. Most models should use <i>Autostep fast</i> . If the blocks that you create change DeltaTime and demand more accuracy, use <i>Autostep slow</i> (which divides the calculated value for DeltaTime by 5). If your custom blocks change DeltaTime but you want to ignore the changes, select <i>Only use entered steps or dt</i> .

Choice	Description
Simulation order	<i>Flow order</i> should be used for most models. For more information, see “Simulation order” on page 86.

### Random Numbers tab



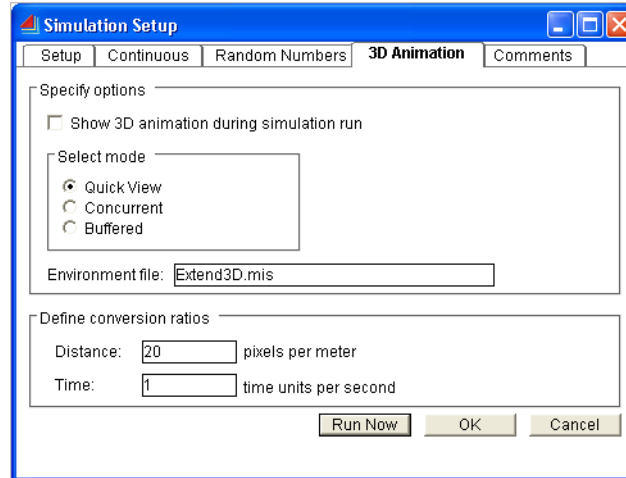
For the table that follows, *consecutive simulation runs* means setting the number of simulation runs in the Setup tab to something greater than 1.


Choice	Description
Random seed	The “interface” for the random number generator. A value of 0 or blank uses a random seed; any other value causes repeatable sequences of pseudo-random numbers. Note: Each block that outputs random numbers will generate its own independent sequence. For more information see “Random seeds” on page 605.
Starting seed used in last model run	Reports the random seed from the previous run.
Reset random numbers for every run	When this option is selected from the popup menu, random numbers are initialized at the start of every simulation run. If the global random seed is blank or 0, a new randomized seed will be generated at the start of every consecutive simulation run. If the global random seed is a positive integer, the same seed value will be used for every consecutive run.
Continue sequence of random numbers	When this option is selected from the popup menu, the sequence of seeds for each consecutive simulation run is continued from the previous run.

Choice	Description
Use database table __Seed for values	When this option is selected from the popup menu, the starting seed for each consecutive run is read from the __Seed table in the selected ExtendSim database. This option allows you to specify exactly which seeds will be used for each simulation run. It also is a convenient method for recording a set of seed values. To specify a set of random seeds, create a table in a database with one field and a number of records equal to the number of simulation runs. The model will access sequential records for each consecutive simulation run, and that record's value will be the seed value for the run. If you leave the records blank or enter zeros, ExtendSim will fill the table with seed values on the first run. It will then use those seed values for each subsequent set of runs.
Check blocks for duplicate random number seeds	At the start of the simulation, all blocks that use random numbers are checked to make sure that no two blocks are using the same seed values. This option is not necessary unless the <i>use block seed</i> option is selected in the individual blocks.
Use recommended random number generator (default)	When checked, ExtendSim will use the recommended <i>Minimum Standard</i> random number generator. When unchecked, ExtendSim will use the optional <i>Schrage random number generator</i> used in previous versions (backwards compatibility). For details about the ExtendSim random number generator see "Random number generators" on page 605.
Antithetic random variates (off by default)	Generates antithetic variates for possible variance reduction using multiple runs. It can be controlled by the Optimizer block (Value library), or by custom blocks, to alternate on and off in alternate runs using the AntitheticRandomVariates global variable. See the ExtendSim Developer Reference for more information. This setting should be off under normal use.



### 3D Animation tab



 3D animation is not available with all products. For complete information about 3D animation and the following settings, see the E3D module that starts on page 389.

Choice	Description
Show 3D animation during simulation run	Opens the E3D window when the model opens. If the window is subsequently closed, it will reopen during model initialization when the model is run. This is the same as selecting the command Run > Show 3D Animation.
Select mode	Defines how the E3D window and the simulation interact. <i>QuickView</i> shows all of the movement of items in the E3D window while the simulation is running. <i>Concurrent</i> is a more realistic animation than QuickView and shows only the movement of items that requires simulation time. The 3D animation runs during the simulation run. <i>Buffered</i> is similar to Concurrent except it runs the 3D animation after the simulation has ended. The selected mode is shown at the top of the E3D window.
Environment file	Selects a file that specifies the appearance and behavior of the background for the E3D window. The default is the Extend3D.mis file; you can also create custom environment files.
Define conversion ratios	Specifies the relationship for distance and time between the E3D window and the simulation. The distance ratio defines how many pixels in the 2D worksheet window represent one meter in the E3D window. The value for the time ratio controls the 3D animation's display speed and is displayed at the top of the E3D window. The time ratio value, and the animation speed, changes if you click the Faster or Slower buttons on the E3D window. In most cases the default values for these settings will not need to be changed.

How To

### Comments tab

Use this tab to enter comments about the simulation run.

## Running a model

In the ExtendSim Tutorial you learned the basic steps in running a model. Some additional points are discussed in the following sections.

### Menu commands and toolbar buttons

You can run, stop, and pause simulations using toolbar buttons or the commands under the Run menu.

- The Run menu commands are discussed at “Run menu” on page 708.
- Buttons to activate the Run commands are shown at “Toolbar buttons” on page 714.

### Running a model multiple times

It is common that you will build a model and run it repeatedly. Running a model multiple times gives a range of values indicating possible outcomes and facilitates model analysis. For example, you would do this when the model has random inputs, for Monte Carlo simulations, or when performing sensitivity analysis.

- As discussed in “Constant values and random variables” on page 57, models with one or more random inputs are called *stochastic* models. Stochastic models are run repeatedly and then analyzed statistically to determine a likely outcome.
- *Monte Carlo simulation* is commonly defined as applying random behavior to a static or dynamic model and evaluating the results over a number of trials or observations. Applying Monte Carlo simulation techniques to a dynamic model is also known as *stochastic* modeling, which is discussed above. You can build both static and dynamic models with ExtendSim. For more information, see “Monte Carlo modeling” on page 47.
- When you perform *sensitivity analysis* on a model, you select a variable for analysis and vary it to determine how much of an impact the variable has on the model as a whole. Sensitivity analysis is discussed in detail in “Sensitivity analysis” on page 568.

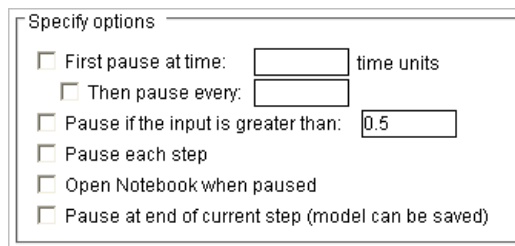
You should plan on running a simulation at least 3-5 times to estimate the variability in output whenever a model contains random variables. Then you can use statistical sample-size calculations (discussed in “Determining the length and number of runs” on page 531) to determine how many additional simulation runs are required to obtain an accurate estimate of the model’s performance. Some tools in ExtendSim for evaluating multiple simulation runs are:

- Histogram blocks (Plotter library) that show the shape and any tendencies in the range of results
- Error Bar plotters (Plotter library) that show how a variable is affected by randomness through the course of the simulation run
- MultiSim plotters (Plotter library) that can retain data and show the results of up to four simulation runs
- Confidence intervals (reported by the Statistics block in the Value library) that show the probability that a certain range captures the true mean for a simulation model result

### Stepping through a model

Sometimes you want to pause a simulation run to observe something of special interest, or step through the run to verify the action. There are several ways to accomplish this:

- Give the command Run > Pause, or click the Pause/Resume button in the toolbar, as the simulation runs. Then give the command Run > Resume or click the Pause/Resume button again to continue execution. This method works well for pausing a run one time to observe something of interest, but is not that useful for stepping through the entire model since it depends on your dexterity.
- Clone the Pause Simulation button from the Buttons block (discussed at “Buttons” on page 507). Then give the command Run > Resume or click the Pause/Resume button to continue execution. To use this method, click the Pause Simulation button each time you want the resumed simulation to pause. This works well if your intention is to pause the simulation once or twice when you see something of special interest.
- Use the Pause Sim block (Utilities library). This block causes the simulation to pause when certain conditions are met; the conditions are specified in the block’s dialog (seen at right). These options give more flexibility than either of the two previous methods; they are particularly useful when stepping through a model. Give the command Run > Resume or click the Pause/Resume button to continue execution.



Pause Sim dialog

- Use the Debugging menu commands discussed on “Debugging” on page 711. This hierarchical menu lets you modify the way a simulation runs and facilitates finding a modeling problem. The three “Step...” commands in the Debugging menu determine how the Step command in the Run menu performs during a simulation run.

See also “Slowing down simulations” on page 533.

### Other points when running models

- You can run multiple models simultaneously. However, the Run > Run Simulation command only activates one model at a time. See “Working with multiple models” on page 533 for how to activate multiple model runs.
- Windows and dialogs, such as the Notebook, a block’s dialog, or a hierarchical block’s worksheet, can be left open when the model runs. However, you cannot run a simulation with a block’s structure open and leaving too many windows open (especially if they have parameters that update constantly) can slow simulation speed.
- The ExtendSim application communicate with blocks in a model and the blocks can communicate with each other by sending and receiving messages before, during, and after a simulation run. This unique communication method allows powerful modeling constructs and results in models that are visually logical and easily understood. See also “How ExtendSim passes messages in models” on page 533.

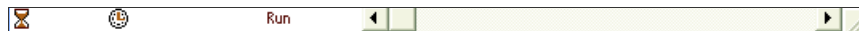
For more information about running models under different circumstances, see the following sections:

- “Animation” on page 551
- “Sensitivity analysis” on page 568

- “Optimization” on page 572
- “Dotted lines for unconnected connections” on page 618
- “Model reporting” on page 620
- “Confidence intervals” on page 567

### Status bar

Once the simulation run begins, ExtendSim shows information in a status bar at the bottom of your model:



Status Bar

When you start the simulation run, ExtendSim displays some initial status information in the form of messages that appear momentarily in the status bar area. Depending on the speed of your computer, you may see the following messages: Wait, Checking Data, or Initializing Data. These messages inform you of ExtendSim’s status as it checks and initializes the model prior to starting the run. On fast computers, the messages may appear too quickly for you to read.

Other messages are displayed in the status bar while the simulation runs:

- The number after the *hourglass* is an estimate of the actual time left in the simulation (expressed as minutes:seconds) so you can determine how long it will run.
- The *clock* shows the current time of the simulation in simulation time units.
- *Run* shows the number of the simulation run when multiple consecutive simulations are running (the numbering of simulation runs starts at 0).

These values are determined by the entries in the Simulation Setup dialog described in “Setup tab” on page 517.

Note that the time remaining shown in the Status Bar is only an estimate. For continuous simulations, it is usually accurate. For discrete event and discrete rate simulations, however, it can be inaccurate, as discussed next.

### Timer inconsistencies (event-based models only)

In discrete event and discrete rate models, the value for the timer hourglass sometimes varies from one moment to the next, especially at the beginning of a run when it is impossible to know when events are going to happen and thus how long it will take to run a model. The estimated time remaining may even increase if the simulation starts to run slower because items are being delayed more or because more events are being generated. The Status Bar will sometimes show the phrase *Initializing Data* during the initial steps of a discrete event simulation if there is a great deal of fast activity (for instance, if you use the preprocessing suggestion shown in “Connections to multiple item input connectors” on page 248).

Since it is only an estimate, do not be concerned about the value in the Status Bar; the model is running correctly regardless of the value shown there. However, if the timer hourglass continues to increase throughout the run and the simulation clock does not advance, it may indicate that there is an infinite loop in the model. Running the model with 2D animation on may help identify the cause of this.

## Blocks that control or monitor simulation runs

In addition to the menu commands and toolbar buttons, several blocks provide customized control over the simulation run. As is true for other blocks, you can clone dialog items from these blocks onto the model worksheet as discussed in “Cloning” on page 504.

### Buttons (Utilities library)



Creates buttons to activate frequently-used commands such as running a simulation, opening a Notebook, and saving the model. Since this block is most often used to create a user interface for a model, it is discussed in “Creating a dashboard interface” on page 506.

### Pause Sim (Utilities library)



Pauses the simulation when certain conditions are met. This gives more flexibility than cloning the Buttons block's Pause Simulation button (discussed on page 507). Give the command Run > Resume or click Pause/Resume to continue execution. See also “Stepping through a model” on page 522

### RealTimer (Utilities library)



Shows the duration of a simulation in real time. It should be placed at the far right side of the model worksheet. At the conclusion of the model run, its dialog opens to display the hours, minutes, seconds, tenths of seconds and ticks (sixtieth of a second) that the simulation took to run.

### Run Model (Utilities library)



Runs the simulation when the *Run Simulation Now* button is pressed. The options in this block give you more flexibility than cloning the Run Simulation button from the Buttons block (discussed at “Buttons” on page 507.)

### Time Sync (Utilities library)



Synchronizes the model to run in real time. It does this by pausing on each simulation step until the amount of simulation time that has passed equals the amount of real time that has passed. This is only effective if the model is running faster than real time. If the model is running slower than real time, the block will have no effect.

The timing starts at the beginning of simulation execution after all of the blocks have been initialized.

## Saving intermediate results

There may be occasions when you need to pause a simulation before it's finished running and save it to continue at a later time. For example:

- To interrupt a long model run when you need to shut down your computer
- To resume a simulation after a startup or warm-up period
- For debugging: pause just before the bug occurs and continue multiple times to try to track down the bug
- To change the random numbers after the model has run for an amount of time

To pause a running model and save it for continuation later:

- ▶ Bring the model window to the front, before all other windows.
- ▶ While the model is running, select File > Pause and Save/Save As.

► Close the model.

When you open the model later, select Run > Continue Simulation to resume.

**⚠** If you are doing this for debugging, do not re-save the model after continuing. Instead, repeatedly run the model from its original paused state.

## Timing

Unless you stop them earlier, all simulations run for a specified period of time. ExtendSim determines the *duration* of a simulation run based on the values entered in the Run > Simulation Setup > Setup tab; the duration is the period from the start time to the end time.

### Continuous simulation timing

In continuous simulations, the duration is divided into intervals or *steps* of equal length, where start time is the first step and end time is the last step. The length of time, in time units, for each step is known as *delta time* or *dt*. The delta time determines how frequently model data is recalculated. For more information, see “Simulation timing” on page 82.

### Discrete event simulation timing

In discrete event and discrete rate simulations, ExtendSim progresses from start time through end time by a series of events. Time progresses from one event to the next and the time between events is unlikely to be equal. The application calculates the time between events internally based on when an event occurs. The number of steps in a discrete event or discrete rate model is the number of events.

### Simulation order (*continuous models*)

The Run > Simulation Setup > Continuous tab allows you to choose the order in which ExtendSim executes block data for continuous models. The choices are *Flow order* (the default), *Left to right*, and *Custom*.

**⚠** It would be unusual to change the simulation order from the default choice, Flow order.

Since this information pertains to running continuous models, it is discussed at “Simulation order” on page 86.

### Time units

Time is the most common unit of measure in simulation. Each model has its own time unit that is managed in the Simulation Setup dialog.

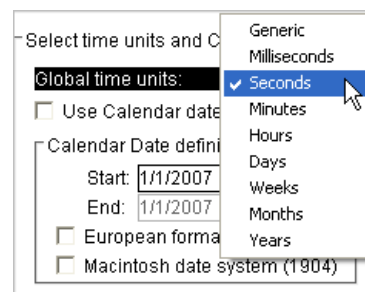
#### Global time unit

The global time unit is the base unit for the entire model, defining the units for the start and end times of the simulation.

There are two ways to specify a global time unit:

- Use the *Generic* time unit to be the default for the whole model.
- Select a specific global time unit, such as hours or seconds, to be the default for the whole model.

Global time units are set by going to the Run > Simulation Setup > Setup tab and selecting and defining the unit of



Global time units popup with Seconds selected

time. Once you select a global time unit, all blocks with time-based parameters, including those subsequently added to the model, use it by default as their local time unit.

**Using the generic global time unit**

By default, the non-specified “Generic” time is a new model’s global time unit. Generic is a conceptual unit of time and has whatever meaning you give it. Generic time is the same for the whole model and for each block that has time-based parameters.

This option is used by most modelers to quickly create a model without having to select a specific time unit. It is also commonly used when the model’s blocks use the same time units as the model.

With the Generic global time unit, time-based blocks do not have a local time unit. Instead, all blocks that include a time-based parameter will specify generic “time units.”

Delay (D):  time units

Block’s time-based parameter when Generic selected as global time unit

You must be sure to maintain all time-based parameters throughout the model in that same unit of conceptual time.

For example, if you have a model that simulates a factory over the course of three hours and your conceptual time unit is minutes, set the end time of the simulation to be 180 in the Simulation Setup dialog and enter parameters based on minutes in block dialogs. The Comments tab in the Simulation Setup dialog is a convenient place for noting what unit of time Generic represents in your model.

 Models cannot use Calendar dates or specify a local time unit if the global time unit is set to Generic.

**Using a specific global time unit**


In the Run > Simulation Setup > Setup tab you can change the default global time unit to be something specific, such as hours or seconds. Using a specific global time unit can add to the understanding of the model, since it allows you to specify parameters based on different time units throughout the model.

When a specific global time unit is selected, that setting becomes the default global time unit for the model as well as the default local time unit within time-based blocks. There will be an asterisk (\*) next to the name of the time unit in the dialog of each block.

Delay (D):

Asterisk indicates time unit in block is the same as global time unit

As discussed in “Local time unit” on page 527, when the model uses a specific global time unit, you can change the local time unit to be anything you want.

 If you change the global time unit in the Simulation Setup dialog, be sure that the new time unit is appropriate for all blocks that use the default time unit. For example, if the global time unit was in hours and you change it to days, a block that used two hours will now use two days if it is set to use the default time unit.

**Local time unit**

You can only change the local time unit in a block if the model is using a specific global time unit, as discussed on page 527. As long as a block’s time unit is in default mode, the local time unit will be the same as the global time unit. For instance, if you change the global time unit in the Simulation Setup dialog, it will also change in every time-based block. However, as soon as you select a different time unit from the popup menu in a block, the local time unit changes from the default to that new time unit.

You can thus choose any unit of time for each time-based parameter in each block in the model. This means you can specify the parameter's time locally as milliseconds, seconds, minutes, hours, days, weeks, months, or years, regardless of the global time unit.

Delay (D):  minutes

Local time unit is minutes; global time is set to a different time unit

- In continuous and discrete event models, a local time unit applies only to its associated time-based parameter. For instance, an Item library block can have two local time units, each associated with a different parameter. In discrete rate models however, the local time unit applies to the entire block. There can only be one local time unit for a discrete rate block.

If a local time unit is specified, ExtendSim will automatically convert parameters based on the local time unit's relationship to the global time unit. If the model is using Calendar date definitions (see "Calendar dates", below) that conversion will be based on the calendar. For example, the actual number of days in the month selected. If the model is not using Calendar dates, the conversion will use the non-calendar definitions entered in the Simulation Setup dialog, as discussed in "Time unit conversions (non-Calendar dates)" on page 529.

Selecting a specific global time unit provides consistency when adding new blocks to the model, since every new block will initially be using the same time unit. However, in some cases you will not want to keep all parameters in default mode. Explicitly selecting a time unit (even the same unit of time as the global time unit) for each local parameter in the model is a safer choice. Then, if you subsequently decide to change the global time unit, you will not accidentally affect the simulation results.

- You can select a time unit from the popup that is not the default but which has the same name as the default global time unit. For instance, you can select *hours* from the popup, rather than *hours\**. You would do that if you do not want the local time unit to change based on a change in the global time unit.

### Calendar dates

Sometimes it is helpful have the duration of simulation expressed in terms of a calendar-based starting and ending time. Calendar dates is a time and date format that corresponds to the calendar and a 24-hour clock. It is an option that you can choose in the Simulation Setup dialog, along with the Start time, End time, and Global time unit entries.

- Calendar dates are not available if the global time unit is Generic, if milliseconds has been selected as the specific global time unit, or if months or years have been selected as the specific global time unit for a discrete rate model. If Calendar dates has been selected, individual Rate library blocks will not be able to select Months or Years as their local time unit.

To enable Calendar dates:

- ▶ Choose Run > Simulation Setup > Setup tab
- ▶ Select **Use Calendar dates**
- ▶ Click the **Start:** field

A calendar appears to select a date and time that represents the beginning of the simulation. ExtendSim will use that information and the other entries you have made in the Setup tab to calculate the *End* date. (Of course, all dates are in simulation time, not in real time.)

There are two other options that affect Calendar date definitions. The *European format* option places the day before the month. The *Macintosh date* option, selected by default for Mac OS, is needed when Excel is set to use the "1904 date system" (see Excel > File > Preferences > Calcula-






tion.) If “1904 date system” is not checked in Excel on the Macintosh, uncheck this option in ExtendSim.

Selecting *Use calendar dates* for a model can also affect blocks that deal with time. For example, if Calendar dates is enabled, you can use the calendar format to create a schedule in the Create block (Item library). An example of this is the block named “Schedule” in the Scheduling Resources model located in the folder \Examples\Discrete Event\Resources and Shifts. (The model is discussed in the section “Scheduling resources” on page 216.)

The following blocks allow you to select Calendar dates in their dialog:

- Create (Item library) when it is set to *Create items by schedule* or *Create values by schedule*
- History and Shift blocks (Item library)
- Lookup Table (Value library) when it is set to *Lookup the: time*
- Plotter blocks such as the Plotter, Discrete Event and Plotter I/O

 Do not use Calendar dates if block dialogs are set to use values smaller than 1 second (such as 0.0005 seconds), as it may affect numerical precision.

### Time unit conversions (non-Calendar dates)

If a model has not been set to use Calendar date definitions when it runs, ExtendSim converts all time-based parameters from their local time units to the specific global time unit based upon constants entered in the Run > Simulation Setup > Setup tab. You can use the default settings or set your own definitions for all the time categories. Specifying that there are 8 hours in a day is an easy way to model a standard 8 hour working day. Note that the default value for a year is 360, not 365.

Non-Calendar Date definitions	
Hours in a day	24
Days in a week	7
Days in a month	30
Days in a year	360

Default constants in the Simulation Setup dialog

For example, assume you are using the default constants and the global time unit for the model is days, but you have selected a local time unit of hours for a block’s time-based parameter. When the model runs, ExtendSim will cause that parameter to be divided by 24, converting it into the appropriate value for one day.

 Time unit conversions are only applicable if the model uses specific global time units and does not use Calendar dates.

## Other Units

### Flow units

Flow units (gallons of liquid, cartons of cereal, rolls of paper, tons of flour, and so forth) can be defined in the Rate library blocks. For more information, see “Units and unit groups” on page 271.

### Length

The Transport and Convey Item blocks (Item library) utilize a unit of distance (feet or meters) to calculate the delay required to move an item from one point to another. Similarly, the Convey Flow block (Rate library) uses a user-defined unit of length to calculate a delay for moving units of flow.

## Length and number of runs

An important consideration when building a model is to determine how long and how often the simulation should be run. Your entries for the *End time* and *Runs* in the Simulation Setup dialog will depend on four factors:

- Whether the system being modeled is *terminating* (has a natural end point) or *non-terminating* (has no obvious end time)
- The period of interest (what portion of time you are modeling)
- Your modeling objectives (estimating performance, exploring alternatives, or other)
- How the samples for statistical analysis are obtained (from running multiple short simulations or by analyzing portions of one very long simulation run)

A brief discussion of terminating and non-terminating systems follows. A comprehensive discussion on this matter is beyond the scope of this manual.

### Terminating systems

Some systems obviously lend themselves to a natural determination of end time. For instance, most service operations have a point at which activities end. In these terminating systems there is an obvious time when no more useful information will be obtained by running the simulation longer. For example, when modeling one day at a walk-in customer service center that is only open 8 hours each day, you could safely set the simulation end time for 8 hours. Since customers would not wait overnight in line for service, the service queue would be empty or cleaned out at the end of the day and further simulation time would be unproductive.

Because terminating systems do not typically reach a continuing steady state, your purpose in modeling them is usually to look for changes and identify trends, rather than obtain system-wide statistical averages. For instance, in the customer service center mentioned above, it is more important to determine the peaks and valleys of activity than to calculate overall averages. Basing your decisions on average utilization in this case could obscure transient problems caused by multiple periods of understaffing.

Since the initial conditions in terminating systems will have an impact on results, it is important that they be both realistic and representative of the actual system. Terminating systems are simulated using multiple runs for short periods of time using different random seeds for each run. As discussed in “Confidence intervals” on page 567, the more frequently a simulation is run, the more confidence you can have in the results.

### Non-terminating systems

A non-terminating system does not have a natural or obvious end time. Models of non-terminating systems are often called *steady-state systems* since, if they are run long enough, the results tend to a steady state. In these situations, simulation runs could go on indefinitely without materially affecting the outcome. Most manufacturing flow systems and some service situations (for example, 24-hour convenience stores, emergency rooms, and telephone service centers) are non-terminating systems.

Systems that have off-shift periods, for instance a manufacturing plant that operates only two shifts a day, may still be considered non-terminating. If the operation does not clear out at the end of the second shift, but instead the first shift starts up where the second shift ended, the system is considered non-terminating and the off shift period is just ignored for modeling purposes.

The important considerations when modeling non-terminating systems involve eliminating the initial bias caused by the warm-up period, deciding how to obtain samples for statistical analysis, and determining the length of the run.

- The *warm-up period* is the period from start-up to when processes operate at their normal or steady-state level. In simulation models, start-up conditions may be unrealistic or nonrepresentative of the actual system and may bias simulation results. To overcome this, you can either wait until after the warm-up period to gather statistics, reset statistics as discussed in “Clear Statistics” on page 566, or run the simulation for a long period of time to “swamp” the biasing effect of the initial conditions.
- To obtain multiple samples for statistical analysis, you could perform repeated runs after eliminating or compensating for the warm-up bias or you could do one extremely long run and calculate statistics on results occurring during various windows of time. As with terminating systems, the greater the number of samples, the higher the confidence in the results.
- The run length of a non-terminating system depends on various factors, including how you obtain samples, your period of interest, and your modeling objectives, as discussed below.

### Determining the length and number of runs

When modeling terminating systems, the length of the simulation run is usually determined by the natural end point of the process being modeled. For instance, the 8 hours that a bank would be open is modeled for 8 simulation hours. For statistical analysis purposes, however, you may want to build a model that looks at a specific time period of a terminating system. For example, you could model the bank’s busiest time period (say from 11 AM to 2 PM) to run multiple times to get a better statistical picture of how the bank operates during that time period.

For non-terminating systems, the length of the run depends on how you decide to obtain your samples (as discussed above) and on your period of interest. Theoretically, a model of a non-terminating system could be run indefinitely. In reality, it is usually simulated until the output reaches an adequate representation of steady-state. For example, you would run a model of a manufacturing operation for a long enough period of time that you feel confident that every type of event happens at least several times. In other situations you might want to limit the run to an artificially short period of time. For instance, you may want to only model the time it takes the manufacturing operation to go from start-up to operating in a normal manner.

The number of simulation runs is determined by statistical sample size calculations and your modeling goals. If your goal is to estimate performance, the number of runs is determined by the required range in a confidence interval. If your goal is to compare alternatives, the number of runs is based on acceptable levels of risk.

For more information on determining the length and number of simulation runs, please refer to a simulation or statistics textbook.

### Speeding up a simulation

The more complex your model, the more important it is to have it run quickly. Models become complex as you add more detail to the workings of each part, or as you run it for longer or, for continuous models only, with smaller delta time increments. Although ExtendSim runs models at extremely high speeds relative to other programs, it can never hurt to think about speed considerations. The following topics discuss some common reasons why a simulation might run slower than it should and how to speed it up.

### Displaying data or movement

The most important thing to remember about running speed is that anything that causes the screen to update will inherently slow down your model. Tips to keep in mind include:

- Only use animation when you need it, since animation will slow your model down more than any other activity.



• Do not keep plotters open when running your model if you are concerned about speed. To keep a plotter closed, select the plotter's dialog tool (shown at right). In the resulting dialog, select either *Show plot at end of simulation* or *Do not show plot*.

- Close dialogs and Notebooks that have parameters updating while the model is running.

### Inefficient settings or block code

In some situations the simulation settings, or a block's code or the settings in its dialog, will cause a model to run slowly.

- *Delta time*. For continuous models, consider using larger delta time increments so that ExtendSim does fewer computations. For example, if each step is a day in your model, consider making each step a week. Of course, this will not work with every model because some calculations are based on the number of steps and will thus be not accurate if you reduce the number of steps.
- *Model profiling*. In some situations a block's code or the settings in its dialog will cause a model to run slowly. For example, you might have 80,000 items arriving in a Queue that is set to prioritize its inputs, or an inefficient section of code in a block you have created. Profiling generates a text file showing the percentage of time that each block spent executing during the simulation run.

To profile a model, select the command Run > Debugging > Profile Block Code, then run the model. A text file is generated with the following information for each block the model uses:

- Block name
- Block Number
- Time spent executing
- Percentage of total time spent executing
- *Display Value block*. The Display Value block (Value library) may cause the simulation to run more slowly. This is because the Display Value block pauses while it shows you information. To speed it up, set its dialog's *Wait* value to 0 ticks or seconds. You can also deselect the *Dialog opens* option if you don't want to see the dialog during the entire model run.
- *Executive block*. The Executive block (Item library) stores information about each item in the model. As the simulation is run, the Executive block allocates additional items in groups of a fixed size when necessary. In the Control tab of the Executive block's dialog, you can specify how many items are allocated at the beginning of the run and the number of additional items allocated when required during the run. The procedure of allocating additional items during the run can slow the simulation down if performed numerous times. Therefore, set the number of items initially allocated to be the maximum number of items that are expected to be in the model at any given time, plus 10%. Note that unnecessarily allocating too many items will take up available memory and can slow down the simulation.

### Other factors that affect simulation speed

- Increase the amount of physical memory (RAM) in your computer. If your computer runs out of RAM, it will use virtual memory, which can slow down your model significantly.
- Using a large number of Value library blocks connected together to perform a simple calculation can slow a model down. Where possible, replace large “webs” of blocks with equation blocks that perform the same calculation.
- For discrete event models, you may be able to scale the number of items you generate. For example, if your model is of a factory floor, instead of generating one item for each object manufactured, you might generate one item for each set of five objects.

### Slowing down simulations

You might want a model to run more slowly to debug it or critically visualize what the model is doing. If this is the case, the best way to slow down your model is to turn on animation. If the model is still running too fast, click the Animation Slower (turtle) button in the toolbar: this causes animated models to run even slower. To speed up the model again, choose the Animation Faster (rabbit) button.

Another example of when you might want to slow down a model is if you want to synchronize the model to real-time events. This might be required if you are receiving real-world data. To accomplish this, refer to the TimeSync block (Utilities library).


If you set it to leave the dialog open while the simulation runs, the Display Value block (Value library) is also useful for slowing down the simulation.

See also “Stepping through a model” on page 522.

### Working with multiple models

ExtendSim can have multiple model windows open and run multiple models simultaneously. This facilitates copying model sections or hierarchical blocks from between models and running models that communicate with each other. You can also run one model in the background while constructing another.

The Run Simulation command only works on the active model. To run multiple models, start the front model running. Then bring each model forward and start their runs. While some models are running, other models can be built.

 The Prioritize Front Model command causes the frontmost model to have processor preference over the background models.

### How ExtendSim passes messages in models

ExtendSim uses a sophisticated messaging architecture to signal blocks into action. While messages can originate either from the ExtendSim application or from individual blocks, it is always a block that is on the receiving end of a message. Different types of messages result in the receiving block doing different types of things.

Since messages have the potential to affect the speed and behavior of simulations, understanding the ExtendSim messaging architecture will help you build more accurate and efficient models and will make debugging models easier.

Model messages can be divided into two categories:

- 1) Messages typically sent from the application to one or more blocks

2) Messages typically sent between blocks.

☞ All types of blocks can receive application messages. Continuous blocks neither receive nor send block messages.

### Application messages

Application messages are usually sent to blocks by the ExtendSim application. There are several types of application messages: Simulation, Model Status, Block Status, Dialog, Connector, Database, OLE, and 3D Animation. They are completely discussed in the Developer Reference.

The major application messages are:

- CheckData
- StepSize
- InitSim
- Simulate
- FinalCalc
- EndSim

The application can send message before, during, and after the simulation run, including:

- At the start of the simulation
- At each step
- When a parameter is changed
- When a dialog is opened
- If a button is clicked
- At the end of the simulation

At the start of the model run, all blocks receive application messages to check their dialogs and connections for consistency and to initialize variables, import data, or allocate any arrays used during the simulation. At the end of the simulation run, blocks are triggered to collect statistics, dispose of any temporary arrays, and update the status of the model. Other application messages are sent only when a specific event, parameter change, or dialog click occurs. The Developer Reference has a complete list of application messages, including CheckData, InitSim, Simulate, EndSim and FinalCalc.

### Link alerts

Link alerts are a special type of application message. If a block is linked to a data source in a global array or ExtendSim database, it will receive a message whenever the value in that data source changes. That message may cause changes in the block's data. The application sends link alert messages in continuous, discrete event, and discrete rate models. In a discrete event or discrete rate model, the alerted block may, in turn, send messages out its connectors.

Managing this message is important because linking blocks to a data source that frequently changes may slow down the simulation. If a simulation model is running slowly, and blocks have links to data sources, consider sending or receiving link alert messages only at the start or end of the simulation. This can be done by modifying the properties of the link so it only sends messages at the appropriate times. (See "Link dialog checkboxes" on page 634).

**Continuous model messaging**

In a continuous model, the most important application message is the *simulate* message that causes code in blocks to execute as simulation time advances. At each step of the simulation, the application sends a simulate message to every block in the model in the sequence defined by the model's simulation order. (Simulation order is determined by the application at the start of the run based on settings in the Simulation Setup dialog, as discussed in “Continuous tab” on page 518.) In the simulate message handler, each block will perform its intended calculation. For example, if *Add* is selected in the Math block (Value library), the block will add all its inputs when it gets a simulate message. Likewise, a Decision block (Value library) will check its inputs and output a true or false value based on the options selected in its dialog.


**Block messages**

Blocks can send messages to each other either through connections or “through the air.” This often happens when one block has calculated a new value and wishes to alert one or more blocks to this change. Continuous blocks, such as those in the Value library, do not typically send or receive block messages. Discrete event (Item library) and discrete rate (Rate library) blocks often send and receive block messages.

**Discrete event block messaging**

In addition to receiving application messages, discrete event blocks communicate with each other using block messages. A variety of messages are sent between blocks during the course of the simulation. These block messages fall into the following categories:


- *Event*. Communication between the Executive and various Item library blocks in the model.
- *Value connector*. Notifies connected blocks that the value of a output connector has changed or requests updated input connector information.
- *Item connector*. Propels items through the model.
- *Block-to-block*. Updates the status of other blocks in the model.

 To avoid modeling errors and simulation speed issues in discrete event models, it is important to understand how the discrete event messaging architecture works. See “Messaging in discrete event models” on page 260 for additional information.

**Discrete rate block messaging**

In addition to receiving application messages, discrete rate blocks intensively use messages in order to initiate the calculation of the effective rates and to propagate information through the blocks. These block messages fall into the following categories:

- *Event*. Communication between the Executive block and various Rate library blocks.
- *Value connector*. Notifies connected blocks that the value of a output connector has changed or requests updated input connector information.
- *Flow connector*. Updates the effective rate through the connected blocks.
- *Rate blocks flow*. Defines the LP area – that part of the model that could be impacted by a change in the effective inflow and outflow rates.
- *Executive flow*. Updates all the blocks in an LP area with a new effective rate.

 The discrete rate technology has a complex and unique messaging architecture. For optimal performance, it is important to understand discrete rate messaging when building rate-based models. For more information, see “Block messages” on page 387.





# How To

## Presentation

Organize and enhance models  
so they communicate your ideas

*“In the modern world of business,  
it is useless to be a creative original thinker  
unless you can also sell what you create.  
Management cannot be expected to recognize a good idea  
unless it is presented to them by a good salesman.”  
— David M. Ogilvy*

Simple models, like the Reservoir shown in the Tutorial, are easy to follow. But as models become larger and more complex, worksheets can become crowded with thousands of blocks, making it harder to communicate what's happening. ExtendSim provides several features that let you organize, explore, and enhance models, such as:

- Using text and graphics to improve model appearance
- Using the Navigator to quickly locate important parts of the model
- Organizing the model into logical sections with hierarchy
- Adding 2D animation to visually represent the model's logic and behavior
- Centralizing model inputs and outputs in an ExtendSim database
- Changing how connection lines are displayed
- Simplifying model appearance by hiding connections and connectors

For additional ideas to enhance a model's appearance, see the chapter "Creating a Custom User Interface" on page 503.

 3D animation is covered in the module that starts on page 389.

## Working with text

Text is used to document a model, create text labels for named connections, and to create text connections for hierarchical blocks.

### Entering text


To add text to your model, double-click the worksheet or Notebook where you want the text to appear and a text box will appear. You can now type your text. You can use the handles in the corner of the text box to resize it.

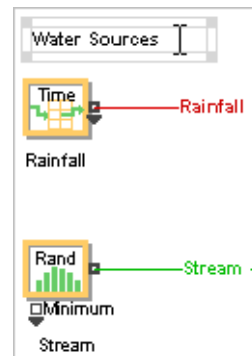
When you've finished entering text, click anywhere else in the window and the box will disappear. To edit an existing text box, select the Block/Text layer tool and double-click the text. (You can also use the Clone layer or All layers tool for this, but using the Draw layer tool will open a new text box.)

### Moving and copying text

To move the text while you're entering or editing it, move your cursor over the border (away from a handle) until it changes to a hand shape, then click and drag the text box to the desired location. If the box is not active, simply move the cursor over the text itself until the cursor changes to a hand, then click and drag.

When the text is highlighted, you can use the Copy and Paste commands, then move the new text where you want it (or click on the window before pasting to position it in that spot). To copy just portions of the text, double-click to make the text box active, then select the text you want. You can also use the Edit > Duplicate command or Ctrl+D to copy and paste model elements slightly offset from the original.

 When you paste text onto the model worksheet or into another text box, it remains editable text. However, if you select and copy an entire text box and paste it to the Notebook, it becomes a picture. To paste editable text into a Notebook, copy the *contents* of the text box, not the text box itself.



Adding text to a model

### Drag and drop text


It is often easier to drag and drop a piece of text than to copy and paste it. To do this, select a piece of text and drag it. When you drag the selection, an insertion point appears at the end of the cursor. Drag the insertion point to the desired location and release the mouse button.

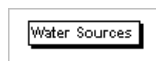
Drag and drop can be used within a text box on a worksheet, a text file, a block's structure window, and between any combination of the above with one exception: it will not work between two separate text boxes. For example, if you double-click on the worksheet to open a text box, enter text into the text box, then select a portion of the text, you will not be able to move that text to another text box using drag and drop. This is because on the model worksheet, you can have only one text box open for editing at a time.

### Formatting text

To type new text with a particular format, select the desired format *before* you start the text box. ExtendSim will remember that format every time you start new text.

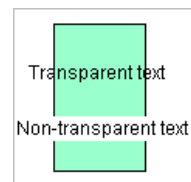
To format existing text, access the text box, select the text, and then choose a command from the Text menu, such as Bold or Justify Center. You can also change the color of text using the Color palette. To do this, select the text in the box then click one of the colored squares in the palette as described in "Patterns and colors" on page 562.

 If you change the format of text within an existing text box, ExtendSim won't use that format as the default for the model. To cause a format to be the default, change it before creating a text box.



The *Border* command adds a border with a drop shadow to the text. This can be a useful way to distinguish descriptive text from block labels or named connections. The text example on the left has a border.

With the *Transparent* command checked, text on top of another object won't show its background color. With the command unchecked, a white rectangle behind the text will overlap other objects. The examples are shown at the right.



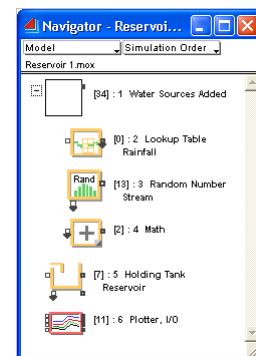
### Navigator

The Navigator is an explorer-like window that can be used for multiple purposes:

- Navigation through a model's hierarchical structure and quick access to block dialogs in large models
- Accessing any databases used in the model
- Adding blocks to the model worksheet, as an alternative to getting them from the Library menu

The Navigator is especially useful when presenting a model to others. Use it to explore the model by drilling down through hierarchical layers to show subsystems or to quickly find blocks and access their dialogs, no matter where they are in the model. See "Navigating through the Reservoir model" on page 38 for an example of using the Navigator to explore a model.

Since it has multiple uses, the Navigator is discussed fully in the How To: Miscellaneous chapter. For more information, see "Navigator" on page 670.




Navigator with hierarchical block expanded

How To

## Hierarchy

As you saw in “Introduction to hierarchy” on page 36, ExtendSim’s hierarchical capability lets you combine basic modeling constructs (such as a group of connected blocks) into a single, higher-level construct, then combine several of those new constructs into an even higher-level construct, and so forth. This process causes portions of a model, or even the entire model, to be implemented in layers, where you can drill down from the highest system level to the smallest detail. For example, the Santee River Basin model on page 9 was created using many layers of hierarchy; without hierarchy the model would be overwhelmingly complex.

The mechanism for creating hierarchical layers in ExtendSim is by building a hierarchical block, a special type of block that usually has a group of blocks nested inside it. These blocks represent a portion of the model, a *subsystem* or *submodel*. Using hierarchical blocks you can nest subsystems in an unlimited number of layers for top-down or bottom-up modeling.

 Top-down modeling is starting with a new hierarchical block and building a model or a series of submodels in it. Bottom-up modeling involves selecting a group of blocks within a model and making them into a hierarchical block. You can create multiple hierarchical layers from the top down or from the bottom up.

### Uses for hierarchy

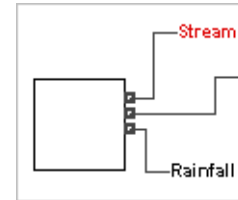
Although you do not have to use them as you build models, hierarchical blocks can help organize models logically, make models appear more attractive, improve productivity, and enhance comprehension.

- Simplify a complex model by grouping areas of the model into hierarchical blocks containing submodels. These submodels can then be reused in other models without having to reproduce all of the connections.
- Instead of showing all the detail, present a model as a few simple but logical steps. To reveal the subsystems within a step, just double-click the hierarchical block.
- To increase productivity and comprehension, create a hierarchical block that represents a process element, then use it in several models that have that element in common. By changing the parameters, each instance of the hierarchical block can be customized for its model.
- When developing a new model, go from the simplest assumptions to more complex ones by creating more and more levels of hierarchy. This helps structure your thinking and makes models easier to follow as they become more complex.
- Although hierarchical blocks usually contain submodels of blocks, they don’t have to. They can also be useful for enhancing and documenting the model, serving as popup windows that reveal pictures, text, and so forth when double-clicked. For an example, see “Help block” on page 514.

## Hierarchical blocks

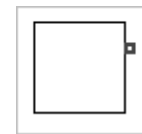
There are two ways to create a hierarchical block:

- 1) Select and encapsulate a group of blocks into a new hierarchical block. This is most often used to organize the complexity of an existing model and is discussed at “Making a selection into a hierarchical block” on page 542.
- 2) Create a new hierarchical block from scratch and build a submodel within it. This is used for top-down modeling as discussed in “Building a new hierarchical block” on page 543.



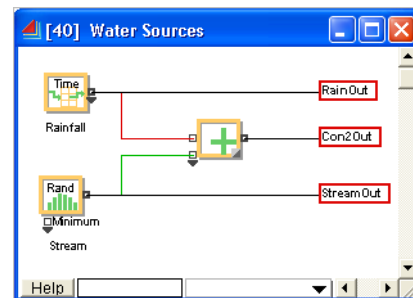
Hierarchical block adds water sources together

In a model, a hierarchical block is represented by its own icon. By default, the icon is a blank white square but you can use drawing tools or paste in a picture to replace the icon. To help identify them in a model, hierarchical blocks can have drop shadows, as determined by a setting in the command Edit > Options > Model tab.



Hierarchical block default icon

You open a hierarchical block by double-clicking it. Instead of seeing a dialog inside, you see the blocks that make up the submodel. The blocks are laid out like a model, with connections, labels, and so on. Depending on how the hierarchical block was created, it might also contain text, pictures, or cloned dialog items that control the blocks within the hierarchical block.



Hierarchical block open

The submodel’s connections with the rest of the model are shown as *connection text boxes* (named connections with red borders around them). In the example to the right, *RainOut* is one of the outputs of the hierarchical block Water Sources.

Hierarchical blocks have two kinds of windows:

- The *submodel pane window*. This is what is displayed if you double-click a hierarchical block. Use this window to add blocks to a model, adjust block and connection line positioning, rename connection text boxes, or add text and graphics to the submodel.
- The *structure window*. This is displayed if you create a new hierarchical block by giving the command Model > New Hierarchical Block or if you select an existing hierarchical block and give the command Model > Open Hierarchical Block Structure. The structure window contains another view of the submodel pane and additional panes for modifying the hierarchical block’s icon, adding Help, and so forth. It is shown on page 544.





### Characteristics of hierarchical blocks

Hierarchical blocks are unique; they have some characteristics of a block and some characteristics of a model worksheet. All hierarchical blocks have the following in common:

- A hierarchical block can contain no blocks, one block, a group of blocks, and other hierarchical blocks. It can also contain text, graphics, cloned dialog items, and pictures.

- Hierarchical blocks can be copied to other areas of the model or to other models and used just like other blocks.
- The settings within a hierarchical block can be changed by double-clicking the hierarchical block, then double-clicking the desired block's icon, or by changing the desired block's cloned dialog items.
- Hierarchical submodel windows are similar to model windows: add blocks from a library, create hierarchical blocks, add graphics, type labels and other text, clone dialog items onto them, and so forth.
- The components of a hierarchical block (icon, connectors, or Help text) can be changed by accessing the hierarchical block's structure window.
- Unlike other blocks, by default hierarchical blocks are saved directly in the model as copies. This characteristic allows them to be treated much like a copy of a portion of the model. You can copy a hierarchical block to another part of your model and make changes to its submodel window without affecting the original hierarchical block. This is known as *physical hierarchy*.
- You can choose to save a hierarchical block in a library, in which case it can be treated much like a regular block. When you make changes to the hierarchical block's structure window and you also choose to update all instances of that block, it is known as *pure hierarchy*.

#### **Important notes about hierarchical blocks**

-  The action of creating a hierarchical block cannot be undone. If you accidentally include more blocks than intended in the hierarchical block, remove them from the block's submodel and put them back into the model using the Cut and Paste commands in the Edit menu.
-  Named connections (discussed on page 560) only connect within one level in a hierarchical model. This means that the data will not flow between levels if you have a named connection on one level and a corresponding named connection in a block at a lower or higher level. To connect between hierarchical levels, use the Throw and Catch blocks in the Value, Item, or Rate libraries, or add connectors to hierarchical blocks. To see how Throw and Catch blocks are used with hierarchy, see "Throw Item and Catch Item blocks for merging item streams" on page 148. For more information on adding connectors to hierarchical blocks, see "Step 4: Add connectors" on page 545.
-  By default a hierarchical block is saved only in the model. To save it in a library, use the Save Block As command, described later in this topic.
-  Because they can contain entire submodels, hierarchical blocks can become quite large. Libraries have a 15MB limit, so you may need to use more than one library to store all the hierarchical blocks for a model.

#### **Making a selection into a hierarchical block**

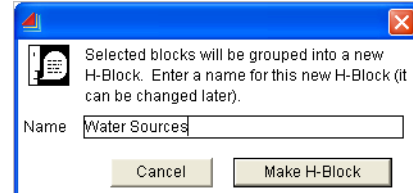
The Tutorial on page 37 showed how to create a hierarchical block named "Water Sources" that grouped three blocks from the Reservoir model. The steps are:

- ▶ Open an existing model or place blocks on a new model.
- ▶ Select some blocks and any desired draw objects by dragging over them or by holding down the Shift key and clicking each item. (Remember, the selection tool used determines what gets selected.)

 Do not select the text labels of any named connections, otherwise communication between the submodel and the model outside will break.

- ▶ Choose the command Model > Make Selection Hierarchical.
- ▶ Enter a descriptive name for the hierarchical block and click Make H-block.

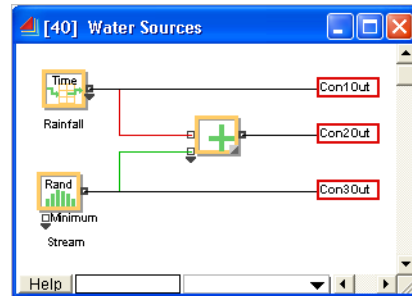
When it creates the hierarchical block, ExtendSim makes all the connections and replaces the selected blocks in the model with the new hierarchical block and its default icon, a square.



Make Selection Hierarchical dialog

- ▶ Double-click the new icon to see the submodel.

Hierarchical blocks can be saved in libraries or just in the model, as discussed at “Saving hierarchical blocks” on page 547. You can also alter aspects of the hierarchical block such as adding blocks, moving or renaming connections, or adding art to the icon; see “Modifying hierarchical blocks” on page 548 for more details.



Open Hierarchical block

### Building a new hierarchical block

The steps in building a new hierarchical block are:

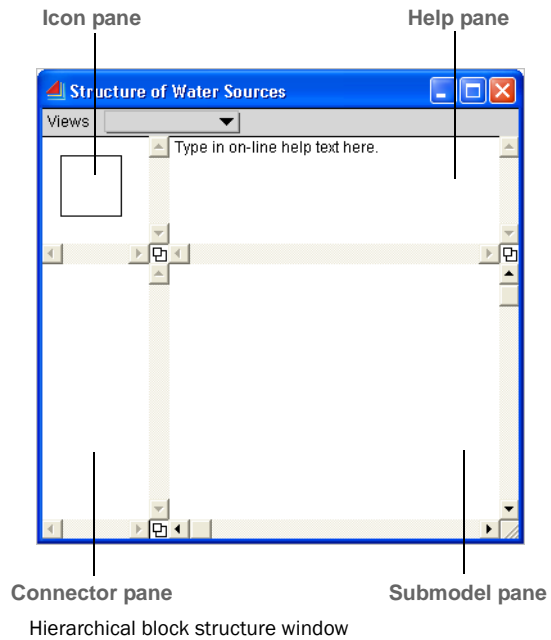
- 1) Open a structure window for the new hierarchical block.
- 2) Build the model in the submodel pane.
- 3) Modify the icon (optional).
- 4) Add connectors to the block's icon.
- 5) Connect the new hierarchical block to the other blocks in the model.

The following example creates a water sources submodel within a new hierarchical block.

#### Step 1: Open a structure window for the hierarchical block

- ▶ Open an existing model or open a new model window.
- ▶ Give the command Model > New Hierarchical Block.

ExtendSim prompts for a name for the hierarchical block, then opens its structure window.



The hierarchical structure window is divided into four *panes*, which are work areas. The upper left pane is the *icon* pane for designing the block's icon and adding connectors. The upper right pane is the *Help* pane for writing the Help text for this block. The lower left pane is the *connector* pane which contains the names of the input and output connectors for the block; it can also be used to rename the connectors. The large pane on the lower right is the *submodel* pane, where the submodel for this hierarchical block will be built. The submodel pane is what appears when you double-click a hierarchical block's icon.

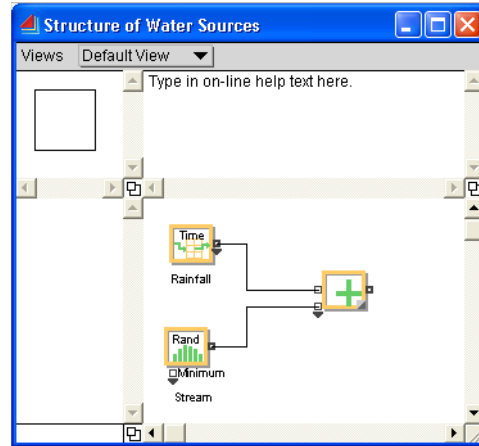


**Step 2: Build the submodel**

To build a submodel in a hierarchical block, add blocks to the submodel pane using the same methods used to build the Reservoir model in the Tutorial on page 24. For instance, use the Library menu to add blocks to the submodel pane (or drag them from library windows) and connect blocks in the normal fashion.

If you prefer, use the Copy and Paste commands from the Edit menu to copy a portion of an existing model into the submodel pane.

For example, the Water Sources hierarchical block in the example at right contains three connected blocks from the Value library: a Lookup Table block labeled Rainfall, a Random Number block labeled Stream, and a Math block set to add its inputs.



Hierarchical block structure window with submodel added

**Step 3: Modify the icon**

The hierarchical block starts with a default icon, a white square. You can modify this icon, for example by changing its shape or color, or you can delete it and create a new one. To do this, draw an icon with the drawing tools in the toolbar, or paste a picture from another program. For information on using the drawing tools, see “Graphic shapes, tools, and commands” on page 561.

**Step 4: Add connectors**

Since the model within a hierarchical block needs to be connected to the outside world, you must add the appropriate connectors. Some hierarchical blocks have both input and output connectors, while others have just one kind. Also, as is true for other blocks, hierarchical blocks use specific types of connectors (value, item, flow, etc.) depending on what they are connected to.

The steps for adding a connector to a hierarchical block are:

- 1) Decide the type of connector to add.
- 2) Add the connector.
- 3) Determine if the connector should be an input or an output connector.
- 4) Connect the blocks in the submodel pane to the connector’s text label.

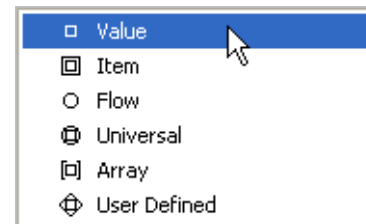
*Choosing a connector type*

When the hierarchical block’s structure window is the active window, the Icon Tools popup button at the right side of the ExtendSim toolbar is available.

The first six items add connectors: Value, Item, Flow, Universal, Array, and User defined. (Connector types are discussed at “Connector types” on page 498.)

*Adding a connector*

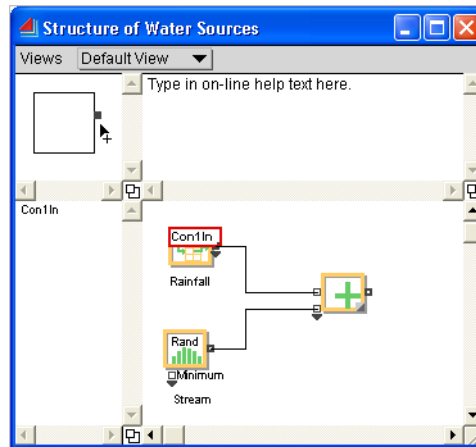
- ▶ To add a connector, select the appropriate type of connector from the Icon Tools popup menu in the toolbar. For this example, select the Value connector.



Portion of the Icon Tools menu

- ☞ The connector type must match the input or output connectors of the block in the submodel pane.
  - ▶ Click in the icon pane at the desired position near the edge of the hierarchical block's icon. For this example, click on the right side of the icon.

This creates the default *Con1In* connector on the icon in the icon pane, lists it in the connector pane, and adds a red-bordered connector text label (similar to a named connection) in the submodel pane.



Value connector added to hierarchical block

- ☞ If you choose the wrong type of connector, just click on that connector on the icon pane to select it, then select the correct connector type from the Icon Tools popup menu. The connector will change to the new type.

*Changing to an output connector*

By default, when a connector is first added to an icon it is an input connector, as indicated by the word *In* at the end of its name. To cause it to be an output connector, the connector name needs to end in **Out**. And, since *Con1* isn't very descriptive, change the connector name to something more relevant, in this case "Water". To do this:

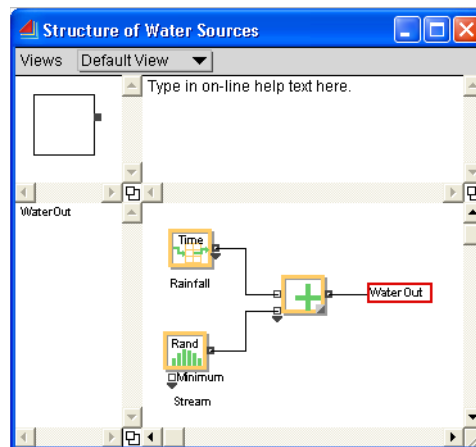
- ▶ Select the connector name in the connector pane (or double-click its text label in the submodel pane) and change its name to *WaterOut*, then hit Enter.

- ☞ You can name a connector anything you want as long as the name ends in *In* or *Out* and is 32 characters or less. Also, every connector must have a unique name.

*Connecting to the submodel*

The next step is to connect the connector text label to the appropriate block:

- ▶ In the submodel pane, move the *WaterOut* text label to the right of the Math block.
- ▶ Drag a line from the output of the Math block to the *WaterOut* text. Once the line thickens, release the mouse and the connection is made.



Connector changed to output connector and added to Math block

- ☞ The Stomach, Absorption, and Bloodstream blocks in the Drug Ingestion model are examples of hierarchical blocks with connectors. The model is located at \Examples\Continuous\Standard Block Models and is discussed on page page 73.

How To

**Step 5: Connect the block in the model**

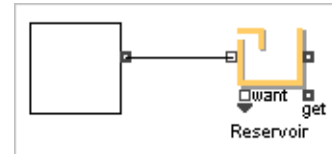
To connect the finished hierarchical block to the rest of the model:

- ▶ Close the structure window.

When you close the structure window of a new hierarchical block, a dialog gives options to save the block, discard changes, or cancel. These options are discussed in the following topic. For this example:

- ▶ Choose **Save to this block only**.

- ▶ Connect the connector on the hierarchical block to other connectors in the model, just as you would any other block. In the example, the hierarchical block is connected to a Holding Tank block.



Hierarchical block connected to Holding Tank

Note that there is only one output from this hierarchical block and it represents the total amount of water from the two sources. You could also add connectors for the individual Rainfall and Stream outputs so this block would more closely resemble the one created in the Tutorial.

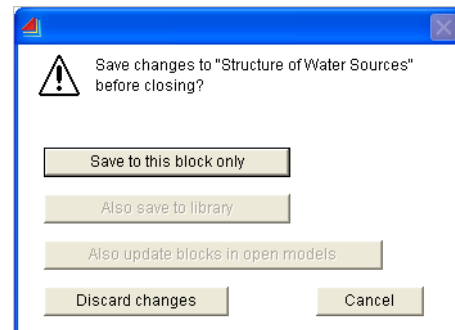
**Saving hierarchical blocks**

By default, hierarchical blocks are saved with the model. You can also save a hierarchical block to a library, which makes it easier to add them to new models.

**Saving hierarchical blocks in a model**

When you make a selection hierarchical, the hierarchical block is automatically saved with the model when the model is saved.

If you make a new hierarchical block, or make changes to the submodel pane window of an existing hierarchical block and then click the Close button in the structure window, the options are to *Save to this block only*, *Discard changes*, or *Cancel*. This dialog does not enable the options for saving the block to a library; if you choose save, the block exists only in the model.



Saving hierarchical block to model only

You can copy such a hierarchical block to other parts of the model or even to other models using the Clipboard. Each instance of the block in the model can be then be made unique by modifying it as described in “Modifying hierarchical blocks” on page 548. Because these hierarchical blocks have not been saved in a library, when one of them is changed, the other blocks do not change.


**Saving hierarchical blocks to a library**

You can save a hierarchical block in a library, but what is saved in the library is only a “snapshot” of the hierarchical block at the time you saved it.

- ☞ If you later modify that hierarchical block on the model worksheet, the changes will not be reflected in the master block in the library unless you specifically tell ExtendSim to save those changes.

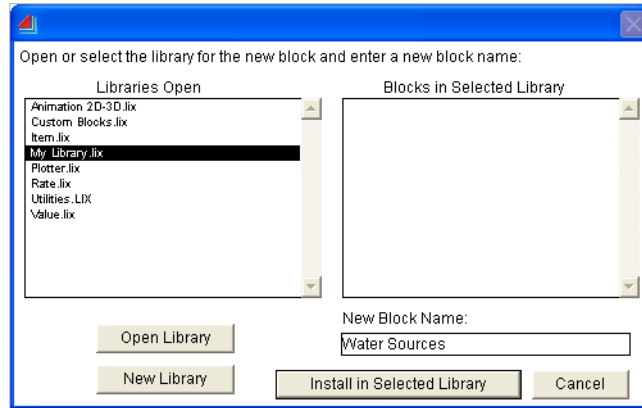
To save a hierarchical block in a library or to cause changes made to a hierarchical block to be reflected in its master block in a library:

- ▶ If it is not already open, open the structure window of a hierarchical block.
- ▶ Choose File > Save Block to Library as... In that dialog, select a new or existing library for the hierarchical block and click **Install in Selected Library**.

 Do not save hierarchical blocks in the libraries that come with ExtendSim.

- ▶ Close the hierarchical block's structure window and choose one of the save options:

- *Save to this block only* affects only this specific instance of the block in the model.
- *Also save to library* affects this instance of the block, the master block in the library, and blocks placed from the library into the active model after the save.
- *Also update blocks in open models* also saves the changes to instances of this hierarchical block in any *open* models.
- *Discard changes*
- *Cancel*



Save Block to Library as... dialog

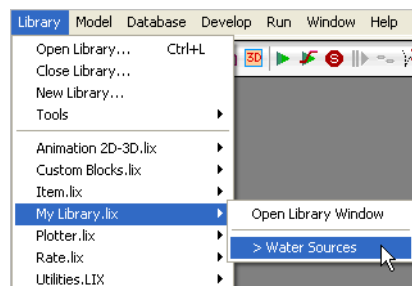
These options are also discussed in “Summary of results of modifying hierarchical blocks” on page 550.

A hierarchical block that is saved in a library has its name listed in the library preceded by the symbol >. Like other blocks, hierarchical blocks that are saved in libraries list the name of the library in their title bar. If no library is listed, the hierarchical block is not saved in a library. You can also get library information by selecting the block in the model window and choosing the command File > Properties.

### Modifying hierarchical blocks

How you modify a hierarchical block depends on whether the block is saved in a library and the type of modification you want to make:

- If you don't want to save the changes to the master block in the library, simply double-click the hierarchical block's icon and change the settings for individual blocks or modify the layout and appearance of the submodel (e.g. add blocks, clone items onto the submodel window, add text and drawings, etc.).



Hierarchical block in Library menu

- To save the changes to the master hierarchical block in a library, you must open the hierarchical structure window. (You must also use the structure window to make any changes to the hierarchical block's icon, connectors, or Help, even if you don't want to save these changes to the library.)

The following indicates where to make changes to a hierarchical block:

	Submodel Window	Structure Window
Add blocks and connections	X	X
Change parameters in submodel blocks	X	X
Add text, drawing objects, and pictures	X	X
Clone dialog items	X	X
Change icon, add alternate views		X
Rename the block		X
Add Help		X
Add animation		X

### Changing the icon

To change the icon to better suit the purpose of the hierarchical block:

- ▶ Open the hierarchical block's structure window.
- ▶ Click the icon in the icon pane.
- ▶ Change the size, color or pattern of the icon, or delete it and create another icon.

To create a new icon, use any of the drawing tools or paste pictures from outside of ExtendSim into the icon pane. See "Graphic shapes, tools, and commands" on page 561, for information on how to use the drawing tools.

 You need to delete the icon before creating a new one, but do not delete any connectors. If you do, ExtendSim will warn you. If you accidentally delete a connector, undo the delete or add another connector. Be sure to check the block's connections in the model.

- ▶ Optionally add alternate views using the Views menu at the top of the icon pane. An example of this is the Markov Chain Weather model discussed on page 50. See "Icon views" on page 496 for more information.

### Moving connectors

The icon pane has an invisible snap grid and the upper left-hand corner of the connector will automatically snap to the grid as it is moved around the icon. To move a connector without using the grid, hold down the Alt (Windows) or Option (Mac OS) key when moving the connector. Use the Model > Align command to align two or more selected connectors.

### Renaming the block

With the structure window of a hierarchical block open, rename it by choosing Develop > Rename Block.

### Adding animation

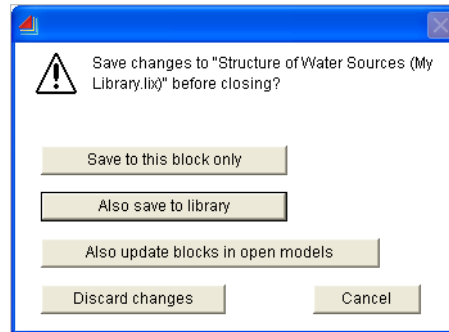
Hierarchical block icons can be animated, as discussed on page 554.

### Summary of results of modifying hierarchical blocks


There are different results when a hierarchical block is modified, depending on whether its sub-model window or structure window was modified and whether the block is saved just in a model or is also saved in a library:

- If you modify a hierarchical block's submodel in the submodel pane window, those changes only apply to that block. Changing a hierarchical block's submodel is similar to changing parameters in a regular block's dialog: the changes affect only that instance of the block on the worksheet and are saved with the model. This is true even for hierarchical blocks that were originally saved in libraries.
- If you modify the structure of a hierarchical block that is not saved in a library, those changes only apply to that block. This has the same result as modifying a hierarchical block's submodel pane window. For example, you can make several copies of that hierarchical block in a model, but when you change one of the copies, the other blocks remain unchanged.
- If you modify the structure of a hierarchical block that has been saved in a library, you can choose how those changes should be reflected:

- Only to this instance of the block on the model worksheet (choose *Save to this block only*).
- Also in the master block in the library, which only affects blocks placed in the model from the library *after* the change has been made (choose *Also save to library*).
- Also in all instances of the block in *open* models (note that this does not affect models that are not open at the time); this is also called pure hierarchy (choose *Also update blocks in open models*).





Saving modified hierarchical block options

 Changing a hierarchical block that is saved in a library can have unintended consequences if the block is used in multiple places. Do not make changes to the master block in the library unless you understand the effect those changes could have on other models that use that block.

- To modify the structure of a hierarchical block saved in a library, copy a fresh hierarchical block from its library window onto the model worksheet, then work on the block directly in the model. Do not work on a hierarchical block that is already in the model or you might overwrite the master hierarchical block with a block that has been customized for a specific purpose in the model.
- If you change the hierarchical block's submodel, rename the block and save it to the library using the new name. This will prevent existing instances of that hierarchical block from being changed.


- Use caution if you change a parameter in a block in a submodel, then save the hierarchical block to the library with the “Also **update blocks in open models**” choice. In that case, each instance of the hierarchical block will have the changed parameter value.

 Changes made to the structure of a hierarchical block that is saved in a library will not be reflected in models that were not open when the change was made. In that case, open the model and replace its hierarchical block with a new one from the library.

 The Constant block (Value library) has an option to *Retain constant if updated from hierarchy*. If that option is checked, the Constant will retain its value when the enclosing hierarchical block is updated from a library and existing instantiations of the block will not be changed inside of existing hierarchical blocks. This is useful if each hierarchical block needs a unique identifier which does not reset if the hierarchical block is updated from a library.

## Animation

When presenting models to others you can show how plotter values or cloned outputs change as the simulation progresses. However, ExtendSim’s animation capabilities give a more visually descriptive representation of what is happening in the model.

 This section of the User Guide discusses ExtendSim’s 2D animation capabilities. The ExtendSim Suite product allows simulations to be run with 3D animation – see the E3D module of this manual for more information.

Models can be animated using:


- Blocks that have built-in animation
  - To animate a block’s icon
  - To animate items traveling in discrete event models
- Specialized blocks for creating customized animation
- Animation functions programmed into blocks you create

To see animation in a model:

- ▶ Choose Run > Show 2D Animation. The command is now checked.
- ▶ Run the simulation.

Due to redrawing, animation tends to slow the simulation. You may want to leave animation on while you are testing, debugging, or presenting your model and then turn it off when you are running the model for quicker results.

Use the Animation Faster (rabbit) and Animation Slower (turtle) buttons to change the speed of the animation.

 If animation is running at its slowest speed, the Animation Slower button will be grayed out. Likewise, if the animation is already running at the highest speed, the Animation Faster button will not be available.



Animation buttons  
Left: Faster  
Right: Slower

### Blocks with built-in animation

Many of the ExtendSim blocks have animation built into them. A block’s online Help says whether it is animated and, if so, which aspects are. The two types of built-in animation are:

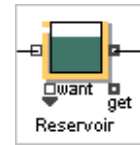
- Animation of a block’s icon
- Animation of items traveling from block to block (discrete event models only)

### Animation on a block's icon

Some ExtendSim blocks show animation on their icons. The amount and type of animation is specific to the block and is described in the block's Help. For example, the Holding Tank block (Value library) shows its contents relative to a minimum and maximum, the Select Item Out (Item library) shows which path items take, and the Convey Flow (Rate library) displays the distribution of its contents as the simulation runs.

- ▶ Open the Reservoir 1 model (ExtendSim7\Examples\Tutorial folder.)
- ▶ Select Run > Show 2D Animation.
- ▶ Run the simulation.

The icon for the Holding Tank shows the level of contents changing as the simulation runs.



Holding Tank icon after animation

By default the option *Automatically set max and min animation* is selected in the Holding Tank's Animation tab. This causes the level to go from 0 to an estimate of the maximum value in the first simulation run; later runs will use an average of the preceding runs' maximum as the level's maximum value. Unselect this option to set your own maximum and minimum levels.

### Animating the movement of items between blocks (discrete event modeling only)

The blocks in the Item library can track the movement of items with *animation pictures* that travel along connections between blocks.

To animate the flow of items along connection lines in a discrete event model:

- ▶ Select the Run > Show 2D Animation command.
- ▶ Choose the command Run > Add Connection Line Animation.
- ▶ Run the simulation.

The Final Car Wash model, located in the folder \Examples\Tutorials\Discrete Event, shows cars moving between blocks and along connection lines.

To see animation pictures also move along the paths of named connections:

- ▶ Select the command Run > Add Named Connection Animation when Show 2D Animation is checked.

#### Selecting an animation picture

When Show 2D Animation is enabled in a discrete event model, pictures representing individual items pass from block to block. Initially, all items are represented by the default animation picture (a green circle). You can cause a block to use any other animation picture included with ExtendSim, existing clip art, or pictures created in an external drawing package. The selection is made in each block's Item Animation tab.

 To find out how to add your own animation pictures, see "Animation pictures" on page 556

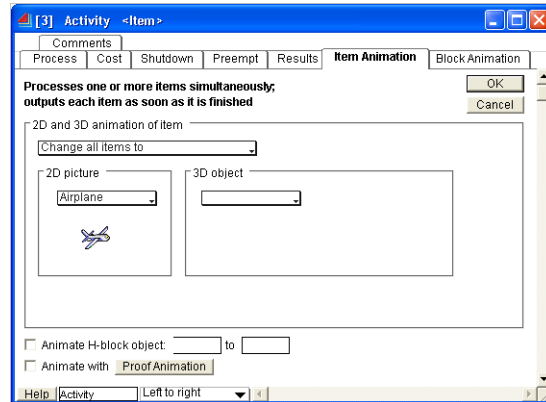


For example, the screenshot at right shows the Item Animation tab from an Activity block (Item library). There are three options for 2D and 3D animation of item:

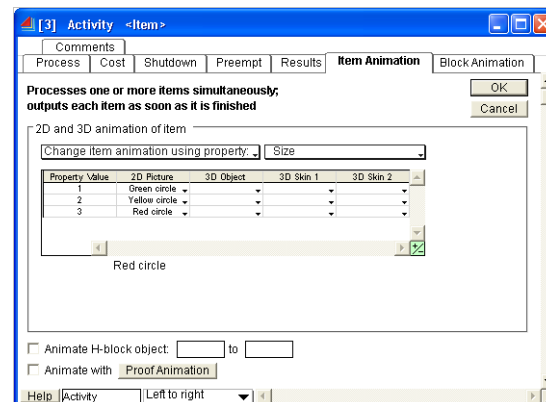
- Do not change item animation. In this mode, items that leave the block would have the same picture they had when they entered.
- Change all items to. This activates the 2D picture popup menu for selecting a picture to represent items as they leave the block. This is especially useful for showing the transformation of one type of item (e.g. cans) into another (e.g. cases of cans).
- Change item animation using property. With this choice, the item's animation will change depending on its property (attribute, quantity, or priority). For this option to work, the item's attribute, quantity, or priority must have been set in a preceding block. Then in this block you create a lookup table with properties corresponding to the desired animation pictures.

To change the animation based on an attribute value:

- ▶ Choose **Change item animation using property** in the popup.
- ▶ From the popup menu to the right of that popup, select a value attribute.
- ▶ Click the green +/- box in the bottom right of the table. In the dialog that opens, enter the total number of attribute values that you want to assign animation pictures to. For example, if you enter "3" and click OK, 3 rows will appear in the table.
- ▶ Enter values in the *Property Value* column, then use the 2D popup menu to select which picture will animate the item when its attribute has that value.



Item Animation tab, changing item picture



Item Animation tab, using value attributes

For information about creating attributes, see "Attributes" on page 115.

As discussed on page 554, the Animate Item block (Animation 2D-3D library) is also useful for changing the animation picture of items that pass through it.

### Blocks for customized animation

There are two ways to use blocks from the Animation 2D-3D library to add custom 2D animation to a model without programming:

- Show animation when an item enters a block or when a block gets a value.

- Animate the icon of a hierarchical block.

### Showing animation in response to model conditions

When connected to other blocks in a model, the Animate Value and Animate Item blocks (Animation 2D-3D library) blocks show customized animation in response to model conditions.

#### Animate Value block

The Animate Value block can be connected to value connectors. There are several choices for how this block animates:

- Display the value that is input to the block.
- Move a level up and down between limits.
- Flash a box, text, picture, or circle when the input goes above a specified value.
- For Mac OS only, play a movie when the input goes above a specified value.

Displaying a value is similar to using the Display Value block (Value library), except the value can be displayed in a selected color.

If you choose to animate with a level, it can be displayed in a specific color and pattern that will move between the top and bottom of the block's icon. Be sure to specify maximum and minimum values that represent the entire range of possibilities.

To flash animation when the input is greater than or equal to a value, select a box, text, picture, or circle. The box and circle can have a color and pattern. Text must only be a few characters long so that it fits within the block's icon. To show an animation picture, select its name from the popup menu.

To play a movie in Mac OS, the movie must be stored in the Extensions folder and must be a QuickTime movie.

#### Animate Item block

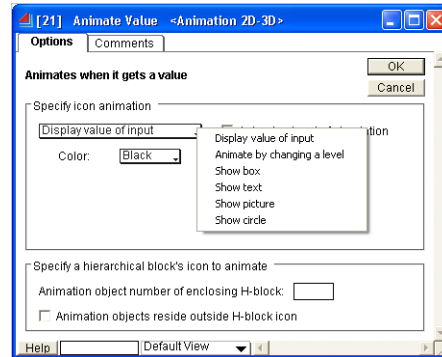
The Animate Item block can only be connected to the item connectors of discrete event blocks. Depending on choices in the Block tab, when an item enters the block a colored box, piece of text, a selected picture, or a selected movie (Mac OS only) is shown on the block's icon. The Item tab is useful for changing the animation picture of the item traveling through the block.

#### Animating a hierarchical block's icon

As shown below, the steps to animate a hierarchical block are:

- 1) Create an animation object on the hierarchical block's icon.
- 2) Include the Animate Item or Animate Value block (Animation 2D-3D library) in the submodel.
- 3) Select options in the Animate Item or Animate Value block's dialog.

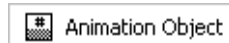
The animation block in the submodel reads the values from other submodel blocks and, using the animation object, controls the hierarchical block's animation based on those values.



Animate Value dialog

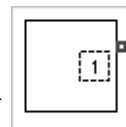
*Step 1: Create an animation object on the hierarchical icon*

- ▶ Open the hierarchical block's structure window:
  - ▶ Select the block in the model.
  - ▶ Choose Develop > Open Block Structure or right-click and select Open Structure.
- ▶ Add an animation object to the icon:
  - ▶ Click the Icon Tools button in the toolbar.
  - ▶ Select the Animation Object at the bottom of the pull-down menu.



- ▶ Click and drag on the icon to draw the animation object to the size you want.


Animation objects display as rectangular shapes on the icon. You can add more than one animation object and you can resize them. Each animation object is assigned a number starting with 1. For example, a hierarchical block's icon with an animation object near the output might look like the image at right.

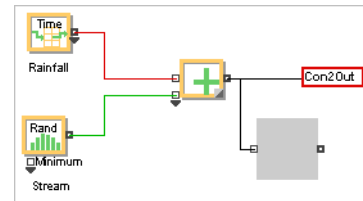


Animation object 1 added to hierarchical block's icon

*Step 2: Include an animation block in the model*

- ▶ Attach the Animate Value or Animate Item block (whichever is appropriate) to the output of the block you want to animate in the submodel. In the example to the right, The Animate Value block is used to animate the sum of the two water sources in the Water Sources hierarchical block.


 The Animate Value block can be connected in parallel and its output does not need to be connected, as shown in the screen shot at right. However, items must pass through the Animate Item block. That block cannot be connected in parallel and its output must be attached to the input of another block. Otherwise an item coming into the Animate Item block would have no way to exit.

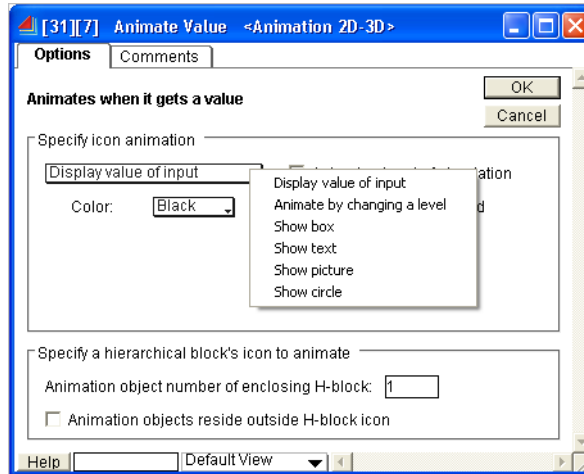


Animate Value block added to submodel

*Step 3: Select options in the animation block's dialog*

- ▶ Choose the type of animation you want from the **Specify icon animation** popup menu,
- ▶ Enter the number of the animation object that the block is to control in the **Animation object number of enclosing H-block** field. For example, to animate object number 1, enter "1" in the field.

 If the animation object number cannot be located on the hierarchical block's icon, ExtendSim will search the icon of the next highest enclosing hierarchical block. If it still fails to locate that animation object number, it will give an error message.



Animate Value dialog

If you create custom blocks, including hierarchical blocks, use this same process to add animation objects to those blocks.

For an example of an animated hierarchical block, see the "Markov Chain Weather model" on page 50 or the "Animating Queue Contents model" on page 141 (that model is not available with ExtendSim CP.)

### Animation functions

If you build your own blocks, ExtendSim offers a suite of functions to design how the block will animate. For example, the Planet Dance model (located in the folder \Examples\Continuous\Custom Block Models) shows three planets orbiting in space. Their orbits are determined by numbers entered in the blocks' dialogs. See the Developer Reference for a description of the animation functions. Also, see "Animating a hierarchical block's icon" on page 554 for an example of how to add animation to a block. You use that same process to add animation to any blocks you create.

### Animation pictures

ExtendSim includes a large number of animation pictures as bitmaps (Windows) or PICT resources (Mac OS). These pictures are used by discrete event blocks to show the flow of items, by the Animate Value and Animate Item blocks (Animation 2D-3D library) to show custom pictures, and in any custom block that calls the appropriate animation functions.

To add your own animation pictures to the available list, they must be of the correct file format and they must be stored in the ExtendSim7/Extensions/Pictures folder. After placing the animation pictures in the Pictures folder, restart ExtendSim and the pictures will be available in animation picture popup menus. For information about these and other extensions, see the Developer Reference.

#### Picture file formats

ExtendSim running on a Windows operating system accepts three kinds of pictures:

- .WMF (Windows MetaFiles)
- BMP (Bitmap)

- A Mac OS PICT (picture) resource file that has been converted to Windows format using the ExtendSim Mac/Win converter.

ExtendSim for the Mac OS accepts only picture resources.

### Displaying messages on a block's icon

The Run > Debugging > Show Block Messages command is a type of animation used by block developers when they step through models to debug blocks. It is discussed further in “Stepping through the simulation” on page 619.

## ExtendSim databases

Databases are very useful when presenting models to others because a model's inputs and outputs can be stored in a centralized location and data can be organized in a logical manner. Each model can have one or more ExtendSim databases for storing, managing, and reporting information. For more information, see “ExtendSim databases for internal data storage” on page 638.

## Connections

Connections are the main method blocks use to send data, items, or flow between each other. ExtendSim has two types of connections, *line connections* and *named connections*. The following discussion shows how to format connection lines and discusses the use of named connections. To learn how to create, move, and delete connections between blocks, refer to “Connecting blocks” on page 27.











- ☞ The Value, Item, and Rate libraries each contain Throw and Catch blocks for sending data without using connections. Use these blocks in situations where you cannot use a line or named connection, such as when you need to send data between hierarchical levels. For instance, see the example at “Throw Item and Catch Item blocks for merging item streams” on page 148.

### Connection lines

ExtendSim provides several formatting options to alter the appearance of connection lines. Some default line options can be specified in the Edit > Options > Model tab. To change an existing line, double-click to select it, then choose a connection line option.

- ▶ Choose Model > Connection Lines to see the menu of options available.

► To use the command, choose an option from the menu and draw a new connection.

right angle	<input checked="" type="checkbox"/> 	Ctrl-Shift-D
straight	<input type="checkbox"/> 	Ctrl-Shift-A
right arrow	<input type="checkbox"/> 	Ctrl-Shift-E
left arrow	<input type="checkbox"/> 	Ctrl-Shift-F
solid	<input checked="" type="checkbox"/> 	Ctrl-Shift-N
dashed	<input type="checkbox"/> 	Ctrl-Shift-O
black	<input type="checkbox"/> Black Connections	Ctrl-Shift-Q
colored	<input checked="" type="checkbox"/> Color Connections	Ctrl-Shift-T
default line types	<input checked="" type="checkbox"/> View Using Defaults	Ctrl-Shift-G
thin	<input checked="" type="checkbox"/> 	Ctrl-Shift-H
medium	<input type="checkbox"/> 	Ctrl-Shift-J
thick	<input type="checkbox"/> 	Ctrl-Shift-K
hollow	<input type="checkbox"/> 	Ctrl-Shift-M

Connection line choices

### Styles

The first set of choices in the menu lets you specify the style of the connection lines, either right angle or straight line.

### Arrows

The second set of choices specifies whether or not you want arrows on your right-angle connections (arrows can *only* be used with right-angle connections). The direction of the top arrow head follows the direction you draw the line when you make the connection; the bottom arrow head follows the opposite direction.

### Attributes

The third choice gives the ability to make the line solid or dashed. (Dashed lines are only available for right-angle connections.)

### Colors

The fourth set of choices in the Connection Lines command deals with the color of connection lines. Like text or drawing objects, connection lines can be any color you choose. There is no limit to the number of different colors you can use for connection lines in a single model. Selecting Black Connections in the Connection Lines command causes all new connection lines to be drawn black regardless of the current color selected in the color palette. Choosing Color Connections will cause all new connections to be drawn using the color currently selected in the color palette.






To change the color of an existing connection line, select the line and click a color in the color palette window. See “Patterns and colors” on page 562 for more information about working with colors.

### Line types

If you don't select the View Using Defaults option, the final connection line section lets you specify the type of line used for connections, such as thin, medium, or hollow. The View Using

Defaults option will cause a connection line to be displayed using its default line type, based on the type of connector. The default connection line types are:

Type of connector	Line type
Value	
Item	
Flow	

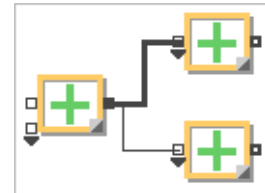
For instance, most discrete event models track flows of items as well as calculate and show values. You can visually differentiate the flow of items from the processing of values by using the default line types. When looking at such a model, it is much clearer which flows are those of items and which are of values. For example, the beginning Car Wash model shown on page 101 uses the default types to specify item and value connection lines.

To specify the default connection type, choose the Model > Connection Lines > View Using Defaults command. You can also choose whether or not to use the default connection types in the Options dialog. When you use this command, the thickness or hollow line types are ignored. You can leave the View Using Defaults command on while you build a model, or only turn it on when you are viewing the model. Note that this is only a view option, not a true line type; if you turn the option off, the line will be thick or hollow depending on the selection in this section.

**How to change line formats**


By double-clicking on a connection line segment you can select all segments between two blocks. This makes it easier to change the entire connection (e.g. line width, color, etc.) at the same time. Any other line segments that may be connected to additional blocks will not be selected.

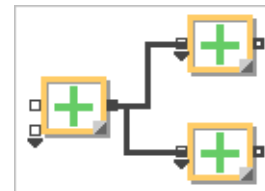
 This works with both regular connection lines and named connections.



Selecting entire connection between two blocks

To select all the connection line segments that connect from a single output, double-click on a line segment while holding down the Alt (Windows) or Option (Mac OS) key. Alternately, you can click on one segment of the connection, then choose the Edit > Select All Segments command.

 Clicking anywhere else on the model worksheet will deselect all the selected line segments.

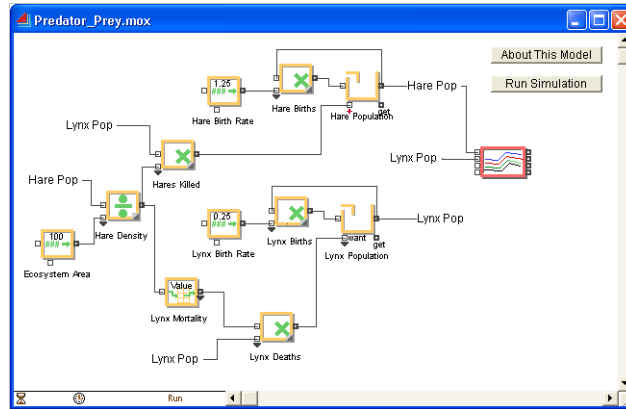


Selecting all line segments

How To

### Named connections

Named connections are text labels that are used to represent one output at many locations in your model. If you have two text labels with the exact same text, you can use these to have the data, items, or flow jump from one part of the model to another. Named connections are often used when you do not want to clutter up your model with many lines. You can place the names near the blocks to which they connect and leave much of the area of your model free from connection lines.



Predator Prey model using named connections

Named connections are not case sensitive and spaces and returns are ignored, but you must use identical spelling in the text names.

By using named connections, you can eliminate a lot of connection lines in your model, making it easier to see which blocks are connected to each other and preventing lines from crossing over blocks or other model elements.

To learn how to create named connections, refer to “Named connection” on page 33.

Named connections only connect within one level in a hierarchical model. This means that the data will not flow between levels if you have a named connection on one level and a corresponding named connection in a block at a lower or higher level. To connect between hierarchical levels, use the Throw and Catch blocks in the Value, Item, or Rate libraries, or add connectors to hierarchical blocks. For more information about using Throw and Catch blocks with hierarchy, see “Throw Item and Catch Item blocks for merging item streams” on page 148. For more information on adding connectors to hierarchical blocks, see “Step 4: Add connectors” on page 545.

#### Show Named Connections command

If you use many named connections in a model, you may accidentally connect a block to the wrong named connection. The Model > Show Named Connections command draws a connection line directly between all named connections. This helps you check that you made the connections that you wanted.

With the Show Named Connections command selected, click on a connection line to highlight the path between the named connections. If you click on one of the connection lines and then use the Edit > Select All Segments command, all the connections for that particular named connection will be highlighted.

### Model appearance

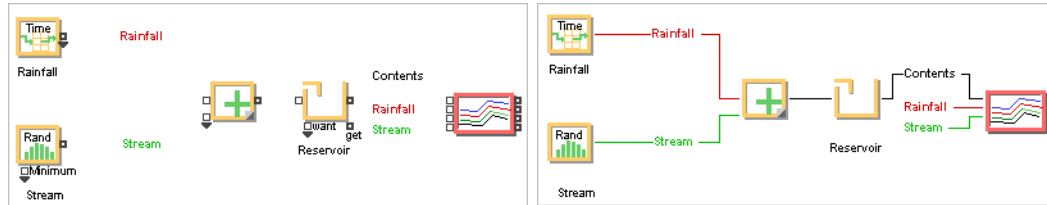
Commands in the Model menu can change how model elements appear.

#### Showing and hiding connections and connectors

Choose the Model > Show/Hide Connections and Model > Show/Hide Connectors commands to display or not display those elements of the model. This can be useful when you have a lot of con-



nections and blocks and you want to present the model to others without so much clutter in the workspace.



Reservoir model with hidden connection lines (L) and hidden connectors (R)

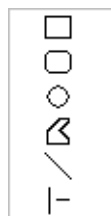
### Changing model styles

ExtendSim provides the capability for block icons to have alternate *model styles* that affect their appearance in a model. This implementation depends on the block developer, so many libraries have only one style - the default Classic style. If a library has multiple styles, use the Model > Change Model Style command to choose a new style. If the library does not have multiple styles, giving the command will have no effect on block appearance. Block developers can create libraries with up to 8 model styles, as discussed in the Developer Reference.

## Graphic shapes, tools, and commands

The six tools in the Shapes menu let you add drawing objects to the model. The Shuffle Graphics tool lets you arrange drawing objects in layers. Use these tools to make your models easier to read or to make them more aesthetically pleasing.

### Drawing objects in the Shapes menu



Use the Rectangle, Rounded Rectangle, Oval, and Polygon tools to add those shapes to your worksheet or Notebook. For example, to add a rectangle, select the Rectangle tool, click in the model where you want one of the rectangle's corners, and drag to the diagonally opposite corner.

The Line tool will draw a line at any angle and the Draw Right Angle Line tool restricts the lines to horizontal and vertical. You can also add colors and patterns to the shapes and lines you draw, as described in "Patterns and colors" on page 562.

**Shapes menu** To resize or reshape an object, click the shape then click the small rectangle in the bottom right and drag to create the desired shape. If you hold down the Shift key as you reshape a rectangle, rounded rectangle, or oval, the shape will become a square or a circle. If you resize a square or circle while holding down the Shift key, it will maintain its proportional measurements.

### Shuffling graphics



By default, new objects are created in front of existing ones. The Shuffle Graphics tool lets you arrange drawing objects (not text) that are on top of each other. Click this tool, then click on the object. If the object is in front, it will be sent behind. If it is not in front, it will be brought to the front. Blocks, text, and cloned dialog items are always in front of any drawing item and therefore cannot be shuffled.

You can also right-click an object with the Draw layer or All layers tool and choose Bring To Front or Send To Back to change the order of the objects.

How To

### Modifying objects

When an object is selected, the Model menu has additional commands available:


- Align left, right, top, or bottom (2 or more objects must be selected)
- Rotate shape
- Flip horizontally/vertically
- Border thickness
- Shape Fill/Border (Depending on the selection in this command's hierarchical menu, the selected color fills the shape or colors the shape's border.)

 One or more objects must be selected for these commands to be enabled.


### Patterns and colors

Every draw object has both a pattern and a color, chosen from the menu in the toolbar. The default is solid pattern and black color. Text can have color but not a pattern.

- ▶ Click the Color and Pattern toolbar buttons to open their respective palette windows.


 Change the color of a piece of text or the fill color of a drawing object by selecting the drawing object on the model worksheet or the text in the text box and choosing a color from the palette. To change the color of the drawing object's border, select the command Model > Shape Fill/Border > Border Color before choosing the color from the palette. If you change the color palette *before* you create the drawing object or start a text box, all new drawing objects and text will be in that color.

When you select a color the hue, saturation, and value (brightness), or HSV, settings for the color are listed at the bottom of the color palette window. This is helpful when you need to assign a color using its HSV definition (in block code for example, as discussed in the Developer Reference.)

 In the pattern palette, black gives a solid pattern, white gives an opaque white pattern, and the "N" signifies no pattern, meaning transparent. Change a drawing object's pattern by selecting the object and choosing a pattern from the palette. If you change the pattern palette *before* you create the drawing object, all new drawing objects will be in that pattern.

### Working with pictures

To add a picture to ExtendSim, create or open the picture in a painting or drawing program and copy it to the Clipboard. With the ExtendSim window open (model worksheet, Notebook, or icon pane) choose Edit > Paste Picture. ExtendSim pastes the picture into the window. For more information, see "Copy/Paste and Duplicate commands" on page 674.

 It doesn't matter what the original file format for the graphic was (JPEG, TIFF, GIF, etc.) Once it has been copied into the Clipboard, ExtendSim treats it as a drawing object.

The screenshots that start on page 9 illustrate a use of pictures in ExtendSim models.

# How To

## Analysis

Making sense of your models

*“Prediction is very difficult, especially if its about the future.”*  
— Niels Bohr

Although simulation is a quantified subject, there is still room for analysis and judgmental procedures. This is especially true when your purpose in building a model is to spot trends or identify patterns of behavior, or when you must make an immediate “go, no-go” decision. How statistically precise your models should be depends on several factors, including the nature of the problem, the importance of the decision, the level of risk you are willing to accept, and the sensitivity of the system to the input data.

One of the main benefits of ExtendSim is that it provides several methods for analyzing the results of simulation runs. This chapter covers those techniques, including:

- Blocks for statistically analyzing data
- Confidence intervals for predicting the true mean value
- Sensitivity analysis to investigate the impact of making changes
- Optimization to determine the best values for a set of parameters
- Using Stat::Fit to determine the appropriate distribution for a set of data
- How to change parameters dynamically
- Plotters for displaying results
- Reporting model details and results

Remember that when you use statistical methods to analyze simulation output, you are performing the analysis only on model results, not on the actual system. It is important to ensure that the numbers entered accurately represent the details of the actual system. The significance and relevance of your analysis will depend on how closely the model’s inputs correspond with real-world data (“garbage in = garbage out”).

### Blocks that calculate statistics

Many ExtendSim blocks automatically calculate relevant statistics and display them on their Results tab. For example, the Activity block (Item library) reports utilization, average wait, maximum length, and so forth.

You can also use blocks in the Value library, such as the Equation and Math blocks, to perform calculations on model outputs.

As described below, other ExtendSim blocks are specially designed to report statistical information for particular types of blocks or for items that pass through the block, or to refine statistical data collection.

### Statistics



The Statistics block (Value library) accumulates data and calculates statistics using a specified *statistical method*. Place a Statistics block anywhere in the model and, depending on selections made in its dialog, it will report statistics for the following blocks and types of blocks:

- Activities (Activity, Convey Item, and Transport blocks in the Item library)
- Mean & Variance block (Value library)
- Queues (Queue, Queue Equation, and Queue Matching blocks in the Item library)
- Resource Item block (Item library)
- Resource Pool block (Item library)
- Mixed blocks (clone drop)

- Workstation block (Item library)
- Convey Flow block (Rate library)
- Tanks (Tank and Interchange blocks in the Rate library)

For the *Mixed type (clone drop)* type, drag cloned output fields (utilization, length, etc.) from the dialogs of any of the supported types of blocks onto the Statistics block icon. This causes the cloned output information to be displayed in the table. For instance, in the screenshot to the right, the length and utilization from an Activity block, as well as the utilization of a Queue block, have been cloned into the Statistics block and are reported in the table.

Block	Block Name	Variable Name	Value	Time	Delete
0	Queue 1	Queue	QueueLength_pr	101	1000
1	Activity 1	Activity	utilization_prm		1000
2	Activity 1	Activity	length_prm	1	1000

Statistics block dialog, discrete event model

### Accumulating data

When a simulation is run, one row of information is recorded in the block's table for each block of the specified type each time an observation is made. Observations can be recorded continuously, which significantly slows simulation time, or at the end of the simulation. They are also made in response to an item or value arriving at the Go universal input connector. For instance, attach a Pulse block (Value library) to the Go connector to force a periodic or scheduled update of statistical information.

The first column of the table in the Statistics block's dialog lists the block's label. For a block without a label, the first column gives the block's name.

To immediately see the current statistics, click the Update Now button; this is also useful if you want to see which blocks will have their statistical information collected during the simulation run.

### Statistical methods

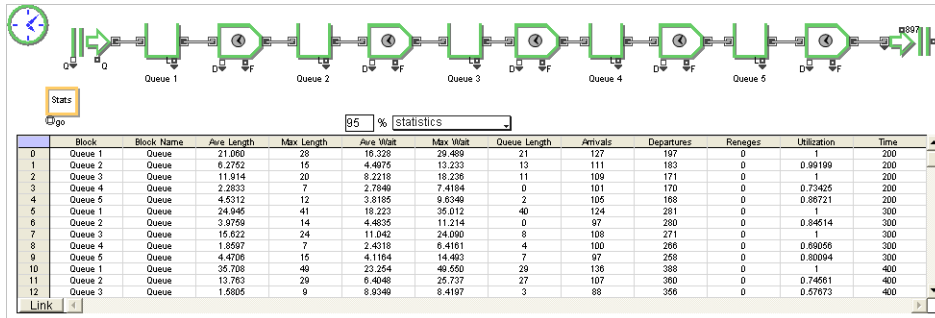
The Statistics block's Options tab allows you to select one of three statistical methods:

- Multirun analysis
- Batch means
- Custom

Choosing one of the first two methods causes specific settings to be selected in the block's Options tab. For the "custom" option, manually select which settings you want.

### Queue Statistics model

The Queue Statistics model uses the Statistics block to gather information about queues. This discrete event model is located in the folder \Examples\Discrete Event\Statistics.



Queue Statistics model

### Clear Statistics



As discussed in “Non-terminating systems” on page 530, when you start a simulation run there is no data in the model. After the model has been running for a while, it gets to the point where it is functioning more like the real system. The interval from when the model starts to when it is functioning in a steady or normal state is called the *warm-up period*.

The Clear Statistics block (Value library) clears the statistics for selected blocks, eliminating the statistical bias of a warm-up period. The type of blocks include:

- Activities (Activity, Convey Item, Transport, Workstation)
- Exit block
- Mean & Variance block
- Queues (Queue, Queue Equation, Queue Matching)
- Resources (Resource Item, Resource Pool)
- Information block
- Rate library blocks

The block can reset statistical accumulators at periodic intervals or in response to a system event. The block can be placed anywhere in the model. Choose in its dialog which types of blocks will have their statistics cleared. The Clearing Statistics example, a discrete event model shown on page 239, uses the Clear Statistics block to restart model statistics after 40 seconds.

### Mean & Variance



The Mean & Variance block (Value library) calculates the mean, variance, and standard deviation of the values that it receives during a simulation run, based on a specified confidence interval. Settings in the block’s dialog allow you to use time weighted statistics, calculate a moving average for a selected interval, and calculate the statistics over multiple runs.

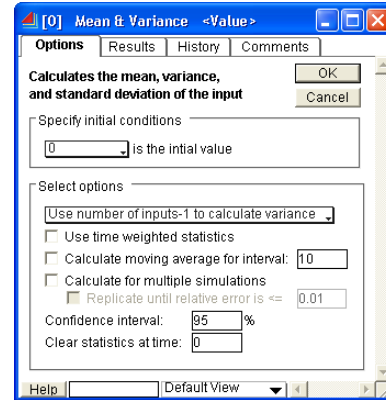
The dialog provides two choices of behavior:

- **Use number of inputs to calculate variance.** The variance is computed as the sum of (valid inputs - mean)<sup>2</sup> ÷ (number of inputs).
- **Use number of inputs-1 to calculate variance.** The variance is computed as the sum of (valid inputs - mean)<sup>2</sup> ÷ (number of inputs - 1).

The Mean & Variance block uses 1/(N-1) as an averaging factor. If an input is a NoValue, it is ignored and does not affect the statistics.

To use this block, connect a value output from of the block of interest to the Mean & Variance block's input.

The Monte Carlo model, located in the folder \Examples\Continuous\Standard Block Models and discussed on page 47, is an example of using the Mean & Variance block.



Mean & Variance dialog

### Information



The Information block (Item library) reports statistics about the items that pass through it. It is useful for counting the number of items and for determining cycle time statistics and the time between item arrivals (TBI). The block reports the average and the current statistics, as well as the minimum and maximum.

The Information block can only be used in models that use item-based blocks, and it must be connected to a block's item output. This block is used to calculate cycle time in the section "Cycle timing" on page 254.

### Cost Stats



The Cost Stats block (Item library) records in a table the input costs and total cost generated by each costing block in a model. This block reports cost information for item-based blocks, such as for a discrete event model. Information can be exported to a spreadsheet. The block can be placed anywhere in the model.


## Confidence intervals

*Confidence interval estimation* tells you, with a given level of probability or confidence, how close the average simulation results are to the *model's* true average or mean. The *confidence interval* is the range within which it is predicted the true mean is located. This is expressed as the probability that the true mean lies within interval  $x \pm y$ , where "x" is the average obtained by multiple observations and "y" defines the outer limits of the interval's range. The *confidence level* is the probability that the true mean will be located within the range. Typical confidence levels are 90%, 95%, and 99%. Notice that, at higher levels of probability, the interval gets wider.

The Statistics and Mean & Variance blocks (Value library) and the Cost Stats block (Item library) provide the option to specify a confidence level in their dialogs. For example, the Statistics block not only summarizes and reports statistical information, but also calculates confidence intervals for the information given various levels of confidence.

To generate a confidence interval where each sample is the result from a single simulation run, in the Run > Simulation Setup > Random Numbers tab do one of the following:

- Randomize the seed by entering a blank or zero random seed.
- Or, select **Continue random number sequence** in the popup.
- Or, select **Use Database table \_\_Seed for values** in the popup.

 To obtain sufficient sample data to determine the confidence interval, multiple observations of each statistic must be made and appended to the table of data. This is shown in the “Run for CI” model located in the \Examples\Tips\Modeling Tips folder. The model runs repeatedly based on the Mean & Variance’s Options tab settings *Calculate for multiple simulations* and *Replicate until relative error is <= 0.01*.

## Sensitivity analysis

Sensitivity analysis allows you to conduct controlled experiments to explore how much of an impact a particular parameter has on model results. ExtendSim’s sensitivity analysis features make it easy and convenient to specify a parameter to investigate and settings to use for the analysis.

### Overview

You use sensitivity analysis to investigate the effect of changing one or more parameters upon an area of interest. Sensitivity analysis works with all numeric parameter fields. It also works with clones of those numeric items. You can add sensitivity to as many dialog values as you like. However, it is recommended that you only vary one or two dialog values at a time so as not to confuse the analysis. Once a parameter has been sensitized, specify a multiple number of runs and run the simulation.

The resulting values for the area of interest are usually plotted. Although you can use any of the standard plotters from the Plotter library (such as the Plotter I/O or the Plotter Discrete Event), it is more common to use a MultiSim plotter (to show up to four runs at a time in one plot window) or an Error Bars plotter (to show the mean and standard deviation of the parameters over the count of runs). The results of varying the parameter value over the selected settings will be displayed as the simulation is run multiple times.

Parameters are sensitized by right-clicking or by using the Edit > Sensitize Parameter command. The Edit > Open Sensitized Blocks command shows all the dialogs for blocks that have sensitivity settings, even if sensitivity analysis is not enabled. This is convenient if you have entered sensitivity settings for many parameters in a large model.

 To utilize sensitized parameters, the number of simulation runs must be greater than one.

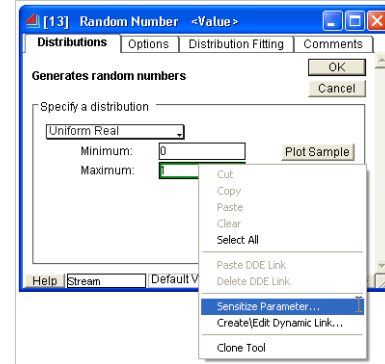
### Steps for using sensitivity analysis

For this example, use the Reservoir 1 model discussed in the Tutorial module.

- ▶ Open the Reservoir 1 model (located in the \Examples\Tutorials folder)
- ▶ So that you don’t overwrite the original model, give the command File > Save Model As and save the model as “Sensitivity”.



- ▶ Open the dialog for the Random Number block labeled “Stream”.
- ▶ Open the Sensitivity Setup dialog by doing *one* of the following to the *Maximum* entry field:
  - Click the entry field and choose the command Edit > Sensitize Parameter.
  - Or right-click the entry field and choose Sensitize Parameter.
  - Or click the entry field while holding down the Control (Windows) or Command (Mac OS) key.



Sensitize Parameter menu

The Sensitivity Setup dialog appears with the *Enable sensitivity* checkbox selected by default.

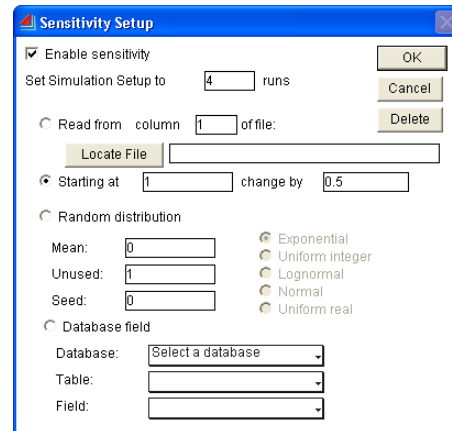
- ▶ Enter **Set Simulation Setup to: 4 runs**.

The number of runs can be set in either the Sensitivity Setup dialog or the Simulation Setup dialog. Each controls the other so that the last value selected in either of them controls the number of runs.

- ▶ Enter **Starting at 1 and change by 0.5**.

This will cause the stream’s flow to increase by 0.5 for each subsequent run.

Parameter values can be sensitized using values from a file, a specified range of values, a random distribution, or values from a database. The choices are described in “Specifying the sensitivity method” on page 570.



Sensitivity Setup dialog

- ▶ Close the Sensitivity Setup dialog.

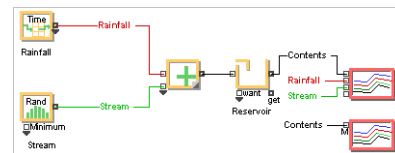
In the Random Number block’s dialog, the parameter field now has a green border around it, indicating that it has a sensitivity setting and sensitivity analysis is active for the parameter.



Sensitized parameter

- ▶ Add a Plotter, MultiSim block (Plotter library) to the right side of the model and add a **Contents** named connection to its input.

This plotter displays up to four runs of data on a single graph, so you can see what impact changing the stream flow has on the contents of the reservoir.

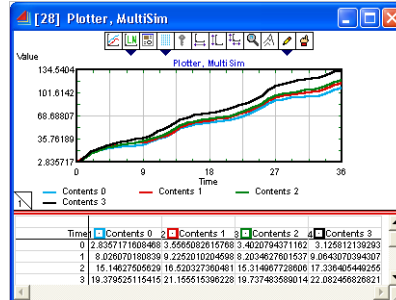


Model with Plotter, MultiSim

- ▶ Ensure that the Run > Use Sensitivity Analysis command is checked (it is checked by default).
- ▶ Run the simulation. ExtendSim runs the simulation four times, from run 0 to run 3.

- ▶ In the MultiSim Plotter, click the AutoScale Y tool to rescale the vertical axis.

You can see the variation between the runs in the four plotted lines. As expected, increasing the stream's flow increased the amount of water in the reservoir. In more complex models, the effect of making a change would not be so obvious.



Multisim Plotter results

### Specifying the sensitivity method

The Sensitivity Setup dialog lets you specify that a sensitized parameter will change from a list in a file, incrementally, randomly, or from based on the value of a database field. The choices are:

Option	Description
Read from column x of file	Assigns the values from a text file. This is the option you will most likely use when performing ad-hoc experiments. If the file has more than one column separated by tab characters, specify the desired column. Starting with the first row of the specified column for the first run, this option uses the value of each successive row in that column for subsequent runs.
Starting at... change by...	Specifies the starting value and the amount of change. By default, the starting value is the same as the parameter's value in the dialog. Increase the variable with a positive number or decrease it with a negative number.
Random distribution	Uses a random distribution to set the parameter. This is an easy way to make a single value in a model change randomly over many simulation runs while keeping the value constant within a single run. Choose from one of the five types of distribution and enter the distribution parameters in the options to the right of the distribution. The <i>seed</i> is the number to use for the random number generator. As in the Simulation Setup dialog, BLANK or 0 for the seed is random.
Database field	Assigns values from the fields in an ExtendSim database. Starting with the first record of the specified field for the first run, this option uses the value of each successive record for subsequent runs.

**How To**

☞ If you are a developer, you can use the *CurrentSense* variable to control the order of use for the values set for specific runs from within a block. For example, you can cause the first simulation run to use row 4 of a file, rather than the default use of row 1.

### Turning sensitivity on and off

You can control sensitivity globally (for the model as a whole) or locally (at the dialog parameter level for a specific block):

- Use the Run > Use Sensitivity Analysis command to turn sensitivity analysis on and off for the model as a whole.
- Enable, disable, and delete sensitivity settings for a particular parameter using the Sensitivity Setup dialog.

When you enter sensitivity settings for a value, sensitivity analysis is enabled as long as the *Enable sensitivity* box is checked in the Sensitivity Setup dialog. If you uncheck the box, a dialog value's

sensitivity is temporarily disabled so that you don't have to re-enter the number for subsequent analysis.

To remove sensitivity settings from a parameter (as compared to simply temporarily disabling the settings by turning off the parameter's Enable sensitivity box), open the Sensitivity Setup dialog using any of the methods discussed on page 569, then click Delete.

- Editing a sensitized parameter in a block's dialog disables the sensitivity settings for that block. When this happens, ExtendSim automatically unchecks the *Enable sensitivity* choice. ExtendSim assumes that if the value is edited, you want to use that new value, not the one that was entered in the Sensitivity Setup dialog. If you want to turn off sensitivity analysis for a parameter for the foreseeable future, open that item's Sensitivity Setup dialog and click the Delete button. This will help prevent accidentally changing the value in a future run of the simulation.

A parameter that has sensitivity settings has a frame inside of it. If sensitivity analysis is active for the parameter (that is, if the Enable sensitivity choice is checked), the frame is green. If the sensitivity analysis is inactive for the parameter or if it is turned off for the model as a whole, the frame is red.

The Edit > Open Sensitized Blocks command shows the dialogs of all blocks with sensitized parameters, regardless of whether or not sensitivity is enabled.

### Reporting the results

In addition to MultiSim and Error Bar plotters, ExtendSim's reporting and tracing features are useful when analyzing output after using sensitivity analysis. The Reporting features show final values and the Trace feature shows values at each step or event. For more information, refer to "Reports" on page 596 and "Model tracing" on page 620.

### Multi-dimensional scenarios

Sensitivity can be enabled on more than one item at a time. For instance, you may want to vary the values of two Constant blocks and see the interaction between the two items. If you set the sensitivity for the parameters with the *Starting at* option, both values will increment at the same rate. For instance, if you have one parameter start at 5 and increment by 1, and the second parameter start at 100 and increment by 50, and the simulation is run seven times the value pairings will be:

Run #	Variable 1	Variable 2
0	5	100
1	6	150
2	7	200
3	8	250
4	9	300
5	10	350
6	11	400

Often, however, you want to look at all the possible pairings of the two (or more) variables. In this example, you would want to run the model 36 times, with the following pairings:

Run #	Variable 1	Variable 2
0	5	100
1	5	150
2	5	200
3	5	250
...	...	...
7	6	100
8	6	150
...	...	...
35	11	400

In order to perform this kind of multi-dimensional analysis, you need to get the values from a file. The most convenient way to do this is to create a file that has two columns separated by a tab character with all the desired pairings. For instance, the file for this example would start with:

```
5 100
5 150
5 200
...
```

In the Sensitivity Setup dialogs for the two parameters, choose **Read from file** and enter the file name. For the first variable, enter **1** for the column number; for the second variable, enter **2** for the column number.

When you run a multi-dimensional analysis, you usually use a Write block (Value library) to write out the values to examine. In the Write block, select a data destination on the Send Data tab and check **Row (or record) index is equal to run number** on the Options tab. If the data takes a long time to transmit, you should check **Only write to (data destination) at the end of the run** on the Send Data tab.

## Optimization

Optimization is a powerful feature that can automatically determine ideal values for parameters in a model. It does this by running the model many times using different values for selected parameters, searching the solution space until it is satisfied that it has found an acceptable solution. It then populates the model with the optimized parameter values.

ExtendSim facilitates optimization by making the optimization algorithm available within a block that can be added to any model to control all aspects of the optimization. Furthermore, having a block do the optimization increases flexibility and opens up the method and source code to users who might want to modify or create their own customized optimization blocks. The Optimizer block (Value library) uses an evolutionary algorithm to reduce the number of times the model has to run before a solution is found.

## How optimization works

Optimization, sometimes known as “goal seeking,” is a useful technique to automatically find the best answer to a problem. The “problem” is stated as an *objective function* or *cost equation* that ExtendSim tries to minimize or maximize to save you going through the tedious process of manually trying different values with each model run.


Like most optimization algorithms, the ExtendSim Optimizer solves models using an initial population of possible solutions. Each solution is explored by running the model several times using different values for some selected parameters, averaging the samples (for stochastic, random models), and sorting the solutions. The best solution sets of parameters are then used to derive slightly different but possibly better solutions. Each new derived solution is called a generation. This process continues for enough generations until the Optimizer determines that there are probably no better solutions in sight. The Optimizer then terminates the simulation runs and populates the model with the best solutions it has found.

The downside to optimization is that the model needs to run repeatedly and this can take a long time with large models. Also, optimization algorithms have an inability to tell when the best solution has been found, or even if the best solution has been attempted. A good approach is to allow the optimization to run for a sufficient number of cases and to then see if the population of solutions has converged. Then try the optimization procedure several additional times to make sure that the answers agree (or are close) and that the first answer is not a false or sub-optimal one. There are no optimization algorithms that are guaranteed to converge to the best answer in a finite time. The more time that you can give optimization to run, the better chance that it will provide the optimum answer.

## Steps for using optimization

The steps needed to optimize a model are listed below. The tutorial that follows illustrates these steps.

- 1) Add an Optimizer block (Value library) to a model.
- 2) Define the form of the objective function.
- 3) Determine which variables the equation needs and “clone-drop” them onto the Optimizer.
- 4) Set the limits for those variables in the Optimizer’s Variables table.
- 5) Derive the equations for the objective function.
- 6) If variables need to be constrained to certain values, add constraint equations.
- 7) Set the Optimizer’s Run Parameters for a random or non-random model, then run the optimization.

 The following tutorial assumes that you know how to clone parameters (see “Cloning” on page 504) and that you are comfortable deriving equations (if not, see “Equation-based blocks” on page 601).

## Optimization tutorial

The model for this tutorial represents a drink stand at a county fair; it has these assumptions:

- Drinks are dispensed from beverage tanks that can range in size from 1000-8000 drinks. (In the example model, 1000 is used for the initial tank size setting.)

- A truck delivers the beverage tank at the start of the day and exchanges it periodically throughout the day. (The model is initially set to get a second tank after 240 minutes.)
- The truck exchanges the beverage tank for a new one of the same size. There is a \$1 per drink beverage charge plus a delivery charge of \$1,000 per tank.
- So that the beverage company will know when to deliver tanks and what size tank you will use for the day, arrangements regarding tank size and delivery frequency must be made at the beginning of the day.
- People purchase drinks according to a random distribution; the cost per drink is \$2.50.
- If the beverage tank becomes empty, you lose an estimated \$100 per minute in sales because customers already in line go to another stand and new customers are discouraged from getting in line.
- If you exchange the tank too often, you lose money because the beverage inside the old tank gets taken away along with the tank.
- This model ignores other expenses, such as labor.
- The model runs for a simulation time period of 480 minutes (8 hours).

The goal of this tutorial is to optimize both how big of a tank you should order and how often the tanks are exchanged. It is a simple continuous model, but a good example to show some of the optimization techniques you will use in any type of large model.

📖 For comparison purposes, the final model for this example, “Optimize 2”, is located in the `\Examples\How To\Optimization` folder.

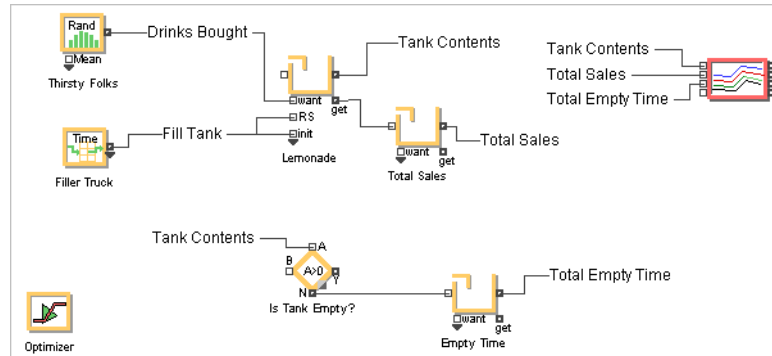
#### **Adding an Optimizer block**

- ▶ Open the Optimize 1 model from the `\Examples\How To\Optimization` folder.

To provide a starting point for the optimization, this model has been populated with initial assumptions for the decision variables: deliveries are repeated every 240 minutes and the tank size is 1,000 drinks.

- ▶ So that the example file isn't overwritten, give the command `File > Save Model As` and save the model as “MyOptimizer”.

► Place an Optimizer block (Value library) in a convenient place in the model.



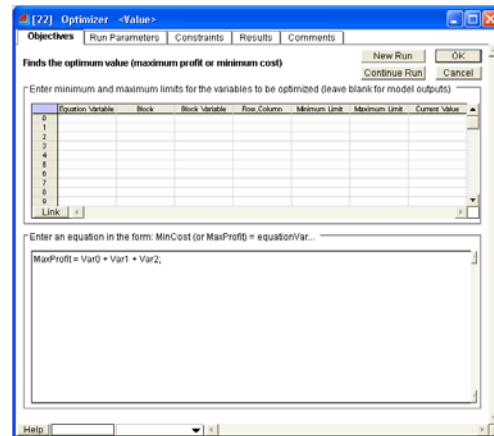
Optimizer block added at lower left of MyOptimizer model

You will be adding most variables to the Optimizer block by doing a clone-drop.

► To get an idea of what it looks like without any entries, open the Optimizer's dialog.

As seen to the right, the Optimize block's Objectives tab has two sections:

- A *Variables table* for entering variables and specifying their limits
  - An *Equation pane* for entering the objective function.
- Close the Optimizer's dialog.



Objectives tab in Optimizer block

**Determining the form of the function**

The Optimizer block will try to reach a goal by changing the values of model variables based on an equation that measures how close the goal is.

The first step is to lay out the form of the objective function. This helps clarify what will be optimized and which factors affect that goal. In most cases, you want to minimize a cost or maximize a profit.

This model tries to maximize the beverage stand's profit. The factors that affect profit are:

- Beverage costs (\$1 per drink plus \$1,000 per tank)
- The revenue for each drink sold (\$2.50 per drink)
- A penalty or reduction in profit if you run out of drinks to sell (\$100 per minute)

Using this information, the form for the objective function is:

$$MaxProfit = \$2.50 * \#sold - \#deliveries * (\$1000 + \#drinks * \$1.00) - time\ empty * \$100$$

**Adding variables to the Optimizer**

The form for the objective function is the basis for the cost equation that you will enter in the Optimizer's dialog. Before you do that, you need to obtain the necessary model variables (called

*decision variables*) so the equation can reference them. In some cases the required variable will be a dialog parameter and in others it will need to be calculated based on a dialog parameter.

An Optimizer block's access to model variables is accomplished by using the Clone Layer tool to drag clones of the desired dialog variables onto the block's icon. This operation, known as *clone-dropping*, adds information about the variable to the Variables table. It also enables the Optimizer block to remotely read and change the value of that variable in the model, so that it can explore possible solutions.

Using the variables nomenclature from the form for the equation ( $\text{MaxProfit} = \$2.50 * \# \text{sold} - \# \text{deliveries} * (\$1000 + \# \text{drinks} * \$1.00) - \text{time empty} * \$100$ ), the steps are:


#### *# sold*

The total number of sales for the day is an output variable calculated by the model; it is directly available in a block's dialog.

- ▶ Open the Holding Tank block labeled "Total sales".
- ▶ Using the Clone Layer tool, drag the **Current level** parameter value (but not its label) onto the closed Optimizer block.
- ▶ When the Optimizer block's icon is highlighted, release the mouse.

This puts the variable into the first row of the Optimizer's Variables table.

- ▶ Close the Holding Tank's dialog.

 The Optimizer block will highlight when a cloned variable can be dropped onto it. Starting with the first row, each cloned item is automatically placed into successive rows of the Variables table.

#### *# deliveries*

Unlike the other required variables for this example, the number of deliveries is not directly available as a dialog item. However, that value can be calculated using the frequency of deliveries, as you will see on page 577.

- ▶ To get the repeat delivery time, open the Lookup Table block labeled "Filler truck".
- ▶ From the Table tab, use the Clone Layer tool to drag the parameter value (but not its label or checkbox) for the **Repeat table every** field onto the closed Optimizer block.

#### *#drinks*

The decision variable for the number of drinks per tank is also located in the Lookup Table block labeled "Filler truck".

- ▶ Clone the data table from the Lookup Table block's Table tab and drag it to the closed Optimizer block.
- ▶ Close the Lookup Table's dialog

#### *time empty*

The model calculates the amount of time that the drink tank is empty; this is an output variable.

- ▶ Open the Holding Tank block labeled "Empty time".
- ▶ Clone the parameter value for **Current level** onto the closed Optimizer block.
- ▶ Close its dialog.



**Setting limits for the variables**

Now that the Optimizer has the necessary variables, you need to enter limits for some of the variables so the Optimizer will know that it should try to change them. (Variables without limits are considered outputs from the simulation and the Optimizer will not try to change them.) For data tables, you also need to specify which cell is to be used in the equation.

- ▶ Open the Optimizer’s dialog so that the Variables table in the Objectives tab is visible.
- ▶ On the first row of the Variables table (“Total sales”) do not enter any limits. This value is not an input to be changed but rather an output value from the Holding Tank. So that it will be more easily understood in the equation, in the Equation Variable column change the variable’s name from **Var0** to **numSold**.
- ▶ On the second row (“Filler truck Repeat\_prm”), which is the time between deliveries, enter a Minimum Limit of **30** minutes (an estimated minimum) and a Maximum Limit of **480** minutes (the simulation end time). Also, in the Equation Variable column change the variable’s name from **Var1** to **deliveryTimes**.
- ▶ On the third row (“Filler truck Data\_tbl”), you need to tell the Optimizer which cell of the data table to use and what its limits are:
  - ▶ Click on the “Filler truck” cell to open the Lookup Table block’s dialog. Notice that the cell that holds the number of drinks per tank is at row 0 and column 1 of the data table (data table rows and columns start at 0). For the Row, Col value in the Optimizer, enter **0,1**.
  - ▶ For this variable’s limits, enter a Minimum Limit of **1000** and a Maximum Limit of **8000** drinks.

☞ Do not enter a decimal point. The absence of a decimal point tells the Optimizer that the value for the number of drinks has to be an integer.

- ▶ Change the variable’s name from **Var2** to **delTankSize**.

On the fourth row (“Tot empty time”) do not enter any limits because this value is a model output. Change the variable’s name from **Var3** to **emptyTime**.

The columns of the Objectives table should now look similar to the table on the right.

	Equation Variable	Block	Block Variable	Row, Column	Minimum Limit	Maximum Limit
0	numSold	Total Sales	contents			
1	deliveryTimes	Filler Truck	Repeat_prm		30	480
2	delTankSize	Filler Truck	Data_tbl	0,1	1000	8000
3	emptyTime	Empty Time	contents			

**Entering the objective function**

Now that the limits have been entered, substitute each variable’s name for the row that holds the value of interest. The result is:

$$\text{MaxProfit} = 2.50 * \text{numSold} - \# \text{deliveries} * (1000.0 + \text{delTankSize} * 1.00) - \text{emptyTime} * 100;$$

The only variable from this equation that has not yet been specified is the **#deliveries** factor, which can be calculated using the **deliveryTimes** variable.

The form for an equation to convert the deliveryTimes variable to the #deliveries factor is:

$$\# \text{deliveries} = \text{int}((\text{endTime} - 1) / \text{deliveryTimes} + 1)$$

The explanation for how this equation is structured is as follows:

How To

EndTime is a global variable representing the end time of the run, in this case 480 minutes. DeliveryTimes is how frequently the deliveries are repeated, or every 240 minutes. If you were to divide the endTime value of 480 by the deliveryTimes value of 240, it would result in 2 deliveries a day, which appears correct. However, if you instead divide by 250, you get only 1.92 deliveries, even though you still have 2 deliveries, one at the beginning of the day and one at 250 minutes. There will always be a delivery at the beginning of the day, so you need to add 1 to deliveryTimes and truncate any fractional result with the `int()` function. This results in 2.92, which when truncated gives 2, the correct answer.

If you had a delivery every 480 minutes, there would be 2 deliveries, but the last one would happen at the same time as the stand closes. To remedy this, reduce the endTime by 1, preventing the 480 minute case (the maximum limit) from causing 2 deliveries.

The entire objective function can now be entered in the Optimizer block's Objectives tab:

- ▶ Delete the default equation from the Equation pane.
- ▶ Define a new variable for the number of deliveries by entering it in the Equation pane (don't forget the semicolons that are needed to end the statement):

```
Integer numDeliveries;
```

- ▶ Below that variable definition, enter the equation that converts deliveryTimes into numDeliveries:

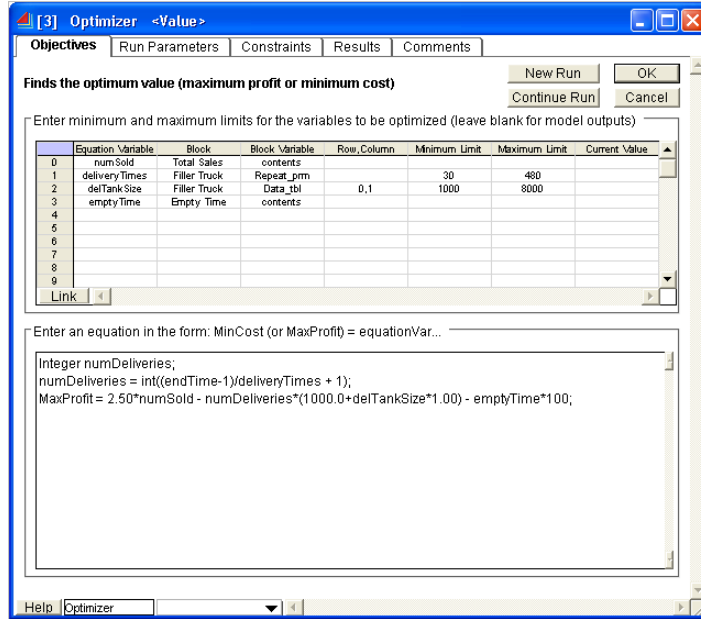
```
numDeliveries = int((endTime-1)/deliveryTimes + 1);
```

- ▶ Below the conversion equation, enter the cost equation:

```
MaxProfit = 2.50*numSold - numDeliveries*(1000.0+delTankSize*1.00)  
- emptyTime*100;
```

The variable definition and the two equations are the objective function for the Optimizer.

The finished Objectives tab for the Optimizer block looks like:



Optimizer dialog, Objectives tab

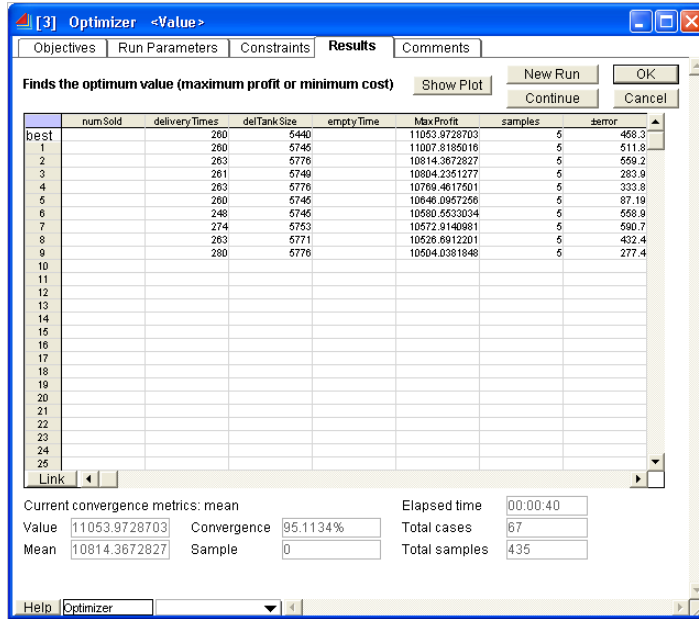
### Running the optimization

- ▶ Open the Optimizer block's dialog.
- ▶ Select the Run Parameters tab. Since this model has random elements, click the *Quicker Defaults* button in the Random model section. This quickly sets up all the parameters for a stochastic (random) model needing multiple samples, but limits the number of samples by default so you can get results more quickly.
- ▶ Run the optimization by clicking New Run in the Optimizer's dialog, clicking the Run Optimization tool on the toolbar, or by giving the command Run > Run Optimization.

While the optimization run is progressing, notice how the MaxProfit value is increasing in the top rows of the table in the Results tab. Notice also how the plotter shows the MaxProfit and convergence values increasing.

- ☞ The optimization will complete faster if you close the Optimizer block. You can leave the Optimization Value plotter open to watch the values converge without slowing the run down.

When the run is finished, the Optimizer opens and displays the Results tab.



Optimizer dialog, Results tab

The best values for the selected parameters (drink size and delivery times) will have already been placed into the model, the drink tank is never empty, and the profit will be maximized. The results of the run as seen on the Objectives and Results tabs are:

- NumSold (total sales) is blank because it has not been changed; it is an output from the model.
- DeliveryTimes (Filler truck repeat prm) = around 250 minutes between deliveries (2 deliveries).
- DelTankSize (Filler truck data\_tbl) = around 5200 drinks per tank delivered.
- EmptyTime (Empty time contents) is blank because it is an output value from the model.
- MaxProfit (on the Results tab) = around \$12,000.

Your results will vary a little from those shown in this guide because of the randomness of the model and because you have set the convergence and samples for a *quick* result.

As mentioned earlier, it is a good idea to run the model with optimization additional times to make sure that the answers agree or are at least close. This provides assurance that the first answer is not a false or sub-optimal.

### Adding constraints

The optimization results indicate that the profit would be maximized if you ordered tanks that hold approximately 5200 drinks and if the second tank were delivered about mid-day. But what would happen if the drink distributor only had specific size tanks and delivery times? By constraining a parameter, values that fall outside the constrained boundaries are not considered as part of the possible solution space. You can constrain parameter values in almost an infinite number of ways by entering constraint equations in the Optimizer's Constraints tab.

There are two types of constraints:

- *Individual constraints* are applied only to the specific variable and cause it to be changed in some manner. For example, an individual constraint could cause all tank sizes to be in increments of 1000.
- *Global constraints* can cause an entire set of parameters to be rejected so that the Optimizer will try a different set. For instance, a global constraint could reject a solution set if the sum of all the variables was greater than a certain value.

For this model the constraints to consider are:

- Delivery tanks only come in sizes that hold 1000, 2000, 3000, 4000, 6000, or 8000 drinks.
- Tanks can be exchanged every 30 minutes except...
- Tanks with 6000 drinks or greater can't be exchanged more often than every 60 minutes.

 For comparison purposes, a model with constraints titled Optimize 3 is located in the \Examples\Tutorial\Optimization folder.

#### **The constrained tank size**

The individual constraint equation to granularize the drink tank sizes recalculates `delTankSize` as follows:

```
// round to delivery amounts (e.g. 1k, 2k, 3k, 4k, 6k, 8k)
if (DelTankSize<=4000) DelTankSize = int(DelTankSize/1000.0 +
0.5)*1000.0;
else DelTankSize = int(DelTankSize/2000.0 + 0.5)*2000.0;
```

 Comments are preceded by `//` and are omitted from the calculation.

This equation puts the tank sizes of 4000 or less into multiples of 1000, and tanks sizes greater than 4000 into multiples of 2000, using an individual constraint. For example, if the Optimizer suggests a size of 3300, the equation would convert it to a potential tank size of 3000. If the suggested size were 5200, the ELSE part of the IF statement would round it up to 6000.

#### **The constrained delivery time**

The individual constraint equation to granularize the delivery times is:

```
// round fillup time to listed delivery times (e.g. 30 minutes)
DeliveryTimes = int(DeliveryTimes/30.0 + 0.5)*30.0;
```

The delivery time is converted to multiples of 30 minute intervals using an individual constraint, similar to how the delivery tank size was converted, above.

#### **Global constraints for 6000 and 8000 drink tanks**

Although the preceding equation constrains the delivery times for most cases, you also need to reject cases where the drink tank is 6000 drinks or more and the delivery time is less than 60 minutes.

Rather than the individual constraints used in the previous two equations, for this equation you need a global constraint. Delivery times for tanks of 6000 and 8000 drinks are not limited to multiples of a specific number as was true for the tank size; they can be any value as long as it is greater than or equal to 60. So the solution space is somewhat unlimited. In addition, it would not be valid to just round up a delivery time that is less than 60. If the equation rounded up all the poten-

tial delivery times below 60 to exactly 60, as the preceding delivery time equation does, it would cause a severe bias toward getting deliveries every 60 minutes. Instead the equation needs to reject the entire solution set if the delivery time is below 60. This will cause the Optimizer to use a new random delivery time to generate a different set of solutions that will be less biased.

- ☞ If you use too many global constraints, or constraints that are too restrictive, the equation would unnecessarily reject almost all cases and the Optimizer could take too long to run or might fail to reach an acceptable solution.

The global constrained equation for deliveries of 6000 or more drinks is:

```
// can't deliver 6000 or more drinks sooner than 60 minutes apart
if (DelTankSize >= 6000 && DeliveryTimes < 60)
    Reject = TRUE;
```

The Optimizer pre-defines “Reject” as a special variable to be used only with global constraints. The Reject variable, if set to TRUE, will reject that case and cause the block to calculate another possible case that could be acceptable. If Reject is not set to TRUE, the current case will be used for the next series of runs.

### ***Enter the constraint equations***

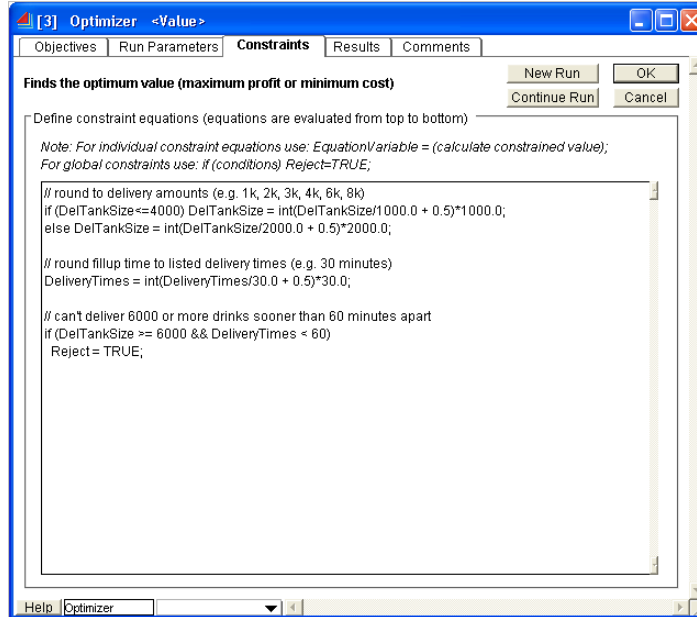
In summary, the equations to enter on the Constraints tab of the Optimizer block are:

```
// round to delivery amounts (e.g. 1k, 2k, 3k, 4k, 6k, 8k)
if (DelTankSize<=4000) DelTankSize = int(DelTankSize/1000.0 +
    0.5)*1000.0;
else DelTankSize = int(DelTankSize/2000.0 + 0.5)*2000.0;

// round fill up time to listed delivery times (e.g. 30 minutes)
DeliveryTimes = int(DeliveryTimes/30.0 + 0.5)*30.0;

// can't deliver 6000 or more drinks sooner than 60 minutes apart
if (DelTankSize >= 6000 && DeliveryTimes < 60)
    Reject = TRUE;
```

After entering the equations in the Optimizer block, the Constraints tab should look like:



Optimizer dialog, Constraints tab

### Running the optimization

Click the New Run button in the Optimizer dialog or give the command Run > Run Optimization. It is a good idea to run the optimization with the new constraints multiple times to make sure that the first answer is not a false or sub-optimal.

Because there are random elements in the model, the results will differ from, but should be close to, the following:

- NumSold (total sales) is blank because it has not been changed; it is an output value from the model.
- DeliveryTimes (Filler truck repeat time) = 270 minutes between deliveries (2 deliveries a day).
- DelTankSize (Filler truck data\_tbl) = 6000 drinks per tank delivered.
- EmptyTime (empty time) is blank because it is an output value from the model.
- MaxProfit = around \$10,500.

Note that the profit may have decreased slightly from the previous optimum results. This is due to the constraints put on the model parameters.

### Using the Optimizer block

This section examines, in detail, how to get the most out of the Optimizer block.

#### Variables table

When a dialog's variable is dragged using the Clone Layer tool onto a closed Optimizer block, the Optimizer block places that variable into its Variables table. This table holds the variables needed in the Optimizer's cost or profit equation. These variables can be of two types:

- Decision variables to be optimized. These need their lower and upper limits entered.
- Output variables from the model. These should have no limits entered.

Both lower and upper limits for a decision variable need to be entered directly in the Variables table. Variables that are outputs from the model, such as *Exited* from an Exit block (Item library), should not have any limits entered or the Optimizer will attempt to incorrectly change them.

☞ If the variable is an integer type, such as a number of machines, **do not enter a decimal point** in the limits. Conversely, variables that need to have real values, such as a delay time that could vary from 1.0 to 2.2, need both limits entered **with decimal points**.

### Specifying data table cells in the Variables table

If a cloned data table is dragged to the closed Optimizer block, the *Row, Col* indexes need to be entered, separated by a comma. Since data tables are zero-based and start at row 0 and column 0, the first cell would start at 0,0.

For example, to use the first row and second column cell of a data table, enter the Row,Col value **0,1** where 0 is the first row and 1 is the second column.

☞ The Variables table's Row,Col column is only used to access a cell within a data table.

### Objective functions

Optimizations can maximize profit, minimize a cost, or approach a constant, depending on the form of the equation. Just like the equation field of the Equation block (Value library), optimization equations can be multi-lined, define variables, have control constructs (i.e. IF-ELSE, FOR...), and call ModL functions. To see more information on equations, see “Equation-based blocks” on page 601.

#### Maximizing profit or minimizing cost

For example, you may want to optimize the number of machines used to maximize the profit. In that case, you can set up an equation:

```
MaxProfit = thruput*dollarsPerUnit - numMachines*dollarsPerMachine;
```

Or, you might want to write the equation as a cost, and minimize that cost:

```
MinCost = numMachines*dollarsPerMachine - thruput*dollarsPerUnit;
```

The variable names used in the MaxProfit or MinCost equations come from the **Equation Variable** column of the Optimizer's Variables table. Depending on how you write the equation, the Optimizer will either maximize or minimize the equation value by changing the values of the decision variables until the correct condition converges.

#### Approaching a constant

A special case for an objective function would be that the result needs to approach a constant K as close as possible. In this case, you can write a profit equation.

For the case where the equation should approach a nonzero constant K, the form would be:

```
// maximizes when equation equals constant
MaxProfit = 1.0 - RealAbs(1.0 - equation/K);
```

Note that the RealAbs(x) function returns the positive absolute value of x.

For the case where the equation should approach zero, the form would be:



```
// this becomes maximum when equation equals zero
MaxProfit = ConstantValue - RealAbs(equation);
```

In this case, ConstantValue should be small, but large enough so that most of the population MaxProfit results are *initially* positive. If ConstantValue is too small, the convergence calculation will fail because there will be both negative and positive values in the final population results. If ConstantValue is correct, all of the values will tend to become positive as the system converges, and the convergence calculation will then be valid. If ConstantValue is too large, the convergence calculation will tend to be insensitive and high all of the time, causing a premature end of the optimization run.

These techniques can be adapted to solve any “approaching a constant” type of problem.

### Run Parameters tab

In most cases, clicking the *default* buttons for the type of model you are optimizing (random or non-random) will quickly set all of the parameters in this tab to useful values.

Two of the parameters are especially important to convergence of the optimization:

- *Maximum Samples per Case*: The maximum number of runs averaged to get a member result. For non-random models, this should be 1. For random models, this needs to be high enough to get a useful mean value at the expense of run time. The Optimizer block starts the number of samples at 1 and increases them with each generation until it reaches the maximum. Sometimes it is useful to reduce the maximum number of samples (possibly to 5), to get a rough idea of the best solution without spending too much time running samples. Most of the time, a useful result will occur and, even if it is not the optimum one, it will be close.
- *Terminate if best and worst within (percent)*: This value is used to examine the current population of results to see if they are within a specified percentage of each other. The default of 0.99 might not be high enough for a precise answer in a noisy model. Increasing this value (i.e. 0.9999) will cause the optimization to continue until the population converges more closely, increasing the likelihood of a more optimum answer at the expense of run time.

### Constraints

When building a model, there are almost always some parameter constraints that have to be satisfied. The two types of constraints are *individual constraints* and *global constraints*. The Optimizer block makes it easy to apply virtually any kind of constraint to a model’s parameters.

#### Individual constraints

Individual constraints are used to change a decision variable’s value if the value has to be limited or if it depends on the values of other decision variables. These constraints are entered as equations, usually with IF or IF-ELSE statements. For example:

```
if (NumQueueSlots > 7)
    NumSlots = NumActivities+3; // change it
```

Sometimes you might need the IF-ELSE form:

```
if (Var2 >= 3)
    Var2 = Var3-Var4;
else// var2 was less than 3
    Var2 = Var5/2;
```

In some cases you just need to modify the value of a variable. For example, if you need to constrain its values to multiples of 0.5 (i.e. 1.0, 1.5, 2.0). You do this by multiplying the variable by 2, adding 0.5 before the Int() function truncates it so that it rounds it to the nearest integer, and then dividing by 2.0, forcing the result to floating point values that are granular to 0.5:


```
Var2 = Int(var2*2.0+0.5)/2.0; // 0.5 Granularity
```

In any case the newly calculated Var2 value will replace the old Var2 value.

#### Global constraints

Global constraints are useful to reject an entire case if any or all of the decision variables don't meet a specific criteria. Global constraints are entered as equations, usually with IF statements, to assign the value TRUE to the variable REJECT if the variables are not within the constraint. They are entered like this example:

```
if (Var4+Var5 > 7)
    Reject = TRUE; // only reject if the sum is too large
```

 Reject is a special optimization variable for use with global constraints. If set to TRUE, it will reject that case and cause the block to calculate another possible case that could be acceptable. If Reject is not set to TRUE, the current case will be used for the next series of runs.

Sometimes you might need a more complex form:

```
if (Var4+Var5 > 7 || Var4 < 2) // the || means OR, && means AND
    Reject = TRUE;
```

In any case, any global constraint will abort that particular case and the Optimizer block will keep attempting to create cases until the global constraint doesn't set REJECT to TRUE. It will try to create 500 new cases before it gives up and prompts the user with an error message. If this occurs, the global constraint is probably faulty.

#### Interpreting results

The Results tab shows the entire population of solutions, sorted with the best one on top (row 0). As the optimization progresses, new and better solutions will replace inferior solutions in the population table.

If the optimization terminates for any reason, either via normal convergence or running for the maximum number of generations, the best solution set found so far is automatically placed in the model.

#### Stat::Fit (Windows only)

In simulation models, it is often useful to characterize a random input (for example, inter-arrival times and demand rates) using a probability distribution. Typically, this involves obtaining historical data that documents the system's behavior, then analyzing the data to determine an appropriate distribution to represent it. There are two advantages to using statistical distributions rather than raw historical data as inputs to a model:

- Values for input random variables are not limited to what has happened historically.
- For continuous distributions, an infinite pool of data exists. With historical data, there is seldom enough collected data to support multiple simulation runs.

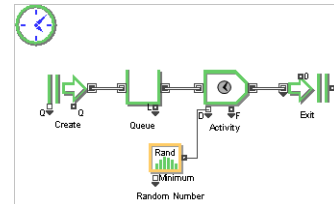
Stat::Fit is a software package from Geer Mountain Software ([www.geerms.com](http://www.geerms.com)) that helps the analyst determine which distributions, if any, offer a good fit for the underlying data. ExtendSim has an interface so you can easily access the power of Stat::Fit.

Stat::Fit is a Windows application included with the ExtendSim AT and ExtendSim Suite products. It can be purchased separately for use with other ExtendSim products.

### Tutorial

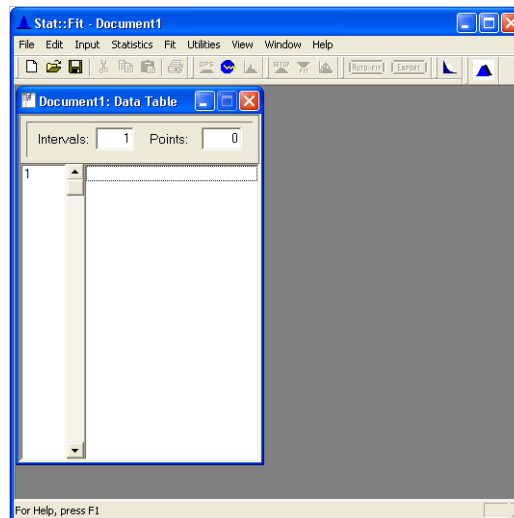
The StatFit Example, shown on the right, is a discrete event model. It has Stat::Fit choose a distribution for a Random Number block (Value library) using a pre-built text file of data.

By default, the Random Number block is set to the Uniform Real distribution with a Minimum of 0 and a Maximum of 1. The 32 historical data points in this project will be used by Stat::Fit to define which distribution and associated parameters might be more appropriate.



StatFit Example model

- ▶ Open the StatFit Example model located in the folder \Examples\Tips\Modeling Tips.
- ▶ In the Distribution Fitting tab of the Random Number block, click **Open Stat::Fit**.
- ▶ The Stat::Fit application should appear on your screen with a new blank document.



Stat::Fit application window

- ▶ In Stat::Fit, select File > Open.
- ▶ Find and select the file **StatFitEx.sfp**. It should be in the same folder (\Examples\Tips\Modeling Tips) as the StatFit Example model.
- ▶ In Stat::Fit, choose Fit > Auto::Fit or click the Auto::Fit button in the toolbar.
- ▶ In the Auto::Fit window, click OK to use the default settings.

After Stat::Fit does some computation, a window appears displaying a list of parameterized distributions that have been ranked according to goodness of fit.

- ▶ In Stat::Fit, choose File > Export > Export Fit or click the Export button in the Stat::Fit toolbar.

- ▶ Select “Extend” under the application window (it should be the default), then choose the desired distribution (there should only be one).
- ▶ Click OK and return to ExtendSim. (You may exit Stat::Fit if you wish.)

The Random Number block’s dialog should reflect the Triangular distribution and parameters Stat::Fit selected.

Additional Stat::Fit documentation is available in Stat::Fit’s Help menu and in the SF Manual V2.pdf file located within the ExtendSim7\Documentation folder.


## Plotters

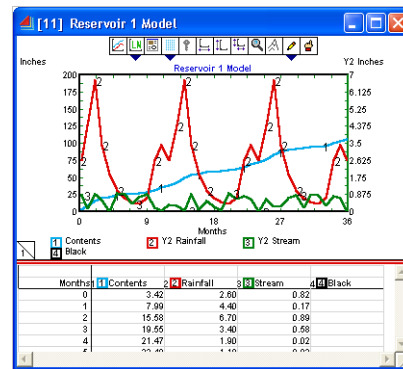
Most of the ExtendSim plotters have many features in common. The Reservoir model in the Tutorial module shows some basics of how to use plotters. This section describes the ExtendSim plotters in more detail and shows how to make plots appear the way you want. All plotters are located in the Plotter library.

Depending on their type, plotters allow you to plot from 1 to 6 traces (plotted lines) at a time. If you build your own plotter using ExtendSim’s ModL language, you can plot up to 100 traces on a single plotter. Your choice of plotter type depends on the type of model you build (continuous, discrete event, or discrete rate) and how you want information plotted (such as batch runs, histogram, and so on). Plotter types are described later in this chapter.

You can have more than one plotter in a model and you can place plotters at any location in the model. Usually plotters open automatically when the simulation is run. If the plotter is set to not open (as discussed in “Plotter dialogs” on page 592), you can open it by double-clicking its icon.

### Plot and data panes


When you open most ExtendSim plotters, instead of the typical dialog there is a plotter window with two panes. The upper pane shows the plot (one or more traces) and the lower pane shows the data for that plot in a table. The two panes are separated by a *split bar*. Initially, about two thirds of the window is devoted to the plot and one third to the data. You can change this by moving the cursor over the split bar until it changes to , and dragging up or down.



Plotter I/O panes

### Plot pane

The plot pane has four pages. If the plotter was open during the simulation run, the four most recent runs will automatically be saved on those pages.

 If the plotter was not opened during the simulation run, only the current plot image is saved in the plot pane. In that case, use the Push Plot tool (described on page 592) to save additional images.

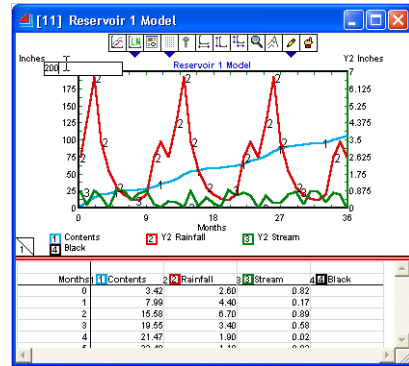
Click on the number at the lower left of the plot pane to see each page. If the Key On/Off tool (described on page 591) was on when the simulation ran, the key will be shown on the pages.

To see the exact numbers that generated a point, scroll through the numbers in the data pane. Or simply put the cursor over the data item you want; the top row in the data pane displays the values that match that point. (In scatter plots, this shows the X and Y values of that point.)

Change the plot axis labels and limits directly in the plot pane, as follows:

- ▶ Select the item to be changed and type the new text. For example, to change the maximum Y-axis value, click that number and enter a new number.
- ▶ To finish, press Enter or click somewhere else in the window.

To change many of the labels and limits, press Tab after entering each new value. If you use the Tab method to change items, the plot will redraw only after you press Enter or click in the window.



Changing maximum Y-axis value

**Data pane**

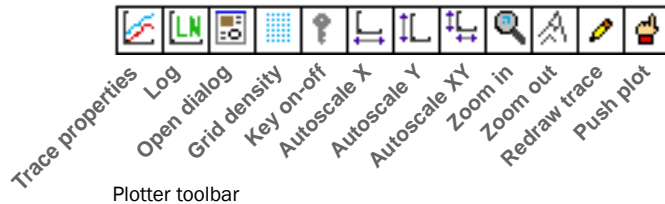
Resize the columns in the data pane by placing the cursor directly over the column divide, and dragging the divide to the desired location while holding down the mouse. You can also tab through the data or use the arrow keys. If you select a section of data, tabbing will move you just through the selection. Notice that only the data for page 1 of the plot is displayed in the table.

If you change the data in the data pane, those changes are reflected in the plot. You can change numbers, paste in rows, and so on. Use this capability to view how various data would be plotted, or to plot a reference line.

The label headers for the columns are changed using the Trace properties tool, described on page 590. The color, pattern, and symbol for each trace is displayed to the left of its label.

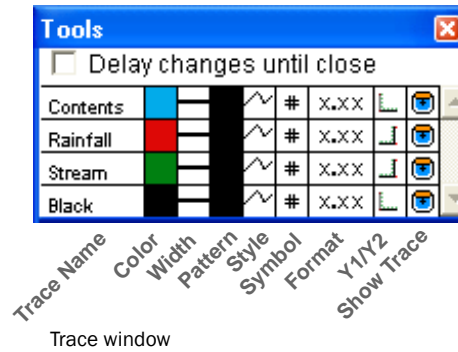
**Plotter tools**

Note the tools at the top of the plot pane. These are used to change the traces and views for the plots, change the labels and formatting for the data being plotted, and to control how and when the data is displayed. Tools with arrows beneath them are popup menus with multiple choices. The toolbar in plotter windows looks like:



**Trace properties tool**

This tool lets you change many aspects of the traces and data. It brings up a small window that looks like:



Trace window

If you check the *Delay changes until close* checkbox, changes you make will occur in the plot only after the Trace properties tool is closed. This is useful when you are making a lot of changes and don't want the traces to be redrawn after each change.

Below the checkbox are nine columns of choices.

The *Trace Name* text at the left describes the traces and is used to label the columns in the data pane. In a discrete event plotter, each trace has two sets of labels: the name of the trace and Time. In a continuous plotter, each trace will only have one label, as shown above. To change a label, click on the text in the Trace Name area and type the text you want.

The *Color*, *Width*, and *Pattern* choices describe how the trace looks. To change one of these items, click on the box and choose from the list. You can select from seven standard colors or choose a custom color. There are also five line widths and four patterns available.

The *Style* choice affects the manner in which the trace is drawn.

- The top style indicates that the data points are connected by diagonal lines (interpolated)
- The middle style (the default) indicates that points are connected by horizontal and vertical lines (stepped).
- The bottom style indicates that you want the points plotted with no lines between them.



Trace style choices

There are also many *Symbol* types, such as dots, squares, and circles, as well as a trace numbering choice (marked #). If you choose a non-numeric symbol, it will be drawn at each point on the trace. If you choose the # symbol, the number for that trace will be spaced evenly along the trace. Except for numbers, each symbol is drawn using the trace color, width, and thickness settings; numbers are drawn in black.

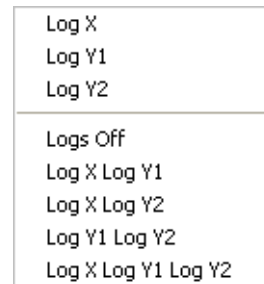
The *Format* option sets the number format that is used for the data in the data pane, including Time for discrete event plotters. The four choices are General, x.xx (decimal), xxx (integer), and x.xex (scientific notation).

The *Y1/Y2* choice tells which axis to plot the numbers against. The default choice, , plots values on the Y1 (left) axis. Clicking this changes it to , the Y2 (right) axis.

The *Show Trace* choice tells whether to display the trace at all. If you click on this choice, it turns to a closed eye and the trace is not drawn. This is useful if you have many traces and you temporarily want to hide one to make the chart clearer or if one trace is on top of another.

**Log tool**

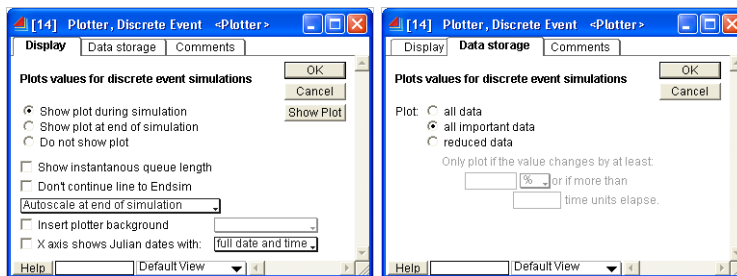
The Log tool lets you choose whether to make one or both axes use a logarithmic scale. Click the tool to display the popup menu.



Log choices

**Open Dialog tool**

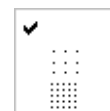
Click the Open Dialog tool to show the plotter's dialog. You use this tool to make changes to the way the plotter operates. For example, to not have the plot show during simulation runs or to plot every nth point. The Plotter, Discrete Event block, for instance, has one tab that controls how and when data is displayed and another tab that allows you to specify the granularity of the plotted data. The choices in the plotter dialogs are described in their Help and in "Plotter dialogs" on page 592.



Display and Data Storage tabs of Plotter, Discrete Event block

**Grid density tool**

The Grid Density tool lets you specify the type of grid behind the plot. Click the mouse on the tool to display the menu. The top choice indicates no grid, the middle choice a light grid, and the bottom choice a dense grid.



Grid choices

**Key on-off tool**


Clicking this tool turns on and off the key at the bottom of the plot pane. The labels, colors, patterns, and symbols that the key shows come from the entries in the Trace properties tool's window. The key is handy if you are not viewing the data table at the bottom of the plotter window or if you copy the plot to another document. If the key is on when you run each simulation, the key will show on each plot page.

**Autoscale tools**


The three autoscale tools are used to manually scale the axes to fit the data after the simulation has run. The first tool scales the X axis to fit all values that were measured in the simulation without changing the Y axis. The second tool scales the Y axes to fit all values that were measured in the simulation without changing the X axis. The third tool scales both axes at once. This is especially useful if a run of the simulation goes off one side of the plot. You can manually autoscale the axes while the simulation runs.


Note that you can also choose to have the data automatically scaled at the end of the simulation run, during the run, or not at all. You do this using the “autoscale” popup menu in the Display tab of the Dialog Open tool. By default, most plotters autoscale at the end of the simulation run.

**Zoom in and Zoom out tools** 

There are many times you want to see a particular part of your plot in more detail, or want to zoom out and look at a larger area of the plot. The Zoom in tool magnifies an area of the plot. After you click this icon, the cursor becomes . Drag the cross-hairs over an area of the plot and release the mouse to zoom in to match the described rectangle. The Zoom out tool zooms out by changing the axes by a factor of 2 in each direction. After using the Zoom in or Zoom out tools, you can use the Autoscale tool or the Edit > Undo command to reset your axes.

**Redraw trace tool** 

The Redraw trace tool lets you change your data on the plot and see the results in the plotter’s data table. Click and hold down the mouse on the tool to display the menu. Choose one of the four traces that you want to change. The cursor becomes a pencil, . You use the pencil to redraw the data that was plotted. After you are done drawing this, you can see the values for the drawn data by looking in the data table. The Redraw trace tool is only available in continuous, non-scatter plotters, like the Plotter I/O.

**Push plot tool** 

Each time you run a simulation with the plotter open, the previous run is automatically saved onto page 2 of the plot. Changes made to the plot after the simulation run (such as autoscale, magnify, and turning the key on) are not automatically saved. Also, if the plot was closed during the simulation run, the plot image from the last run will not be saved to page 2. After you change the view of a plot or run a simulation with the plotter closed, you may want to save a copy of that plot in one of the plotter’s plot pages. To do that, click the Push Plot tool.

For example, assume that you are looking at a magnified portion of the plot and you want to save that magnified view before running the simulation again. Use the Zoom In tool, then click the Push Plot tool to keep a record of that view. Then choose Edit > Undo Zoom in to restore your original axis values, and run the simulation again.

When you use this tool, the current plot image is saved onto page 2. The previously saved images are each pushed up one page so that the image that was on page 4 is discarded.

**Plotter dialogs**

ExtendSim comes with many types of plotters, as seen in “Types of plotters” on page 593. Plotters have a dialog that you can access by clicking on the Open dialog tool at the top of the plotter. Choices that some of the plotters have in common are:

Choice	Description
Show plot during simulation	Determines if the plotter window is automatically opened during the simulation. For example, you probably do not want the plotter to open if you are running the model with animation on.
Show plot at end of simulation	Display plotter window only when simulation has been completed.
Do not show plot	Plotter window remains hidden until the plot icon is double-clicked.



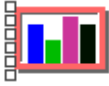
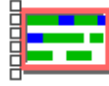




Choice	Description
Show instantaneous queue length	For discrete event plotters, plots additional data to show when a queue changes length in zero time. For example, when an item enters and leaves the queue during one event.
Don't continue line to Endsim	For discrete event plotters, stops drawing the plot trace at the last event, not at the End Time of the model run.
Show plot	Brings the plot window to the front if it is already open, and opens it if it is closed.
Autoscaling	Automatically resizes the Y-axis of the plot based on the maximum and minimum values observed. The plot can automatically scale during the simulation, at the end of the run, or not at all.
Insert plotter background	When checked, enables a popup for selecting a background for the plot pane. You can copy custom plotter backgrounds into the Extensions\Pictures folder. They must be windows metafiles (.wmf) or bitmaps (.bmp) for Windows or PICT resources in a Macintosh resource file for Mac OS.
x axis shows Calendar dates with	If <i>Use Calendar dates</i> is checked in the Run > Simulation Setup > Setup tab, this option expresses time units on the plot pane in Calendar date format, displaying full date and time, no time, or time only. For more information, see "Calendar dates" on page 528.
Plot every nth point	In continuous plotters, lets you specify that the plotter should only draw the nth point of the data it receives. This is useful if there are many data points and the points are all very close together, or to make the plot draw faster.
Data storage tab	The choices on this tab are to plot all data, all important data, or reduced data. This is useful if you have plotters that store too much data, as some of it may be redundant. The Reduced data choice samples from the incoming stream, conserving memory.

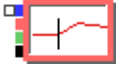

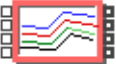




### Types of plotters


Plotters are located in the Plotter library. As indicated in the following table, each plotter has a distinct purpose. Some can only be used in continuous models ("C"), some only in discrete event or discrete rate models ("DE/DR"), others in both ("Both").

- ☞ Discrete Event plotters are used in discrete event models. However, since they plot values not items, you must connect a value output (as opposed to an item output) to the Discrete Event plotter. This tells the plotter what information about the items or about the status of the model you want to plot.

The MultiSim and Error Bar plotters are specifically designed to be used when you run multiple simulations for Monte Carlo or sensitivity analysis. You choose the number of times you want the simulation to run in the Run > Simulation Setup dialog.

Plotter	Description	Model
Bar Chart 	<i>Windows only.</i> Displays a bar graph of up to six input values. The bars can change instantaneously or at regular intervals. Unlike other plotters, the bar chart does not record any data. It uses the IOComp ActiveX control to display the graph.	Both
Gantt Chart 	<i>Windows only.</i> Plots up to six Gantt chart bars. Each bar displays one of the following types of information: <ul style="list-style-type: none"> <li>• The binary status of a variable (on or off)</li> <li>• The level of a variable (the height of the bar varies with the level)</li> <li>• The status of an Activity or Workstation block (Item library)</li> </ul> It uses the IOComp ActiveX control to display the graph.	Both
Histogram 	Creates a histogram of all the values it receives. Each bin counts the number of data values that fall within its range, or the amount of time that the incoming value was in that bin (if the Time Weighted option is used). The number of bins and the overall minimum and maximum range values are specified in the plotter's dialog. The maximum and minimum can be the entire range received or specified numbers. You can specify in the dialog whether to plot the data from each step or, if you run multiple simulations, to plot the final values of each run. The dialog also displays the number of points that fall within the specified range.	Both
Plotter, DE Error Bars 	Shows the mean and standard deviation of a value. This plotter is used when running multiple discrete event or discrete rate simulations, such as for Monte Carlo modeling or for sensitivity analysis. The time line is divided into a number of equal intervals specified in the dialog. You can also choose to use time-weighted statistics. The plotter calculates the average mean of the value over each interval and from this determines the mean and standard deviation over all of the runs.	DE/DR
Plotter, DE MultiSim 	Accumulates the values from up to four runs of a discrete event or discrete rate simulation on a single plot pane and table. In the dialog, you can choose to only plot values when they change or to plot all values. You can also specify by number which trace to show the current run on and whether to automatically increment the trace number.	DE/DR
Plotter, Discrete Event 	Gives plots and tables of data for up to four value inputs in discrete event and discrete rate models. Both the value and the time the value was recorded are shown in the data table for each input. In the dialog you can specify whether to plot values only when they change or to plot all values. Use the <b>Show instantaneous length</b> option if you attach an input to the <b>L</b> connector of a queue-type block and you want it to report on items that arrive and depart on the same time step (these are items that stay in the queue for zero time).	DE/DR

Plotter	Description	Model
Plotter, Error Bars 	Similar to the Plotter DE Error Bars except it is only used in continuous models. As opposed to the Plotter DE Error Bars, this plotter calculates the mean and standard deviations for the value at the exact point or step in time rather than its average over the time interval.	C
Plotter, FFT 	A specialized plotter used by electronic engineers, it plots both the data that is input and the FFT (Fast Fourier Transform) of the data. When you double-click the FFT icon, it shows a dialog where you can specify the number of FFT points and choose an FFT window.	C
Plotter, I/O 	Gives plots and tables of data for up to four time-associated inputs. In the dialog, you can specify that only every “nth” point is plotted and you can choose whether or not to autoscale the plot during the simulation. The output connectors allow the data generated in one simulation to be used as an input to another simulation. To do this, run the simulation, then copy the plotter into the new model and connect from its output connectors.	C
Plotter, MultiSim 	Accumulates the results of up to four runs of a continuous simulation on a single plot pane and table. You can specify by number which trace to show the current run on and whether to automatically increment the trace number.	C
Plotter, Scatter 	Shows two sets of data plotted as x,y value pairs. You must connect both the x and y inputs of at least one pair in order to plot data.	Both
Plotter, Scatter (4) 	The same as the Plotter, Scatter except that you can plot up to four sets of data. You must connect both the x and y inputs of at least one pair in order to plot data.	Both
Plotter, Strip 	Shows a moving strip chart of a specified number of data points plotted over time. The strip chart moves left as the data appears on the right side of the plotter. In the dialog, you can specify the numbers of points to show on the chart. Because it uses less memory than the Plotter, I/O, this is especially useful if you are running a long simulation and only need to monitor the current conditions.	C

Plotter	Description	Model
	Shows two sets of data plotted as x, y value pairs for a specific number of points. You can specify how many points to show at a time (the worm width) in the dialog. This is like the Plotter Scatter except that points are deleted. Use this if you are generating a great deal of data but do not need to see all of it. You must connect both the x and y inputs of at least one pair in order to plot data.	Both

### Copying plotted information


You can copy the information from the plotter to the Clipboard. If the plotter is the front-most window, simply choose Edit > Copy Plot to copy the picture of the plot into the Clipboard. To copy the data, you must select it, then choose the Copy command.

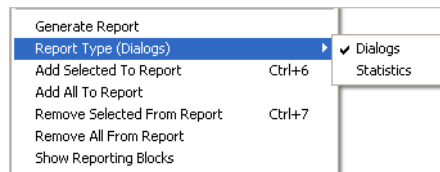
### Clearing plotted information

When the plotter is the active window, clear all the data associated with the current plotter by selecting the Edit > Clear All Plots command. This is useful to reduce the size of a model for distribution or archiving.

## Reports

The report commands in the Run menu are for generating custom reports of model data. The commands let you choose just the blocks you want in the report, or you can choose to report on all the blocks in a model.

 If you build your own blocks and want to use the Report features for debugging, add special code to the blocks as described in the Developer Reference.



Report options

Once the report is generated, you can view or edit the text file in ExtendSim or export the data into another application for analysis or presentation. For example, if you have a discrete event model with queues, you might want to check each queue to see what the maximum queue length was. You could simply look in the report for those queues.

### Types of reports

There are two types of reports that can be generated:

- The Dialogs report includes the final values for the input and output parameters of every chosen block, as well as the information in each block's comments field. Since this report includes the values and settings for all of the parameters in each of the selected blocks, it is a good tool for documenting a model.
- The Statistics report includes the final values for the output parameters only. This report arranges the statistics in tabular form, allowing for easier comparison of block results and exporting to spreadsheets.

 Only one type of report can be generated for a given run.


The currently selected type of report is displayed as part of the Report Type command in the Run menu. Both report types organize the data first by block category, then by block number, allowing you to easily locate the results for a particular block. The reports are saved as text files and open

automatically when the simulation is finished. If you perform a multi-sim run by setting the *Runs* parameter in the Simulation Setup dialog to a number greater than 1, each report is appended to the current report file so that you can compare reports from earlier runs. However, if you perform two consecutive single runs, the reports will be written to separate files or, if you specify the same file name, written over.

Note that model reporting only tells you the final values of the blocks in the simulation. If you want to see the values of the blocks during the simulation, use the tracing commands described in “Model tracing” on page 620.

### Generating reports

Before generating a report, you need to choose which blocks to report on. To include some blocks in a report, select the blocks and choose Run > Add Selected to Report. To have all the blocks in a model be included, choose Run > Add All To Report. To report on fewer blocks in the next simulation run, select the blocks you want out of the report and select Run > Remove Selected from Report; to start over on your selection of blocks, choose Run > Remove All from Report.

 It's highly recommended that you do not include a Plotter block in a report. Plotters write out all report data to a very large file.

You can see which blocks are being reporting on by choosing Run > Show Reporting Blocks; each block in the report displays the word *Report* on its icon.

Reports are opened, closed, and edited just like any other text file in ExtendSim.



Icon for block in report

### Steps for reporting

- ▶ Choose the blocks you want to report, as discussed above.
- ▶ Select a report type using the Run > Report Type command.
- ▶ Choose the command Run > Generate Report.
- ▶ Run the simulation to generate the report. ExtendSim prompts you for a name for the report file.

### Reporting example

For example, the following (edited) reports were generated for Reservoir 1 model's Random Number block for the first two simulation runs.

#### Dialogs report

```
ExtendSim Dialog Report - 8/27/2007 2:24:51 PM Run #0
Block Name: Random Number Block Number: 13 Block Label: Stream
```

```
DistType = Uniform Real
Argument1_prm = 0
Argument2_prm = 1
UseBlockSeed_chk = UnChecked
RandomSeed_prm = 14
RandomResult_prm = 0.396125656921534
GenerateOnce_chk = UnChecked
```

```
*****
```

```
ExtendSim Dialog Report - 8/27/2007 2:24:51 PM Run #1
Block Name: Random Number Block Number: 13 Block Label: Stream
```

```

DistType = Uniform Real
Argument1_prm = 0
Argument2_prm = 1
UseBlockSeed_chk = UnChecked
RandomSeed_prm = 14
RandomResult_prm = 0.371057238775359
GenerateOnce_chk = UnChecked

```

**Statistics report**

```

ExtendSim Statistics Report - 8/27/2007 2:26:01 PM Run #0
INPUTS_____
Block Label                Number Name                Result
-----
Stream                      13      Random Number0.22989
*****


ExtendSim Statistics Report - 8/27/2007 2:26:01 PM Run #1
INPUTS_____
Block Label                Number Name                Result
-----
Stream                      13      Random Number0.70181

```

For the Statistics report, “Inputs” is the block’s category in the Library menu.

You can edit the report files just as you would any text file in ExtendSim; this is described in “Text files” on page 663. If you program custom blocks, you can specify what data gets written in the report as discussed in the Developer Reference.

When printing the Statistics report onto a standard 8.5" by 11" sheet, to ensure that the entire page width is printed, use the Text command to change the font from the default size **12** to size **9**. The default size is set for easy on-screen viewing, but will exceed the page margins.

 Reports can also be used as a debugging tool, but you can get more detailed information for debugging by performing model tracing. See “Model tracing” on page 620.

# How To

## Math and Statistical Distributions

Working with equations and distributions

*“I know that two and two make four - & should be glad to prove it too if I could –  
though I must say if by any sort of process I could convert 2 and 2 into 5  
it would give me much greater pleasure.”*  
— George Gordon Noel Byron

ExtendSim provides an extensive palette of tools for integrating mathematical equations and randomness in a model. You can even control the exact moment that calculations will take place. This chapter discusses:

- Blocks that provide mathematical functionality
- Using Equation blocks to create custom equations
- Incorporating randomness in models
- Selecting a probability distribution
- Determining when inputs should be integrated or summed

☞ See also the How To chapter titled “Analysis” on page 563.

### Blocks that represent functions

ExtendSim libraries are toolkits of blocks for quickly building a graphical representation of model logic. As shown below, some blocks have specific mathematical functionality and perform calculations automatically based on settings in their dialogs. Other blocks provide even more flexibility and perform calculations based on equations you enter in their dialogs; they are discussed on “Equation-based blocks” on page 601.

☞ For information about blocks that perform statistical analysis, see “Blocks that calculate statistics” on page 564.

The following blocks may be used in any type of model to provide mathematical functionality based on dialog selections:

#### **Decision (Value library)**



Compares the value at one input to the value at another input and reports a result. For example, use this block to determine if one model value is greater than, less than, or equal to another value during the simulation run.

#### **Integrate (Value library)**



Provides different integration methods to integrate the input value over time. You can also set an initial value in the dialog.

#### **Math (Value library)**



Calculates a mathematical, financial, logical or trigonometry function depending on the option selected in its dialog. Set the block to add a number to its input value and output the result. Or have it calculate the exponent of the input value. Provides over 30 functions from a popup list.

#### **Mean & Variance (Value library)**



Calculates the mean, variance, and standard deviation of the input. You can set an initial value in the dialog and select options to calculate a moving average, use a specified confidence interval, and use time weighted statistics.

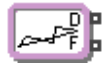


**Random Number (Value library)**



Generates random numbers for the distribution selected in the dialog. Select from over 30 distributions or use an empirical table to create a custom distribution. This block is discussed more in “Random numbers” on page 604.

**Data Fitter (Utilities library)**



Uses matrix techniques to obtain a least mean square curve fit to a set of data. Enter or import the data into the dialog’s data table and select a fitting function to solve for.

In addition to the blocks listed above, most blocks automatically calculate and report statistical information during or at the end of a simulation run. For example, the Math block (Value library) reports the results of the selected mathematical calculation in its Options tab and the Queue block (Item library) displays statistics about the queue length, average wait time, utilization and so forth in its Results tab.

**Other options**

There may not be an ExtendSim block that provides the specific function or equation that you want. Or, you may want to combine the functionality of several blocks into one. Some possible solutions are:

- Select several blocks and make them into a hierarchical block, as discussed in “Hierarchy” on page 540.
- Add features to an ExtendSim block by modifying the structure of a block (its dialog and code) as discussed in the ExtendSim Developer Reference.
- Use the Equation block (Value library) or the Equation(I) or Queue Equation blocks (Item library) to directly combine functions or to obtain behavior not available in other blocks. These blocks are discussed in the next topic.

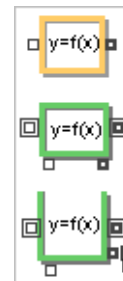
**Equation-based blocks**

The equation-based blocks calculate values for models based on formulas and ModL code entered in their dialogs. There are three equation-based blocks:

- Equation block (Value library)
- Equation(I) block (Item library)
- Queue Equation block (Item library)

These blocks provide access to over 1,000 internal functions; you can also use operators to enter logical statements, write compound conditions, and specify loops. The equation can be as simple as performing a mathematical operation on the value from an input connector or it could be as complex as a full programming segment. The equation is automatically compiled when you click OK in the block’s dialog.

The Equation(I) and Queue Equation blocks can only be used in non-continuous (discrete event and discrete rate) models. They typically perform calculations when items arrive or depart. Although it is a continuous block, you can use the Equation block in non-continuous models. This is common when you want the equation to calculate independent of item status.



Equation (top), Equation(I), and Queue Equation blocks

How To

**Overview**

An equation-based block takes input variables, uses those values in an equation, and outputs the results of the calculation. Equation-based blocks are similar to the formula bar of a spreadsheet. Most of the usual components (operators, values, functions, and so on) are the same. There are two differences - instead of a cell reference, these blocks have input and output variables that are identified by name in the equation, and the results of the equation can be output to different destinations.

**Equation components**

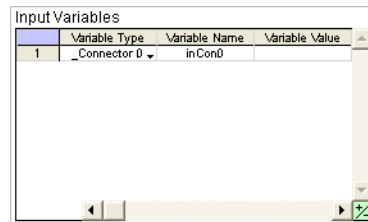
The components of an equation are the inputs, the equation itself, and the outputs. The dialogs of equation-based blocks have separate tables for specifying the input and output variables and a scrollable pane for entering the equation.

**Input variables**

Input variables are the model values used in an equation. Each row in the Input Variables table (shown on right) has a popup menu for selecting the type of input variable and a field for its name and value.

Add rows to the table by clicking the green +/- grow button in the table's bottom right corner; you can select a different input option for each row.

The Variable Type popup provides several options for input variables, as shown below. Each option is described fully in the blocks' Help.



Input Variables table

Input Variable Type	Equation	Equation(I)	Queue Equation
Input Connector	X	X	X
Database value	X	X	X
Database pointer	X	X	X
Static first run initialization	X	X	X
Static multiple run initialization	X	X	X
Attribute of item		X	X
Item quantity		X	X
Item priority		X	X
Item index		X	X
Batch size		X	X
3D object ID		X	X
Item's arrival time			X
Best result			X

**How To**

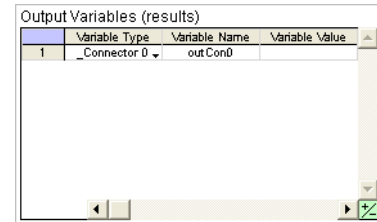
For the Variable Name you can use the default names, assign names that have more relevance to the model, or select a name from a popup, depending on the type of variable selected. The type of input variable selected also determines the options for the Variable Value field: enter a value directly, select the value from a database location, and so forth.

**Output variables**

Output variables are where the results are recorded when the equation is calculated. Each row in the Output Variables table (shown at right) has a popup menu for selecting the type of output variable and a field for its name and value.

Add rows to the table by clicking the green +/- grow button in the table's bottom right corner; you can select a different output option for each row.

The Variable Type popup provides several options for output variables, as shown below. Each option is described fully in the blocks' Help.



Output Variables table

Output Variable Type	Equation	Equation(I)	Queue Equation
Output connector	X	X	X
Database value	X	X	X
Attribute of item		X	X
Item quantity		X	
Item priority		X	X
3D object ID		X	X
Queue rank			X
Select connector			X

For the Variable Name, use the default name, assign a name that has more relevance to the model, or select a name from a popup, depending on the type of variable selected. The Variable Value field reports the results for the selected output variable. In the Equation(I) block there is also a column to specify what should happen to the result if an output is needed but there is no item to trigger the equation's recalculation.

**Equation**

An equation is a list of commands to be executed on one or more input variables, resulting in one or more output variables. In the equation pane, enter an equation that uses the names of the input and output variables and any of ExtendSim's built-in functions and operators. The equation must be of the general form **output = equation;** (the semicolon at the end is required).

For example, an equation that uses the values from input connectors named Input1 and Input2 and outputs the result to an output connector named OutputA would be:

```
if (Input1 > Input2)
    OutputA = 2;
```

```
else
    OutputA = 5;
```

The equation does not have to use all the names assigned to the input and output variables. However, if an input connector is connected, ExtendSim assumes that you will want to use it in the equation. If you don't use it, ExtendSim will give a warning when the simulation runs. ExtendSim will also warn if the equation uses a connector that is not named or is not connected.

☞ Equation-based blocks can calculate separate results and output them using any number of output variables.

**The timing and control of equation calculations**

The equation-based blocks evaluate their equations:

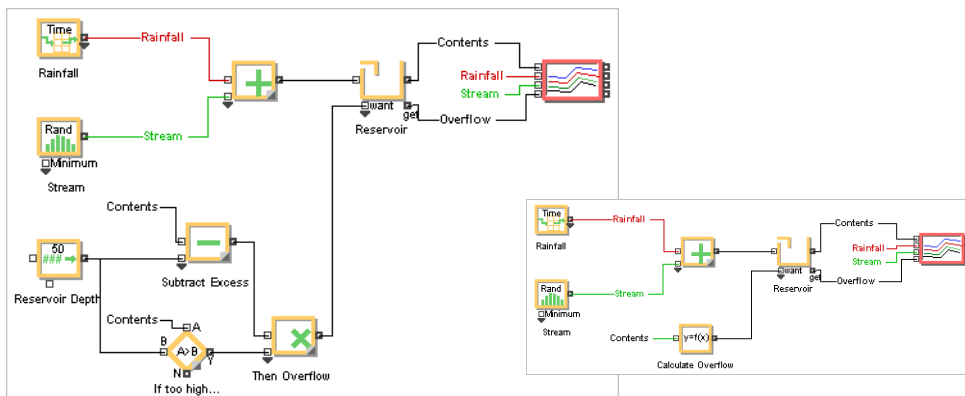
- *Equation*. In a continuous model, at every step. In a discrete event or discrete rate model, when a message is sent to its value input or output connectors. The frequency of calculations can also be customized in the block's Options tab.
- *Equation(I)*. Each time the block gets an item at its item input connector.
- *Queue Equation*. Each time an item arrives or leaves and when a message is sent to one of the value input connectors.

The timing of equation calculations can have significant consequence in a simulation, affecting model behavior. To control when the equation is evaluated so that extraneous messages aren't generated, see the Step Message block (Utilities library) or the Pulse block (Value library).

**Equation block example**

The topic "Simplifying the model" on page 68 demonstrates how one Equation block can replace four blocks that calculate and remove overflow from a Holding Tank block.

How To



Original Reservoir model with overflow calculations (left) and using Equation block (right)

**Random numbers**

The ability to include randomness and show dynamic aspects through time is one of the most valuable characteristics of a simulation experiment. Introducing randomness into a model mimics the patterns and unpredictability of the real world, increasing model accuracy. Since most models have randomness, it is important to understand random numbers.

ExtendSim has blocks, features, and functions that provide randomness in models. For example, the Random Number block (Value library) and the Create and Shutdown blocks (Item library) calculate random numbers using random distributions accessed through ModL functions. You can also specify random settings when using Sensitivity Analysis and for values in the ExtendSim database, and developers can directly access several random number functions.

### Random number generators

The random functions produce random numbers based on a repeatable algorithm known as a *pseudo random number generator*. This generates the uniform random numbers used in the distribution functions. These 32-bit functions are seed-based and update their seed after being called. ExtendSim supports two types of random number generators:

- The recommended, default generator known as the minimum standard random number generator initially by Lewis, Goodman, and Miller, using new coefficients by Gerald P. Dwyer, Jr. (See Numerical Recipes in C, 2nd edition, pp.279 “A portable and reasonably fast minimum standard random number generator that uses Schrage's algorithm” and L'Ecuyer - Comm. of the ACM, Oct. 1990, vol. 33 and L'Ecuyer and Cote, ACM Transactions on Mathematical Software, March 1991.)
- An optional generator based on Schrage, “A More Portable Fortran Random Number Generator,” ACM Transactions on Mathematical Software, Vol 5, No. 2, June 1979, pages 132-138.

The first type of generator is specified by default in the Run > Simulation Setup > Random Numbers tab and is highly recommended for building models. The second generator is used mainly for backwards compatibility, so that models developed before ExtendSim 4.0 will retain results that are consistent with those older versions of ExtendSim.

### Random seeds

A *random number stream* is a sequence of random numbers; the numbers in the stream are derived based on *seed* values. The pseudo random number generator is the internal mechanism in ExtendSim which calculates the numbers in the stream.

ExtendSim provides independent random number streams with the ability to specify a seed so that sequences of random numbers can be repeated. A random number is generated based on a seed.

You can specify a seed for the model as a whole in the Run > Simulation Setup > Random Numbers tab. A seed value of 0 or blank uses a random seed; any other value causes repeatable sequences of pseudo-random numbers. This gives repeatable results, allowing you to determine exactly how changes affect the model.

ExtendSim automatically assigns a separate seed to each block in the model that generates random numbers. To specify your own seed, enter a value in the *Use block seed* field in the dialog of those blocks. Any number entered as a seed in a block dialog will result in an independent random number stream that will not change across runs. Since each stream of random numbers is based on a seed, and you can have a separate seed for each block that generates random numbers, random number generation in ExtendSim is independent. The blocks that generate random arrivals or numbers are the Create and Shutdown blocks (Item library) and the Random Number block (Value library).

Each cell in an ExtendSim database can have an independent seed for the selected distribution. The database seed is dependent on what is set in the Run > Simulation Setup > Random Numbers

tab. The seed field in the Sensitivity Setup dialog is independent of the seed that is set in the Random Numbers tab.

### Resetting random numbers for consecutive runs

The Run > Simulation Setup > Random Numbers tab has three options for when a model is run repeatedly:

- Reset random numbers for every run
- Continue random number sequence
- Use Database table \_\_Seed for values

These options are described in “Random Numbers tab” on page 519.

## Probability distributions

In real life you cannot know exactly when an event is going to occur until it happens. For example, you do not know when the next customer will enter your store. However, by using the correct *statistical distribution* you can approximate what happens in the real world.

A distribution (also known as a *probability distribution* or a *random distribution*) is a set of random values that specifies the relative frequency with which an event occurs or is likely to occur. ExtendSim’s random number distributions express both a probability that something will occur and a range of values that specify the maximum and minimum value of occurrence.

Distributions represent the data observed in real-world situations. When you gather data for a simulation model, it is seldom in a useful form. By “filling in the gaps,” distributions help to compensate for information which was overlooked during data collection. For example, distributions account for extreme or outlying values which may have been missed during typically short data-gathering intervals. Stochastic models use distributions as a handy method for converting data into useful form and inputting it into models.

### Characteristics of distributions

The functions that produce a distribution have one or more parameter arguments which define and control its characteristics. The most important characteristics are a distribution’s *shape*, its *spread*, and its *location* or *central tendency*. Shape is often used to identify distributions; for example, the bell-shaped curve of a normal distribution is widely recognized. Shape can be characterized according to *skewness* (leaning to one side or another) and *kurtosis* (whether it is peaked or flat). You specify the characteristics for the selected distribution by the values you enter for these arguments.

### Choosing a distribution

Using random numbers means either choosing the theoretical distribution that best describes the variability of the raw data, describing the data using a user-defined or *empirical* distribution (such as the empirical distribution in the Create block), or fitting known data to a distribution. As seen below, there are many distributions in ExtendSim and it also has the ability to interface with external distribution-fitting software, as discussed in the next topic.

The choice of one distribution over another is not an exact science, but rather is dependent on the type and extent of the data which is gathered, the detail required for the process being modeled, and (in the case where little data is available), informed guesswork. If the data does not fit any of the distributions described below as “typical” for your process, but fits a distribution which is not typical, go with what your data tells you. It is usually better to use an approximate distribution than it is to keep a value constant.

### Distribution fitting

There are also software applications which fit data to distributions. Use these tools in situations where there is empirical data you want to model using random distributions, but the ExtendSim distributions do not exactly fit. These products can help find the statistical distribution that best emulates the real-world data.

Some ExtendSim packages come with the Stat::Fit distribution fitting application (see “Stat::Fit (Windows only)” on page 586.) The Random Number block has a Distribution Fitting tab from which to launch a distribution fitting package, analyze empirical data, and determine the appropriate statistical distribution for a given data set.

### ExtendSim distributions

When there is sparse or no data, this guide of common uses may help you select a plausible distribution in the Create, Shutdown, or Random Number blocks, within the Sensitivity Setup dialog, or when formatting a cell in the ExtendSim database:

Distribution	Definition
Beta	Distribution of random proportion, such as the proportion of defective items in a shipment, or time to complete a task.
Binomial	The number of outcomes in a given number of trials. Most often used to express success/failure rates or the results of experiments, such as the number of defective items in a batch or the number of customers who will arrive who are of a particular type.
Cauchy	Used to represent the ratio of two equally distributed parameters in certain cases or wildly divergent data as long as the data has a central tendency. It has a sharp central peak but broad tails that are much heavier than the tails of the Normal distribution.
Chi Squared	Used in statistical tests but, since it does not have a scaling parameter, its utilization is somewhat limited. It is a subset of the Gamma distribution with $\beta = 2$ and $\alpha = \nu/2$ .
Constant	This does not produce a random number, but a constant value which does not change. Used when there is exactly the same amount of time between arrivals or as a method to reduce the effects of randomness in the early stages of model building.
Empirical	Used to generate a customized or user-defined distribution with a special shape when the probability of occurrence is known. The options are: <i>discrete</i> (the block will output the exact values given in the table); <i>stepped</i> (values in the table will be used as probabilities of ranges of data); and <i>interpolated</i> (the probability distribution will be interpolated between the data points).
Erlang	Frequently used for queueing theory to represent service times for various activities or when modeling telephone traffic.
Exponential	Primarily used to define intervals between occurrences such as the time between arrivals of customers or orders and the time between failures (TBF) or time to repair (TTR) for electrical equipment. Also used for activity times such as repair times or the duration of telephone conversations.

HOW TO

<b>Distribution</b>	<b>Definition</b>
Extreme Value Type 1A	describes the limiting distribution of the greatest values of many types of samples. Used to represent parameters in growth models, astronomy, human lifetimes, radioactive emissions, strength of materials, flood analysis, seismic analysis, and rainfall analysis. Its peaked shape is always the same but it may be shifted or scaled.
Extreme Value Type 1B	Describes the limiting distribution of the least values of many types of samples. Represents parameters in growth models, astronomy, human lifetimes, radioactive emissions, strength of materials, flood analysis, seismic analysis, and rainfall analysis.
Gamma	Typically used to represent the time required to complete some task. The distribution is shaped like a decaying exponential for shape (2) values between 0 and 1. For shape values greater than 1, the distribution is shaped like a bell curve skewed towards the low end.
Geometric	Outputs the number of failures before the first success in a sequence of independent Bernoulli trials with the probability of success on each trial. Typically used for the number of items inspected before encountering the first defective item, the number of items in a batch of random size, or the number of items demanded from an inventory.
HyperExponential	Usually used in telephone traffic and queueing theory.
Hypergeometric	Describes the number of defects, $x$ , in a sample of size $s$ from a population of size $N$ which has $m$ total defects. It is used to describe sampling from a population where an estimate of the total number of defects is desired. It has also been used to estimate the total population of species from a tagged subset.
Inverse Gaussian	Originally used to model Brownian motion and diffusion processes with boundary conditions. It has also been used to model the distribution of particle size in aggregates, reliability and lifetimes, and repair time.
Inverse Weibull	Describes several failure processes as a distribution of lifetime. It can also be used to fit data with abnormal large outliers on the positive side of the peak.
Johnson SB	Used in quality control to describe non-normal processes, which can then be transformed to the Normal distribution for use with standard tests. It is a continuous distribution that has both upper and lower finite bounds.
Johnson SU	Used in quality control to describe non-normal processes, which can then be transformed to the Normal distribution for use with standard tests. It is an unbounded continuous distribution.
Laplace	Used in error analysis and to describe the difference of two independent, and equally distributed, exponentials.
Logarithmic	Describes the diversity of a sample, that is, how many of a given type of thing are contained in a sample of things. For instance, this distribution has been used to describe the number of individuals of a given species in a sampling of mosquitoes, or the number of parts of a given type in a sampling of inventory.



Distribution	Definition
Logistic	Most often used a growth model: for populations, for weight gain, for business failure, etc. Can also be used to test for the suitability of such a model, with transformation to get back to the minimum and maximum values for the Logistic function. Occasionally used in place of the Normal function where exceptional cases play a larger role.
Log-Logistic	For Shape = 1, it resembles the Exponential distribution. For Shape < 1, it tends to infinity at Location, and decreases with increasing X. For Shape > 1, it is zero at Location, and then peaks and decreases.
LogNormal	Often used to represent the time to perform an activity (especially when there are multiple sub-activities), the time between failures, or the duration of manual activities. This distribution is widely used in business for security or property valuation, such as the rate of return on stock or real estate returns.
Negative Binomial	Number of failures before Sth success. P specifies the probability of success.
Normal	The well-known Gaussian or bell curve. Most often used when events are due to natural rather than man-made causes, to represent quantities that are the sum of a large number of other quantities, or to represent the distribution of errors.
Pareto	Represents the income distribution of a society. It is also used to model many empirical phenomena with very long right tails, such as city population sizes, occurrence of natural resources, stock price fluctuations, size of firms, brightness of comets, and error clustering in communication circuits.
Pearson Type V	A distribution typically used to represent the time required to complete some task. The density takes on shapes similar to lognormal, but can have a larger “spike” close to $x = 0$ .
Pearson Type VI	A distribution typically used to represent the time required to complete some task. A continuous distribution bounded by zero on the left and unbounded on the right.
Poisson	Models the rate of occurrence, such as the number of telephone calls per minute, the number of errors per page, or the number of arrivals to the system within a given time period. Note that in queueing theory, arrival rates are often specified as poisson arrivals per time unit. This corresponds to an exponential interarrival time.
Power Function	A continuous distribution with both upper and lower finite bounds. It is a special case of the Beta distribution with $q = 1$ . The Uniform distribution is a special case of the Power Function distribution with $p = 1$ .
Rayleigh	Frequently used to represent lifetimes because its hazard rate increases linearly with time, e.g. the lifetime of vacuum tubes. This distribution also finds application in noise problems in communications.
Triangular	Usually more appropriate for business processes than the uniform distribution since it provides a good first approximation of the true values. Used for activity times where only three pieces of information (the minimum, the maximum, and the most likely values) are known.

How To

Distribution	Definition
Uniform Integer	Describes a integer value that is likely to fall anywhere within a specified range. Used to represent the duration of an activity if there is minimal information known about the task.
Uniform Real	Describes a real value that is likely to fall anywhere within a specified range. Used to represent the duration of an activity if there is minimal information known about the task.
Weibull	Commonly used to represent product life cycles and reliability issues for items that wear out, such as the time between failures (TBF) or time to repair (TTR) for mechanical equipment.

These distributions and their arguments are described more fully in the Help of blocks that use them.

### Integration vs. summation in the Holding Tank block

As discussed on page 85, the Holding Tank block (Value library) has integration capabilities. The block's dialog gives the option to either sum or integrate its input. In general, a good rule of thumb is:

- Integrate when the value going into the Holding Tank is based on simulation time units, such as a rate. For example, you would integrate when the Holding Tank block's input represents dollars per year or gallons per hour.
- Sum when the value is to be added to the block at each step or dt calculation. For example, you would sum when the Holding Tank block's input is orders or people.

Summing adds the given input to the total at each step, regardless of the time units. Integration considers the input to be spread evenly over each time unit; at each step integration adds a portion of the input to the total. For example, the following table shows the effect of inputting \$2000 to the Holding Tank block when the time units are in years and dt (delta time) is set to 0.25 (for 1/4 of the year).

As seen in the table, if the Holding Tank is set to sum its inputs, it would be the equivalent of adding \$2000 to the account every quarter. If the Holding Tank is set to integrate, it would be the equivalent of adding \$500 per quarter.

Time	Step	Summed	Integrated (delay)	Integrated (no delay)
0	0	2000	0	500
0.25	1	4000	500	1000
0.50	2	6000	1000	1500
0.75	3	8000	1500	2000
1	4	10000	2000	2500

- *Summation* occurs at each step so there is an amount calculated at time 0. Since summation treats its input as an amount, the entire 2000 is added at each step.

- The *integrated (delay)* choice treats its input as a rate and calculates a new result at the next step. Because this backward Euler integration occurs during the interval between steps, there is no amount at time 0.
- The *integrated (no delay)* choice also treats its input as a rate. However, it calculates a new result at the current step. Because its integration occurs at each step, there is an amount at time 0.

It is also important to note that if you subsequently change the delta time to something other than 0.25, the total amount in the summed Holding Tank would be different from the amounts shown above, but the total amount in the integrated Holding Tank would remain the same.

Your choice of integration methods depends on the model:

- You would generally use the *integrated (delay)* choice when there is only one integrating block in the model, when the integrating blocks are not interdependent or cross-coupled, or when there is no feedback.
- In models with more than one integrating block, where the integrating blocks are interdependent or cross-coupled, the feedback between the blocks is usually a correction factor. If this feedback is delayed, the system may correct too late or overcorrect, causing the model results to become unstable. This is often observed as a graph where the traces oscillate with increasing magnitude as time progresses. The *integrated (no delay)* choice compensates for the feedback delays by outputting results one step earlier. For example, the Predator\_Prey model discussed on page 72 is an example of interdependent Holding Tank blocks.



# How To

## Debugging Tools


Learn how ExtendSim can  
help find errors in your models

*“If debugging is the process of removing bugs,  
then programming must be the process of putting them in.”*  
— Unknown author

Creating a model is not so fool-proof that your work is finished once the model has been built. Two important steps in any simulation project are *verification* of simulation results (compare the results to what was intended or expected) and *validation* of results (compare the model to the real system).

ExtendSim provides several methods for detecting problems in models and for debugging models that are not working as expected, including:

- Hints for debugging models
- Verifying results at each step of the model-building process
- Specific blocks used for debugging
- Getting the information you need to debug a model
- Using the Find command to easily locate blocks or dialogs that require attention
- Dotted lines to show unconnected connection lines
- Running with animation to see if a model is behaving as expected
- Using the Notebook to keep track of critical parameters in one location
- Stepping through the model as it progresses
- Show Simulation Order command
- Model tracing for comparison with the real system

 This chapter is concerned with debugging a model, not with debugging custom blocks. The ExtendSim Developer Reference discusses in detail how use the ExtendSim Debugger to find bugs in blocks you create.

## Debugging hints

Efficient debugging of a simulation requires an organized, logical approach. Following the following steps will shorten the process:

- Duplicate the bug. Fix the random number stream so that the problem occurs the same way and at the same time whenever the model is run.
- Describe the bug. Defining the difference between the correct behavior and the observed behavior can lead to insight into the source of the problem. This also helps in formulating a strategy for locating the source of the bug.
- Assume the bug is yours. The vast majority of modeling errors are caused by the modelers themselves. It makes sense to start with the most likely source of the bug.
- Divide and conquer. Determine the source of the bug. And, determine where the bug is not. Build the simplest model that duplicates the error. This will make the model run faster and there will be fewer variables to consider in the debugging process.
- Think creatively. Bugs don't always come from the expected locations. If the source of the problem is not immediately evident, you may be looking at only a symptom. Look at other places in the model that could be the actual source of the error.
- Leverage tools. ExtendSim comes with a variety of tools for debugging a simulation model. For discrete event models, adding a History block (Item library) or Record Message (Utilities

library) block can provide insight into the operation of the model. In continuous models, writing a sequence of values to the ExtendSim Database or to a global array is an easy way to record the values at a specific point in the model. Trace files are also useful at this point.

- Start heavy debugging. Focus on the problem at hand and on how it can be fixed
- Learn and share. This may be a problem that could occur in someone else’s models. Share your experiences with other modelers through the ExtendSim E-Xchange or the ExtendSim Academic E-Xchange as discussed in “User forums” on page 8.

### Verifying results as you build a model

One of the most efficient methods for debugging models is to verify that the model is working correctly at each step during the model building process. It is a lot easier to find problems as you create each section of the model than to try to debug a finished model. You can use almost any of the debugging features discussed in this chapter, but the two most common methods to debug at each step are by examining connector information and cloning dialog items.

#### Connector information

Connectors provide helpful information when you are debugging models. Run the model at any point in the model building process. As the model runs, or at the end of a run, hover the cursor over a connector to see its name and current value. You may also see additional information depending on how the block is programmed.

#### Cloning dialog items

To focus on a particular parameter, clone it to the model worksheet and watch it change as the simulation runs. You can also turn animation on to compare the cloned parameter to the behavior of the surrounding blocks. Cloning dialog items is discussed at “Cloning” on page 504.

### Blocks for debugging

ExtendSim libraries have many blocks that are useful for debugging models either during or after a simulation run, including:

Block	Library	Category	Use
Display Value	Value	Inputs/Outputs	Displays the value of its input connector at each simulation step on the block’s icon and in the dialog. In the dialog, set the time between displays.
Notify	Value	Inputs/Outputs	Plays a sound or stops the simulation when its input is $\geq 0.5$ . Optionally displays a message set in its dialog.
Statistics	Value	Statistics	Displays information about all blocks of a certain type, such as all queues or all activity blocks.
History	Item	Information	Displays statistics and history (arrival time, priority, and attributes) of items that pass through it.
Information	Item	Information	Counts items that pass through it and reports the time between item arrivals and cycle time statistics.
Record Message	Utilities	Discrete Event Tools	Records messages sent over value connectors in a discrete event or discrete rate model.

How To

Block	Library	Category	Use
Item Messages	Utilities	Discrete Event Tools	Records the messages sent over item connectors.
Find and Replace	Utilities	Information	Finds specified dialog items and replaces their values. Drag a clone of a dialog item onto this block's icon to search for similar dialog items.
Pause Sim	Utilities	Model Control	Causes the simulation to pause when certain conditions are met. Click Resume to continue execution
any plotter	Plotter		Add a plotter block any place in a model and connect it to the values you want to track.

### Measuring performance to debug models

ExtendSim provides several methods for obtaining model information when debugging models:

- Dialog boxes display data pertinent to the specific block and in some cases automatically perform statistical calculations. For instance, the dialog of the Queue block (Item library) reports utilization and maximum queue length as well as the number of arrivals and departures.
- You can clone dialog parameters to the model window or to the Notebook to create customized reports and control panels, as shown in “How to clone a dialog item” on page 504.
- Many of the blocks in the libraries have value output connectors that give direct access to specific information. For example, the U output connector on the Activity block outputs utilization values. You can attach any value output to a plotter to display information about model performance. You can also attach value outputs to value inputs on diagnostic-type blocks, such as to the Display Value block (Value library) to display information about that output.
- Plotter blocks, from the Plotter library, conveniently display graphs and tables of data over time. Plotters are useful not only for showing results but for identifying trends and anomalies. You can choose what you want plotted and how you want it displayed, and you can use as many plotters in a model as you want. See “Plotters” on page 588 for more information.
- Animation shows the flow of items in a model, levels of values, etc. ExtendSim blocks have built-in customizable animation; you can also add custom animation using the Animate tabs in block dialogs or by using blocks from the Animation 2D-3D library. Animation is especially useful for verifying a model since it can show if portions of the model are operating as expected. Since animation can slow model performance considerably, it is common that you would use animation in the early stages of model-building or for presentations. See “Animation features for debugging” on page 618 for more information.
- There are numerous blocks that can be used for debugging models and verifying results. For instance, the Notify block (Value library) can stop the simulation and notify you when its input goes above or below a specified level. The Information block (Item library) provides information about the output of the block it is connected to (the interval between arrival times, how many items are currently present at the output, and so forth).



- Sensitivity analysis allows you to vary a parameter incrementally, randomly, or in an ad hoc manner to determine how sensitive model results are to changes in one variable. See “Sensitivity analysis” on page 568 for more information.
- Running simulations multiple times, such as for Monte Carlo simulations, gives ranges of values indicating the possible outcomes for the model. See “Running a model multiple times” on page 522 for more information.
- The Run > Generate Report command, discussed in “Model reporting” on page 620, instructs ExtendSim to generate a text file of the final model results. You can report on all the blocks in a model, or use menu commands to specify which blocks are included in the report. Reports are especially useful for outputting to other applications, such as statistics packages, for further analysis.
- The Statistics block (Value library) reports and statistically evaluates results. For example, it can display information about every queue-type block in the model and calculates the confidence intervals based on the results. As discussed in “Clear Statistics” on page 566, the Clear Statistics block resets statistical accumulators at random intervals or in response to a system event; this is used to eliminate statistical bias during the warm-up period.
- The Utilities library contains two blocks that are useful for debugging discrete event models. The Record Message block, when connected between two value connectors, shows all of the messages, the values transferred, and whether the message came in the input or output connector. The Item Messages block records the message communication between two item connectors. See “Messaging in discrete event models” on page 260 for a detailed discussion of the item-based messaging system.

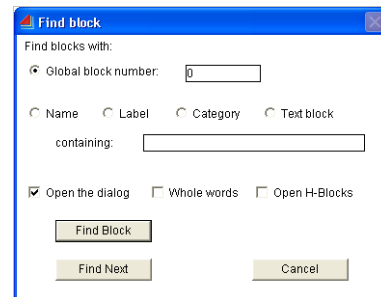
### Find command

If a model is large, it might be difficult to find all the blocks by sight. For example, you may see in the Trace file that a particular block didn't get the expected input. The Edit > Find command lets you locate a block by its global block number, name, label, or category, or find a text block by its global block number or by the text within it.

*Global block numbers* are unique, permanent identifiers for blocks and text blocks. *Name* means the name of the block in the library menu (e.g. Equation). Block *labels* are defined by the user in the block dialog and are especially useful to find types of blocks. *Category* refers to how the block is classified in the Library menu (e.g. Inputs).

The debugging blocks described above list the global block number in case you have many copies of a block in the model. Use global block number and the Edit > Find command to quickly scroll to a block and select it.

- ☞ You can also use the Find and Replace block, listed in the table of debugging blocks above, to find blocks. It is even more useful for finding specific dialog items in the located blocks, so you can replace their parameter values.



Find dialog

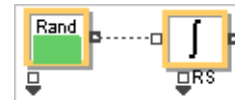
HOW TO

## The Source Code Debugger

The ExtendSim Source Code Debugger is the ultimate model debugging tool. With the debugger you can step through every statement, inspect the value of every variable, and view the sequences of messages sent from one block to another. Using the debugger requires that you understand the ModL language and how blocks interact with each other. For more information, see the Developer Reference.

### Dotted lines for unconnected connections

If a block is not getting an input or is not generating an output when you think it should, it may not be connected properly in the model. This can happen when connections run underneath blocks when you thought that they were connected to the blocks. ExtendSim shows incomplete connections as a red dotted line. To fix these, delete the incomplete connections by double-clicking them and reattach them to where they are supposed to be. Clicking one segment selects that segment; double-clicking a segment selects the entire connection line.



Incomplete connection

### Animation features for debugging

Running the model with 2D animation on is useful for debugging. If the animation goes by too quickly for debugging, slow down the process with the Animation Slower (turtle) button in the toolbar. To resume speed, use the Animation Faster (rabbit) button.

To learn more about how models, blocks, and connection lines are animated see “Animation” on page 551.

### Animating the model

If you know how a block is animated (see the block’s Help), you can watch its icon to determine if something is not acting as expected. For example, if a block indicates the status of its contents, you can use that information to debug the model.

Discrete event models have additional animation capabilities and can show the flow of items along connection lines and between named connections. This gives a visual representation of what is happening in the model.

### Animating item properties (discrete event models only)

Changing an item’s animation depending on a property (attribute, quantity, priority) is helpful when debugging models. For example, in a block’s Item Animation tab you can visually differentiate between types of items by setting separate animation objects for each attribute value. Then observe the individual items as they flow through the model.

### Notebook

The Notebook is a convenient way to see the final values for several dialog items in a model or to compare inputs to outputs. This is useful in debugging because you can group the dialog items by what their final values are expected to be.

For more information, see “Notebooks” on page 508.

## Stepping through the simulation

The toolbar in the application window has buttons that help if you are debugging a model. When the simulation is running, the Stop, Pause, Animation Off/On, Animation Faster, and Animation Slower buttons are available.



Debugging buttons while running

When you click the Pause button, the simulation pauses, the Pause button changes to the Resume button, and the Step button becomes available.



Debugging buttons when paused

Instead of using the Pause command, you can use the Pause Sim block (Utilities library). It has additional options for triggering when to pause the simulation.

The Step Entire Model, Step Next Animation, and Step Each Block commands in the Run > Debugging menu tell ExtendSim how far to go when the Step button is clicked after pausing.

- The Step Entire Model command starts at the selected block and runs until that block would be executed again. This is a good way to examine what happens in the intervals between when a block is called.
- If you have animation turned on, the Step Next Animation command tells ExtendSim to step until the next animation change. In models where there are many steps between animation changes, this option makes going from visible change to visible change much faster.
- The Step Each Block command causes the Step button to simply go to the next block. It is usually used in conjunction with the Show Block Messages command, otherwise what is happening in the model will not be apparent.

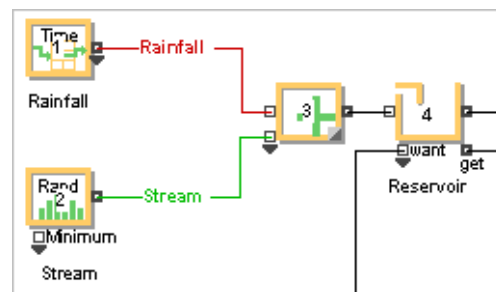
If you select Run > Debugging > Show Block Messages when you are stepping through a simulation run, the block that is active will be highlighted with the current message name written on it. If that block is not currently visible, the window will automatically scroll to the block if Scroll To Messages is checked. You can also choose to only show the On Simulate messages.

The Run > Debugging > Pause at Beginning command automatically pauses the simulation after the initial model processing (initialization, error checking, etc.) but before the first step. This gives you a chance to step from the very beginning without having to guess when to click the Pause button.

## Show Simulation Order command

ExtendSim normally determines the order that blocks in a continuous model are executed by following the path of connections. If a continuous model is behaving in an unexpected manner, you may want to see explicitly the order ExtendSim is using to execute calculations in a model. Selecting the Model > Show Simulation Order command puts a small number on each block indicating its order of execution.

Although also accurate for groupings of continuous blocks in non-continuous models, this



Section of Reservoir model showing simulation order

display will be inaccurate for discrete event (Item library) and discrete rate (Rate library) blocks in those models. This is because discrete blocks can generate block-to-block messages and override the system's simulation order.

### Slow simulation speed

There are many reasons why a simulation would not run as quickly as you might expect. Some common causes, and the methods to detect and avoid them, are discussed in “Speeding up a simulation” on page 531.

### Model reporting

The report commands in the Run menu are for generating customized reports of data to help debug models. However, model tracing (discussed in the next topic) can offer more detailed information so you will probably use that option more for debugging, and the reports option for analysis.

To learn more about how to generate reports in ExtendSim, see “Reports” on page 596.

### Model tracing

Model tracing is useful for finding anomalies that occur as the simulation runs. The model tracing commands act like the reporting commands, but the output is much more extensive. A trace text file shows the details of block values at every step or event in the simulation.

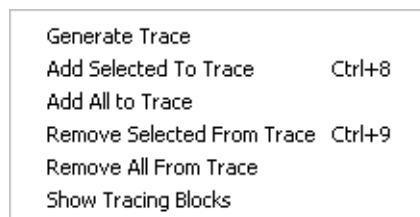
Because of the large amount of information generated by the model tracing commands, most people don't use model tracing often. However, tracing is a highly effective method for following a single block or a few blocks to watch for values that do not match expectations.

The trace is saved as a text file and opens automatically at the end of the run. If the *Runs* field in the Simulation Setup dialog is greater than 1, each consecutive trace is added to the end of the file so that you can compare traces from earlier runs.

### Generating traces

The tracing commands are located in the Run > Debugging menu. To generate a trace:

- ▶ Select the blocks you want included in the trace:
  - ▶ To specify individual blocks for tracing, select them and choose the **Add Selected To Trace** command.
  - ▶ To get a trace on every block, choose **Add All To Trace**.



Tracing commands

- ▶ Choose the Run > Debugging > Generate Trace command to create a trace the next time you run the model.
- ▶ Run the simulation to see the trace results. ExtendSim prompts you for a name and location for the trace file. Trace reports are opened, closed, and edited just like any other text file in ExtendSim.
- ▶ To trace fewer blocks in the next simulation run, select the blocks you want out of the trace report and select Remove Selected from Trace. To start over on the selection of blocks, choose Remove All from Trace.


You can see which blocks are included in the trace by choosing Show Tracing Blocks; each traced block in the model displays the word *Trace* on its icon.

### Tracing example

For the Math block from the Reservoir 1 model, the top of the tracing report is:

```

ExtendSim Trace - 8/16/2007 3:48:57 PM
Run #0
### calculation at Math number 2. CurrentTime:0.
ValuesIn = 2.6
ResultOut = 3.1550072961999
----- Step #1 -----
### calculation at Math number 2. CurrentTime:1.
ValuesIn = 4.4
ResultOut = 5.3917648062033
----- Step #2 -----
### calculation at Math number 2. CurrentTime:2.
ValuesIn = 6.7
ResultOut = 7.1088551154443
----- Step #3 -----
### calculation at Math number 2. CurrentTime:3.
ValuesIn = 3.4
ResultOut = 3.8298759283776
  
```

 If you build your own blocks and want to use the Trace features for debugging, add special code to the blocks, as described in the Developer Reference.



# How To

## Data Management and Exchange

Managing and transferring data within ExtendSim  
and between ExtendSim and other applications

*“It is a capital mistake to theorize before one has data.”*  
— Sir Arthur Conan Doyle

ExtendSim provides a variety of standards-based options for managing and sharing data internally and with other applications. This chapter covers:

- User interfaces for exchanging data within ExtendSim and between ExtendSim and external data structures:
  - Copy/Paste
  - Importing and exporting
  - Read and Write blocks
  - DDL to dynamically link to internal data structures
  - DDE links to external data structures
- Internal structures for storing and managing data
  - ExtendSim databases
  - Global arrays
  - Dynamic arrays
  - Embedded objects
  - Linked lists
- Exchanging data with external applications
- Blocks for accessing and managing data
- How data source types are indexed and organized
- Transferring data between ExtendSim and devices
- Standard communication technologies:
  - Text files
  - DDE
  - ActiveX/COM/OLE
  - ODBC
  - DLLs and Shared Libraries
  - FTP
  - Mail Slots



This chapter is concerned with the communication of data. For information about working with 2D graphic objects or pictures, see “Graphic shapes, tools, and commands” on page 561 or “Copy/Paste and Duplicate commands” on page 674.

### User interfaces for data exchange

The following sections discuss methods for exchanging data within ExtendSim and between ExtendSim and external applications and devices. Some of these methods (such as Dynamic Data Linking) are specific to ExtendSim, and some (such as DDE Linking) are industry standards for inter-application communication. As discussed below and shown in the table that follows, each method has its own specifications and properties:


- The flow of data can be one-directional or two-directional. With two-directional data flow, a data change at the source affects data at the target and a change at the target affects data at the source. One-directional data changes at the target do not affect the source.



- Some methods require that two windows be open at the same time. Copy/Paste requires only one window; you do not need to leave the ExtendSim dialog or application file open once the data has been copied. On the other hand, DDE linking requires that windows in both ExtendSim and the other application be open to complete the process.
- A communication method that has live links sends a data update message along with the data. This means that as soon as the data at the source changes, the target is dynamically notified and can take the appropriate actions to react to the new value.
- Dynamic data linking is only used to access ExtendSim internal data structures, such as ExtendSim databases and global arrays. Other methods can be used either internally or externally (for instance, copy/paste from one ExtendSim dialog to another or copy/paste from ExtendSim to Excel).

The information is summarized in the following table.

Interface	See Page	Data Direction	Open Window(s) Required	Live Link	ExtendSim or External Application
Copy/Paste	625	One way	1	No	Both
Import/Export	626	One way	1	No	Both
Read/Write blocks	628	Two way	1	Possible	Both
Dynamic Data Link (DDL)	629	One way	1	Yes	Internal
Dynamic Data Exchange (DDE) Link	636	Two way	2	Yes	External

 Most of the above methods are supported by an underlying industry-standard technology, as discussed in “Data source indexing and organization” on page 661.

### Copy/Paste

The Clipboard is useful for passing information within ExtendSim and between ExtendSim and other applications on the same computer. The copy/paste action can occur before a simulation run, after the run, or when the simulation is paused. While the Edit menu’s Copy, Paste, and Duplicate commands work the same as in most applications, there are a few items you should note:

- When copying/pasting data tables or database tables:
  - To copy all the data in a specific column of a data table or database table, click in the column’s *title field* so that the entire column is selected. Then give the command Edit > Copy Data.
  - To copy all the data from all the columns in a data table or database table, click in the *upper left corner* of the table so that the entire table is selected, then give the command Edit > Copy Data.
  - To paste data into a data table or database table, click in the upper left corner, in the upper left cell, or in a desired starting cell and give the command Edit > Paste Data. The table cells will be populated downward to the extent of the data in the Clipboard or the number of rows in the data table.

- When an ExtendSim database or database tables are selected, the Copy Database/Paste Database commands provide a convenient method to copy an ExtendSim database or tables from one model or one database to another. The command Edit > Duplicate Database is useful for duplicating an ExtendSim database within a model, so you don't have to create a new database structure.
- Regarding "Allow data table titles copying" in the Edit > Options dialog:
  - This option is off by default, because you typically would not want to also copy table titles when copying data from a data table.
  - To also copy row and column titles when copying tables from Plotter blocks, select the option "Allow data table titles copying" in the Edit > Options dialog. ExtendSim will prompt for the titles to be copied. This is helpful when you are copying into a word processing program for presentation.

### Importing and exporting data

Importing and exporting is similar to copying/pasting in that both methods involve copying data from one location to another. However, importing/exporting provides additional capabilities:

- Some types of importing/exporting can be automated; copy/paste always requires user interaction.
- You can import/export while the simulation is running.
- It might be faster than reading and writing data because you only import data once at the beginning of the simulation or export once at the end, rather than reading and writing frequently.
- The information can be accessed even if it originated on a different computer or an external device.

All the data that is imported/exported gets copied as one piece (a local copy). This means that, whether the exchange happens before, during, or after a simulation run, all the data is made available to the target at the same time. This differs from, and is usually faster than, the Reading/Writing method (discussed later in this chapter) where the data is transferred piece by piece during a simulation run.

Data can be imported from or exported to external applications such as spreadsheets and database programs. Internally, you can import data to or export data from dialog or plotter data tables, ExtendSim database tables, and global arrays.

In some cases the data is imported/exported directly within ExtendSim or directly between ExtendSim and the other application. In other cases it is imported/exported in the form of text files, a standard communication technology discussed in "Text files" on page 663.

Importing/exporting is one-directional; changing data at the target does not affect data at the source and vice versa. The data can reside locally, be remotely accessed over a network, or accessed via the internet using FTP protocols.

ExtendSim provides the following methods to import or export data:

- 1) The File menu has Import Data and Export Data commands for exchanging data using text files, as shown in the example that follows.

- 2) The Data Import Export block (Value library) enables ExtendSim databases or global arrays to directly import data from or export data to spreadsheets, external databases, or an FTP site. Use of this block is described in detail on page 661.
- 3) Read and Write blocks (Value library) can import or export text files, ExtendSim database tables, global arrays, or Excel worksheets as a local copy. See “Read and Write blocks” on page 659.
- 4) You can use Database menu commands to:
  - Create a new, or replace an existing ExtendSim database from an imported database text file. The text file can be generated by a different ExtendSim database or by an SDI Industry database.
  - Export an entire ExtendSim database as a text file. The file can be imported into a different ExtendSim database in the same model or in a different model
  - Import tables from an ExtendSim or SDI Industry database as text files to an ExtendSim database.
  - Export tables as text files from an ExtendSim database to a different ExtendSim database.
- 5) The ExtendSim DB Add-In works with database text files to transfer data between an ExtendSim database and Microsoft Excel. It is described fully in “Excel Add-In for ExtendSim databases”.

**How to import data using the File menu**

The example that follows uses a menu command to import a text file of data into the Reservoir model that was discussed in the Tutorial module. The imported data will be used as the monthly rainfall values. The steps are:

- ▶ Open the model and save it under a different name:
  - ▶ Open the Reservoir 1 model located in the folder \Examples\Tutorials.
  - ▶ So that you don’t overwrite the original file, save the model as **Reservoir Import**.
- ▶ To import data into the data table:
  - ▶ In the Lookup Table block, click in the title field of the data table column labeled **Rain (inches)**; this selects that entire column.

 You must select the table column(s) to enable the Import Data or Export Data menu commands.

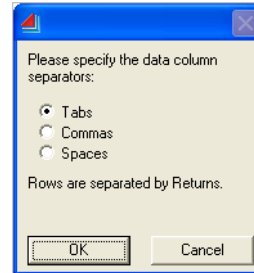
- ▶ Give the command File > Import Data.
- ▶ For Windows, choose **Files of type: Text File** as the type of file to open.
- ▶ Find and open the “Rainfall.txt” file, located in the folder \Examples\How To.



- ▶ Since you are only importing one column of data, in the delimiter dialog (shown on the right) click OK to select the default settings and import the data. (Delimiter settings are discussed later in this section.)

This action replaces the data that had been in the “Rain (inches)” column with a local copy of the source data. This exchange is one-directional and not live; if the data in the originating text file is subsequently changed, the text file must be re-imported to affect the change in the block.

Use this same process to import data into any data table, ExtendSim database table, or global array that is linked to a data table.




Delimiter dialog

### **How to export data using the File menu**

Exporting data using a menu command is similar to importing data, except in reverse. For example, to export results from the Reservoir Import model:

- ▶ Select the data to export:
  - ▶ In the Reservoir Import model from above, open the Plotter I/O block.
  - ▶ Click in the upper left corner of the plotter’s data table so that all columns are selected.
- ▶ Export the data to a text file:
  - ▶ Give the command File > Export Data.
  - ▶ In the dialog that appears, name the file **Export Text**, then click Save.
  - ▶ In the delimiter dialog that appears, click OK to select the default settings and export the data. (Delimiter settings are discussed later in this section.)
- ▶ To see the exported file:
  - ▶ Give the command File > Open
  - ▶ For Windows, choose Text File or All Files as the type of file to open
  - ▶ Locate and open the file named Export Text. It will contain the three columns of data from the plotter.

 To also copy row and column titles from Plotter blocks, select the option “Allow data table titles copying” in the Edit > Options > Miscellaneous tab. ExtendSim will prompt for the titles to be copied. This is useful when exporting to a word processing program, but including titles could cause data errors if the text file is imported into a spreadsheet or an ExtendSim table.

### **Where to get more information**

For more information about:

- Creating, saving, and using text files, see “Text files” on page 663.
- How to import and export data using the Data Import Export block, go to page 661.
- Read and Write blocks, see “Read and Write blocks” on page 659.


### **Read and Write**

ExtendSim blocks can read data from internal and external sources, and write data to internal and external sources, during a simulation run. The blocks for reading and writing are the Read and

Write blocks in the Value library and the Read(I) and Write(I) blocks in the Item library. The Read and Write blocks read data from or write data to ExtendSim databases, global arrays, Excel workbooks, and text files. The Read(I) and Write(I) blocks read data from or write data to ExtendSim databases.

These blocks are powerful tools for reading data from internal and external sources and writing data into internal or external data destinations.

- In contrast to the importing/exporting described above, the Read and Write blocks are usually used to exchange data piece-by-piece as required during the run, rather than by one exchange of the entire data set. Even though it may slow simulation speed, models may need to write and read data while a model is running if the data that is written is in turn read and used in another part of the model.
- The Read and Write blocks can create a one-way live link between the target and the source. If you write to a location in an ExtendSim database or global array, ExtendSim will send update messages to all the Read blocks in the model linked to that location. When the Read block receives the alert message, it will act on the fact that the data value changed, making the link 'live'. Writing to a cell in an Excel spreadsheet, or to a text file, will not have the same effect.

 A potential disadvantage of using the Read and Write blocks is that reading and writing the data each time the block is calculated can have a significant impact on simulation speed. If the data to be read or written will not change over the course of the simulation, you should import/export the data as a local copy in the block before or after the run. If the data is not going to change at all, consider using another method, like a one-time copy/paste or import/export.


For more information, see “Read and Write blocks” on page 659.

### Dynamic linking to internal data structures

ExtendSim supports a comprehensive proprietary internal data linking method called *dynamic data linking (DDL)*. This application-level protocol tracks which dialog items (parameters and data tables) are linked to which internal data structures. The internal structures that dialog items can be dynamically linked to are:


- ExtendSim database tables
- Global arrays
- Dynamic arrays

The DDL technology has a user interface so you can easily link data tables and parameters in any part of a model to ExtendSim database tables or global arrays. Dynamic arrays can only be linked to dialog items through ModL code.

 Dynamic data links are live and bidirectional, so the value of a linked dialog item can change immediately when the value of the data source changes, and vice versa.

You can use menu commands to dynamically link a dialog parameter or data table to an ExtendSim database or global array. The clone of a parameter or data table can also be dynamically linked. Linking a clone has the same effect as linking the original dialog item - both the original and the clone will be linked to the data structure.

The sections that follow show how to create a dynamic link between dialog items (parameters and data tables) and internal data structures (ExtendSim database tables and global arrays) through the user interface. To learn how to use ModL code to dynamically link dialog items to internal structures, see the Developer Reference.

 There is overhead associated with the process of accurately updating a linked dialog item with a data structure. If you overload a model with links that frequently change, the model's performance can suffer. Consider the following suggestions: 1) Just link the data that is important to a model and not link every piece of data to every possible source. 2) Import values into a Read block (Value library) for use in the model, so they don't have to be continuously updated. 3) Send or receive link alert messages only at the start or end of the simulation as discussed in "Link dialog checkboxes" on page 634.

### **Linking a parameter to an internal data structure**

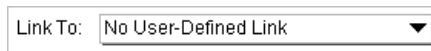
Unless it is already linked for sensitivity analysis (see "Sensitivity analysis" on page 568) or through the DDE link feature (see "DDE links (Windows only)" on page 636), a dialog parameter can be dynamically linked to a specific cell of an internal data structure.

Parameter fields that are dynamically linked to an internal data structure are outlined in light blue. (Parameter fields are outlined in green for active sensitivity analysis, red for inactive sensitivity analysis, and yellow for DDE linking (Windows only) to an external application.)

#### *How to link a parameter to an ExtendSim database*

The following example uses the Reservoir 1 model from the Tutorial module. It describes how to dynamically link the Holding Tank's "Initial Content" parameter to a cell in a database.

- ▶ Open an existing model and save it under a different name:
  - ▶ Open the Reservoir 1 model located in the folder \Examples\Tutorials.
  - ▶ So that you don't overwrite the original file, save the model as **ReservoirDBLink**.
- ▶ In the ReservoirDBLink model, open the Holding Tank dialog.
- ▶ Open the Link dialog by doing one of the following:
  - ▶ Click in the Holding Tank's **Initial contents** parameter field and give the command Edit > Create/Edit Dynamic Link
  - ▶ Or, right click the **Initial contents** parameter field and choose Create/Edit Dynamic Link.
- ▶ In the Link dialog's top popup menu (shown at right with its default setting of "No User-Defined Link") choose **Link to: Database Table**.



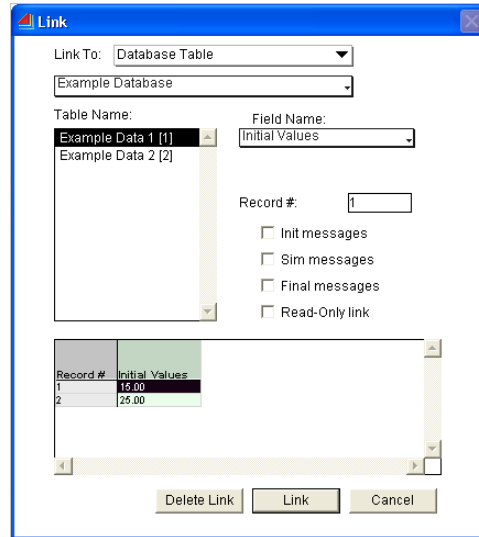
Link dialog popup menu, default option

- ▶ In the popup menu that appears below the Link popup, select the database named **Example Database**. (The Reservoir 1 model already had a database, so you can concentrate on learning how to create a dynamic link. For information on creating a database, see “How to create an ExtendSim database” on page 640.)

- ▶ Select the table named **Example Data 1**.

The field “Initial Values” is selected by default in the Field Name popup menu, because there is only one field in this database table.

Notice that the *table viewer* in the lower section of the Link dialog displays two records containing initial values.



Link dialog for Database Table

- ▶ Do one of the following:
  - ▶ Enter **Record #: 2**.
  - ▶ Or, select the initial value of **25.00** by clicking on that cell in the table viewer; the setting is in the second row of the table viewer.
- ▶ Click the **Link** button to close the dialog and establish the link.
- ▶ Save the model.

In the Holding Tank’s dialog there is now a light blue frame around the “Initial contents” parameter, indicating that the field is dynamically linked. Mousing over the parameter field displays the source and location of the linked cell. To open the Link dialog for viewing or changing the linked settings, click in the parameter field and give the command Edit > Create/Edit Dynamic Link.

When the simulation is run the Reservoir will start with an initial contents of 25 inches; it gets that value from a live link with the database.

- ☞ Since the link is a live link, changing the value of the “Initial contents” parameter in the Holding Tank will change its record in the database table, and vice versa. If you do not want that behavior, select the Read Only option in the Link dialog.

*How to link a parameter to a global array*

The following example uses the Reservoir 1 model from the Tutorial module. It describes how to dynamically link the Holding Tank’s “Initial Contents” parameter to a cell in a global array.

- ▶ Open an existing model and save it under a different name:
  - ▶ Open the Reservoir 1 model located in the folder \Examples\Tutorials.
  - ▶ So that you don’t overwrite the original file, save the model as **ReservoirGALink**.
- ▶ In the ReservoirGALink model, open the Holding Tank dialog.
- ▶ Open the Link dialog by doing one of the following:
  - ▶ Click in the Holding Tank’s “Initial contents” parameter field and give the command Edit > Create/Edit Dynamic Link

HOW TO

► Or, right click the “Initial contents” parameter field and choose Create/Edit Dynamic Link.

► In the Link dialog’s top popup menu choose **Link to: Global Array**.

► In the popup menu that appears, select **Array name: Example Array 1**. (The Reservoir 1 model already had a global array, so you can concentrate on learning how to create a dynamic link. For information on creating a global array, see “How to create and use a global array” on page 653.)

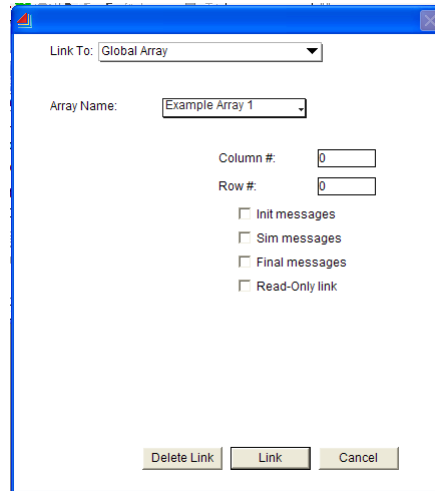
► Since global arrays are zero-based, enter **Column #: 0** and **Row #: 1**. This selects the cell that contains the initial value of **25.00**.

► Click the **Link** button to close the dialog and establish the link.

► Save the model.

In the Holding Tank’s dialog there is now a light blue frame around the “Initial contents” parameter, indicating that the field is dynamically linked.

Mousing over the parameter field displays the source and location of the linked cell. To open the Link dialog for viewing or changing the linked settings, click in the parameter field and give the command Edit > Create/Edit Dynamic Link.



Link dialog for Global Array

When the simulation is run the Reservoir will start with an initial contents of 25 inches; it gets that value from a live link with the global array.

☞ Since this is a two-way live link, changing the “Initial contents” parameter in the Holding Tank will dynamically change the value of its record in the database table. To disable that behavior, select the Read Only Link checkbox in the Link dialog when you create the link.

### Linking a data table to an internal structure

Many ExtendSim data tables have a *Link* button in their lower left corner. If the Link button is not present for a table, the table does not support dynamic linking.

A data table linked to an internal data structure will have its upper left corner in light blue rather than the grey of the table header.

- If the source is a database (*DB*) or global array (*GA*), those initials will be displayed in the corner
- If the link is to a dynamic array there will be no initials.

If a data table is linked to a database, double-clicking the DB initials in the upper left corner will open the linked database table’s viewer.

⚠ Because ExtendSim data tables and global arrays are zero-based, but databases are one-based, a data table’s cell at row 0, column 0 (the top, leftmost cell) is linked to the database table cell at field 1, record 1 (the top, leftmost cell) or the global array cell at row 0, column 0 (also the top leftmost cell). For detailed information, see “Indexing and organization” on page 662.



*How to dynamically link a data table to a database table*

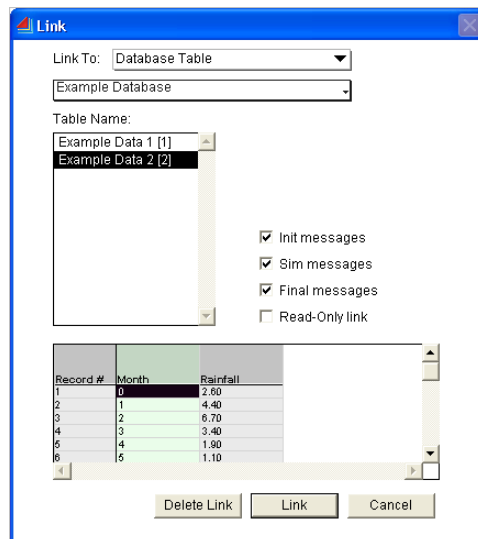
The following example uses the ReservoirDBLink model from the previous section that showed how to dynamically link a parameter to a database. This example shows how to link a data table to a database table. (So you can concentrate on learning how to do this, the model already has a database table with the necessary data. For information on creating a database, see “How to create an ExtendSim database” on page 640.)

- ▶ In the ReservoirDBLink model you created on page 632, open the Lookup Table block’s dialog.
- ▶ Open the Link dialog either by clicking the **Link** button in the lower left corner of the data table or by selecting the data table and giving the command Edit > Create/Edit Dynamic Link.
- ▶ In the Link dialog (shown at right):
  - ▶ Select **Link to: Database Table**.
  - ▶ In the popup menu that appears choose **Database name: Example Database**.
  - ▶ Select the table named **Example Data 2** from the list of tables.

Notice that the data for the selected table is displayed in the viewer pane at the bottom of the Link dialog. This is the information that will populate the Lookup Table block’s data table.

- ▶ Click the **Link** button to close the Link dialog and establish the link.
- ▶ Save the model.

In the Lookup Table dialog notice that the upper left corner of the data table now displays the initials “DB”, indicating that the data table is linked to a database table. Mousing over that corner displays information about which database and table it is linked to.



Link dialog for Database Table

When the simulation is run, the model will get rainfall data from a live link with a database.

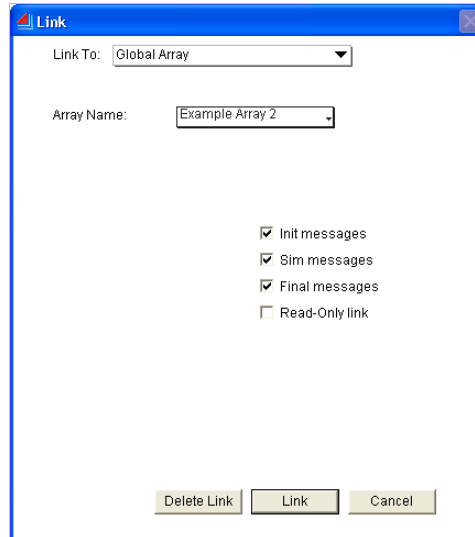
- ☞ Since this is a two-way live link, changing the value of one or more cells in the block’s data table will change the values in the database table. To disable that behavior, select the Read Only Link checkbox in the Link dialog when you create the link.

*How to link a data table to a global array*

The example that follows uses the ReservoirGALink model from the earlier section that showed how to dynamically link a parameter to a global array. This example shows how to link a data table to a global array. (So you can focus on learning how to do this, the model already has a global array with the necessary data. For information on creating a global array, see “How to create and use a global array” on page 653.)

- ▶ In the ReservoirGALink model, open the Lookup Table block’s dialog.
- ▶ Open the Link dialog either by clicking the *Link* button in the lower left corner of the data table or by selecting the data table and giving the command Edit > Create/Edit Dynamic Link.

- ▶ In the Link dialog:
  - ▶ Select **Link to: Global Array** (as shown on the right).
  - ▶ In the popup menu that appears select **Array name: Example Array 2**.
  - ▶ Click **Link** to close the dialog and establish the link.
- ▶ Save the model.



Link dialog for Global Array

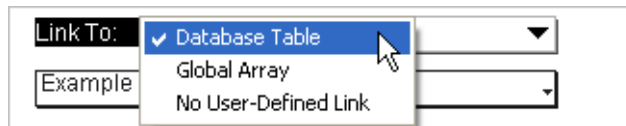
As signified by the initials *GA* in its upper left corner, the data table in the Lookup Table block is now dynamically linked to the global array. Mousing over the GA initials displays the name of the linked global array.

When the simulation is run, the model will get rainfall data from a live link with the global array. (To make the difference more obvious, the global array has 6.6 inches of rainfall for the first month, rather than the 2.6 inches in the original Reservoir 1 model.)

☞ Since the link is a two-way live link, changing the value of one or more cells in the block's data table will change the values in the global array, and vice versa. If you do not want that behavior, select the Read Only option in the Link dialog.

### The Link dialog

When you dynamically link a dialog item, the Link dialog opens and displays the **Link To** popup menu listing potential data sources.



☞ When the Link dialog is first opened, the Link To popup menu displays the current linked data structure (if the parameter or data table is already linked), or the term “No User-Defined Link” (if it is not linked).

The Link To popup menu has three options: No User-Defined Link, Global Array, and Database Table. The No User-Defined Link setting is informational only. The Database Table and Global Array options are active options for changing, or adding, a dynamic link to the parameter or data table. If you select either of those options, the dialog displays additional popup menus and fields to further define the data source.

⚠ A parameter or data table can only have one data source; if you change the option in the Link To popup menu, the previously linked data source will be unlinked.

### Link dialog checkboxes

Outside of a simulation run, blocks receive messages whenever dynamically linked data changes at the source. To maximize simulation speed you may not want these update messages sent during certain phases of a run. As shown on page 631 and page 632, when the Database Table or Global Array option is selected in the Link dialog, the dialog displays four checkboxes:

- Init messages
- Sim messages
- Final messages
- Read-Only link


These allow you to control the link's data update behavior, as discussed below.

The first three link options (Init messages, Sim messages, and Final messages) control whether or not blocks receive update messages when linked values change during specific points in a simulation run. For example, the "Init messages" option determines whether a block receives a message if a linked value changes during the InitSim message.

These three options default to on. Turning an option off prevents ExtendSim from sending data update messages to the linked blocks if the data source is updated during that phase of the simulation. For example, if you know the dynamic link only updates during the FinalCalc message, it will reduce the messaging and speed up the simulation to turn off the Init messages and Sim messages options.

When the Init messages and Final messages options are turned on, linked blocks will receive an additional message during InitSim and/or FinalCalc, causing an update of the linked dialog items during those phases.

The fourth checkbox is the Read-Only link option, which is off by default. If checked, the linked data can only be changed from the data structure side. In that case, you will not be allowed to edit the parameter or the data table directly and the live link is only one-way (from the internal structure to the data table or parameter).

 Use these options carefully, since they can dramatically change linking behavior. You would probably only want to uncheck a message option (Init messages, Sim messages, or Final messages) if you are sure that the source data won't change during that message.

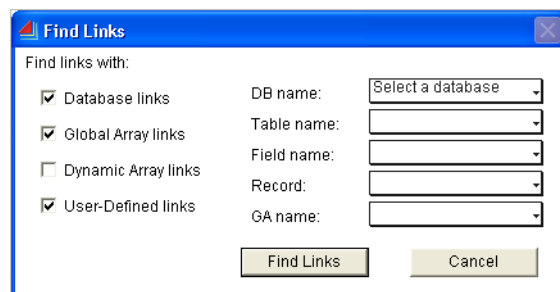
### Finding linked dialog items

The command Edit > Open Dynamic Linked Blocks opens the Find Links dialog, shown on the right. This is useful for examining and locating linked dialog items.

The options in the dialog allow you to selectively choose which types of links you want to open. Choosing *Database links*, *Global Array links*, or *Dynamic Array links* will specify whether or not those types of links are located. The *User-Defined links* checkbox determines

whether to locate the links defined by the user through the user interface or (if the checkbox is unchecked) only the links that were defined by block developers through ModL code. The DB selectors and the GA popup menu on the right of the dialog specify which database or global array structure will be searched for. (Leaving the DB name or GA name popup menus to the default choices seen above will find any linked database or global array.)

When the Find Links button is clicked, ExtendSim opens the dialog boxes of all the blocks with the specified types of links.



Link dialog for Global Array

HOW TO

☞ If the model is large and you specify all types of links this command can open many block dialogs. It is usually advisable to search for specific types of links, so as not to have more dialogs open than needed.

### DDE links (Windows only)

One method of sharing data between ExtendSim and another application is by creating a *DDE link* between the two applications. ExtendSim supports two types of DDE links:

- 1) As a client application, importing data into ExtendSim by linking an external application's file to an ExtendSim model.
- 2) As a server application, exporting data from an ExtendSim model to an external file.

The external application and ExtendSim communicate with each other in a type of data conversation, with one application (the server) sending the data and the other application (the client) responding. A DDE link is a special registered Clipboard format that identifies the piece of data in the conversation. These are “hot” links, causing the server to simultaneously notify the client of the change and send the changed data to the client whenever the value of the specified data item has changed.

☞ This communication is unidirectional; data changes made in the server file affect the client file, but not vice versa.

The underlying technology that supports DDE linking is Dynamic Data Exchange (DDE), a Windows operating system protocol through which applications can exchange data. It is described fully at “DDE (Windows only)” on page 666. Microsoft Excel is an example of an external application that support DDE links.

☞ For data to be exchanged, DDE linking requires that the windows of both the server and the client files be open.

### Creating a DDE link

DDE links from external applications are created and managed using DDE Link commands in ExtendSim's Edit menu. Use those commands to link a value or values from the external application to parameters and data table cells in ExtendSim.

- To create an outgoing link from ExtendSim, use the Edit > Copy command in ExtendSim and the external application's linking commands, such as Paste Special.
- To create an incoming link to ExtendSim, use the Edit > Paste DDE Link command. This command will only be active if all of the following is true:
  - The external application supports DDE links.
  - Both the external file and the ExtendSim model have been named and saved.
  - Both the external file and the ExtendSim model are open.
  - The Clipboard contains data that has been copied from the external application.
  - You have selected a location in the ExtendSim model to paste to.

Parameters or cells that have been linked with this command are outlined in yellow. ExtendSim's DDE links are saved with the model. When the model is opened, ExtendSim will attempt to re-establish the associated links.

☞ An Excel worksheet embedded in an ExtendSim model can also be linked to ExtendSim data via the Paste DDE Link command.

### How to create a DDE link to ExtendSim

The following example uses a Microsoft Excel workbook and the Reservoir 1 model from the Tutorial module. It shows how to link a value from Excel to the Holding Tank's "Initial contents" parameter.

- ▶ In Excel:
  - ▶ Open a new workbook.
  - ▶ Enter the value **10** at row 1, column 1.
  - ▶ Save the workbook as **My Workbook**.

 The workbook must be named and saved before the data is copied.

- ▶ Copy the value (10) from cell; this places a 10 in the Clipboard.
- ▶ Leave the Excel workbook open!
- ▶ In ExtendSim:
  - ▶ Open the Reservoir 1 model located at \ExtendSim7\Examples\Tutorials.
  - ▶ So that you don't overwrite the original file, save the model as **ReservoirHotLink**.
  - ▶ In the Holding Tank block, click in the **Initial contents** parameter field.
  - ▶ Give the command Edit > Paste DDE Link. A yellow outline appears around the parameter field, indicating that the link has been created.
  - ▶ In the Holding Tank's dialog, click OK to save changes and close the dialog.

When you open the Holding Tank's dialog, the number 10 is in its Initial contents field. Mousing over the field shows the location of the originating data.

### How to create a DDE link from ExtendSim

To export data from an ExtendSim model to another application:

- ▶ In ExtendSim:
  - ▶ Open a model.
  - ▶ Select the dialog parameter or data table values you want linked.
  - ▶ Select Edit > Copy.
- ▶ Refer to the other application's documentation to determine how to create a DDE link. Typically, you would choose Paste Special or Paste Link from the client application's Edit menu.


Any changes made to the parameter in ExtendSim will now be reflected in the other application. To remove the link, refer to the other application's documentation.

### Managing DDE links

To change a linked value, change the value in the Excel workbook, then save the workbook. For instance, the Holding Tank's Initial contents value in the previous example will change once the value in My Workbook has been changed and the file has been saved. To remove linking from a parameter, use the Edit > Delete DDE Link command.

The Show DDE Links command opens all the dialogs that are linked using that technology in a model. If linked applications are open and the links don't appear to be working correctly, or if the

server application has been closed, the Refresh DDE Links command attempts to reestablish existing links between ExtendSim and the external application.

 If you directly change the value of an ExtendSim parameter or cell that has been linked, the field will still be outlined in yellow but the model will use the value you entered instead of the link value. The model will automatically refresh the link if the DDE value is changed and the external file is saved.

### **Updating remote references**

Creating a DDE link works in a fairly straightforward manner with the following exception: Excel doesn't support reconnecting links in embedded objects when they are reopened by default. There is an option called Update Remote References in Excel's Options dialog that will enable this behavior. The following Excel VBA macro will turn this flag on:

```
Private Sub Workbook_Open()  
    With ThisWorkbook  
        .UpdateRemoteReferences = True  
    End With  
End Sub
```

Unfortunately the value of this option does not seem to be saved in embedded Excel worksheet objects. This means that the macro above should be used in worksheets that are embedded into ExtendSim models if they use DDE link connections to share data with ExtendSim.

## **Internal data storage and management methods**

This portion of the chapter provides information about ExtendSim internal data structures. Other parts of this chapter discuss methods (such as dynamic data linking or ExtendSim blocks) for exchanging data with the internal data structures. For information about using ModL programming to interface with internal data structures, see the Developer Reference.

As discussed in the following sections, ExtendSim provides several internal structures for storing data for use in a model. See:


- ExtendSim databases that start on this page
- Global arrays starting on page 652
- Dynamic arrays on page 654
- Embedding an object that starts on page 655
- Linked lists as discussed on page 657

Access to internal data sources is accomplished through a user interface feature called dynamic data linking (DDL), by the use of data access blocks, or by programming in ModL. Databases and global arrays are accessible by the modeler and by block developers; dynamic arrays and linked lists are only accessible by block developers coding in ModL.

### **ExtendSim databases for internal data storage**

A database is a centralized repository for data. The ExtendSim graphical simulation database (GSDB) feature allows you to create internal relational databases for storing, managing, and reporting model data. ExtendSim databases also provide a convenient interface between models and external applications, such as spreadsheets and external databases.

Each model can have multiple databases associated with it and each database can contain multiple tables of data. Each table has fields that represent columns of data, records that represent rows, and cells that are the intersection of a field and a record.

 Following the standard convention for databases, ExtendSim databases are organized by field and record (similar to being organized by column and row.) However, data tables, arrays, and external spreadsheets are organized by row and column. This is important to remember if you transfer data between ExtendSim databases and other data structures such as global arrays.

ExtendSim databases are stored with the model. They automatically open when the model opens and are automatically saved or closed when the model is saved or closed.

Using ExtendSim's data access blocks or the dynamic data linking interface you can link model parameters to a database without writing any code. In this manner, parameter-based models can be easily changed to database-driven models.

### How this section is organized

Because the ExtendSim database feature is such an extensive system, this section of the manual is divided into several topics:

- Advantages of using an internal database.
- Ways to create an ExtendSim database and methods for exchanging data with it.
- How to create an ExtendSim database.
- Establishing a parent/child relationship between two database table fields.
- Managing databases, tables, and fields.
- Database dialogs and popup menus.
- Using an Excel Add-In for ExtendSim databases.

For additional information, also see these sections of the chapter:

- “Dynamic linking to internal data structures” on page 629.
- “Blocks for data management and exchange” on page 659.

### Advantages of using internal databases

There are many advantages to using an internal database. For instance, you can:

- Separate the data from the model for better project and experiment management.
- Organize information in a logical fashion, either within one database or across several databases.
- Create, view and manipulate data by product type, location, components, or any other common characteristic.
- Provide a centralized location for information that is used in several parts of a model.
- Get easier access to different sets of data depending on model needs.
- Use database tables to provide outputs and reports throughout the model.
- Reuse common sets of data from one model to the next.
- Import model inputs from, or export results to, external applications.

While using a database to manage data is indispensable for large models, the user interface makes it convenient to use an ExtendSim database even for small models.

### **Creating and interacting with internal databases**

There are three ways you can create an ExtendSim database:

- 1) Through the user interface, using menu commands to create a new database, as illustrated starting on page 640, or by importing a database text file from an ExtendSim or SDI database.
- 2) Use database functions in an equation-based block from the Value or Item library. These blocks are discussed in “Equation-based blocks” on page 601.
- 3) Program with ModL code. For more information, see the Developer Reference.

ExtendSim provides the following methods for models to interact with databases;

- 1) Through the user interface, using dynamic data linking to establish live links between dialog items and databases. This is demonstrated in “Dynamic linking to internal data structures” on page 629.
- 2) Using Read and Write blocks (Value library) or Read(I) and Write(I) blocks (Item library) to exchange data between a model and a database. This is illustrated in “Read and Write blocks for accessing a database” on page 645.
- 3) Using database functions in an equation-based block from the Value or Item library to manage data. These blocks are discussed in “Equation-based blocks” on page 601.
- 4) Programming using ModL code. See the Developer Reference for more information.

### **How to create an ExtendSim database**

The following example shows how to use menu commands to create a database for the Reservoir model that was shown in the Tutorial module. While a database is obviously unnecessary for the Reservoir model, the purpose of the example is to show how to create an internal database.

As illustrated in the example that follows, the steps are:

- 1) Open a new or existing model
- 2) Create a new database for the model.
- 3) Add tables and fields to the database.
- 4) Add database records.
- 5) If the database will be used to input values to the model: enter, paste, or import values for the database cells.

#### ***Opening an existing model***

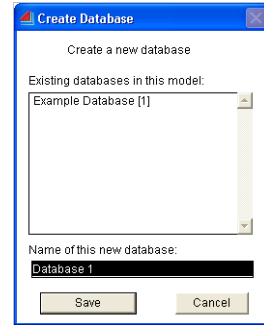
- ▶ Open the model Reservoir 1 from the \Examples\Tutorials folder.
- ▶ So that you don't overwrite the original file, save the model as **ReservoirDB**.



**Starting a new database**

- ▶ With the model window active, choose the command Database > New Database.

The Create Database dialog opens with a field for naming the database. (Note that an existing database is already listed in the scrollable field at the top of the dialog. That Example Database is used for illustrating features in other sections of this chapter.)



Create Database dialog

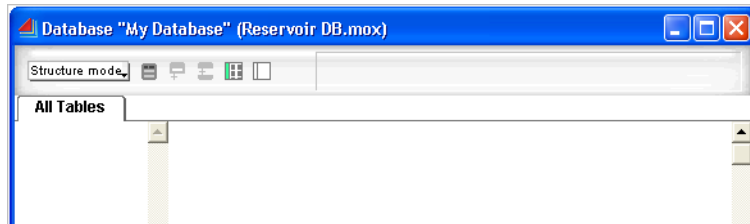
- ▶ ExtendSim assigns each database a unique index number, starting with 1. As seen for the Example Database, that number is appended to the database name and enclosed in brackets. Index numbers are used for referring to databases in equations or ModL code.

- ▶ Name the new database **My Database**.

The name can be anything you want as long as it is not used by another database and does not exceed 63 characters. Database names are not case sensitive; they can contain spaces.

- ▶ Click Save to close the dialog and save the new database.

The database window opens in structure mode, as shown at right. The header for the window displays the name of the database and its associated model. The structure window is



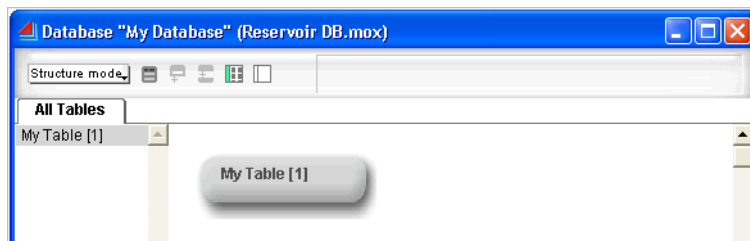
Database window

where you create database tables, add fields to the tables, and create relationships between fields.

**Adding tables and fields**

- ▶ With the database window active, create a new table using one of these methods:
  - ▶ Give the command Database > New Table.
  - ▶ Or, right click on the database window.
  - ▶ Or, click the New Table button in the database's toolbar.
- ▶ In the dialog that opens, name the table **My Table** and click Save.

The new table is placed in the All Tables tab. The name of the table is listed in the table list pane at the left of the tab and the database structure is shown in the tables pane on the right of the tab. As



Adding new database table

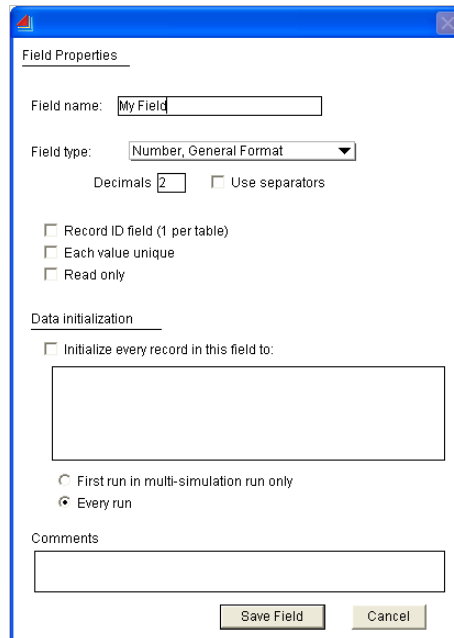
How To

is true for databases, after its name each table is assigned an index number enclosed in brackets.

- ▶ Create a new field using one of the following methods:
  - ▶ Select the table named My Table and give the command Database > Append New Field.
  - ▶ Or, right click the table and choose Append New Field.
  - ▶ Or, select the table and click the Append New Field button in the database's toolbar.

▶ In the Field Properties dialog (shown at right):

- ▶ Name the field **My Field**.
- ▶ Choose the default choices: **Field type: Number, General Format** and **Decimals: 2**.
- ▶ Click Save Field.



Field Properties dialog

**Creating records**

To access the table so you can create records, do one of the following:

- ▶ Double-click the table in the database window to open the viewer for that table.
- ▶ Or, double-click the table name in a tab's list of tables to open the viewer.
- ▶ Or, select the table and change the popup menu in the database window's upper left corner from Structure mode to Viewer mode. This opens a viewer pane for that table within the database window.

▶ Click the Append New Records tool in the database toolbar, or give the command Database > Append New Records.

▶ In the dialog that opens, enter **2** for the number of records and click Add Records.

**Entering values for the cells**

- ▶ In the viewer, enter **10** as record #1 and **20** as record #2.
- ▶ If the table viewer window is open, close it.
- ▶ Close the database window and Save the model. Databases are saved with the model when the model is saved.

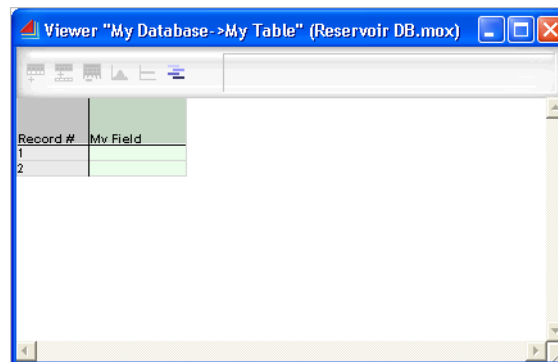


Table Viewer showing two records

Notice that My Database is listed at the bottom of the Database menu. If the Navigator is opened to Database List mode, the database will also be listed there. Making a copy of the ReservoirDB model causes the copy to also have a database named My Database.

How To

### Establishing Parent/Child relationships

To illustrate another powerful database feature, the following example adds a new table to the database created in the previous section and makes the table's field the parent for the field in the original table. This will establish a parent/child relationship, causing My Table to get its values from the new table.

Establishing a parent/child relationship between fields is optional but powerful. It is useful for many situations, such as providing a components list for a specific product, a selection of custom colors for a particular model of car, or the properties of a material. Having a parent/child relationship limits a field's set of data to what is present in the parent. Instead of entering data directly into the child field, you select the data from a popup data selector that shows all the possible values from the parent field.

Parent fields have a red background and tables that have parent fields are listed in the database window's tabs in red; child fields have green backgrounds and tables that have child fields are listed in green.

#### How to create a parent/child relationship

The following example builds on the ReservoirDB model created in the example "How to create an ExtendSim database" on page 640. It assumes you know how to add tables, fields, and records to a database.

- ▶ Open the ReservoirDB model you created earlier.
- ▶ Open the window for My Database by selecting My Database at the bottom of the Database menu. Or use one of the methods discussed in "Opening a database window" on page 646.
- ▶ Create a new table and name it **Parent**.
- ▶ Select the Parent table and:
  - ▶ Append a new field and name it **Contents**.
  - ▶ Leave the other settings in the Field Properties dialog at their default settings and click Save Field.
- ▶ Double-click the Parent table to open the viewer window.
  - ▶ Add 4 records to the Parent table.
  - ▶ Enter the values **5, 8, 10, and 15**, one value for each record.
  - ▶ Close the viewer window so that the database window is active.

Notice that each field in a table has *connector points* on its left and right sides. These are used for establishing parent/child relationships.

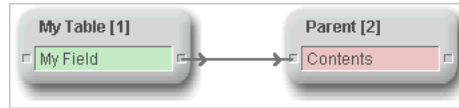
- ▶ Draw a line *from* a connector point for My Field *to* a connector point for Contents.

 Draw the line so the arrow points to the parent, the source of the data.

Since the child field has existing data, the Parent/Child Relationship dialog opens with options for how the data in the child field should be handled.

- ▶ In the Parent/Child Relationship dialog, select the default (top) choice and click Set Relationship. (The dialog is discussed later in this section.)

This establishes the Contents field as a parent to the field named My Field. Parent fields have a red background and are listed in red; child fields have green backgrounds and are listed in green.



Parent Child relation between fields

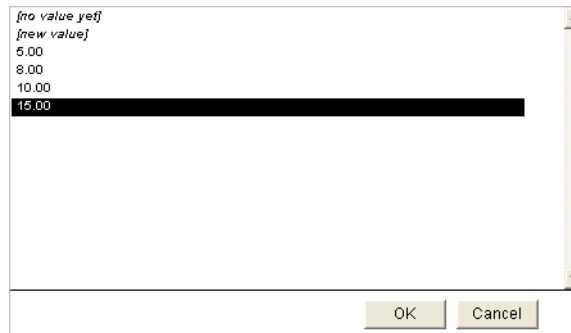
▶ Switch the database window to Viewer mode so you can select values for the records in My Field.

- ▶ Select My Table in the table list pane at the left of the All Tables tab.
- ▶ Click the arrow in record #1.
- ▶ In the popup selector that appears, select the value 15.

The child cell for field 1, record 1 is now set to 15; it gets that value from the parent field. If the parent value changes, the child value will change (but not vice versa.)

**The Child popup selector**

In Viewer mode, when you click the down arrow in a child cell it displays a popup menu with 3 types of options. The choices are to select a value (whichever number is listed), add a “new value”, or choose “no value yet”. Choosing “new value” allows you to place a new value in the cell; the value is appended to the parent field. The “no value yet” option leaves the cell blank.

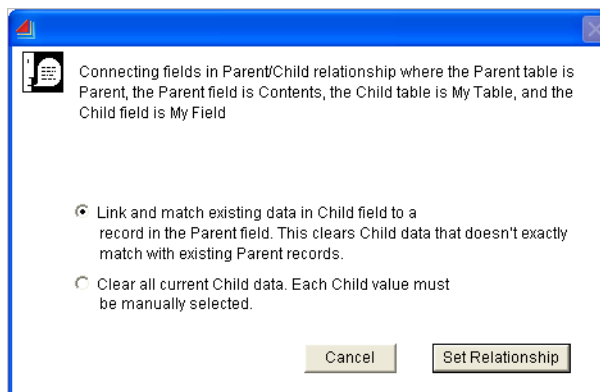


Child popup selector

**The Parent/Child Relationship dialog**

If there is existing data in a child field, this dialog opens and allows you to arbitrate what happens when the relationship is made. If there is no existing data, the dialog does not open. The options are:

- Link and match existing data in Child field to a record in the Parent field. If existing data in the child matches data in the parent, the data is left unchanged. Otherwise, it is cleared and the record is left blank so the value can be manually selected.
- Clear all current Child data. This option clears all the existing data in the child.



Parent Child relationship dialog

How To

### Linking a database to data


Databases are used to store data for use in the model or to store model outputs. A model can access a database's information by dynamically linking to it, by using data access and equation-based blocks, and through ModL programming.


The following section discusses how to dynamically link a dialog item to a database. Using Read and Write blocks to access database data is described below.

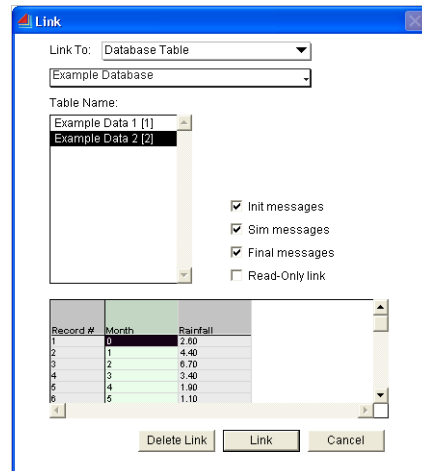
#### Dynamic data linking

The steps for dynamically linking a block's dialog item to an ExtendSim database table are:

- 1) Create an ExtendSim database (see "ExtendSim databases for internal data storage" on page 638.)
- 2) Link a parameter or data table, as illustrated in "Dynamic linking to internal data structures" on page 629.
- 3) In the Link dialog, select **Link To: Database Table**.
- 4) Choose a database from the popup menu of available databases.
- 5) Select a table from the list of **Table Names**.
- 6) For parameters only, choose a field from the **Field Name** popup, then enter a number in the **Record #** field. Or, select the field and record by clicking on a cell in the table viewer at the bottom of the Link dialog.

 The table viewer displays the contents of the database table you are linking to. It is also useful for selecting the database field and record for a parameter.

 Dynamic linking should be used judiciously. Avoid overloading models with dynamic links to data sources that get frequently updated; it could slow simulation performance as messages are sent to linked blocks every time their linked data source is modified.



Link dialog for Database Table

#### Read and Write blocks for accessing a database

Instead of dynamically linking a dialog's parameter or data table to a database cell or table, the Read and Write blocks (Value library) and Read(I) and Write(I) blocks (Item library) can access a database cell, row, or column for use in a model. These data access blocks are useful when you don't want to, can't, or shouldn't dynamically link a dialog item. The advantages of using these blocks rather than dynamic linking is:

- Using data access blocks gives more control over which database values get used, and where and when they are used.
- Database usage is clarified visually. For example, instead of a blue outline around a parameter field inside a block's dialog, it is more obvious when the database value is coming from a Read block.

- Dynamic linking is fixed point-to-point. While you can change the value at the data source, you can't dynamically control which database or table is accessed during a simulation run or in a series of runs.
- In some blocks there is no dialog parameter or data table you can directly link to,
- It is easier to perform mathematical calculations on the database values.
- The data can be more easily accessed at several places in the model.
- It is more convenient for hierarchical blocks, allowing access to a different record for each instance of the hierarchical block in a model.

The Monte Carlo model, located at \Examples\Continuous\Standard Block Models and described on page 47 is an example of using Read and Write blocks to exchange data with an ExtendSim database. For more information about these blocks, see “Read and Write blocks” on page 659.

### Database management

This section provides additional information about using and managing ExtendSim databases, database tables, fields, records, and data.

#### Opening a database window

To open a database's window so that you can change its structure or data, do one of the following:

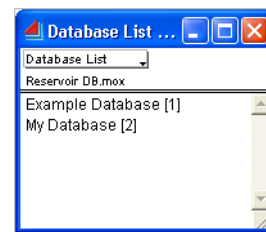
- ▶ Select the database from the databases listed at the bottom of the Database menu.
- ▶ Or, open the Navigator, select its Database List mode, and double-click the name of the database in the list.
- ▶ Or, select Window > Database List to open the Navigator in Database List mode, then double-click the name of the database in the list.

#### Opening a database list

To access the Database List, give the command Window > Database List. The database list is used to open a database or when copying, renaming, or deleting a database.

#### Copying, renaming, or deleting a database

To copy or duplicate a database, select it in the database list and give the command Edit > Copy (or Duplicate) Database. Give the Paste command in the database list or on the model worksheet; you can paste into the current model or a different model. Pasting or duplicating places a copy of the database in the database list with a different index number and a modified name (if the database is copied into the same model's Database List) or with the same name (if the database is copied into a different model).



Database list

- ☞ Giving the Paste command on the model worksheet places a copied ExtendSim database in the Database List and makes it available to the model. It does not put a visual representation of the database on the worksheet.

To move a database from one model to another, select the database in the Database List and use the Edit > Cut Databases command, then paste the database into the other model.

To rename a database, select it in the Database List and give the command Database > Rename Database. To delete a database, select it in the Database List and give the command Edit > Clear Database.

**Importing or exporting a database**

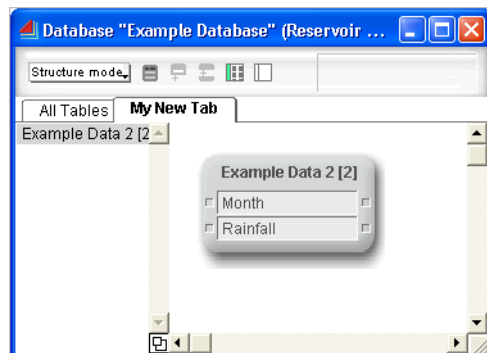
The command Database > Import New Database creates a new ExtendSim database by importing an entire database from an exported ExtendSim or SDI Industry database. In the dialog, select the database text file to import. This command imports all the tables, fields, records and so forth from the exported file and creates a new database. If you choose the name of an existing database, it will be replaced. To import tables to an existing database, such that the database tables are appended at the end of the database, see the Import Tables command. See also “Excel Add-In for ExtendSim databases” on page 650.

The command Database > Export Database is only active when an ExtendSim database window is active. This command exports the entire ExtendSim database into a text file. You do this so you can import the database into another model, send the database to another user, or to prepare a database text file for use by the ExtendSim DB Add-In for Excel (discussed on page 650). To export only specific tables from a database, see the Export Selected Tables command, discussed below. To enable this command, bring a database window to the front, as described under Database menu, above.

 Exported databases can only be imported to ExtendSim or to the ExtendSim DB Add-In for Excel.

**Managing database tables and using tabs**

Tabs are useful for organizing tables by category. The All Tables tab will always contain a list of all of a database’s tables in its table list pane. You can also list some of those tables in other tabs. The database window must be in structure mode to add a tab. To create the tab, either double-click the blank area of the tab bar (that is, to the right of the All Tables tab) or give the command Database > New Database Tab. Clone a table to the new tab by selecting the table in the tables pane and giving the command Database > Clone Selected Tables to Tab. In the dialog that appears, select a tab to clone the table to.



Database window showing new tab with cloned table

Import and append tables into the current database by giving the command Database > Import Tables when a database’s window is open. This imports a database text file of tables from an ExtendSim or SDI Industry database.

Export selected tables by using the Database > Export Selected Tables command. The tables can be exported as text files from one ExtendSim database to a different ExtendSim database.

To show/hide tables in the tables pane, unselect the table in the list of tables in any tab in the database window.

To copy a table from one database to another, select the table in the database window, give the command Edit > Copy Tables, and paste the table into the same or a different database’s window.

To move tables from one database to another, use the Edit > Cut Tables command instead of the Copy Tables command.


**Managing fields**

Fields are formatted using the Field Properties and Field Type dialogs, discussed in “Database dialogs and popup menus” on page 648.

To reorder fields in a table, with the database window in Structure mode, click and drag the fields.

To resize a field’s column width, put the database window in Viewer mode. Then hover the cursor over the column divider until it turns into a resize cursor, then click and drag to resize.

Use the Sort Table tool in the Viewer toolbar to sort a table’s records by up to three fields; each field can be sorted in either ascending or descending order. For example, use this to sort a table’s records first by last name, then by first name, and then by city.

 To create a field with mixed formats (for instance, with both strings and numbers), set the field as a string type in the Field Properties dialog. Enter numbers or strings into the cells.

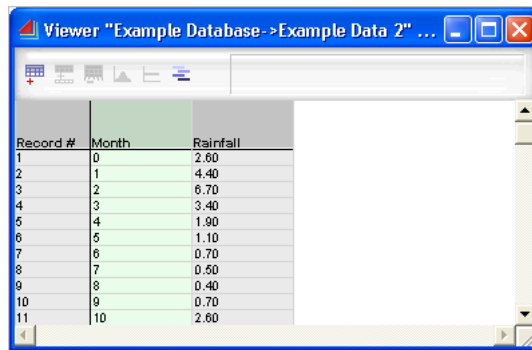
**Editing data**

To access data for editing, in the database window do one of the following:

- ▶ Double-click a table in the database window to open its Viewer window.
- ▶ Or, select the table in the tables pane and change the database window’s mode from Structure to Viewer.

The records and fields for the selected table will be displayed.

To make a cell random, select the cell and choose the Make Cell Random tool in the Viewer toolbar. By default, all cells are constant.



Database table viewer window

**Finding data**

To find a string or number within a database, with the database structure window open, give the command Edit > Find. This command opens the Find dialog. To only search a single table, select the table before giving the Edit > Find command.

**Database dialogs and popup menus**

Most of the database dialogs are self-explanatory. The following are more unusual or complex and require some explanation.

**The Field Properties dialog**

When you add a field to a database table, the Field Properties dialog presents formatting and other options for the field, as described in the following table.

Option	Description
Field name	Enter any unique name, up to 63 characters. Spaces and characters are allowed.
Field type	Format for the field. See the following table.

How To



Option	Description
Decimals	For general format or scientific numbers, the number of digits to display. For currency and percent, the number of digits to the right of the decimal point.
Use separators	Inserts commas and periods to separate digits.
Record ID field	The field used as an index when searching for a record in this table.
Each value unique	When checked, the number or string for each record in this field must be unique.
Read only	When checked, prevents the data from being changed after it has been entered.
Initialize every record in this field to:	If checked, initializes the data for all the records in the selected field to the entered parameter value. If the parameter is left blank, initializes the field's records to a blank.
First run...Every run	Specifies when the record initialization takes place.

**Field type popup menu**

The following table describes the options for the “Field type” popup menu in the Field Properties dialog.

Field type	Description
Number	An integer or double-precision floating point number. Choose general, scientific, percent, currency or integer.
String	An alphanumeric string up to 255 characters
Date/Time	Calendar date and time
Data address	Describes the coordinates of a cell in a database table
Boolean checkbox	Places a checkbox to indicate yes or no, true or false, on or off, etc.
List of tables	Provides a popup menu of all the tables in the database, so you can select a table.

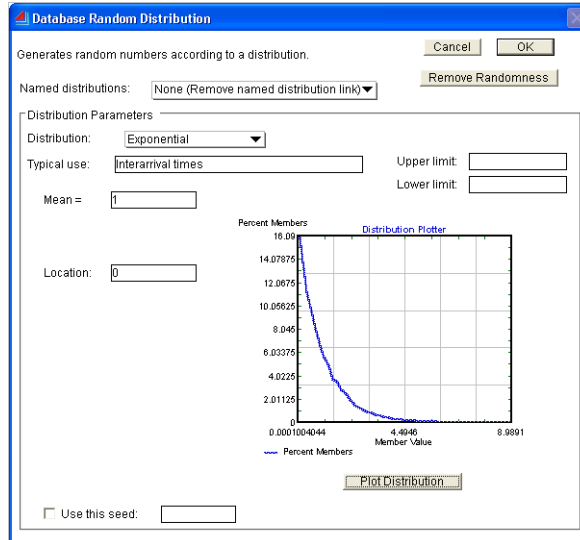
### Database Random Distribution dialog

By default all database cells are constant, but they can be formatted as random numbers. To make one or more cells random, select the cells in the Viewer and choose the Make Cell Random tool in the toolbar. Or right-click the cell and choose Make Random. This opens the Database Random Distribution dialog, shown at right.

The random distributions in this dialog are the same as for the Random Number block (Value library). For a complete list and short description, see “Choosing a distribution” on page 606.



A special feature of this dialog is that you can assign a name to a distribution in a database, allowing the distribution to be selected by custom name in that database. For example, enter typical values for an empirical table and name it Machine Time Distribution. Or give a custom name to an exponential distribution with specific arguments.



Database Random Distribution dialog

### Excel Add-In for ExtendSim databases

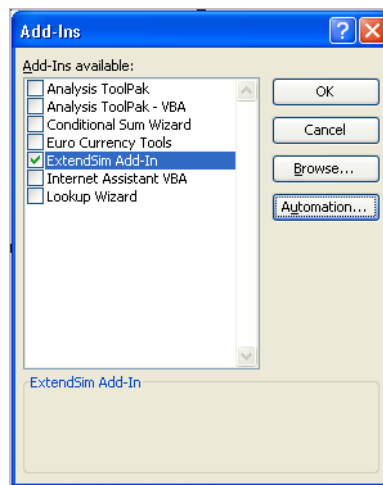
The ExtendSim DB Add-In allows an exported ExtendSim database text file to be imported into Excel. The data and structure of the ExtendSim database can thus be edited in Excel, outside of the ExtendSim application. After editing, the database can be exported from Excel as an ExtendSim database text file and imported into ExtendSim.



The ExtendSim DB Add-In is only included in the ExtendSim Suite and ExtendSim AT products. It may be purchased separately for use with other ExtendSim products.

To install the add-in into Excel:

- ▶ Follow Excel’s instructions for installing an Add-In. These will differ depending on the version of Excel and the platform you’re installing on.
- ▶ In the Add-Ins dialog, click the Browse button and navigate to the ExtendSim DB Add-In.xla file; it is located in the ExtendSim\Extensions folder.
- ▶ Select the ExtendSim DB Add-In.xla file and click OK. This returns you to the Excel Add-Ins dialog and puts ExtendSim DB Add-In in the list of available Add-Ins. (Make sure the checkbox next to ExtendSim DB Add-In is checked.)



Excel Add-Ins dialog

How To

- ▶ Click OK to close the Excel Add-Ins dialog. This action installs a new Add-In menu command called “ExtendSim DB”.

To export an ExtendSim database file from ExtendSim so that Excel can import it, see “Importing or exporting a database” on page 647. The exported file will be in text file format.

To import the exported ExtendSim database file into Excel, give the command ExtendSim DB > Import Database in Excel. After importing the file, you can modify data in the tables, change the number of records (rows) or fields (columns), change the field types, modify named and empirical distributions. Although it is done infrequently and is much easier to do in ExtendSim, you can even add or change relationships in the database.

- To modify field characteristics, click the Ungroup (+) button at the left of a table. This expands the field properties so you can change them as desired.
- To add rows or columns to a table that already has one or more rows and columns, copy an existing row or column in the table and paste rows at the bottom of the table or paste columns at the right side of the table. This will enlarge the table. Then enter any needed data into the new rows or columns. If the table doesn't have any rows, enter row numbers below the Field identifier. The screenshot to the right has two rows (numbered 1 and 2) below the Field identifier.
- To change relationships, enter the correct table and field names into the Indexed Fields table at the bottom of the All Tables tab.

Table	Example Data 1	
Notes		
Fields	Initial Values	
	1	15
	2	25

☞ When changing the number of rows in empirical distributions, be sure to also change the value for the Number of Empirical Rows.

When you are finished making changes, export the database file using the Excel command ExtendSim DB > Export Database. Then import that file into ExtendSim using the command Databases > Import New Database as discussed in “Importing or exporting a database” on page 647.

Table	Example Data 1	
Table Index	1	
Table in Tabs	1	
Tab Locations	“(20,20)”	
Table Format	0	
Access	read_write	
Format	general	
Decimal	2	
Comma	no	
Unique	no	
Record ID	no	
Do Initialization	no	
Init First Run Only	no	
Initializer		
Notes		
Fields	Initial Values	
	1	15
	2	25

Grouped and ungrouped tables

### Monte Carlo model

The Monte Carlo model (located in the folder \Examples\Continuous\Standard Block Models) uses an ExtendSim database named “Scenarios” to store the inputs and results of the model. Using a database allows the model to experiment with different scenarios. Rather than dynamically linking model parameters and data tables to a database, the model uses Read and Write blocks (Value library) to provide the interface to the database. The Read blocks get random values from a database and use them to represent model parameters where the actual values are not known with certainty.

**How To**

### Other internal data storage and management methods

While ExtendSim databases will probably be your primary method of storing and managing data, ExtendSim provides several other methods. These are:


- Global arrays discussed starting on this page
- Dynamic arrays on page 654
- Embedded objects starting on page 655

- Linked lists discussed on page 657

### Global arrays

A global array is a two-dimensional (row and column) array of data that is accessible by any block in a model. Like ExtendSim databases, global arrays are user-accessible structures for internally storing and managing data. Use global arrays to share information between blocks when a direct connection is either inconvenient or impossible, to store information that can be accessed by a row and column index, or to exchange data with external sources.

Global arrays provide many of the same modeling advantages that ExtendSim databases do; see “Advantages of using internal databases” on page 639, but they differ from databases in some important aspects. Compared to databases, global arrays have a simpler data structure and take less time to access data. However, global arrays aren’t as flexible as databases, which allow mixed data types, randomness, and parent/child relationships. It is also easier to add or remove fields in a database than columns in a global array, and creating a global array requires adding a block to the model.

 Unless you are using custom blocks that create global arrays, a Data Source Create block must be in the model to initially create the global array. Since the global array is saved with the model, the block can be removed after the array has been created.

While it is common to use a database for managing and exchanging data, global arrays are handy in situations where:

- You just want a simple array structure of a single data type. For instance, to create an integer array with 10 columns and 20 rows of data for use in a model.
- You program blocks that can use a more simple data structure than a database requires, and you don’t need or want the data storage to be visible to the block user.


### Creating a global array

There are three ways to create a global array:

- 1) Use the Data Source Create block (Value library) as illustrated later in this chapter.
- 2) Through global array functions in an equation-based block from the Value or Item library. These blocks are discussed in “Equation-based blocks” on page 601.
- 3) Programming with ModL code. For more information, see the Developer Reference.

You can name a global array anything you want as long as the name is fewer than 32 characters and it does not begin with the underscore character (`_`). Global array names are not case sensitive; spaces between characters are allowed.

A global array can only be of one data type - either integer or real (if the data type is selected through the user interface), or integer, real, or string (if the data type is selected through ModL code.) Each model can have one or more global arrays associated with it and each global array can have multiple rows and columns. Global arrays are stored and saved with the model, even if the Data Source Create block that created the array is deleted.

 Following the standard conventions, ExtendSim databases are organized by field and record (similar to being organized by column and row.) However, arrays are organized by row and column. This is important to consider when transferring data between databases and global arrays.

**How to create and use a global array**

The example that follows uses the Data Source Create block to create a global array for the Reservoir 1 model. As illustrated below, the steps are:

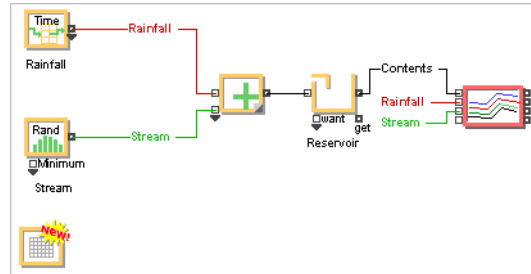
- 1) Place a Data Source Create block in a model.
- 2) Build the global array in the Data Source Create block.
- 3) Populate the array with data.
- 4) Determine how the model exchanges data with the array.

*Using an example model*

- ▶ Open the model Reservoir 1 from the \Examples\Tutorials folder.
- ▶ So that you don't overwrite the original file, give the command File > Save Model As and save the model as **ReservoirGA**.
- ▶ Place a Data Source Create block (Value library) at any convenient location on the model worksheet.

*Creating the global array*

- ▶ In the dialog of the Data Source Create block:
  - ▶ Choose **Type: Global array**.
  - ▶ In the "Create a new array" frame, click the Integer Array button, indicating that you want the values to be integer.
  - ▶ In the three dialogs that appear, name the array **My Array**, give it **12** rows, and give it **2** columns. Click OK after each dialog entry.



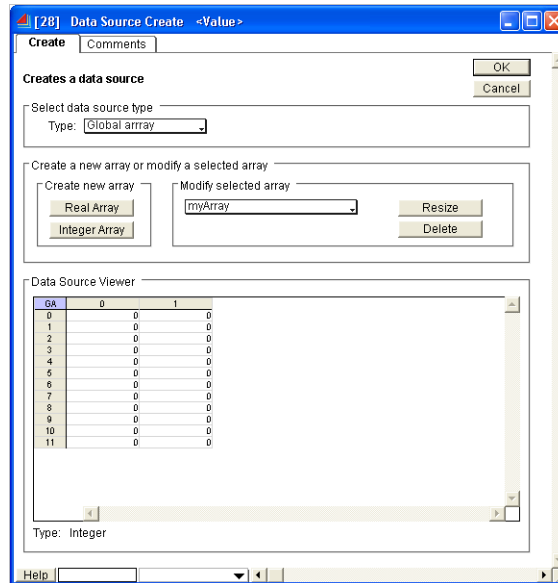
ReservoirGA model

Notice that Viewer for the selected array in the Data Source Create block now displays 12 rows and 2 columns. The table's upper left corner is light blue, indicating that it is a data source, and it has the initials "GA", indicating that the data source is a global array.

*Populating the array with data*

Global arrays can be used as inputs to a model or to store model results.

- A common method of populating an array with data for use in a model is to enter values or copy data directly into the Data Source Create block's Viewer table when the array is created. You can also use the Data Import Export block (Value library) to import data to the array from a spreadsheet or external database, the web, or a text file.
- To store model results, it is most common to use a Write block (Value library) to send data to a global array.



Data Source Create after creating myArray

The Data Init block (Value library) can be used to initialize a global array table, column, row, or cell; it is described on page 660. The Data Specs block (Value library) gathers information about the global array and puts it into a report; it is discussed in page 661.

*Exchanging data with a global array*

ExtendSim provides the following methods for models to interact with global arrays:

- 1) Through the user interface, using dynamic linking to establish live links between dialog items and global arrays. See the examples "How to link a parameter to a global array" on page 631 or "How to link a data table to a global array" on page 633.
- 2) With data access blocks, such as the Read and Write blocks (Value library), to exchange data between a model and a global array. Data access blocks are discussed on page 659.
- 3) Using global array functions in an equation-based block from the Value or Item library to manage data. These blocks are discussed in "Equation-based blocks" on page 601.
- 4) Programming using ModL code. See the Developer Reference for more information.

**Dynamic arrays**

A dynamic array is multi-dimensional internal data structure that is only accessible through programming. A model user can resize a data table that is linked to a dynamic array and can link the data in the table to an ExtendSim database or global array. However, the dynamic array itself cannot be accessed outside of ModL code.

A data table linked to an internal data source will have its upper left corner in light blue rather than the grey of the table header. If the source is a database (DB) or global array (GA), those ini-

How To


tials will be displayed in the corner. If the source is a dynamic array, the upper left corner will not have any initials.

Since dynamic arrays are created and managed through ModL code, see the Developer Reference for more information.

### Embedding an object (Windows only)

Embedded objects allow applications to have new behaviors and functionality without needing to custom program the behavior. An embedded object is an external software component created with one application and embedded into a document created by another application. The component could be an embeddable part of an existing application (such as a Worksheet object from Excel), or a component specifically created for embedding (such as a Graphics Server ActiveX control). Embedding the object ensures that the object retains its original format and that it can be modified with the original program. The underlying technology for embedded objects is OLE (see “ActiveX/COM/OLE (Windows only)” on page 665).


ExtendSim’s support of this functionality allows model authors and block developers to incorporate objects and ActiveX controls that can do things that might otherwise be difficult or impossible in ExtendSim.

 For purposes of this manual, the word “objects” will mean both “component objects” and “ActiveX controls”.

As illustrated below, an object can be embedded into:

- A model worksheet
- A block’s dialog (if you program the block in ModL)

When an object is embedded into a model worksheet, ExtendSim creates a *container object*, a worksheet object similar to a block or a hierarchical block. Each container object has a block number so that ExtendSim functions can reference it. A container object can be selected, moved around the screen, and deleted in the same manner as a block.

 If clicking a container object causes any response other than selection, you need to select the command Edit > Design Mode to move it, as discussed on page 688.

Double-clicking an embedded object activates features of the originating program. For instance, double-clicking an embedded Excel Chart provides access to Excel tools for selecting a chart type and configuring its display.

A typical use of embedded objects would be to insert an Excel worksheet into an ExtendSim model, then use DDE Linking (discussed on page 636) or the Value library’s Read and Write blocks (discussed on page 659) to exchange data between the model and the embedded object. This gives ExtendSim the added capabilities of Excel, without opening the Excel application.

### How to embed an object into a worksheet

The example that follows embeds an Excel worksheet into an existing model and uses the Write block to send results to the worksheet.

- ▶ Open the Reservoir 1 model located in the folder \Examples\Tutorials.
- ▶ So that you don’t overwrite the original file, save the model as **ReservoirEmbed**.
- ▶ Select Edit > Insert Object. The Insert Object dialog appears. It lists the embeddable objects that are available on your machine. The dialog is shown on page 657.

- ▶ In the dialog, select the button **Create New** (the default choice) and choose **Object Type: Microsoft Office Excel Worksheet**. Then click OK to close the dialog and place the embedded object in the model.

If the Excel worksheet object is in an inconvenient location, click elsewhere on the model window to deselect it, then click back on the worksheet to select and move it.

- ▶ Delete the Plotter I/O block on the right side of the model.
- ▶ Add a Write block (Value library) to the model where the Plotter block was. Then open its dialog:
  - ▶ In the “Select destination type” frame, choose **Send data to: Excel workbook**, check **Use embedded workbook**, and select **Embedded workbook list: Worksheet in ReservoirEmbed**.
  - ▶ In the “Specify Excel coordinates” frame, select **Sheet: Sheet 1**, **Write: one row**, and **Col end: 4**.
  - ▶ Leave the other settings at their defaults and close the Write block’s dialog.
- ▶ To the left of the Write block, add a Simulation Variable block (Value library) to the model.
  - ▶ In its dialog, select **Current time**.
  - ▶ Connect the output of the Simulation Variable block to the top input of the Write block. In the finished model, this block will output current simulation time to the Excel worksheet.
- ▶ Draw a connection line from the remaining inputs of the Write block to the Contents, Rainfall, and Stream named connections. (Be sure to make a complete connection; if a line segment is dotted, it is not connected.)
- ▶ Add a second Simulation Variable block (Value library) to the model. In its dialog, select **Current step** and specify that it adds **1** to the result.

Adding 1 is necessary because ExtendSim is zero-based and Excel is one-based, as discussed in “The mailslots feature allows communication between two ExtendSim applications running on different computers on the same local area network (LAN). Since mail slots involve programming, they are described in the Developer Reference.” on page 668.

- ▶ Extend the variable connector at the bottom of the Write block so that the R (row) input is displayed. Then connect the output of the second Simulation Variable block to the R input.

This causes the Write block to write the data to a new row at each step.

- ▶ Run the model.

The Excel worksheet should show 4 columns of data. Simulation time is the first column and the reservoir contents, rainfall amount, and stream contributions are the other three columns.

To access Excel’s formatting tools, double-click the embedded Excel object. This places Excel commands and tools within the ExtendSim application window. Use these tools and commands to format the object’s data, chart the data, and so forth.

### **How to embed an object into a dialog**

Block developers can embed objects into the blocks they create. They do this by selecting the “Embedded Object” type of dialog item when creating the block. This places a rectangle in the dialog editor for embedding an ActiveX control or other kind of object. The embedded object is



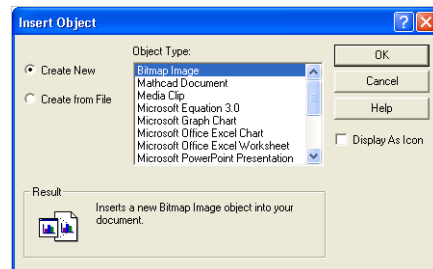
referenced in the block's code via a combination of its dialog item name and container object block number, using ModL functions as described in the Developer Reference.

The Bar Chart and Gantt Chart blocks (Plotter library) are examples of Graphics Server ActiveX controls embedded in ExtendSim block dialogs. For more information on ModL programming to control embedded objects or how to use ExtendSim as an Automation Client or Server, see the Developer Reference.

### The Insert Object dialog

The Insert Object dialog appears when the command Edit > Insert Object is given; it is used for selecting an object for embedding. A scrollable lists displays the insertable objects that are available for your machine.

If “Create New” is selected in the dialog, ExtendSim will create a new object of the indicated type with no data associated with it—basically an empty document of the specified type. Selecting “Create from File” creates an embedded object with the data from that file as the starting point. If the “Display As Icon” box is checked, the new object will only display in ExtendSim as an icon, rather than as the control's interface.



Insert Object dialog

The use of the Insert Object dialog will have different meanings depending on which ExtendSim window is open. If a model or hierarchical worksheet window is open, issuing this command will insert a Container Object on the model. If a dialog with an Embedded Object dialog item is open, it will insert the embedded object into that dialog item.

### Linked lists

A linked list is an internal data structure that allows the construction and manipulation of complex lists of data. It can be accessed only through programming. Linked lists are queue-like structures of multiple types that maintain internal pointers between different elements. This enhances the sorting for complex structures by speeding the movement of the elements within the list.

ExtendSim implements linked lists:

- In the queue blocks (Item library)
- Extensively in the Rate library
- With the Linked List functions described in the Developer Reference.

Since this involves programming, linked lists are described in the Developer Reference.

## Exchanging data with external applications

The means by which two or more applications communicate with each other and share data are collectively known as *interprocess communications (IPC)*. Some communication methods have previously been discussed in this chapter. For example, the Clipboard provides convenient data sharing between applications running on the same operating system. The following section focuses on more extensive communication methods which allow ExtendSim to directly communicate with other applications and with ExtendSim models running on other computers, *while the simulation is running*. This allows ExtendSim to work on a wide variety of tasks jointly with external applications such as:

- Spreadsheets
- External databases
- Other applications, such as word processors and statistics packages

There are several ways you can incorporate IPC into models, including:

- With blocks from the Value library discussed on page 659.
- Using the DDE Link feature (Windows only) described on page 636.
- Embedding objects (Windows only) as seen on page 655.
- If you program, using the OLE and IPC functions discussed in the Developer Reference.

An application that communicates with another application can be categorized as either *client* or a *server*. A client application requests a service from some other application and a server application responds to the client's request. Many applications, such as ExtendSim, can act as either a client or a server depending on the circumstances.

### Spreadsheets

You may want to use spreadsheet data as the input for an ExtendSim model. Likewise, you may want to send output data to a spreadsheet for further analysis or presentation. For instance, if you have a spreadsheet that performs calculations on large amounts of data, you may want that spreadsheet to respond dynamically as you model real world conditions in ExtendSim.

ExtendSim facilitates communication with spreadsheets by:

- The Command, Data Import Export, Read, and Write blocks (Value library). These blocks can send data to and from spreadsheet applications, send one value to a specified cell in a spreadsheet and request the recalculated value from another cell, or send commands or a macro to a spreadsheet. See “Blocks for data management and exchange” on page 659.
- DDE Link commands in the Edit menu. For more information, see “DDE links (Windows only)” on page 636.
- The ExtendSim DB Add-in, which allows an exported ExtendSim database text file to be imported into Excel. The data can then be stored in Excel, edited and exported to an external database, or edited and exported back to ExtendSim as a database text file.
- Import Data and Export Data commands in the File menu to create text files that can be imported into a spreadsheet or to import a text file that has been exported from a spreadsheet. For more information, see “Importing and exporting data” on page 626.
- OLE and IPC functions to facilitate DDE or ActiveX/COM/OLE communication with spreadsheets. For more information, see the Developer Reference.

### External databases

In many situations, the historical data for a model is stored in an external database. ExtendSim models can share information with external database applications:

- With the Data Import Export block (Value library) to send data to or from ODBC database applications. This block can import a block of data to, or export a block of data from, an ExtendSim database or global array. See “Data access blocks” on page 660.

- By using Excel as an intermediary application between an ExtendSim database and an external database. In this case, the data is imported from the external database to Excel, where it is manipulated. The data is then exported from Excel to ExtendSim as a database text file using the ExtendSim DB Add-in.
- Using the ExtendSim ODBC functions to initiate an SQL query or perform other functions. For more information, see the Developer Reference.

### Blocks for data management and exchange

Several ExtendSim blocks facilitate data storage and management. Some are used to establish and control dynamic links between ExtendSim and internal or external sources. Others are useful for importing or exporting data between ExtendSim and external applications.

In general there are several uses for data management and access blocks:

- Sharing information between blocks when a direct connection between them is inconvenient.
- Storing information which can be accessed by a row and column index.
- Accessing existing spreadsheets, either embedded in the model or imported as a file.
- Accessing existing databases using ODBC/SQL.
- Accessing internet-based data.

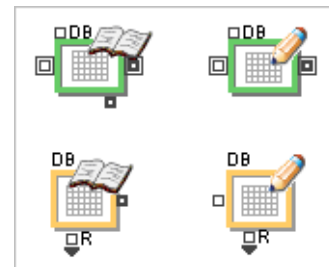
Most of the following blocks allow block-level access to data structures without having to program with the ModL functions. An additional group of blocks are designed to help block developers, serving as templates for custom-built data management blocks.

#### Read and Write blocks

The Read and Write blocks (Value library) and the Read(I) and Write(I) blocks (Item library) are designed for exchanging model data with a data structure. These data access blocks make it easy to get data into or send data out of models.

Both of the Read blocks lookup information found in a specified data source; they can report the information found on their value output connectors. The Read(I) block can also store the information as an attribute on items passing through.

Both of the Write blocks send information to a specified data destination; the information to be sent can come from their value input connectors. The Write(I) block can also send the attribute information that has been found on items passing through.



Top: Read(I) and Write(I) blocks  
Bottom: Read and Write blocks

#### Communicating with data structures

Depending on which library (Value or Item) the Read/Write blocks are from, they communicate with different types of data structures.

##### *Read and Write blocks (Value library)*

As determined by selections in popup menus in their dialogs, the Read and Write blocks in the Value library exchange data with:

- ExtendSim database

- Global array
- Excel workbook
- Text file
- Local table (information that is entered the block's data table)

*Read(I) and Write(I) blocks (Item library)*

The Read(I) and Write(I) blocks in the Item library exchange data with an ExtendSim database; data is updated only when an item arrives at the block.

**Addressing the data structure**

By definition Read and Write blocks are required to be interfaced to some type of data structure. The pieces of information that are required to fully and properly specify where the information is to be read from or written to (the “address”) depends on the type of data structure chosen, as shown in the following table.

	ExtendSim Database	Global Array	Microsoft Excel	Text File
File name			X	X
Database name	X			
Table, array, or sheet	X	X	X	
Field or column	X	X	X	X
Record or row	X	X	X	X

For instance, the fully specified address for accessing an Excel file would include the file name, sheet, column, and row.

**Interface methods**

Once you have specified a data structure type to communicate with and determined what the fully qualified address is, the next step is to determine where the data structure address is entered:

- In the block's dialog or through value input connector for the Read and Write blocks (Value library)
- In the block's dialog, through value input connector, or by using attributes on items that pass through the Read(I) and Write(I) blocks (Item library)

**Triggers**

The following table indicates the mechanism that will trigger when the data is sent or received.

	Read	Write	Read(I)	Write(I)
Connector message	X	X		
Beginning of run only	X			
End of run only		X		
Item arrival at block			X	X

**Data access blocks**

The following blocks are useful for creating global arrays and text files and for accessing data from internal or external sources.

How To

### Data Source Create



The Data Source Create block (Value library) provides an easy method to create or modify global arrays or text files. You can also use this block to resize or delete a global array. For an example of how this block is used, see “How to create and use a global array” on page 653.

### Data Import Export



The Data Import Export block (Value library) is used for importing data from a data source to an ExtendSim database or global array or for exporting data from an ExtendSim database or global array to a data source. The supported data sources include:

- Microsoft Excel
- ODBC technology (Windows only)
- FTP site (Windows only)
- Text files

### Data Init



The Data Init block (Value library) uses a data source to initialize a target with values. A popup menu in the block’s dialog allows you to choose as a target a database’s table, field, record, or cell or a global array’s column, row, or cell. The data source can be a value, a database’s table, field, record, or cell, or a global array’s column, row, or cell.

Each row of the block’s table defines one initialization record for one target. Each initialization record can be set up to initialize the data it refers to at every run, the first run only, or to not initialize. Each cell in a given row of the table contains interface items that allow customization of that particular initialization record.

### Other blocks for modelers

#### Data Specs



The Data Specs block (Value library) reports information about a selected ExtendSim database or global array. You can assign a name to the specification report, select its components through popup menus, initialize the outputs, and display the specification’s name and/or values on output connectors.

#### Command



When triggered by a value at its “send” input, the Command block (Value library) sends a command, such as an Excel macro or DDE command, to a spreadsheet. You can select a command to send and choose when the command is sent.

### Blocks for developers

The Utilities and ModL Tips libraries contain several blocks to assist block developers, such as the Object Mapper block in the Utilities library.

## Data source indexing and organization

Communicating between various types of data sources has been greatly assisted by standardized technologies, such as the ability of diverse applications to exchange data through a standard text file format. However, it is important to keep in mind that each standard has its own conventions. This can cause data-confusion when transferring data from one type of source to another.

### Transferring data between a data table and a spreadsheet

ExtendSim data tables have zero-based indexes and are organized by row and column –data starts at row 0, column 0. Spreadsheets are also organized by row and column, but they are one-based – data starts at row 1, column 1. When transferring data from an ExtendSim data table to an Excel worksheet, the row and column numbers for Excel must both be increased by 1 compared to their location in the ExtendSim data table.

### Transferring data between a spreadsheet and a database

Databases are one-based like spreadsheets, but are organized by fields and records (equivalent to columns and rows) rather than being organized by the spreadsheet convention of rows and columns.

#### Indexing and organization

The following table lists indexing and organization conventions for different data source types.

Data Source Type	Indexing	How organized
Data Tables	0-based	Row/Column
Spreadsheets	1-based	Row/Column
Internal Databases	1-based	Column/Row (Field/Record)
External Databases	1-based	Column/Row (Field/Record)
Arrays	0-based	Row/Column
Text Files	N/A	Row/Column

### Communicating with external devices

In some situations, you may want to obtain data for a model directly from an external piece of equipment. For instance, when modeling a chemical process you might want to read the temperature of the actual process and compare it to simulated results.

If you create your own blocks with ModL code, there are two methods you can use to communicate with external devices such as scientific equipment and other hardware:

- Dynamic Link Libraries (DLLs) on Windows or Shared Libraries for Mac OS. DLLs and Shared Libraries are segments of code written in a language other than ExtendSim's ModL, such as Visual Basic or C++. Their standardized interface provides a method for linking between other languages and ModL. They can also be used to perform complex calculations utilizing specialized hardware. For more information, see “DLLs and Shared Libraries” on page 668 and the Developer Reference.
- Serial port functions on Windows. To pass data through serial devices, use the serial port functions. These functions can read and write any data (including real-time data) to and from the computer's serial ports. This is useful for transmitting and receiving data on a modem, for example. For more information, see the Developer Reference.

## Technologies for communication

Many technologies have been developed as industry standards to allow applications to share and access data both internally and with each other independent of programming language, operating system, and file type. ExtendSim supports most of the standard types of communication technologies including:

Technology	ExtendSim Use or Implementation
<b>Text Files</b>	Import and export of raw data or ExtendSim databases, Report and Trace files.
<b>ActiveX/COM/OLE</b>	Embedding controls and objects. The Gantt Chart (Plotter library) has an embedded ActiveX control.
<b>DDE</b>	DDE Link menu commands.
<b>ODBC</b>	Import from, or export to, ODBC databases using a "Data Import Export" block (Value library).
<b>DLLs (Windows) Shared Libraries (Macintosh)</b>	LPSolve used with the Rate library
<b>FTP</b>	Import from, or export to, files located on a server or on the web using a "Data Import Export" block (Value library).
<b>Mail Slots</b>	Functions for inter-computer communication

In addition to the uses listed above, these technologies are supported for block development. This is described in the Developer Reference.

### Text files

A text file (also known as an *ASCII* file) is a file of unformatted information. Text files contain written text with the styles removed and/or numerical data separated by some delimiter or separator (such as tabs or spaces).

You can create text files in another application such as a spreadsheet, database, or word processing program, then read those text files into ExtendSim. Or create text files in ExtendSim, then read them into the other programs.

Text files contain text, data, or both data and text. There are many uses for text files. For example:

- To supply internal data as the basis for analysis when using Sensitivity Analysis.
- To share data with an external database or spreadsheet program such as Microsoft Access or Excel.
- To output model results to external applications, such as spreadsheets or word processing programs, for presentation or analysis.
- Text files of information are generated for model Reports or Traces.
- Most files transmitted from minicomputers and mainframes are text files.

Text files can reside locally, be remotely accessed over a network, or accessed via the internet using FTP.

### Creating and opening text files

Some ExtendSim blocks automatically create text files. For example, selected blocks automatically output information in the form of a text file when the Generate Report command is chosen for a run. If you program custom blocks, you can include ModL functions to have the blocks create or read text files. In addition, there are two methods to directly open or create a text file in ExtendSim:

- 1) Use the File menu commands to create or open text files. This allows you to look at report files, modify data input files, and so on, without having to open another application.
- 2) Use the Create or Open button in the dialogs of the Read and Write blocks (Value library) when Text File is selected as the data type. This is useful if you want to create a file of data for use in a model. The Read and Write blocks are discussed on page 659

### Working with text files

- The File > Import Data and File > Export Data commands are useful to exchange text file data from internal or external sources with dialog and plotter data tables or ExtendSim databases or global arrays. To see how these commands are used, see “Importing and exporting data” on page 626.
- The Read and Write blocks (Value library) provide an easy method for working with text files in a model. With these blocks text files can reside locally in the model or be remotely accessed over a network. These blocks also give added capability with text files in that they allow you to select which column and row of data to access. They are discussed in “Read and Write blocks” on page 659.
- When building custom blocks, use the file I/O functions described in the Developer Reference to read and write text files.

### How to create a text file

This example shows how to use menu commands to create a file with text and data:

- ▶ Give the command File > New Text File.
- ▶ In the Text File window, enter the information shown on the right into two tab-delimited columns. For example,
  - ▶ Type **January**
  - ▶ Click the tab key
  - ▶ Type **2.6**
  - ▶ Click the enter or return key to start a new row
  - ▶ Continue entering the information, clicking the tab key after each month to separate the information into two columns and clicking the enter key after each value to start a new row of information.
- ▶ Give the command File > Save Text File As and name the text file **My Text**.

January	2.6
February	4.4
March	6.7
April	3.4
May	1.9
June	1.1
July	0.7
August	0.5
September	0.4
October	0.7
November	2.6
December	3.4



☞ Because it is unformatted, the information in the text file will not be lined up as shown in the screen shot above. It is important that you do not try to line up the data by adding extra tabs as that will affect how the data is interpreted. There must be one, and only one, tab character separating the columns.

#### **Delimiting text file data**

Columns in text files can be separated by a tab, a space, or a comma. Carriage return characters are always used to separate rows. When creating text files for use as inputs to ExtendSim blocks or other applications, consider how the blocks or programs want the data presented. For example, some ExtendSim blocks can take data columns separated a tab character, a space, or with another character such as a comma. On the other hand, Sensitivity Analysis and many external applications will only read files where the columns are delimited by tab characters. In general, it is best to use tab characters.

Also, remember that programs expect only one delimiter (one tab, one space, or one comma) between columns. For example, don't put in extra tab characters in order to make columns line up visually; if you do, the program is sure to get confused.

#### **Changing text file font and size**

Use the commands in the Text menu to *temporarily* improve the readability of a text file, for example, by increasing the size of the text. However, Text menu changes are discarded when you close the text file.

To permanently change the font or font size used when viewing text files, give the command Edit > Options > Model tab and use the **Text file font** option.

#### **ActiveX/COM/OLE (Windows only)**

*ActiveX* is Microsoft's set of object-oriented programming technologies and tools. There are two main uses of the ActiveX technology in ExtendSim:

- ActiveX/OLE automation
- ActiveX/controls and embedded objects (COM)

#### **ActiveX/OLE automation**

ActiveX automation is the process of using ExtendSim's OLE functions, or the scripting environment of another application, to communicate with, exchange data with, or control another application. This is, for example, the technology used by the Read and Write blocks to communicate with Excel.

When you use the OLE functions inside ExtendSim to communicate with another application, ExtendSim is the Client in the automation communication and the other application is the server. When the other application is using its scripting environment to communicate with ExtendSim, ExtendSim is the Server.

When you develop custom blocks, automation is probably the most powerful tool that you can use for interapplication communication. It does, however, come with a cost. Developing code to use automation to control another application requires expertise with OLE/COM as well as knowledge of the object model of the target application.

An object model is essentially a list of the methods and objects that an application supports with regard to ActiveX automation. ExtendSim has a simple yet powerful object model that is described in some detail in the Developer Reference. Other applications have more or less complex object models. Excel, one of the more common target applications, has a quite complex object model; it

is quite complex to deal with. The Object Mapper block (Custom Blocks library) is a tool that can be useful in learning more about the object model of an application or ActiveX control.

Automation is also the most common tool used to communicate with embedded objects or ActiveX controls when they are added to the ExtendSim application. The Bar Chart and Gantt Chart plotters (Plotter library) use ActiveX automation through ExtendSim OLE function calls to communicate and exchange data with the embedded objects.

### **ActiveX controls and embedded objects**

The main ActiveX technology is *COM* (Component Object Model). COM is the framework for developing and supporting *ActiveX controls* and *component objects*.

While the terms “component objects” and “ActiveX controls” are sometimes used interchangeably, an object is often considered to have a source application that it derives from and a control is usually defined as a stand-alone object that doesn’t necessarily have an application behind it. For purposes of this manual, the words “object” or “embedded object” applies to either a component object or an ActiveX control.

A component object is an identifiable part of a larger program that provides a particular function or group of related functions; it is roughly equivalent to a Java applet. ActiveX controls can be created using one of several languages or development tools, or with scripting tools. In implementation, an ActiveX control is a dynamic link library (DLL) module. It runs in what is known as a container, an application program that uses COM program interfaces.

*OLE* (Object Linking and Embedding) is Microsoft’s framework for a compound document technology for combining text, sound, animations, controls and so forth into a document. Each object is an independent program component that can interact with the user and communicate with other objects.

Whereas OLE provides services for the compound document that users see on their display; COM provides the underlying services of interface negotiation, life cycle management (determining when an object can be removed from the system), licensing, and event services (putting an object into service as the result of an event that has happened to another object.)

ExtendSim supports embedding of COM/OLE objects and ActiveX controls in two places in the application:

- At the worksheet/model level, where they can be included as a container object and be manipulated much like blocks and other types of worksheet objects.
- At the block dialog level, where they can be inserted into a special type of dialog item called an embedded object.

The Gantt Chart (Plotter library) uses an embedded ActiveX control. The Read and Write blocks (Value library) uses OLE/COM to communicate with other applications. ExtendSim also has many OLE/COM functions to facilitate ActiveX controls and COM objects.

ExtendSim can be a container for objects embedded using the Insert Object menu command or through ModL programming. It also has ModL functions to support its use as either an OLE automation client or server.

### **DDE (Windows only)**

Dynamic Data Exchange (DDE) is a Windows operating system protocol through which applications can exchange data. DDE allows applications to form DDE links from one application (DDE server) to another (DDE client) and obtain data in real time. As data changes at the server, the

server sends the new data to the client for processing. Once the link is established, the applications exchange data without further user involvement.

Two applications linked by DDE are said to be engaged in a conversation. The server is the application that initiates the conversation; the client is the application that responds to the server. Unlike embedded objects, this conversation takes place between two windows.

DDE linking can be used for one-time data transfers and for continuous exchanges in which applications send updates to one another as new data becomes available. It is most appropriate for data exchanges that do not require ongoing user interaction. Some areas where it is useful are:

- Performing data queries, such as an ExtendSim model requesting information from an Excel worksheet.
- Linking to real-time data, such as scientific instruments, process control, or stock market updates
- Creating compound documents, such as an ExtendSim model that includes an Excel chart.

In ExtendSim, DDE technology is implemented through:

- Menu commands that establish a connection between two pieces of data. (See “DDE links (Windows only)” on page 636.)
- The Command block (Value library) for sending instructions to a spreadsheet, causing it to execute a macro. (See “Command” on page 661.)
- IPC functions that provide more flexibility than using just menu commands to create DDE links, such as the ability to set a value in Excel without establishing a link. (See the Developer Reference.)

### ODBC/SQL

Open DataBase Connectivity (ODBC) provides a standard application programming interface (API) method for accessing database data independent of programming language, operating system, and database system. The goal is to enable access to data from any application, regardless of which database management system (DBMS) is handling the data. It does this by inserting a middle layer between an application and the DBMS that translates data queries into commands understandable by the DBMS.

The ODBC API allows applications to access data in DBMS using SAG SQL (SAG = SQL Access Group; SQL - Structured Query Language) as the standard for requesting information from a database. SQL allows a single application to access different database management systems.

ODBC offers connectivity to a wide variety of data sources, including relational databases and non-relational data sources such as spreadsheets and text files.

 To communicate using ODBC, use the “Data Import Export” block (Value library) or the ExtendSim ODBC functions.

### FTP

File Transfer Protocol (FTP) is used to connect two computers over the Internet or an intranet for the purpose of transferring data and commands. In an FTP environment, one computer acts as a server and the other acts as a client.

As a client, ExtendSim initiates a connection to the server. Once connected, ExtendSim accesses a specified file on the server and either imports data from it or exports data to it.

ExtendSim implements FTP by:

- The Data Import Export block (Value library) can open a file on a remote computer and access the data for use in an ExtendSim model.
- Internet Access functions provide more flexibility than using the Data Import Export block. See the Developer Reference.

### **DLLs and Shared Libraries**

Dynamic-Link Libraries (DLLs on Windows) and Shared Libraries (Mac OS) are segments of code written in a language other than ExtendSim's ModL. This standardized interface provides a method for linking between ModL and other languages. When and if a DLL or Shared Library file is needed, it is loaded into memory and run by ExtendSim.

DLLs and Shared Libraries can contain code, data, and resources. They are used to provide access to functions that are already written in another language or to solve problems that might be difficult or impossible to solve in ModL. Use a DLL or Shared Library to calculate some function, perform a task, access application programming interface (API) calls, or even control external hardware devices.

ExtendSim implements DLLs and Shared Libraries by:

- The Rate library uses the LPSolve DLL or Shared Library.
- ExtendSim functions allow you to call the external code segments from within a block's ModL code and perform operations. For example, a block can pass data to the DLL or Shared Library, cause the DLL or Shared Library to calculate using that data, and get the results back from the DLL or Shared Library. For more information, see the Developer Reference.

### **Mailslots (Windows only)**

The mailslots feature allows communication between two ExtendSim applications running on different computers on the same local area network (LAN). Since mail slots involve programming, they are described in the Developer Reference.

# How To

## Miscellaneous

Other important features  
that didn't fit somewhere else

*“All truths are easy to understand once they are discovered;  
the point is to discover them.”  
— Galileo Galilei*

This chapter covers some additional ExtendSim features and topics, including:

- Using the Navigator to explore models and to access library or database windows
- Printing the model worksheet, Notebook, and block dialogs
- Copying model elements within ExtendSim and between ExtendSim and other applications
- Tool Tips
- Changing parameters dynamically
- Model sharing using the Lock Model command and the LT-RunTime version of ExtendSim

## Navigator

The Navigator is an explorer-like window that provides easy access to different aspects of ExtendSim. As discussed later in this section, it has three modes:

- *Model Navigator mode.* Lists the blocks in the active model, showing block icons, names, labels, global block numbers, and any hierarchical levels. In this mode the word “Model” is displayed in the Navigator window’s leftmost popup menu.
- *Database List mode.* Provides a list of the databases the active model uses, if any. In this mode the word “Database List” is displayed in the Navigator window’s leftmost popup menu.
- *Library Window mode.* Lists the blocks, including their icons and names, for a selected library. In this mode all open libraries are listed alphabetically below the line in the Navigator window’s leftmost popup menu.

## Opening the Navigator

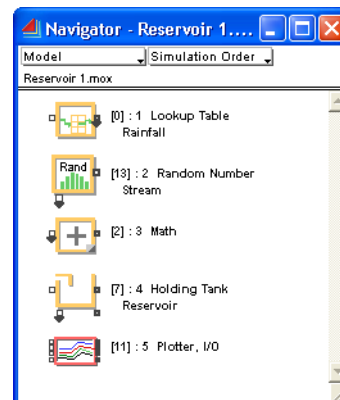
To open the Navigator:

- ▶ Select Window > Navigator or click the Open Navigator tool in the Toolbar.

By default, the Navigator opens in Model Navigator mode, with the word “Model” selected in the window’s leftmost popup menu. The name of the active model is listed at the top of the window and below the Navigator’s leftmost popup menu. Each block’s icon is shown, and its information (global block number, name, and label) is displayed to the right of the icon. If no model is open, the Navigator opens in default “Unknown doc” mode.

A Navigator’s window has two popup menus:

- The popup menu on the left is for switching between modes - Model Navigator (the default mode), Database List, or Library Window. For instance, the Navigator window for the Reservoir 1 model is shown above, set to Model Navigator mode.
- The popup menu on the right (Model Navigator mode only) is for changing the order for how the model’s blocks are listed in the window. You can sort by *Block Label*, *Block Name*, *Block Number*, hierarchical *Depth*, or *Simulation Order* (the default order).



Navigator for Reservoir 1 model

To switch from one mode to the other, select the mode in the popup menu on the top left of the Navigator window.

- ☞ You can have multiple Navigator windows open at the same time, all set to the same or different modes. To open a new Navigator window, Navigator windows that are already open cannot be in Model Navigator mode.

### Model Navigator mode

The Model Navigator mode is especially useful for large models with many blocks or when a model has several hierarchical layers. You use this mode to directly locate a block and open its dialog or to access submodels by drilling down through the hierarchical layers. For an example of using the Navigator in Model Navigator mode, see “Navigating through the Reservoir model” on page 38.

### Database List mode

The Navigator displays the list of databases for a model (if any) and is one of the methods for opening a database window, as well as copying, duplicating, and pasting databases between models.

- ☞ All the ExtendSim databases used in a model are automatically opened when the model opens. The Navigator is used to open a database’s window, not the database.

To open a database window using the Navigator:

- ▶ Open a model that uses a database, such as the Reservoir 1 model located at \Examples\Tutorials.
- ▶ Open a Navigator.
- ▶ In the Navigator’s leftmost popup menu, select **Database List**.  
Any ExtendSim databases used in the model are listed below the model name.
- ▶ To open a database’s window, double-click a database from the list.

Choosing Window > Database List is equivalent to choosing the Database List mode in the Navigator. For more information about creating and using ExtendSim databases, see “ExtendSim databases for internal data storage” on page 638.

### Library Window mode

If a library is selected in the Navigator’s leftmost popup menu, the Navigator displays an alphabetical list of blocks for that library. You can add blocks to a model from this list; this is often quicker than using the Library menu.

To open a library window and add blocks to a model:

- ▶ Open a model, if one is not already open.
- ▶ Open the model’s Navigator.
- ▶ In the Navigator’s leftmost popup menu, select a library from the open libraries listed below the line.

This action opens the library window for the selected library, showing block icons and names in alphabetical order.

- ▶ Select a block’s icon in the library window. (To select multiple blocks at once use the Shift key.)
- ▶ Drag and drop the block onto the model worksheet.

Opening a library window directly (by choosing Library > Open Library, then selecting the Open Library Window command at the top of the selected library's submenu) is equivalent to choosing the Library Window mode in the Navigator.

- ☞ If a block has been compiled with external source code, it will be listed in the library window with the designation *CM* (code management) on the right side of its icon. If a block has been compiled with debugging code, its name and any additional information will be listed in the library window in red and the block will display in the model with a red border around its icon. Compiling options are used by block developers and are discussed in the Developer Reference.

## Printing

### Selecting what to print

ExtendSim gives you control over what appears on the printed page. The following table lists what can be printed, depending on which window is active:

Active Window	Printed
Model worksheet	A picture of the worksheet as it appears on the screen. You can choose how many hierarchical layers to print and to optionally print the contents of the dialogs and plotters.
Dialog	A picture of the dialog. If the dialog has a data table, you can select whether or not to print all of the data in the table. You can choose to print just the top tab or all tabs.
Plotter	The plot and, optionally, the data table.
Navigator	The contents of the Navigator as it appears on the screen.
Notebook	The contents of the Notebook as it appears on the screen.
Hierarchical block worksheet	A picture of the worksheet as it appears on the screen. If you print more than one layer, it will also print the model worksheet.
Help text	The text from the current Help choice.
Text file	The text in the file.
Library window	A picture of the library window.
Structure window	All the panes from the structure window and the dialog editor window of a block (see the Developer Reference for more information).

### The Print command

ExtendSim decides which type of item to print based on what type of window is active when you choose the Print command. For instance, if a plotter is active, the command becomes File > Print Plotter.

When you select the Print command, if a Notebook or a Navigator window is active, the system's standard Print dialog appears. If another window (such as the model worksheet) is active, two dialogs open in sequence: an ExtendSim Print dialog and then the standard Print dialog.

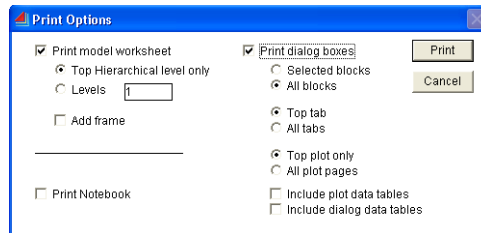
- ExtendSim's Print dialog is where you select what will be printed.



- After you have selected what to print, the system's standard Print dialog allows you to specify particulars about the print job. For example, you can set the number of copies to be printed and (for model worksheets and Notebooks) the range of pages.

**Worksheet, dialog, or plotter active**

When you give the Print command with the worksheet, dialog, or plotter active, an ExtendSim Print dialog opens.



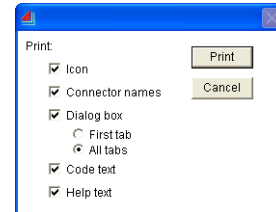
ExtendSim Print dialog

The available options are:

Option	Description
Print model worksheet	Prints the contents of the worksheet.
Top hierarchical level only	Only shows the worksheet's top level (not the contents of hierarchical blocks).
Levels	Specifies how many levels of hierarchical blocks to print (each level will print on a separate page).
Add frame	Puts a border around the printed worksheet.
Print Notebook	Prints the contents of the Notebook.
Print dialog boxes	Prints the contents of the dialogs.
Selected blocks	Specifies that only the dialogs of selected blocks will be printed.
All blocks	Specifies that the dialogs of all blocks in the model will be printed.
Top tab	Prints only the top tab of the dialogs.
All tabs	Prints all tabs in the dialogs.
Top plot only	Prints only the visible plot.
All plot pages	Prints all four of the plot pages.
Include plot data tables	Also prints the data tables in the plotters.
Include dialog data tables	Also prints the complete data tables in the dialogs.

### Structure or dialog window active

When a block's structure or dialog window is the active window, choosing the Print command gives you a special ExtendSim Print dialog before the standard Print dialog. This special dialog is shown on the right. The structure and dialog windows are only open if you are working on the structure of a block - its code, dialog, icon or help.



### Printing and Print Setup hints

If you print the model worksheet, a hierarchical block's structure window, or the Notebook, the standard Print dialog allows you to print all or only some of the pages. The File > Show Page Breaks command displays page breaks in an active worksheet or Notebook window. It also shows the page numbers in the upper left corner of the displayed pages to help you choose which pages to print. If the Show Page Breaks command is selected for a model, that setting is saved with the model but is not saved for the model's Notebook.

The Model > Show Block Numbers command causes block numbers to appear on their icons. Block numbers are unique identifiers for each block and appear in the title bar of the block's dialog. Choosing this command before you print the model worksheet and dialog boxes will help to match dialogs of blocks with their icons in the model.

ExtendSim will automatically print default headers and footers on each page. The Options dialog's Misc tab contains a preference to enable or disable the printing of headers and footers (see "Options" on page 688).

## Copy/Paste and Duplicate commands

The Clipboard is useful for passing information within ExtendSim or between ExtendSim and other applications. The information that is passed will be either in the form of ASCII text or graphics (Windows: a Windows metafile; Mac OS: PICT graphics). ExtendSim uses the Cut, Copy, Paste, and Duplicate commands in the Edit menu just like other programs. You can see the contents of the Clipboard with the Edit > Show Clipboard command.

Before you choose the Edit or Duplicate command, it is important to select the tool from the toolbar which will allow you to copy the desired items. For example, choosing the All Layers tool and frame-selecting a section of a model causes all items in the frame (blocks, drawing objects, clones, etc.) to be selected and available for copying. However, if you use the Block/Text Layer tool, only blocks and text will be selected.

### Copying within ExtendSim

#### Blocks

If you copy and paste or duplicate blocks within ExtendSim, the Clipboard also holds block parameters and connections. This allows you to duplicate portions of models, including the variables in the dialogs of the blocks, to another section of a model or to other ExtendSim models.

#### Drawing objects and text

Copy or duplicate drawing objects and stylized text from one section of a model to another section or from one model to another. To copy text, select the text as a block, rather than selecting the text within the text box.

### Data

Data can be copied from parameter entry fields and data tables and pasted into other entry fields or data tables.

- ☞ When copying data tables within ExtendSim, *Allow data table titles copying* should not be selected on the Misc tab of the Options dialog. If it is, the titles will be added to the data and will cause the original data to be displaced.

### Copying from ExtendSim to other applications

#### Data

Data from parameter fields and data tables can be copied from ExtendSim and pasted into other applications such as word processors and spreadsheets. When you copy data, ExtendSim puts the data into the Clipboard as unformatted text.

For more information, see “Copy/Paste” on page 625.

#### Pictures

Plot panes, Notebooks, parts of models, and dialogs are available as pictures to be copied from ExtendSim into other programs:

- Copy an ExtendSim plot pane to other programs for use in reports or presentations. To do this, simply click in the plot pane, then choose Edit > Copy Plot.
  - Notebooks and models contain various types of items such as drawing objects, cloned dialog items, and so forth. Before choosing the Copy command, it is important to select the tool from the toolbar that will allow you to copy the desired items. For instance, to copy an entire Notebook, choose the All Layers tool from the toolbar. Then frame-select the entire Notebook (or choose Edit > Select All) and choose Copy.
  - To copy all or parts of a block's dialog, choose the Clone Selection tool from the toolbar and select the dialog items you want to copy using any conventional method (frame-select, shift-select, etc.). Then choose Edit > Copy to Picture. To paste the contents of the Clipboard into Microsoft Word, for example, choose Edit > Paste Special.
- ☞ Copying an ExtendSim picture into the Clipboard changes it into a drawing object. To paste the contents of the Clipboard into the receiving document, it is common to use the Edit > Paste Special command.

When it is not possible to copy parts of a model directly using the Copy command (such as for the entire plotter window), do the following:

- Windows: Use the Ctrl + Prnt Scrn key combination to save the entire screen as a bitmap. Or use a capture program to capture specific parts of the screen.
- Mac OS: Use the Command + Shift + 3 key combination to save the selection as a picture. Or use a capture program to capture specific parts of the screen.

### Copying from other applications to ExtendSim

#### Data

You can copy data from other applications into an ExtendSim parameter field, data table, text file or database. However, if there is a lot of data it is usually better to import the data into a data table

or into an ExtendSim database. For more information, see “Exchanging data with external applications” on page 657.

### **Pictures and text**

Pictures and text can be copied into ExtendSim in color or black-and-white to use on model windows, Notebooks, hierarchical windows, or in the icon pane of a block’s structure window. Create a picture with a painting or drawing program and copy the picture to the Clipboard. Then paste the copied picture into an ExtendSim window with the Edit > Paste Picture command.

☞ It doesn’t matter what the original file format for the text or graphic was (JPEG, TIFF, GIF, etc.) Once it has been copied into the Clipboard, ExtendSim treats it as a drawing object.

In the model and Notebook windows, ExtendSim will paste the text or graphic object wherever you last clicked on the window. Pictures and text pasted from another program into ExtendSim become a drawing object which can be resized and repositioned using the Draw Layer tool.

Pictures are treated like other drawing elements. For example, use the Draw Layer tool to drag a picture around the window and resize it however you want. Or use the Model menu commands to rotate or flip the picture. To delete a picture, select it with the Draw Layer tool and choose Edit > Clear or press the Backspace or Delete key. Pictures that are copied, like objects created with ExtendSim’s drawing tools, always go behind ExtendSim blocks and text.

☞ Pictures can be proportionately resized. To do this, hold down the Shift key as you reshape the picture. It will resize with all dimensions proportional to the original.

### **Tool tips**

Tool tips can be turned on or off in the Edit > Options dialog. They help to quickly identify block information without having to open the block’s dialog.

- Resting the cursor above a block causes a small window to appear containing the block’s name, its local and global numbers, and (if *Include additional block information* is selected) the first sentence of the block’s online help.
- Placing the cursor over an input or output connector gives information about it, such as its name and the number of items that have exited or the block’s constraining rate.
- Tool tips also exist to identify the variable name for block dialog items. This can be helpful when programming custom blocks and is discussed further in the Developer Reference.

### **Changing parameters dynamically**

Another name for the variable or constant number you input in a model’s blocks is “parameter.” Parameters that do not change during the entire simulation run are *static*. It is more likely that you would want to have a model parameter change *dynamically*.

One of the most powerful features in ExtendSim is the ease with which you can dynamically change the parameters of a block during the course of the simulation. This is usually done to represent the value of the parameter as a function of something else in the simulation model. You may also want to change a parameter to allow someone else to “game” (modify a value manually) to see how a change impacts the simulation.

### **Methods**

There are a number of means for changing parameters dynamically:

- The most common method is to use the connector that corresponds to the dialog parameter field. Connect this to a block that calculates the desired value, such as a Random Number or

Lookup Table block (Value library). This approach visually shows the connection between the parameter value and where the value comes from. In general, this is also computationally efficient since only the blocks connected to the parameter are recalculated.

- A second approach is to link the parameter field with a record in an ExtendSim database. To do this, right-click the parameter and select *create/edit dynamic link*. (You can also create this link programmatically by creating a custom block that links the parameters of other blocks with an ExtendSim database or global array.) Whenever the linked value in the database changes, the parameter will change as well. This method hides the relationship between the parameter and where it is calculated, however it is very useful for centralizing the location of certain parameters in the database or for broadcasting the same value at the same time to a number of blocks throughout the model.
- In a discrete event model, some blocks (such as the Activity or Equation (I) block) allow you to specify a property value for a parameter. Using an item's property is an easy way to make the parameter value a function of the item in the block.
- Sensitivity analysis allows you to change a parameter from one simulation run to the next. Sensitized parameters change values at the start of each new run. They can change randomly, based on a value in a file, or by a fixed increment. Changing parameters is useful for sensitivity analysis and Monte Carlo simulation. To utilize sensitized parameters, the number of simulation runs must be greater than one. For a complete discussion, see "Sensitivity analysis" on page 568.
- Some blocks allow you to specify that a parameter is random. While this is not directly changing the parameter dynamically, this feature allows you to simulate a parameter that changes randomly each time the block is recalculated.
- There are other methods, such as calling the `SetDialogVariable` function in an equation block or custom block, but these are generally just used for specific situations.

## Sharing model files

You may want to provide access to models you have built, without giving users the ability to change the models. ExtendSim provides two methods for preventing changes being made to models:

- Locking a model to prevent anything other than parameter changes when the model is run in the full version of ExtendSim.
- Using the LT-RunTime version of ExtendSim to open and run models.


### Locking the model

Locking a model is especially useful when giving the model to others who may accidentally move or delete blocks or anytime you want to prevent changes to a model's layout. The Model > Lock Model command prevents any modification of a model other than changing dialog values. Since this command hides many of the tools in the toolbar, the user also cannot add or change connection lines, drawing elements, and so forth.

This command does allow the user to save the model and any dialog value changes. If they give the command File > Save Model As, the new model will also be locked.

When this command is selected, a dialog opens for entering an optional password to further protect the model. To lock a model without using a password, leave the password fields blank and click OK. (Note that, if a password is not used, any user can unlock the model.) To unlock a model, simply choose the Lock Model command again and enter the password, if any.

The Lock Model dialog also has a *Lock H-Blocks* checkbox. Select this to prevent a user from double-clicking a hierarchical block to see the underlying submodel.

 Once you have locked a model with a password, you must have that password to unlock the model. Always make unlocked copies of models before you lock them.

### The ExtendSim LT-RunTime version

If you develop models for others, the intended user may not have a need for the full version of ExtendSim. For instance, you might want to distribute a model as part of the response to a Request for Proposal or to showcase your consulting capabilities to potential clients. Or you may want to provide models to your company's sales staff so they can show customers the outcomes of various equipment or service options. The ExtendSim LT-RunTime version provides a low-cost, convenient method for distributing the models you build (and your libraries, if you program).


The LT-RunTime version allows users who do not have the full version of ExtendSim to:

- Run models of any size
- Enter or import parameter values into a model
- See and export results
- See animation in the model, if the model includes that feature
- Build small models for a limited time (up to 75 blocks for up to 180 days)
- Save changes (except to models that are locked)
- Print models, dialogs, and so forth

 To prevent end users from using the LT-RunTime version to make changes to your models, use the Lock Model command discussed on page 677.

The limitations of the LT-RunTime version are:

- The model-building capability is limited to models of 75 blocks and expires after 180 days. While a model larger than 75 blocks is running, or after the expiration period, the application changes to RunTime mode. At that point, end users can only run models. They will no longer be able to build a model, add or remove blocks from models, or alter connections between the blocks.
- Libraries converted to RunTime format cannot be used for model building. So that the end user can run models you give them, but not use your libraries to build models, convert your libraries to RunTime format using the full version of ExtendSim (see “Convert Library to RunTime Format” on page 697).
- The scripting functions that place or create blocks are not enabled.
- As with any other application, distribution and usage rights are limited.

 The ExtendSim LT-RunTime version is licensed to a single user for that user's personal use on a single computer. For information about any other use, such as distributing an LT-RunTime version on a CD or from your web site (“right to distribute”), or distributing ExtendSim's functionality or use to others on an intranet or through the internet (“ASP license”), contact Imagine That Inc.

# Reference

## Menu Commands and Toolbars

A reference section describing  
each menu item and tool

*“I wish it, I command it. Let my will  
take the place of a reason.”  
— Juvenal*


This chapter explains all the commands that appear in the menus and the circumstances in which you might use them. At the end of this chapter is a description of the tools in the application toolbar and ExtendSim database.

### ExtendSim menu (Mac OS only)

The ExtendSim menu is used to Quit ExtendSim or to Hide its windows. An additional choice, **About ExtendSim**, opens a window that lists the ExtendSim product, version and serial number and the registered user of the license. If you open the About ExtendSim window, click on the window to close it.

### File menu

The File menu lets you open, save, close, and print model and text files. Most of the commands in this menu act just like they do in other applications.

 Library files are opened from the Library menu, shown on page 695. Databases open when the model opens, as described at “Database menu” on page 700.

#### New Model

Opens an untitled model window.


#### New Text File

Opens an untitled text file window. You can use this to create text files for the Read and Write blocks, as input to sensitized blocks, or for any other ExtendSim feature that uses a text file as input. See “Text files” on page 663 for more information.

#### Open

Opens an existing model or text file. (Any libraries and databases that the model uses open when the model opens. To open a library file, use the Open Library command described on page 695.) You can also open a model by double-clicking the model file; this action will launch ExtendSim before opening the model.

You can have any number of models open at the same time; open models are listed at the bottom of the Window command. When ExtendSim opens the model, it also opens any libraries that are used by the model.

 If ExtendSim can't find a library used by a model, the message **Searching for library...** will appear as described in “Searching for libraries and blocks” on page 491. If ExtendSim cannot find all of the blocks used in the model, the missing ones will be replaced with text blocks. When this occurs, ExtendSim will open the model as **Model-x** to prevent accidentally saving over the old model.

#### Close

Closes the active window. For example, closes a model if the model worksheet is the active window or closes the model's Notebook, if the Notebook is the active window.

File	Edit	Text	Library	Model	Database
New Model				Ctrl+N	
New Text File				Ctrl+T	
Open...				Ctrl+O	
Close				Ctrl+W	
Revert Model					
Save Model				Ctrl+S	
Save Model As...					
Update Launch Control...					
Import Data...					
Export Data...					
Import DXF File					
Show Page Breaks					
Print Setup...					
Print				Ctrl+P	
Network License					
Properties... (Alt+Enter)				Ctrl+I	
1 3D Bank Line 1.mox					
2 Airline Security 2 Bob.mox					
3 Airline Security.mox					
4 3D Bank Line start.mox					
5 Model-2.mox					
Exit				Ctrl+Q	



### Revert Model/Revert Text File

Depending on whether a model or a text file is the active window, reverts the model or text file to the version saved on disk, discarding any changes since the last save. ExtendSim warns you before it completes this command.

### Save Model and Save Model As

Saves the model in the active window to disk. Choose Save to save the file under the current name or Save As to give a new model a name or to save a model under a new name.

- ☞ If a crash occurs during the save process, your original file could get corrupted. To protect against file corruption, prior to saving any new changes ExtendSim makes a copy of the previously saved version of the model and saves it as **ModelName.BAK**. To recover a model from a backup file, add a .MOX extension after the .BAK, then open the backup file from the File menu. You may choose to have ExtendSim automatically save backup files or delete them when a save is successful (see “Options” on page 688).

### Save Text File and Save Text File As

Saves the text file in the active window to disk. Choose Save to save the file under the current name or Save As to give a new text file a name or to save a text file under a new name.

### Update Launch Control (Windows only)

Select this command to cause the currently active Extend or ExtendSim application to open when a model (.mox) or library (.lix) file is double-clicked. This is only applicable if you have more than one instance of the Extend or ExtendSim application on your computer and you want to cause a specific instance of the application to open when you double-click files. For example, you would use this command if you have both ExtendSim 7 and Extend 6 installed, and you want model files to automatically launch ExtendSim 7 when they are double-clicked.

The application in which this command was last given will control the launching of the model and library files. To switch applications that will launch those files, just give the command in the Edit menu of whichever application you want to be in control of launching.

### Import Data Table

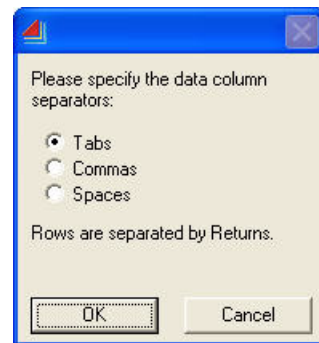
Copies data from a text file into the *selected* table. After choosing the file to be imported, the Column Separator dialog appears.

You must specify how the columns in the text file are delimited (separated); rows are automatically separated by returns. Most text files that are exported from other applications have columns delimited by tabs; check the format of the text file before choosing Import Data. For more information, refer to “Importing and exporting data” on page 626.

- ☞ A table from a database, dialog or plotter must be selected before this command can be used.

### Export Data Table

Copies data from a *selected* table to a text file. It works the opposite of the Import Data command. After you give the file name, ExtendSim puts up the column separator dialog (as shown in the previous section), so you can specify what type of separator to use in the text file. You can read the text file in ExtendSim or in a



Column Separator dialog

word processing or spreadsheet application. For more information, refer to “Importing and exporting data” on page 626.

 A table from a database, dialog or plotter must be selected before this command can be used.


### Import DXF File (Windows only)

Imports CAD drawings in standard DXF format from AutoCAD v13 or earlier or other CAD programs. The drawing becomes a graphic image which can be used as a background picture in the model, the notebook, or on an icon.

### Show Page Breaks

Causes ExtendSim to draw a set of page boundaries on the active window and place page numbers in the upper left hand corner of each page. These page boundaries show where page breaks will occur if you print the window. Since the size of a page is dependent upon the settings in the Print Setup (Windows) or Page Setup (Mac OS) command, it is recommended that you make your Print or Page Setup choices before showing page breaks.

When you choose Show Page Breaks, this menu item has a check mark next to it. To hide page breaks, select this command again.

 This command works independently for each active window (each model worksheet, Notebook, dialog window, and so forth). If the command is selected for a model, that setting is saved with the model but is not saved for the model's Notebook.

### Print Setup (Windows) and Page Setup (Mac OS)

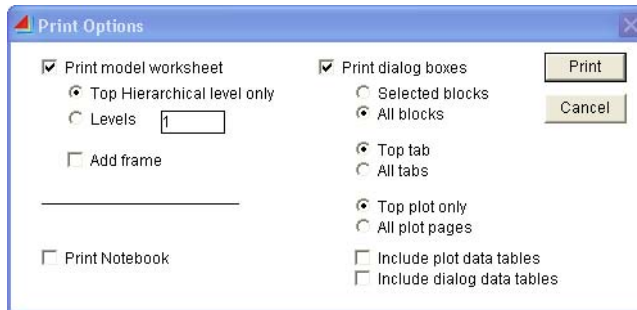
Sets the printing parameters (paper orientation, reduction or enlargement, etc.) for the printer you are using. Choose this command after changing printers or whenever you want to change the printing parameters. Printing parameters are saved with a model.

### Print

Prints various ExtendSim documents. If the worksheet, dialog, or plotter is the active window, ExtendSim first displays the Print Options dialog.

As discussed in “Printing” on page 672, you can choose to print the model itself, its Notebook, and/or the dialogs of the blocks in the model. For models with hierarchical blocks you can print just the top level or any number of hierarchical levels. Selecting Add frame causes ExtendSim to print a border around the worksheet.

With block dialogs, you can specify to print them for just the selected blocks or for all of the blocks in the model. You can also choose to print just the top tab or all the tabs in block dialogs. Since some dialogs have long data tables, you can specify whether or not to print the entire data table. If you choose Show Block Numbers or Show Simulation Order from the Model menu before you choose Print, the blocks will print with that information on them.



Print Options dialog

For plotters, you can print just the top plot page or all pages, and you can also print the (usually lengthy) data tables from the plotters.

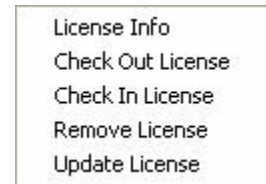
If you program, the structure and dialog windows have their own print dialog that lets you choose to print the ModL code, the dialog window, the connector names, the icon, or the help text.

The Print command also lets you print individual windows, such as the Notebook, the dialog, or the plotter window. When one of these windows is the active window, the Print command will print the contents of that window.

**Network License** (Windows only; network license only)

This command is only enabled if the ExtendSim application is a concurrent-user network license.

Network licenses require that the ExtendSim Network License Manager software be installed on a server as a Service. The ExtendSim application and files are then either installed on the server (such that client computers use ExtendSim in memory) or on the hard drives of individual client computers (such that the client computers use a local installation of ExtendSim).




Network License menu

**License Info** (Network license only)

Provides information about the network license, such as the maximum number of concurrent users allowed, the number of current users, utilization, and the name of the server that has the license management file.

**Check Out License** (Network license only)

Allows you to temporarily check out a license on a client computer, such as for a laptop that is to be used offsite. Checking out a license reduces the maximum number of concurrent users the server will allow. That number will be restored when the license is properly checked back in.

 To permanently remove a client computer from the license network, see the Remove License command, below.

**Check In License** (Network license only)

After reconnecting the client computer to the license network, use this command to return a license that had been temporarily checked out from the network. This also restores the number of concurrent users the server allows.

**Remove License** (Network license only)

Use this command to permanently remove a client computer from the license network. This action deletes the information that would enable the client installation of ExtendSim to connect to the server's license management application. You can then uninstall ExtendSim from that client. (If you don't uninstall ExtendSim from the client computer, the next time ExtendSim is launched it will not be able to run unless you supply the connection information.)

 To temporarily remove a client computer from the license network, see the Check Out License command above.

**Properties**

Shows information about the selected block or blocks, including Controls, hierarchical blocks, size, X-Y pixel location, creation and modified dates, and which library (if any) the selection came from.

**Five most recent models or text files**

The five model or text files with which you have most recently worked are listed near the bottom of the File menu. To open, select one of the files.

**Exit/Quit**

Leaves ExtendSim. If there are any model files with unsaved changes, you are first prompted whether you want to save them.

**Edit menu**

The Edit menu contains the standard Cut, Copy, and Paste commands as well as commands for directly linking with internal data sources and external applications, performing sensitivity analysis, and specifying global options.

**Undo**


Reverses the most recent action. You can undo commands, moving blocks, and so on.

**Cut**

Removes the selected item (such as a block, some text, or numeric data from a data table) and places it on the Clipboard. You can see the current contents of the Clipboard by choosing Edit > Show Clipboard.

**Copy**

Copies the selected item to the Clipboard. You can see the current contents of the Clipboard by choosing Edit > Show Clipboard. Copying is useful for duplicating parts of a model as well as for exporting to other applications. You can copy a single block, a piece of text, a group of blocks and text, graphical objects, or numeric values from a data table. You can also copy sections of the Notebook or dialog box as a picture.

 The items copied (blocks and text, drawing objects, etc.) depend on the selection tool used to make the selection.

Copying data and pictures is discussed in “Copy/Paste” on page 625.

**Paste**

Copies the contents of the Clipboard to the model. If the Clipboard contains text, a block, or a graphic item, the copied item is placed at the insertion point. For example, if you copy a block, click on the model worksheet, and then use the Paste command, the block will be pasted where you clicked the mouse. If there is no insertion point, the item is placed in the upper left corner of the window.

**Clear**

Removes the selected item. The menu changes to indicate what is selected, such as “Clear Data” or “Clear Blocks”.

Edit	Text	Library	Model	Database	E
Undo				Ctrl+Z	
Cut				Ctrl+X	
Copy				Ctrl+C	
Paste				Ctrl+V	
Clear					
Delete Selected Records					
Select All				Ctrl+A	
Duplicate				Ctrl+D	
Find...				Ctrl+F	
Find Again				Ctrl+G	
Replace				Ctrl+=	
Replace, Find Again				Ctrl+H	
Replace All					
Enter Selection				Ctrl+E	
Create/Edit Dynamic Link...					
Open Dynamic Linked Blocks				Ctrl+6	
Sensitize Parameter...					
Open Sensitized Blocks				Ctrl+5	
Paste DDE Link					
Delete DDE Link					
Show DDE Links					
Refresh DDE Links					
Insert Object...					
Design Mode					
Object					▶
Show Clipboard					
Options...				Ctrl+;	

### Delete Selected Records

Removes the selected records from the currently active ExtendSim database table.

### Select All

Selects all the items, such as all the blocks in a model or all the text in a field. The items selected (blocks and text, drawing items, and so on) depend on the selection tool chosen in the tool bar.

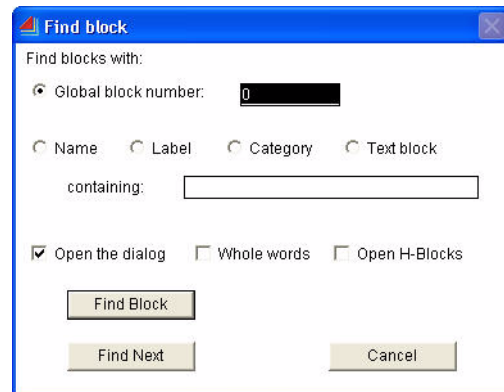
### Duplicate

Makes a copy of the selected item and puts it near the original item. This is faster and often more convenient than copying and pasting an item.

### Find

This command displays a different dialog depending on whether or not text (such as text within a text box or within the code pane of a block's structure window) is selected.

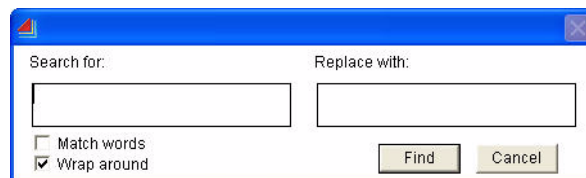
If text is not selected, the dialog to the right appears. This dialog finds a block by its global block number, name, label, or category, or locates a text block by its global block number or by the text contained within the text block. *Global block numbers* are unique, permanent identifiers for blocks and text blocks. *Name* means the name of the block in the library menu (e.g. Equation). Block *labels* are defined by the user in the block dialog and are especially useful to find types of blocks. *Category* refers to how the block is classified in the Library menu (e.g. Inputs).



Find block dialog

You can also choose to open the dialog of the block once it is located. This command is useful for large models when you are looking for specific blocks.

If text is selected (such as within a text box or within the code pane of a block's structure window), this command presents the following dialog.



Find text dialog

In this dialog, enter the text you want to search for in the **Search for:** edit box. The **Match Words** option tells ExtendSim to match whole words only. In that situation, for example, **boxer** would not be found by a search specifying **box** as the **Search for:** string. The **Wrap Around** choice tells the command to begin searching from the top of the ModL code if it did not find the text before the end.

The **Replace with:** edit box is used for finding and replacing at the same time (see Replace).

ExtendSim searches for text starting at the current selection and going to the end of the text. If the specified string is found, the string is selected. If it is not found, the insertion point is not moved.

- ☞ See the Find and Replace block (Utilities library) for additional block search capabilities, including the ability to find a dialog value and replace it with another value.

### Find Again

Repeats the most recent find operation using the same search string.

### Replace

Replaces the currently selected text with the text in the **Replace with:** edit box from the Find dialog. (Note that you can select text directly, in which case the text contained in the **Search for:** box in the Find dialog may not be the same as the text you have selected and are replacing.) If there is no text in the **Replace with:** edit box, the selected text is deleted.

### Replace, Find Again

Replaces the currently selected text with the contents of the **Replace with:** box in the Find dialog, then performs another Find command. Note that, if there is no text in the **Replace with:** edit box, the selected text is deleted.

### Replace All

Replaces every instance of the text in the **Search for:** box with the text in the **Replace with:** edit box in the Find dialog.

- ⚠ You cannot undo a Replace All command, so use it with caution.

### Enter Selection

Chooses the selected text to be the search string. This puts the text in the **Search for:** edit box of the Find dialog and also determines the search string to be used for the Find Again command. This is useful if you want to find the next instance of some text that is already in the code.

### Create/Edit Dynamic Link

Links a block's parameter field or data table to an *internal* data structure (ExtendSim database or global array). This action creates a two-way dynamic link between the data structure and the dialog item. Dialog parameters can be linked to a cell in a global array or database table; dialog and plotter data tables can be linked to a global array or database table. Although you can link data tables to a database table, you cannot link an individual cell in a data table to a cell in a database.

- ☞ The Create/Edit Dynamic Link command is only used for linking parameters and tables to *internal* data structures. Use the DDE link commands, discussed at "Paste DDE Link (Windows only)" on page 687, to link parameters and cells to *external* applications

When you give the Create/Edit Dynamic Link command, a dialog appears with a popup menu to select the data structure. Depending on the structure selected, other options are presented. To delete a dynamic link, click the linked parameter or select the linked data table and give the Create/Edit Dynamic Link command. In the dialog, choose Delete Link.

The command is equivalent to right-clicking the dialog item and choosing Create/Edit Dynamic Link, or (for a data table) clicking the Link button. Dynamically linked parameters are outlined in light blue. Dynamically linked data tables display the words DB (database) or GA (global array) in their upper left corner. For more information, see "Dynamic linking to internal data structures" on page 629.

- ☞ Another type of internal data source is a *dynamic array*, which is implemented through a block's code. You cannot use the Create/Edit Dynamic Link command to link to a dynamic array. However, if a data table is already linked to a dynamic array, its upper left corner will be blue.

### Open Dynamic Linked Blocks

Depending on options selected in the dialog, opens the dialogs of all blocks that have a parameter field or data table dynamically linked to an internal data structure (ExtendSim database, global array, or dynamic array). For more information, see “Finding linked dialog items” on page 635.

### Sensitize Parameter

If a dialog parameter is selected, this command opens the Sensitivity Setup dialog, which lets you set values for sensitivity analysis. An alternate method of opening the Sensitivity dialog is to click on the dialog parameter once while holding down the Control (Windows) or Command (Mac OS) key.

A parameter that has sensitivity settings has a frame inside of it. If sensitivity analysis is active for a parameter (that is, if the **Enable sensitivity** choice is checked in the Sensitivity Setup dialog), the frame is green. If the sensitivity analysis is inactive for the parameter or if it is turned off for the model as a whole, the frame is red.

Sensitivity analysis is discussed in “Sensitivity analysis” on page 568.

### Open Sensitized Blocks

Opens the dialogs of all blocks that have sensitized parameters. This is useful for finding which blocks are used in the scenario. The command opens dialogs with sensitized parameters even if sensitivity has been disabled for a parameter or is not active for the entire model. For more information, see “Sensitivity analysis” on page 568.

### Paste DDE Link (Windows only)

Copies the contents of the Clipboard to the selected parameter fields or data table cells. This action creates a DDE link so that data from an *external* application is live-linked to ExtendSim.

This command will only be active if all of the following is true:

- The external application supports linking through DDE
- The external file has been named and saved, but is open
- There is data in the Clipboard that has been copied from the external application
- You have selected a dialog parameter or group of data table cells in ExtendSim to paste to

Parameters or data table cells that have been linked with this command are outlined in yellow. For more information, see “DDE links (Windows only)” on page 636.

- ☞ The DDE Link commands only relate to ExtendSim's interaction with *external* applications. See the command “Create/Edit Dynamic Link” on page 686 for linking parameters and tables to ExtendSim's *internal* data structures.

### Delete DDE Link (Windows only)

Unlinks the selected parameter field or cell from the external application. This action does not change the value in the field or cell, but the field or cell will no longer be linked to the other application.

**Show DDE Links** (Windows only)

Opens any block dialogs that have DDE links to external applications.

**Refresh DDE Links** (Windows only)

If linked applications are open and links appear to be working incorrectly, this command will attempt to reestablish existing links between ExtendSim and the external application.

**Insert Object** (Windows only)

Brings up a list of registered embeddable objects - OLE component objects or ActiveX controls from an external application that can be inserted into an ExtendSim worksheet or block's dialog. For information about the options in this dialog or about embedded objects in general, see "Embedding an object (Windows only)" on page 655. For information about OLE or ActiveX, see "ActiveX/COM/OLE (Windows only)" on page 665.

**Design Mode** (Windows only)

The way embedded objects behave in and out of Design Mode is dependent on the type of object. In general, this command causes the external source application to open when an embedded object is double-clicked. For an embedded object that is activated by a single click, Design Mode changes the single-click behavior from activation to selection, allowing the embedded object to be selected and moved without activating it.

For example, clicking a Graphics Server control object when you are not in Design Mode will have the effect of "in-place-activating" it. In Design Mode, double-clicking the object opens a Property Pages Graphics Server dialog for customizing its settings.

**Object** (Windows only)

Some embedded objects from external applications have options that can be supported in ExtendSim. Selecting an object enables this menu command. The contents of it submenu depend upon the type of embedded object; many objects do not have options.

**Show Clipboard**

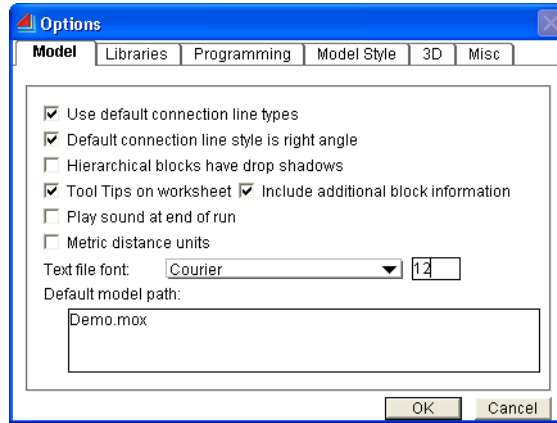
Shows the contents of the Clipboard in its own window.

**Options**

Lets you specify how you want ExtendSim to behave. Note that this is different from the Simulation Setup command which only affects the running of a specific model. The Options command controls actions for all models and has six tabs: Model, Libraries, Programming, Model Style, 3D, and Misc.



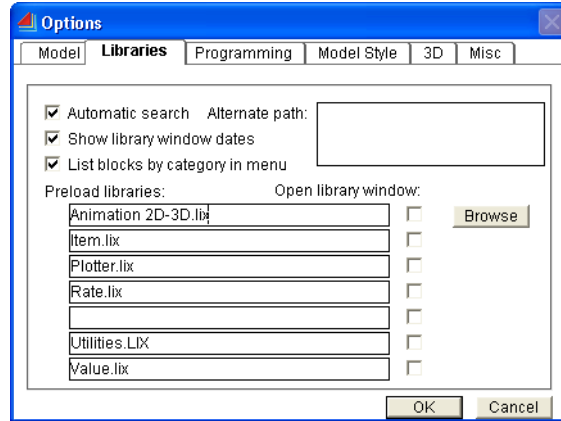
**Model tab**



Model tab of Options dialog

Option	Description
Use default connection line types	Specifies whether or not ExtendSim uses the default connection line types when creating new connections (see “Connection lines” on page 557).
Default connection line style is right angle	Sets the default for the Connection Lines command from the Model menu to be right-angle connections instead of straight line connections.
Hierarchical blocks have drop shadows	Determines whether there is a shadow around hierarchical blocks.
Tool Tips on worksheet	Causes the block name, number, and library to be displayed when the cursor is hovered over a block on the model worksheet. If you hover the cursor over a connector, the connector name and value are displayed. Note that Help captions for the tool bar stay on even if this choice isn’t selected.
Include additional block information	In addition to the block name, number and library, when a cursor is hovered over a block this command will cause additional information, such as a description of the block, to be displayed.
Play sound at end of run	Causes ExtendSim to play the default system sound at the end of every simulation run.
Metric distance units	Specifies that default distance or length units in the Convey Item and Transport blocks (Item library) is meters instead of feet. In the Convey Flow block (Rate library), distance and length units are specified directly in the block’s dialog and this setting is ignored.
Text file font	Lets you specify the font and size of the characters for viewing and printing text files.
Default model path	If a model pathname is entered here, that model will open when ExtendSim is launched (when ExtendSim is installed, Demo.mox is the default model.) If no path is specified, ExtendSim will look for the model in the application directory or folder.

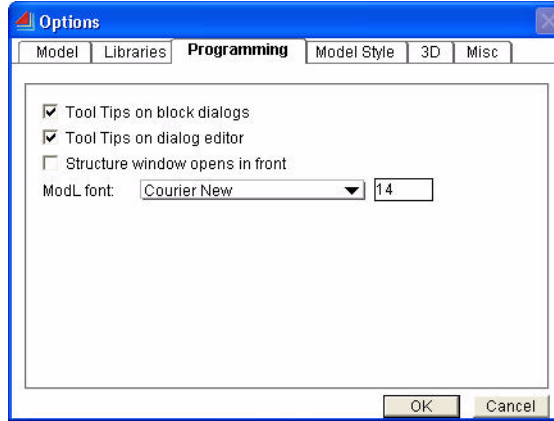
**Libraries tab**



Libraries tab of Options dialog

Option	Description
Automatic search	Causes ExtendSim to automatically find and open the libraries used in a model. If this is not selected, ExtendSim prompts you for the location of each library that a model uses. See also “Substituting one library for another” on page 495.
Alternate path	By default, ExtendSim searches for libraries in the Libraries folder. This choice specifies an alternate path for library searches. See also “Library searches” on page 491.
Show library window dates	Displays the modified and compiled dates for each block in the library windows.
List blocks by category in menu	Causes blocks in each library to be listed in hierarchical menus by category. Deselect this choice if you want all of the blocks listed alphabetically.
Preload libraries	Enter names of libraries you want automatically opened when ExtendSim starts. Type in the name or use the Browse button to locate a library and cause its name to be entered in the selected field. Note that ExtendSim will still ask for the location of any libraries located outside of the active application’s Libraries folder.
Open library window	Opens the window of the library listed directly to the left of the checkbox when ExtendSim is started.

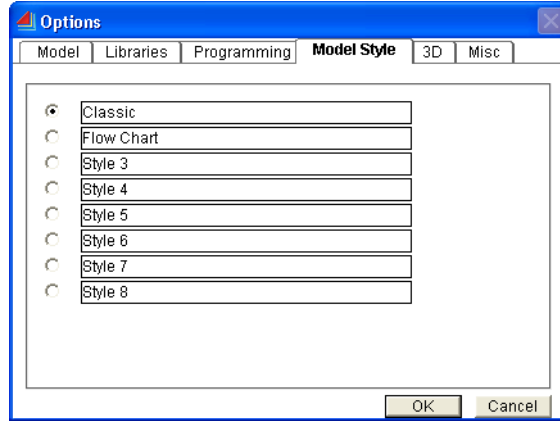
**Programming tab**



Programming tab of Options dialog (Windows)

Option	Description
Tool Tips on block dialogs	Displays the variable/message dialog names when the cursor is hovered over dialog items in a block's dialog.
Tool Tips on dialog editor	Displays the variable/message dialog names when the cursor is hovered over dialog items in a block's dialog editor window.
Structure window opens in front	Opens the structure window in front of its dialog editor window. Deselect this choice to cause the dialog window to open in front.
ModL font	Specifies the font and size of the characters for viewing and printing the text in the code pane of the structure window.

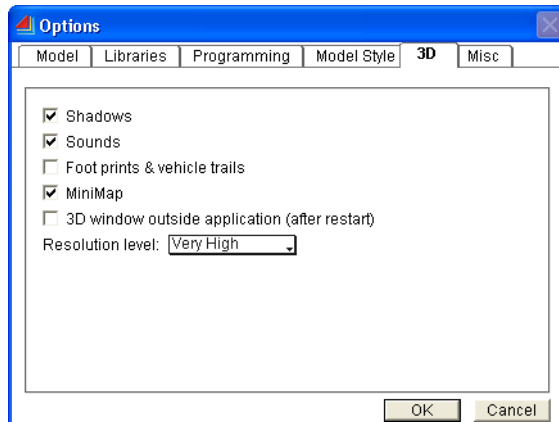
**Model Style tab**



Model Style tab of Options dialog (Windows)

If a block developer provides for this feature, blocks can have alternate styles that affect their appearance in a model. Model styles are saved as indexes so the developer can rename the style and it will still work on a different computer. The radio button selects the style that will be used as a default when a new model is created. This does not affect models that are open or already built.

**3D tab**



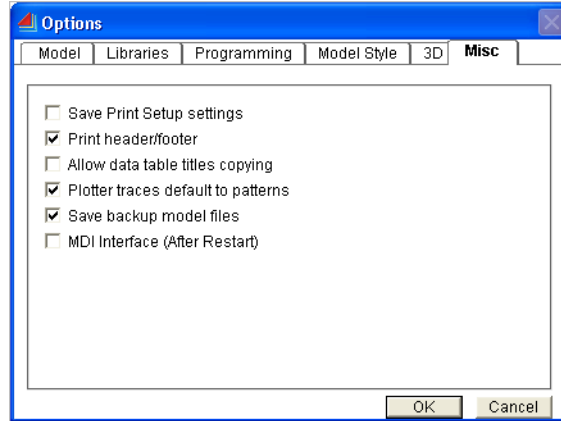
3D tab of Options dialog

Option	Description
Shadows	Displays shadows on objects in the E3D window. Can cause animation to slow down, since calculating shadows for many objects can be computationally intensive.
Sounds	Turns on the sounds for 3D objects, such ambient sounds, footfalls, vehicle tire squeals, and sounds produced by calling the E3DPlaySound function.

Option	Description
Foot prints & vehicle trails	Displays foot prints following the movement of people objects and vehicle trails that follow vehicle movement.
MiniMap	Displays a reduced map of the entire E3D window, including location information. Especially useful when building a model.
3D window outside application (after restart)	<b>Windows only.</b> Specifies whether or not the E3D window is a child window of the ExtendSim application. If checked, the E3D window will be a separate window that can be behind or in front of the ExtendSim application window. Otherwise, the E3D window is contained within the application window, like the model worksheet. Note: This option only becomes active after ExtendSim is restarted.
Level of detail	Controls the level of detail with which 3D objects are displayed in the window. You would only need to change from the default setting of Very High if there are many objects on the screen and the display is slow.

The 3D options only apply to ExtendSim products that include 3D animation. For more information about the 3D animation, see the E3D module that starts on page 390.

*Miscellaneous tab*



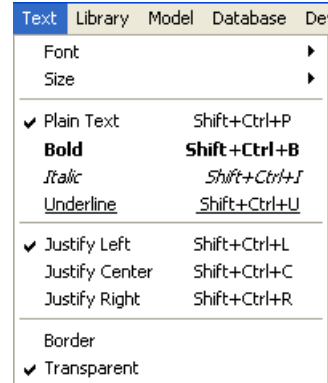
Misc tab of Options dialog

Option	Description
Save Print Setup settings	<b>Windows only.</b> Causes models to retain the settings specified in the Print Setup dialog. Caution, this may not transfer successfully from one computer to another because of differences in printer drivers.
Print header/footer	Prints the header and footer information for all files.
Allow data table titles copying	Specifies whether or not the Copy command copies row and column titles when copying from a data table. Select this option if you want to copy the titles, such as when you are copying to another program. Do not select this if you are copying into another table within ExtendSim.
Plotter traces default to patterns	Draws plotter traces with a pattern, allowing traces to be visible on a black and white monitor.
Save backup model files	When a Save command is given, renames the existing model to “Model-Name.BAK”, then saves the model. This is a precautionary measure, so the original model file won’t be overwritten when it is saved. See also “Save Model and Save Model As” on page 681.
MDI Interface (after restart)	The MDI (Multiple Document Interface) is off by default.

## Text menu

The Text menu is used to set the style of text in the model and to temporarily set the style of text in text files. The choices are the same as in most applications. The **Border** command draws a shadowed border around the text box. The **Transparent** command causes the background of the text to be transparent. If the Transparent command is not selected, the background of the text is white. For more information about using text, see “Working with text” on page 538.

- ☞ If you make changes to the settings in the Text menu before you enter text, those changes will apply to all text entered after the changes. To change only an existing piece of text, select the text within its text box, then change the Text menu settings. In that case, Text menu changes will apply only to that existing piece of text, and not to text you subsequently enter.

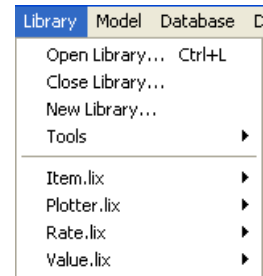


## Library menu

ExtendSim opens libraries automatically when you open models. To open or close a library manually, use the Library menu. For more information about libraries and their usage, see “Using libraries” on page 490.

### Open Library

Opens a library file. This causes the library’s name to appear in the Library menu in alphabetical order. To view the blocks in the library, click the Library name and drag down to the name of the library. When the library name is selected, blocks will be listed to the right of the menu either alphabetically or by category, depending on the option selected in Edit > Options > Libraries tab. The first choice in each list is **Open Library Window** to open a window listing the blocks in the library, as discussed at “Library windows” on page 493.



### Close Library

Closes an open library. This command displays a dialog of the open libraries; select the library or libraries to be closed from the list and click Close. Libraries that are in use cannot be closed. To close multiple libraries, Shift select the libraries (or use the Ctrl (Windows) or Command key (Mac OS)), then click Close.

### New Library

Creates a new library. When you select this command, you will be presented with a dialog to specify the name of the new library.

### Tools

Allows you to protect the code of blocks, set library versions, convert libraries to RunTime format, and edit the Startup Screen for the LT-RunTime version.

Other than the two RunTime commands, the following commands are of interest only to block developers.

Open All Library Windows
Compile Open Library Windows
Compile Selected Blocks
Add Debug Code to Open Library Windows
Remove Debug Code in Open Library Windows
Add External Code in Open Library Windows
Remove External Code in Open Library Windows
Protect Library...
Set Library Version...
Convert to RunTime Library...
RunTime Startup Screen Editor...

Library Tools choices

#### ***Open All Library Windows***

Opens all the library windows for libraries currently open in ExtendSim.

#### ***Compile Open Library Windows***

Causes all libraries whose library windows are open to be recompiled. To activate the command, open the library window for the desired library.

#### ***Compile Selected Blocks***

Causes the selected blocks in the library to be recompiled. To activate the command, open the library window for the desired library and select the desired blocks.

#### ***Add Debug Code to Open Library Windows***

Causes all libraries whose library windows are open to be recompiled with Debugging code. To activate the command, open the library window for the desired library and select the desired blocks.

#### ***Remove Debug Code in Open Library Windows***

Causes all libraries whose library windows are open to be recompiled without Debugging code. To activate the command, open the library window for the desired library.

#### ***Add External Code in Open Library Windows***

For each library whose library window is open, moves the source code for each block in the library into a separate text file. This is useful for source code control. For more information, see the Developer Reference.

#### ***Remove External Code in Open Library Windows***

Moves the external source code back into each library's blocks, for all libraries whose library windows are open. For more information, see the Developer Reference.

#### ***Protect Library***

Removes the ModL source code from all the blocks in a library so users cannot access block code. This is discussed in the ExtendSim Developer Reference.



 You would only use this command to protect libraries of blocks you build yourself. Do not use this command with the libraries that are included with ExtendSim.

#### **Set Library Version**

Allows you to set the long and short version strings for the library. This is useful if you are programming your own libraries and are concerned about version control.

#### **Convert Library to RunTime Format**

Changes a copy of a selected library to RunTime format. This removes the ModL code, as discussed in the Protect Library command, above. It also limits the use of the library:

- A RunTime formatted library can be read by an ExtendSim LT-RunTime version to run models, but that library cannot be used to build models. (The ExtendSim LT-RunTime version is discussed on page 678.)
- The full version of ExtendSim cannot read libraries that are in RunTime format. Thus RunTime formatted libraries cannot be used to run or build models in the full version.

Converting to RunTime format provides an easy method for providing libraries to others for evaluation or model running, while preventing those libraries from being used to build models.

#### **RunTime Startup Screen Editor**

Allows you to customize the startup screen of an LT-RunTime version of ExtendSim so that users know who to contact if they have questions. To use this command, you must have an ExtendSim LT-RunTime application installed on your hard drive. Note that you cannot personalize the startup screen of the Demo-Player version.

#### **MacWin Conversion** (Mac OS only)

The libraries and extensions that come with ExtendSim are already formatted correctly for your operating system. However, if you build your own libraries or create your own extensions, and want to transfer them to a computer running a different operating system, you must convert the files to the appropriate operating system format. Use this command on a Macintosh computer to convert libraries and extensions to the specified operating system format, either Windows or Mac OS.

#### **List of libraries**

As libraries are opened, they will be listed at the bottom of the Library menu in alphabetical order.

## Model menu

The commands in this window affect the parts of the model window and the way that the window is viewed.

### Make Selection Hierarchical

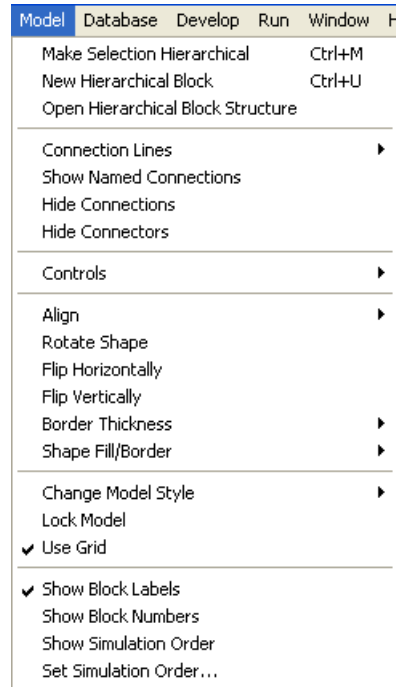
Encapsulates selected blocks into a single hierarchical block and replaces those blocks on the model worksheet with the hierarchical block. This is described in “Hierarchy” on page 540.

### New Hierarchical Block

Starts a new hierarchical block. This prompts you for the name of the new hierarchical block, then opens a blank hierarchical block structure window for building the model. This is described in “Building a new hierarchical block” on page 543.

### Open Hierarchical Block Structure

Opens the structure of a hierarchical block so you can edit its icon or Help. Note that the hierarchical block must be selected on the model worksheet. This command is equivalent to holding down the Alt (Windows) or Option (Mac OS) key while double clicking a hierarchical block on the model worksheet. See “Modifying hierarchical blocks” on page 548.



### Connection Lines

Sets the format of the connection lines. These are described in detail in “Connection lines” on page 557.

### Show Named Connections

Shows the connections between named connections. This is useful to show data flow in complex models with many named connections. Named connections are discussed at “Named connections” on page 560.

### Hide Connections

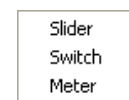
Hides the connecting lines between blocks. This is a cosmetic change that is mostly used to enhance presentations. Select the command again to show the connecting lines.

### Hide Connectors

Hides the connectors visible on blocks. This is a cosmetic change that is mostly used to enhance presentations. Select the command again to show the connectors.

### Controls

Allows you to add a control to the model. You can add a Slider, a Switch, or a Meter. These are described in “Controls” on page 509.



### Align

If two or more graphic objects, pieces of text, clones or blocks are selected, allows you to align the objects on the left, right, top, or bottom.

Controls

### Rotate Shape

Rotates the selected shape or picture by 90 degrees clockwise. Especially useful when creating rotated views for icons.

### Flip Horizontally/Flip Vertically

These commands flip (reverse) the shape or picture either horizontally or vertically.

### Border Thickness

Puts a border of specified thickness, or of no thickness, around shapes. The default is a black border; change border color using the Shape Fill/Border command, below.

### Shape Fill/Border

Determines whether the color selected from the Color Palette is used to fill the shape or to color its border. See also Border Thickness, above


### Change Model Style

If a library developer has implemented this feature and blocks have more than one style, this command changes the visual style of blocks in the entire model. The command can also be accessed by right-clicking the model. To set a default model style for new models, go to Edit > Options and select the Model Style tab.

 If the developer has not implemented this feature in the libraries, changing model styles will have no effect on the model. Most ExtendSim v7 libraries do not use this feature.

### Lock Model

Prevents any modification to a model other than changing dialog values. Can also be used to lock a hierarchical block. Locking models and hierarchical blocks is useful if you are giving the model to others who are unfamiliar with ExtendSim's features and may accidentally move or delete blocks. For more information, see "Locking the model" on page 677.

 Once you have locked a model with a password, you must have that password to unlock the model. Always make unlocked copies of models before you lock them.

### Use Grid

Creates an invisible snap grid on the model worksheet, Notebook, or icon pane to help you draw, move, and resize objects and blocks. The grid spacing is 4 pixels. While the grid is enabled, you can snap the upper left hand corner of an item to the grid. To override the grid, hold down the Alt (Windows) or Option (Mac OS) key as you move the object.

### Show Block Labels

Shows the block labels below the blocks in the model. For more information, see "Working with block dialogs" on page 29.

### Show Block Numbers

Puts the block numbers in square brackets on the blocks in the model. Block numbers are unique, permanent identifiers. Each block and text block in ExtendSim has a global block number. The blocks inside a hierarchical block show two numbers - the first is the block's global block number and the second is the block's local number within the hierarchical block.

### Show Simulation Order

Puts the number of the block's order of execution on the blocks in the model. Hierarchical block internals have their own simulation order relative to the parent block. Simulation order is discussed at "Simulation order" on page 86.


Because blocks can override the system's simulation order, this display may be inaccurate for discrete event (Item library) and discrete rate (Rate library) blocks that send block-to-block messages.

### Set Simulation Order...

Sets the number of the block's order of execution in the model. See "Simulation order" on page 86.


## Database menu

You can create relational databases in ExtendSim to store data for use in a model and to store model outputs. Databases are especially useful for managing data in complex models.

 Databases are stored with the model. Databases automatically open when the model opens and are automatically saved or closed when the model is saved or closed.

To bring a database window to the front, do one of the following:

- Click its name at the bottom of the Database menu.
- Select the Navigator tool and choose its Database List mode. Then double-click the database you want to access.
- Go to Window > Database List and double-click a database in the list.

 The tools for the database window are shown on page 714. For more information about creating and using ExtendSim databases, see "ExtendSim databases for internal data storage" on page 638.

### New Database

Opens a dialog for naming and creating a new ExtendSim Database for the model. The dialog also shows a list of the model's current databases so you can avoid using a duplicate name (ExtendSim will warn you if you try to use a duplicate name.) After naming the database, select OK to go to the database window.

### Import New Database

Creates a new ExtendSim Database by importing an entire database from an exported ExtendSim or SDI Industry database. In the dialog, select the database text file to import. This command imports all the tables, fields, records and so forth from the exported file and creates a new database. If you choose the name of an existing database, it will be replaced. To import tables to an existing database, such that the database tables are appended at the end of the database, see the Import Tables command, below.



### Export Database

Exports the entire ExtendSim Database into a text file. You do this so you can import the database into another model, to send the database to another user, or to prepare a database text file for use by the ExtendSim Excel Add-In. To export only specific tables from a database, see the Export Selected Tables command, below. Exported databases can only be imported to ExtendSim or to the ExtendSim Add-In for Excel. To enable this command, bring a database window to the front, as described under Database menu, above.

### Rename Database

Renames an existing ExtendSim database. The database renaming dialog shows a list of the model's current databases, so you can avoid duplicating a name (ExtendSim will warn you if you try to duplicate a name.) To enable this command, open an existing database by clicking its name at the bottom of the Database menu when the model window is active or select a database in the list when you go to Window > Database List.

### New Table

Creates a new table for an ExtendSim Database and adds it to the list of tables in the database's All Tables tab. The dialog that appears displays a list of the database's existing tables. After naming the table, click OK. To enable the New Table command, bring a database window to the front, as described under Database menu, above.

- ☞ To delete a table from a database, select the table and use the Delete or Backspace key or go to Edit > Clear Table. A dialog displays the delete options and, if there is data dynamically linked to the table, warns you before deleting it.

### Import Tables

Imports tables from an exported ExtendSim or SDI Industry database and appends the tables to an existing database. To create an entirely new database using imported files, see the Import New Database command, above. Give the Import Tables command after opening an existing database. In the dialog, select a file to import. This command imports all the tables from the exported file and places them below the existing tables. If you end up with unneeded tables, you can delete them; see also the Export Selected Tables command, below. To enable this command, bring a database window to the front, as described under Database menu, above.

- ☞ To import data into a specific table from a tab delimited text file, such as from Excel, go to File > Import Data.

### Export Selected Tables

Creates and exports a ExtendSim Database text file containing only the selected tables. Use this command instead of the Export Database command, when you want a file that contains only a portion of the database. To enable the command, select one or more tables in a database window.

### Rename Table

Renames the selected table in an ExtendSim Database. A dialog opens with a list of the database's existing tables and a field for entering the new name. To enable this command, bring a database window to the front, as described under Database menu, above. Then select a table in the database window.

### **New Tab**

Places a tab in the database window, to the right of any previous tabs. This is equivalent to double-clicking the blank area to the right of existing tabs. To enable this command, bring a database window to the front, as described under Database menu, above.

### **Rename or Delete Tab**

Allows you to rename or delete the tab that is in the active database window. This is equivalent to double-clicking the tab to rename or delete it. To enable this command, bring a database window to the front, as described under Database menu, above. Then select a tab to bring it to the front.

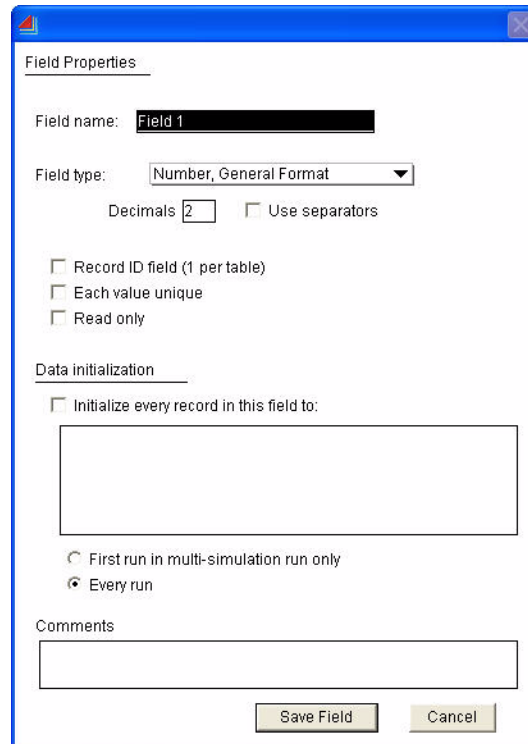
### **Clone Selected Tables to Tab**

Clones the table or tables to another tab. Select one or more tables, then give the command. In the dialog, select the tab to copy the table to. (Or right-click a selected table and choose “clone table to tab”.) To enable this command, bring a database window to the front, as described under Database menu, above. Then bring a tab to the front and select the tables to clone using the Block/Text layer tool.

- ☞ A cloned table will behave exactly like the original table and will change when the original table changes. Likewise, changes made to the cloned table will be reflected in the original table. Cloning is different than copying and pasting a table using the Edit menu. Tables that are copied and pasted are neither linked to each other nor to the original table.

### Append New Field

Creates a new field for the selected table and puts the field below any other fields in the list.



Field Properties dialog

The dialog gives options for setting the field's properties, such as its name, type, and initialization. To enable this dialog, bring a database window to the front, as described under Database menu, above. Then select a table and give the command.

- To delete a field, use the Delete or Backspace key or go to Edit > Clear Fields. A dialog displays the delete options and, if there is data dynamically linked to the field, warns you before deleting it.

### Insert New Field

Works like the Append New Field command, except inserts the new field above the selected field in the table.

### Append New Records

Creates new records for the selected field and puts the records below any other records in the field. A dialog appears requesting the number of records to add. To enable the command, select the table when the database window is in structure mode or select the table from the list when the database window is in viewer mode.

- To delete a record, use the Delete or Backspace key or go to Edit > Clear Record. A dialog displays the delete options and, if there is data dynamically linked to the record, warns you before deleting it.

### Insert New Records

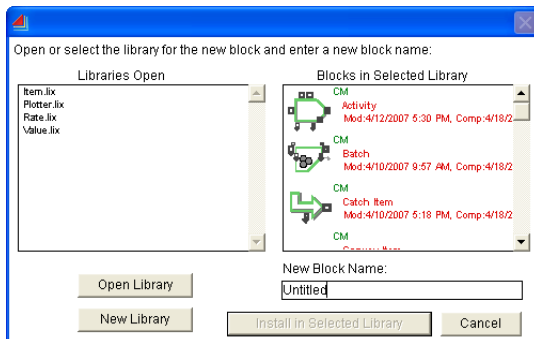
Works like the Append New Records command, except inserts the new record above the selected record.

### Develop menu

The Develop menu is used when creating or modifying blocks in libraries. See the ExtendSim Developer Reference to learn how to create your own libraries of blocks and for additional information on using these commands.

### New Block

Creates a new block. In the initial dialog, you specify the block's name and the library in which you want to save the block. Use the Open Library or New Library buttons in the dialog to open or create a library for the new block.



New Block dialog

After naming the block and selecting the library, you are presented with the default dialog and structure windows. For a quick overview on building a new block, see the Developer Reference.

### Open Block Structure

Opens the structure of the selected blocks. This command is equivalent to holding down the Alt (Windows) or Option (Mac OS) key while double clicking a block in the library window or on the model worksheet.

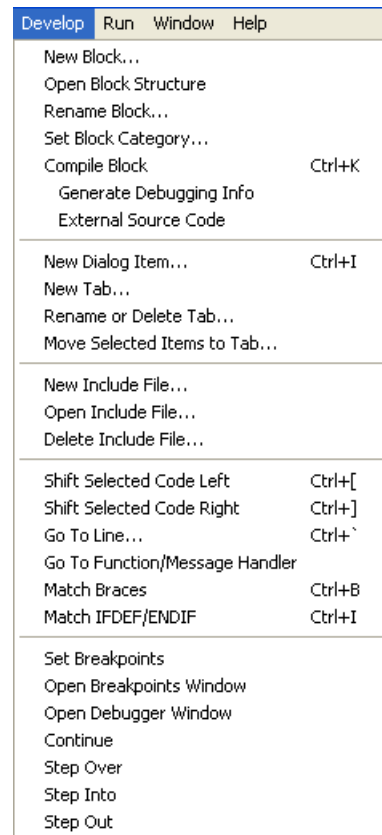
### Rename Block

Changes the name of the block, including a hierarchical block. The block must be either selected in the library window or its structure window must be the active window.

### Set Block Category...

Sets a block's category. This serves two functions:

- To organize blocks within the Library menu by functionality.
- To organize blocks within statistical reports by functionality.





This command is only available when the block's structure window is the active window. If **Show category in Library menu** is not checked in the Set Block Category dialog, the block name will be listed in alphabetical order directly under the library's name in the Library menu. Note that you cannot change the categories of ExtendSim blocks.

### **Compile Block**

Compiles the ModL code for the block. This is useful for checking the syntax of the code that you are working on without having to close the structure or dialog editor windows. The Compile command is only available when the structure window of a block is the active window. You can also choose to compile the block with debugging information and with external source code, as discussed below.

### **Generate Debugging Info**

When the block is compiled, generates debugging information for the block. This allows you to debug block source code using breakpoints, watch points, and so forth. Blocks with debugging code run slower. Their names and any additional information will be listed in the library window in red and they will show in a model window with a red border around them.

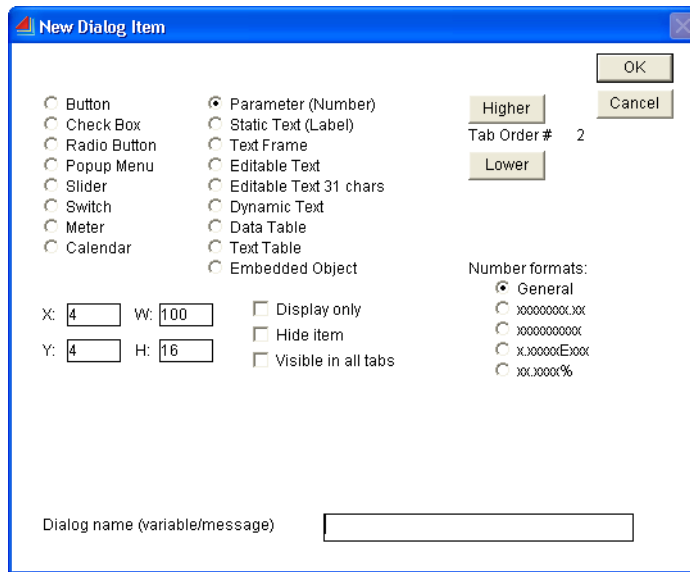
To remove debugging code from a block, open the block's structure window and uncheck the Generate Debugging Info command, or right click the block on a model worksheet and select Remove Breakpoints. To remove debugging code for an entire library, see "Remove Debug Code in Open Library Windows" on page 696.

### **External Source Code**

When the block is compiled, generates a text file containing the block's source code. This is used for version control. If a block has been compiled with external source code, it will be listed in the library window with the designation **CM** (code management) on the right side of its icon.

### New Dialog Item

Adds a dialog item (such as a button, some text, and so on) to the block's dialog window. This command is only enabled when a block's dialog window is the active window. Dialog items, and the use of this command, are covered in detail in the ExtendSim Developer Reference.



**New Dialog Item dialog**

### New Tab

Adds a new tab to the block's dialog window. This is equivalent to double-clicking the blank area to the right of existing tabs. This command is only enabled when a block's dialog window is the active window.

### Rename or Delete Tab

Allows you to rename or delete a tab. This is equivalent to double-clicking the tab or rename or delete it. This command is only enabled when a block's dialog window is the active window.

### Move Selected Items to Tab

Allows you to move the selected dialog item to a specific tab in the block's dialog. This command is only enabled when a block's dialog window is the active window and a dialog item is selected in that window.

### New Include File

Creates a new Include file window. This command is only enabled when a block's structure or dialog window is the active window.

### Open Include File

Opens an existing Include file window. This command is only enabled when a block's structure or dialog window is the active window.

**Delete Include File**

Deletes an Include file. This command is only enabled when a block's structure or dialog window is the active window.

**Shift Selected Code Left**

Moves the selected lines of the ModL code one tab stop to the left. This command is only enabled when a block's structure window is the active window.

**Shift Selected Code Right**

Moves the selected lines of the ModL code one tab stop to the right. This command is only enabled when a block's structure window is the active window.

**Go To Line**

Allows you to quickly move to a specific line in the block code. This command is only enabled when a block's structure window is the active window.

**Go To Function/Message Handler**

Jumps to where the function or message handler is defined in the code. Select the function or message handler name where it is used in the code, then give the command. This is equivalent to holding down the Alt (Windows) or Option (Mac OS) key while double clicking the function or message handler name, or right-clicking on the function or message handler name. This command is only enabled when a block's structure window is the active window and the function or message handler name is selected in the code.

**Match Braces**

Highlights the area between the start and end of braces or parentheses in block code. Click after the opening brace or parenthesis, then give the command. This is equivalent to right-clicking after the brace or parenthesis. This command is only enabled when a block's structure window is the active window.

**Match IFDEF/ENDIF**

Highlights the area between a #IFDEF or #IFNDEF and its corresponding #ENDIF in block code. Click after the #IFDEF, then give the command. This is equivalent to right-clicking after the #IFDEF. This command is only enabled when a block's structure window is the active window.

**Set Breakpoints**

Opens the Set Breakpoints window for the selected block or brings it forward into view if it is already open. This window shows the source code for the block, along with a breakpoint margin on the left of the window. If needed, the block is automatically recompiled in debugging mode. This command is only enabled when a block is selected on the model worksheet.

**Open Breakpoints Window**

Opens the global breakpoints window, showing all breakpoints from all blocks in the model. This command is only enabled when a model window is the active window.

**Open Debugger Window**

Opens the source debugger window or brings it forward if it is already open. This command is only enabled when you are debugging the source code of a block.

### Continue

Continues execution from a breakpoint. This command is only enabled when you are debugging the source code of a block and you are stepping through the code.

### Step Over

Steps over a function call when stepping after a breakpoint. This command is only enabled when you are debugging the source code of a block and you are stepping through the code.

### Step Into

Steps into a function call when stepping after a breakpoint. This command is only enabled when you are debugging the source code of a block and you are stepping through the code.

### Step Out

Steps out of a called function to return to the caller when stepping after a breakpoint. This command is only enabled when you are debugging the source code of a block and you are stepping through the code.

## Run menu

The Run menu lets you modify the way your simulation runs, show 2D and 3D animation, and generate model reports. A hierarchical menu at the end provides commands for debugging models. For more information about running models, see “Model Execution” on page 515.

### Run Simulation

Starts a simulation. The command first checks every block in the model to see that it has been compiled.

You can run multiple models at the same time; processor speed will be divided equally between the simulations. See also “Prioritize Front Model”, below.

### Continue Simulation

This continues a simulation run in an opened model that was previously paused and saved during a simulation run. See “Saving intermediate results” on page 525.

### Run Optimization

Runs an optimization. Alerts the user if there is no Optimizer block (Value library) on the model. See “Optimization” on page 572.

### Simulation Setup

Modifies the start time, end time, and other settings for a simulation run, and provides options for setting random number seeds and showing 3D animation. The dialog is described in detail in “Simulation setup” on page 516.

Run	Window	Help
Run Simulation		Ctrl+R
Continue Simulation		
Run Optimization		
Simulation Setup...		Ctrl+Y
<input checked="" type="checkbox"/> Prioritize Front Model <input checked="" type="checkbox"/> Use Sensitivity Analysis		
Show 2D Animation <input checked="" type="checkbox"/> Add Connection Line Animation Add Named Connection Animation Show 3D Animation		
Launch Proof Launch StatFit		
Generate Report Report Type (Dialogs) ▶		
Add Selected To Report		Ctrl+6
Add All To Report		
Remove Selected From Report		Ctrl+7
Remove All From Report		
Show Reporting Blocks		
Stop		Ctrl+.
Pause		Ctrl+/,
Step		Ctrl+,
Resume		Ctrl+-
Debugging ▶		

**Prioritize Front Model**

If running multiple models simultaneously, choose this to cause the frontmost model to have processor preference over the background models. Running multiple models is discussed at “Working with multiple models” on page 533.

**Use Sensitivity Analysis**

Causes ExtendSim to use sensitivity analysis settings when you run the simulation. Only enabled if a dialog parameter value has sensitivity settings. For more information, see “Sensitivity analysis” on page 568.

**Show 2D Animation**

Causes blocks in the model that have animation to animate when the simulation is run. This is discussed in “Animation” on page 551. Note that some blocks can show some animation, such as text on the icon reporting final values, even if Show Animation is not selected. You can also choose to animate along connection lines or between named connections, as discussed below.

**Add Connection Line Animation**

This option controls whether or not 2D animation will be displayed along connection lines in discrete event models. If this is on, blocks from discrete event libraries, such as the Item library, will display their item animations as discussed in “Animating the movement of items between blocks (discrete event modeling only)” on page 552. If it’s off, only animations on block icons will be displayed. This command requires that Show 2D animation be checked and is only available for discrete event models.

**Add Named Connection Animation**

For named connections (text labels indicate the path and connection line does not appear on the worksheet), this option will cause the animation picture to travel in a straight line between the two text labels. If this option is turned off, the animation picture will disappear when it has reached a text label and reappear at the matching text label. This command requires that Show 2D animation be checked and is only available for discrete event models.

**Show 3D Animation**

Opens the E3D window for showing 3D animation. If the window is subsequently closed, it will reopen when the simulation run begins. This is the same as choosing Run > Simulation Setup > 3D Animation tab and selecting “Show 3D animation during simulation run”. Note that only models using libraries that support 3D animation, such as the Item library, will animate in this window. For more information, see the E3D module that starts on page 389.

**Show Movies (Mac OS only)**

Causes blocks that have QuickTime movies to show their movies when you run the simulation. In addition, the Animate Value block (Animation 2D-3D library) has a dialog option to play a movie when it gets a value at its input. Movies must be stored in the Extensions folder and be QuickTime movie files. This command requires that you have QuickTime installed.

**Launch Proof (Windows only)**

Opens the Proof Animation application. This command is only enabled if Proof Animation is installed. Proof Animation is only available with certain ExtendSim products.

### Launch StatFit (Windows only)

Opens the Stat::Fit application. This command is only enabled if Stat::Fit is installed. Stat::Fit is included with the Windows version of the ExtendSim AT and ExtendSim Suite products; it can be purchased separately for use with other ExtendSim products. For more information about Stat::Fit, see “Stat::Fit (Windows only)” on page 586.

### Generate Report

Causes ExtendSim to generate a report file when the simulation is run. ExtendSim will prompt for a name for the new report file when the model is run. Report files show final results of the simulation for selected blocks. They are described in “Reports” on page 596. Use the commands that follow to specify which type of report to generate and which blocks to include in the report.

### Report Type

Allows you to choose which report type, **Dialogs** or **Statistics**, to generate when Generate Report is checked and the model is run. The current selection is shown in parenthesis following the command.

### Add Selected To Report

Causes blocks selected in the active model window to be included in the report when Generate Report is checked and the model is run.

### Add All To Report

Causes all blocks in the active model window to be included in the report when Generate Report is checked and the model is run.

### Remove Selected From Report

Causes blocks that have been selected in the active model window to be removed from the report.

### Remove All From Report

Causes all blocks in the active model window to be removed from the report.

### Show Reporting Blocks

Causes blocks that have been included in a report to show the word **Report** on them in the model window.

### Stop

Stops the simulation. This is the same as the Stop button in the toolbar. As an alternative, you can hold down the Control (Windows) or Command (Mac OS) key while pressing the period (.) key.

### Pause

Halts the simulation temporarily. This is the same as the Pause/Resume button in the toolbar. When you give this command, the word **Paused** appears in the model’s status bar until the simulation is resumed. To restart the simulation, give the Resume command, below, or click Pause/Resume.

### Step

Steps the simulation depending on which mode (Step Each Block, Step to Next Animation, or Step Entire Model) is selected in the Debugging menu as discussed at “Debugging” on page 711. This is the same as the Step button in the toolbar.

## Resume

Restarts a paused simulation. This is the same as clicking the Pause/Resume button in the toolbar.

## Debugging

This hierarchical menu lets you modify the way your simulation runs and facilitates finding a modeling problem. The three “Step...” commands in this menu determine how the Step command in the Run menu performs during a simulation run. The Trace commands are used to generate a Trace file of the values for each selected block in the active model at each step of the simulation. For debugging hints, see “Animation features for debugging” on page 618.

Pause At Beginning
<input checked="" type="checkbox"/> Step Each Block
Step To Next Animation
Step Entire Model
Show Block Messages
Only Simulate Messages
Scroll To Messages
Generate Trace
Add Selected To Trace      Ctrl+8
Add All to Trace
Remove Selected From Trace    Ctrl+9
Remove All From Trace
Show Tracing Blocks
Profile Block Code
<input checked="" type="checkbox"/> Show Debug Messages

Debugging menu

### Pause At Beginning

If this is enabled, pauses the simulation after it starts so that the user can step through the run from step zero. When the simulation is paused, the word **Paused** appears in the model’s status bar.

### Step Each Block

Controls the behavior of the Step command or button so that you can step through a simulation block by block. This command is only active when the Pause command or Pause button have been activated.

### Step Next Animation

Controls the behavior of the Step command or button so that you can step through a simulation, pausing only at animation changes. In models where there are many steps between animation changes, this option makes going from visible change to visible change much faster. This command is only active when the Pause command or Pause button have been activated.

### Step Entire Model

Controls the behavior of the Step command or button so that you can step through an entire cycle of all blocks in the model. Each Step command starts at the selected block and continues the simulation run until the execution order returns to the original block. This command is only active when the Pause command or Pause button have been activated.

### Show Block Messages

Used in conjunction with the preceding Step Each Block, Step Next Animation, or Step Entire Model commands to show system messages on the blocks. When you are stepping through a simulation and choose Run, the simulation automatically pauses between step cycles (defined by the three preceding commands.) This command causes the block that is active to be highlighted with the current message name written on it. If that block is not currently visible, the window will automatically scroll to the block if Scroll To Messages (see below) is checked. Block messages are discussed “Block messages” on page 535.

### Only Simulate Messages

When the Show Block Messages command is active, causes only the messages that occur during the Simulate stage of the run to be shown. This saves time by not showing all the initial and final messages.

**Scroll To Messages**

When the Show Block Messages command is active, causes the window to scroll automatically, following the path of messages so that the user can quickly step through the model.

**Generate Trace**

Causes ExtendSim to generate a trace file during the simulation. ExtendSim will prompt for a name for the new trace file when the model starts running. The content of the trace file depends on which blocks have been selected to be included. For more information, see the following commands and “Model tracing” on page 620.

**Add Selected To Trace**

Causes blocks that have been selected in the active model window to be included in the trace file. Trace files are generated if Generate Trace is checked when the model is run.

**Add All To Trace**

Causes all blocks in the active model window to be included in the trace file. Trace files are generated if Generate Trace is checked when the model is run.

**Remove Selected From Trace**

Causes blocks that have been selected in the active model window to be removed from the trace file.

**Remove All From Trace**

Causes all blocks in the active model window to be removed from the trace file. Use this before starting a new type of trace.

**Show Tracing Blocks**

Causes blocks that have been included in a trace to show the word **Trace** on them in the model window.

**Profile Block Code**

Generates a text file showing the percentage of time that each block spent executing during the simulation run. See “Inefficient settings or block code” on page 532 for more information about profiling a model.

**Show Debug Messages**

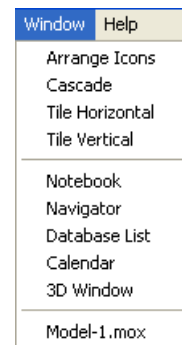
Allows the user to turn on and off the messages created using the debugMsg() function.

**Window menu**

In addition to the standard (Windows only) commands for arranging the windows in ExtendSim, the Window menu lists ExtendSim windows such as the Notebook and Navigator. The window also lists all open models, dialogs, and library windows at the bottom. To bring a window to the top of your workspace, select it from the menu.

**Notebook**

Opens or brings forward the Notebook for a model. This is the same as choosing the Open Notebook tool. Notebooks are used for controlling model parameters, reporting simulation results, and documenting the model. The words (**has data**) after the Notebook command indicates it has content. For more information, see “Notebooks” on page 508.





### Navigator

Opens or brings forward a Navigator window for a model. This is the same as choosing the Open Navigator tool. The Navigator has three modes: an explorer-type interface for the model; a list of databases used in the model, and a library window for open libraries. For more information about the Navigator, see “Navigator” on page 670.

### Database List

Opens or brings forward a window showing the databases (if any) used in the model. This is the same as choosing the Open Navigator tool and selecting Database List in the Navigator’s popup menu. Databases and their usage are discussed at “ExtendSim databases for internal data storage” on page 638.

### Calendar

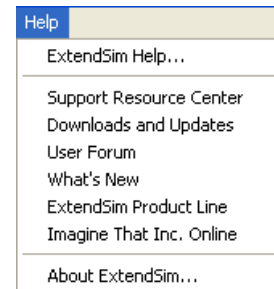
Opens or brings forward a Calendar. If you run a model and are using Calendar dates in the model, the calendar will automatically scroll from the beginning simulation date until the end. This window is only available if Use Calendar Dates is selected in the Setup tab of the Run > Simulation Setup command.

### E3D Window

Opens or brings forward the E3D window for showing 3D animation. This is the same as selecting the Open E3D Window tool. The command is only enabled if your ExtendSim product has 3D animation capabilities. For more information, see “The E3D environment” on page 396.

## Help menu

This menu has commands for accessing ExtendSim Help and useful web sites.



### ExtendSim Help

Shows a list of available Help topics. Select a topic from the list and click Help for more information. You can search for key words within any Help topic.

### Support Resource Center

Uses your current browser to open an ExtendSim web page with many types of support resources, including example models, FAQs, and downloadable manuals.

### Downloads and Updates

Uses your current browser to open the ExtendSim product updates web page.

### User Forum

Uses your current browser to open the ExtendSim user group home page.

### What’s New

Uses your current browser to open the ExtendSim web page describing the features in the newest ExtendSim release.

### ExtendSim Product Line

Uses your current browser to open the ExtendSim web page that describes the ExtendSim products and versions.

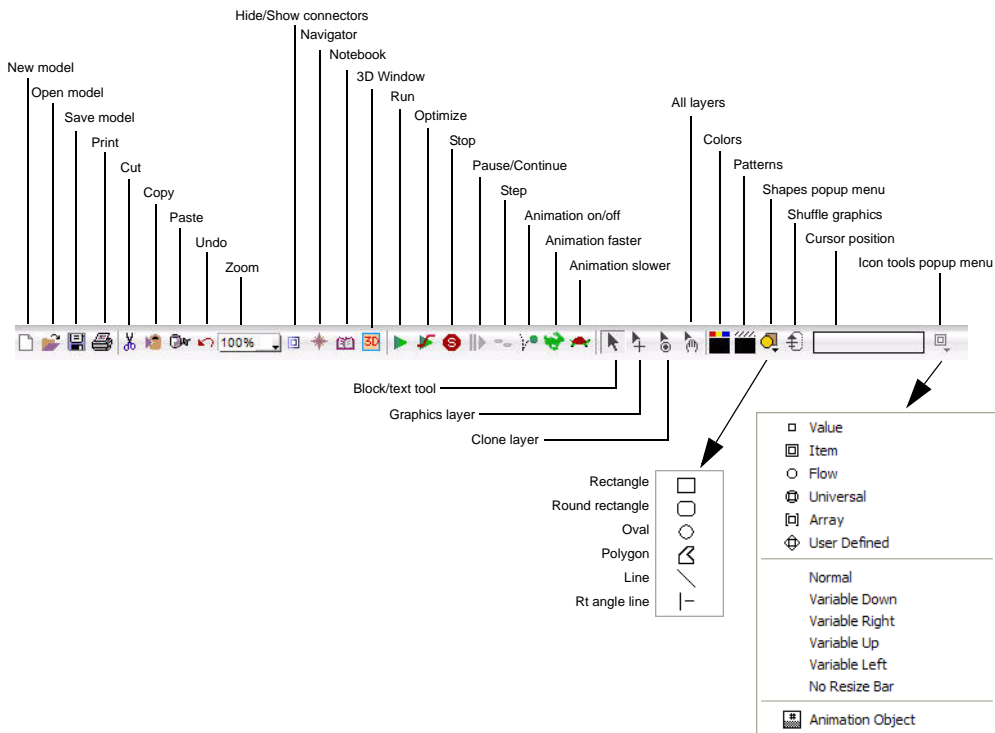
### Imagine That Inc. Online

Uses your current browser to open the Imagine That, Inc. home page at [www.thatinc.com](http://www.thatinc.com).

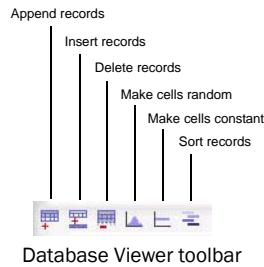
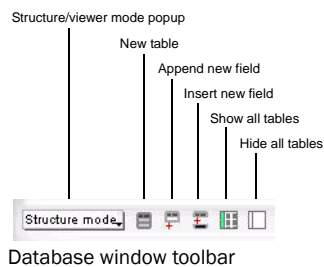
### About ExtendSim (Windows only)

Opens a banner window that lists the ExtendSim product, version and serial number and the registered user of the license. Click on the window to close it.

### Toolbar buttons



### ExtendSim database tool bars



# Reference

## Value Library Blocks

A detailed description of the building blocks in the Value library

This chapter provides tables of the blocks in the Value library, separated by category. Each Value library block has an icon that represents its function, predefined input and output connectors for quick model building, and a dialog for entering parameters and viewing results.

The tables have brief descriptions and are useful to get an idea of a block's functionality in your model. For more details about the usage of a block:

- Click the Help button in the lower left of the block's dialog
- Look in the index of ExtendSim's online Help for the block's name




### Submenus


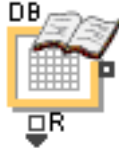
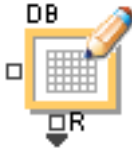
The blocks are listed by the block categories displayed in hierarchical submenus of the Value library menu:

- Data Access: Accessing global data
- Holding: Accumulating or storing values
- Inputs: Generating values
- Math: Calculating values
- Optimization: Finding the best solution
- Outputs: Writing data to files or display
- Routing: Routing or deciding which value to use
- Statistics: Calculating mean, variance

### Data Access



The blocks in this category are used to access and store data in your models.

Block	Function
Data Import Export 	Imports and exports data from and to external data sources such as Microsoft Excel, ODBC compatible databases, text files, and files from the internet via FTP.  The data can reside in ExtendSim database tables and global arrays.
Data Init 	Defines any values needed to initialize multiple database tables and global arrays before a simulation run starts.
Data Source Create 	Creates, resizes, deletes, and views global arrays and text files. Global arrays are general purpose arrays available anywhere in the model. Text files are ascii files containing text data, used as a data source, that can be saved on the computer's local hard drive or on the network.  Note: If you are interested in creating, editing, or viewing database tables, use the commands in the database menu.

Block	Function
<p>Data Specs</p> 	<p>Outputs selected specifications on data sources. Data sources can be either ExtendSim databases or global arrays.</p> <p>Each row in the table defines a specification whose value will be output on the corresponding variable output connector on the block.</p>
<p>Read</p> 	<p>Reads data from a data source to be used in a model. The data sources supported are the ExtendSim database, global arrays, Excel workbooks, Text Files, and local tables.</p> <p>You can specify whether you want to read a single number or a row or column of data and you can specify when the data should be read.</p>
<p>Write</p> 	<p>Writes data from a model to a data destination. The data destinations supported are: ExtendSim databases, global arrays, Excel Workbooks, Text Files, and Local Tables.</p> <p>You can specify whether you want to write a single number or a row or column of data, and when the data should be written.</p>

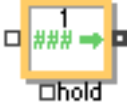



## Holding

The blocks in this category are used to accumulate or store contents.

Block	Function
<p>Holding Tank</p> 	<p>Accumulates the total of the input values, and allows you to request an amount to be removed if it is available. You can also choose to allow a request that would make the contents go negative (such as an overdraft).</p> <p>You can specify that the inputs are summed or integrated.</p>
<p>Wait Time</p> 	<p>Holds its inputs for a specified amount of simulation time (the delay) before passing them to the output. This block works like a conveyor with slots: values come into a slot, advance position based on an advance in simulation time, then exit when their slot reaches the end of the conveyor.</p> <p>Note: This block should not be used in a discrete event model.</p>


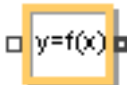
## Inputs


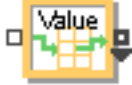



The blocks in this category generate values to be used as inputs for other blocks.

Block	Function
	<p>Generates a constant value at each step. You specify a constant value in the dialog (the default constant is 1.0). This block is typically used for setting the value for inputs to other blocks. For example, you can use it for a steady flow of fluid, cash, or a delay time value.</p> <p>If the ValueIn input on the left is connected, the input value is added to the constant in the dialog and the sum of those two numbers is output.</p>
	<p>Outputs a true value (1) at specified times, and a false value (0) at all other times. In the dialog, you specify the time between outputting true values (the delay or time out); the dialog value is overridden by the D connector. The R connector resets the block back to the beginning of the delay period.</p>
	<p>Generates random integers or real numbers based on the selected distribution. You can use the dialog or the three inputs, 1, 2, and 3 to specify arguments for the distributions. You can select the type of distribution or use an Empirical Table. The Empirical distribution uses a table to generate a discrete, stepped, or interpolated distribution.</p>
	<p>Outputs the value of a simulation variable. It is usually used in conjunction with a decision-type block, for example, to halt a process after current time reaches a certain value. The variables you can use are: current run number, current step, current time, end time, number of runs, number of steps, start time, time step, and random seed.</p>

## Math


The blocks in this category are used to perform mathematical calculations and functions.

Block	Function
	<p>Makes a decision and outputs TRUE or FALSE values based on the inputs and defined logic. The dialog lets you perform the following tests comparing A to B: greater than, greater than or equal to, equal to, less than, less than or equal to, and not equal. You can also test for A being an invalid number (noValue). The block can be set to use hysteresis.</p>
	<p>Outputs the results of the equations entered in the dialog. You can use ExtendSim's built-in operators, functions, and some or all of the input values as part of the equation. The equations can have any number of inputs and any number of outputs.</p>

Block	Function
<p>Integrate</p> 	<p>Integrates the input values over time using either Euler or Trapezoidal integration methods. If present, an initial value is added to the outputs.</p>
<p>Lookup Table</p> 	<p>Acts as a lookup table (x in and y out) that are used to calculate what the output value would be for the given input. Input values can come from an input connector (the default) or can be simulation time. The output can be discrete, interpolated or stepped.</p>
<p>Math</p> 	<p>Performs a selected mathematical operation on its inputs and outputs a result.</p>
<p>Max &amp; Min</p> 	<p>Determines the maximum and minimum values from among the values input. The dialog shows the maximum and minimum values and the input connectors they came from. The block outputs the maximum or minimum values and the respective connector number.</p>
<p>Time Unit</p> 	<p>Converts the value at the input connector from one time unit to another.</p>




## Optimization

The block in this category is used to find the optimum values for your simulation.

Block	Function
<p>Optimizer</p> 	<p>Searches for the best set of model parameters that maximizes profit or minimizes cost, given parameter limits and any entered constraints. Uses evolutionary strategies that are similar to genetic algorithms.</p>





## Outputs

The blocks in this category output data to files or to display.

Block	Function
Command 	Sends Excel macro or general DDE commands to a spreadsheet application when triggered by the “Send” connector.
Display Value 	Displays the value at the input connector on each simulation step. This is useful for debugging models and scripts because you can display the value of a block's value output connector at any time.
Notify 	Notifies the user when an event occurs. The notification can take the form of playing a sound, stopping the simulation, or querying the user for a new input value.

## Routing

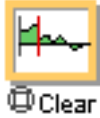


The blocks in this category route values or decide which values to use.

Block	Function
Catch Value no group 	Outputs values thrown from a Throw Value block. You specify in the dialog of the block which throw block(s) this block is connected to. This block is typically used for passing a value from one point in a model to another without using connectors, in conjunction with Throw Value blocks.
Select Value In 	Selects its output value to be one of its inputs according to the value of the select connector. Up to 253 inputs can be used.
Select Value Out 	Sends its input value to one of its outputs according to the value of the select connector. Up to 253 outputs can be used.
Throw Value no group 	Throws a value to one or more catch blocks in the model. You specify in the dialog of the block which catch block(s) this block is connected to. This block is typically used for passing a value from one point in a model to another without using connectors, in conjunction with Catch Value blocks.



## Statistics

The blocks in this category report and clear statistics on various blocks.

Block	Function
Clear Statistics 	Clears the statistics in various blocks in a model at a specific time or event. Useful in reducing the effects of warm-up in a model.
Mean & Variance 	Calculates the mean, variance, and standard deviation of the values input during the simulation.
Statistics 	Collects common statistics from blocks in a model into a single table. You can select which types of blocks will have their information collected. The choices are Activity, Mean & Variance, queue, Resource Item, Resource pool, or mixed.



# Reference

## Item Library Blocks

A detailed description of the building blocks in the Item library

This chapter provides tables of the blocks in the Item library, separated by category. Each Item library block has an icon that represents its function, predefined input and output connectors for quick model building, and a dialog for entering parameters and viewing results.

The tables have brief descriptions and are useful to get an idea of a block's functionality in your model. For more details about the usage of a block:

- Click the Help button in the lower left of the block's dialog
- Look in the index of ExtendSim's online Help for the block's name

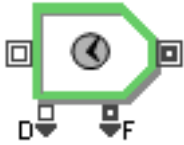


### Submenus


The blocks are listed by the block categories displayed in hierarchical submenus of the Item library menu:

- Activities: Processing items
- Batching: Joining and dividing items
- Data Access: Accessing global data
- Information: Getting information about items
- Properties: Assigns and displays properties for items
- Queues: Holding, sorting, and ranking items
- Resources: Representing items as resources
- Routing: Moving items to the correct place
- Executive: Needed in every discrete event and discrete rate model to handle events

### Activity



The blocks in this category are used to process items in the model.

Block	Function
<p>Activity</p> 	Holds one or more items and passes them out based on the process time and arrival time for each item.
<p>Convey Item</p> 	Behaves as a conveyor (accumulating or non-accumulating) that moves items from one location to another.
<p>Transport</p> 	Transports item from one point to another based on distance and speed information. Useful in creating models with 3D animation as this block is the primary way to model multiple 3D objects moving at the same.

Block	Function
<p>Workstation</p> 	Behaves as a workstation that has both processing and queueing aspects.



### Batching

The blocks in this category are used to join and divide items.

Block	Function
<p>Batch</p> 	Allows items from several sources to be joined as a single item. Useful for synchronizing resources and combining various parts of a job ("kitting").
<p>Unbatch</p> 	Produces multiple items from a single input item. This block can be used to disassemble a kit, break a message packet into component messages, route the same message to several places, or distribute copies of invoices.





### Data access

The blocks in this category are used to access and store data in your models.

Block	Function
<p>Read(I)</p> 	Reads data from a database when an item arrives. You can define an indefinite number of reads to be made by the block when an item passes through.
<p>Write(I)</p> 	Writes data to a database when an item arrives. You can define an indefinite number of writes to be made by the block when an item passes through.

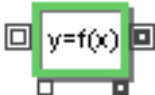


## Information

The blocks in this category provide information about the blocks in your model.

Block	Function
<p>Cost By Item</p> 	Views and displays the cost of the cost accumulators that pass through it. By using a sorting attribute or the row connector, the throughput, average cost, and total cost can be calculated for different item types.
<p>Cost Stats</p> 	Reports Statistics for costing blocks in a model. Place this block anywhere in the model and it will report the following statistics for all costing blocks in the model: Block Number (or label, if a label is entered in the block), Block Name, Cost Per Item, Cost Per Time Unit, and Total Cost.
<p>History</p> 	Views and displays information about the items that pass through it. You specify which properties will be displayed. Properties can be attributes on the item, priority values, or other more obscure values that are available on the item.
<p>Information</p> 	Reports statistics about the items that pass through it, such as cycle time and TBI (Time Between Items).

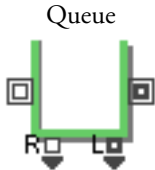
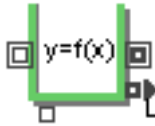

## Properties

The blocks in this category assign and display item properties.

Block	Function
<p>Equation(I)</p> 	Calculates equations when an item passes through. The equations can use multiple inputs and properties of the item as variables, and the result(s) of the equations can be assigned to multiple outputs and properties of the item.
<p>Get</p> 	Displays and outputs properties from items that are passing through. The property value is shown in the dialog and output at the value output connector. You can specify multiple properties and multiple output connectors.
<p>Set</p> 	Sets the properties of items passing through the block from input connectors, values in the dialog, or databases.




## Queues



The blocks in this category hold, sort, and rank items.

Block	Function
 <p>Queue</p>	<p>Queues items and releases them based on a user selected queuing algorithm, such as Resource pool queue, Attribute value, First in first out, Last in first out, and Priority. Options include renegeing and setting wait time.</p> <p>If you need more advanced control over the queuing algorithm, consider using the Queue Equation block, below.</p>
 <p>Queue Equation</p>	<p>Queues items and releases them based on the results of user entered equations. The result(s) of the equations can optionally be assigned to properties of the item</p>
 <p>Queue Matching</p>	<p>This block is useful for matching one type of item with another, such as in reassembling parts in the correct order or to insure that subassemblies are correctly matched with each other. Has a specified number of internal queues for holding items in separate groups. Releases a group when there is downstream capacity and the group requirements have been met.</p>

## Resources






The blocks in this category represent items as resources.

Block	Function
 <p>Resource Item</p>	<p>This block holds and provides items (cars, workers, orders, and so forth) to be used in a simulation. It can be used as part of an open or closed system.</p>
 <p>Resource Pool</p>	<p>This block holds resource pool units to be used in a simulation. These units limit the capacity of a section of a model. For example, this could be used to represent a limited number of tables at a restaurant. The Resource Pool block works with the Queue block in Resource Pool mode and the Resource Pool Release blocks.</p>
 <p>Resource Pool Release</p>	<p>Releases a resource back to its resource pool as an item passes through.</p>



Block	Function
<p>Shift</p> 	<p>Generates a schedule over time which can be used to change the capacity of other blocks in the model. Multiple Shift blocks can be connected together to create complex shift patterns. For example the typical 40 hour work week can be built with two connected Shift blocks, the first containing the work days, the second contains the work hours.</p>
<p>Shutdown</p> 	<p>Generates shutdown information for activities in the Item library and valves in the Rate library according to inputs or distributions specifying time between failures and time to repair.</p>

### Routing

The blocks in this category move items to the correct place.


Block	Function
<p>Catch Item</p> 	<p>This block catches items sent by Throw Item blocks without using connection lines. Any number of Throw Item blocks can send items to a Catch Item block.</p>
<p>Create</p> 	<p>Provides items or values for a discrete event simulation at specified interarrival times. Choose either a distribution or a schedule for the arrival of items or values into the model.</p>
<p>Exit</p> 	<p>Passes items out of the simulation. The total number of items absorbed by this block is reported in its dialog and at the value output connectors.</p>
<p>Gate</p> 	<p>Limits the passing of items through a portion of the model, either by demand or by using a sensor connector to monitor how many items are in a section of the model.</p>
<p>Select Item In</p> 	<p>Selects items from one input to be output based on a decision.</p>



Block	Function
<p>Select Item Out</p> 	<p>Selects which output gets items from the input, based on a decision</p>
<p>Throw Item</p> 	<p>This block throws items to a Catch block without using connection lines. Any number of Throw blocks can send items to a single Catch block. You could use the Throw and Catch blocks instead of using Combine blocks, even from inside one hierarchical block to inside another one.</p>

### Executive

The block in this category is needed in every discrete event and discrete rate model to handle events.

Block	Function
<p>Executive</p> 	<p>This block must be placed to the left of all other blocks in discrete event and discrete rate models. It does event scheduling and provides for simulation control, item allocation, attribute management, and other discrete event and discrete rate model settings.</p>



# **Reference**

## **Rate Library Blocks**

A detailed description of the building blocks in the Rate library





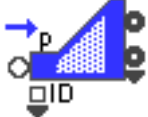
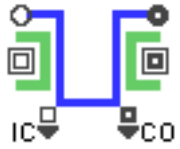
This chapter provides a brief description of the blocks in the Rate library. There are no block categories used in this library. For more details about the usage of a block:



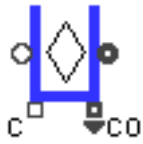


- Click the Help button in the lower left of the block's dialog
- Look in the index of ExtendSim's online Help for the block's name

### Block descriptions

The following table is useful to get an idea of a block's functionality in your model. More details about usage of a block can be obtained in two ways:

- Look in the index of ExtendSim's online Help for the block's name
- Click the Help button in the lower left of the block's dialog

Block	Function
	<p>Prioritizes the flow going through it, and thereby allows you to specify a preference for where the model's flow should be directed. The bias from a Bias block is by definition stronger than the bias from any Merge or Diverge block.</p>
	<p>Receives flow from non-connected Throw blocks. The Throw Flow and Catch Flow blocks (and the Merge and Diverge blocks in certain modes as well) can be used to move flow without the use of connection lines.</p>
	<p>Changes the flow unit of measurement. Blocks that are connected together through flow connections and share the same flow unit are called a unit group. The Change Units block uses the conversion factor to create a new unit group. This causes the blocks downstream of the Change Units block to be in a unit group different from its upstream blocks.</p>
	<p>Add a delay to the flow available at the output of the block. The Convey Flow is FIFO and can be accumulating or non-accumulating.</p>
	<p>Distributes the input flow to two or more outputs. Systems modeled using discrete rate technology frequently have one flow stream that needs to be split (or diverged) into multiple streams (referred to as branches). It has seven different rule-based options that define how the inflow will be distributed across the outputs.</p>
	<p>Acts as a Tank interfacing between Flow and Items. The Tank acts as source, intermediate storage, or sink.</p>

Block	Function
<p data-bbox="435 422 500 449">Merge</p> 	<p>Merges flows from multiple inputs into one output. Systems modeled using discrete rate technology frequently have multiple flow streams (referred to as branches) that need to be merged into one stream. It has seven different rule-based options that define how the inflow will be merged from all inputs.</p>
<p data-bbox="435 600 500 627">Sensor</p> 	<p>Displays potential upstream supply and potential downstream demand in a Flow branch.</p>
<p data-bbox="440 743 493 770">Tank</p> 	<p>Acts as source, intermediate storage, or sink. As a residence type block the Tank has the capacity to hold defined amounts of flow as time advances. If a Tank has no outflow connection, by definition it is being used as a sink. Conversely, if a tank has no inflow connection, by definition it is being used as a source.</p>
<p data-bbox="402 932 506 959">Throw Flow</p> 	<p>Sends flow to non-connected Catch blocks. The Throw Flow and Catch Flow blocks (and the Merge and Diverge blocks in certain modes as well) can be used to move flow without the use of connection lines.</p>
<p data-bbox="440 1052 493 1079">Valve</p> 	<p>Controls, monitors, and transfers flow. This block places an upper bound on the rate at which flow is allowed to pass through. Goals can be set up to control flow.</p>



# Reference

## Utilities Library Blocks

A detailed description of the building blocks in the Utilities library

This chapter provides tables of the blocks in the Utilities library, separated by category. Each Utilities library block has an icon that represents its function, predefined input and output connectors for quick model building, and a dialog for entering parameters and viewing results.

The tables have brief descriptions and are useful to get an idea of a block's functionality in your model. For more details about the usage of a block:

- Click the Help button in the lower left of the block's dialog
- Look in the index of ExtendSim's online Help for the block's name


### Submenus

The blocks are listed by the block categories displayed in hierarchical submenus of the Utilities library menu:

- Developer Tools: Provides information about OLE objects
- Discrete Event Tools: Managing and reporting on discrete event models
- Information: Getting information about the model
- Math: Performing mathematical calculations
- Model Control: Controlling how the model runs
- Time: Working with time functions



### Developer Tools

The block in this category provides information about OLE objects.



Block	Function
	Gets information about IDispatch properties and methods exported by activeX controls or objects that have been embedded or automated. It is useful for those needing to navigate the object model of an outside application or an embedded object via Extend's OLE programming capabilities.

### Discrete Event Tools

The blocks in this category are used to manage and report on discrete event models.

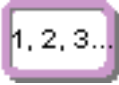
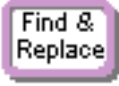
Block	Function
	Displays detailed information about the messages used to pass items. This block is "invisible" to the items and messages and passes each message it gets on to the next block. Also see the Record Message block, below.
	Views and optionally initializes the contents of a queue via tables. Connect the input connector to the length (L) connector on a queue (this is the only type of connector that can be connected to this block).



Block	Function
Record Message 	Records all the messages that pass between two value connections in a discrete event model. Also see the Item message block, above.
Stop Message 	Transfers the value from the input to the output without relaying any discrete event messages. The output connector is set equal to the input connector, but the message is not sent out through the output connector.


### Information

The block in this category provides general model information to the user.

Block	Function
Count Blocks 	Calculates the number of blocks of each type in the model.
Find and Replace 	Drag a clone of a dialog item onto this icon to search for similar dialog items. You can also manually enter search criteria into the dialog.


### Math






The blocks in this category perform mathematical functions.

Block	Function
Data Fitter 	Uses matrix techniques to obtain a least mean square curve fit to a set of data. Both the data and the fitted curve are output during a simulation through the output connectors (D and F, respectively).

### Model Control



The blocks in this category provide various means of controlling the simulation run.

Block	Function
Buttons 	Creates a pushbutton interface for a model. An equation is executed whenever the button is pushed. The button itself can be cloned to the model worksheet, notebook, or hierarchical block to create a user interface for the model.

Block	Function
<p>Feedback</p> 	<p>Helps resolve feedback calculation order issues in continuous models. This is only for advanced use and only to be used in situations where you know that you have a calculation problem that is based on the simulation order of a feedback loop.</p>
<p>Pause Sim</p> 	<p>Causes the simulation to pause when certain conditions are met.</p>
<p>Popups</p> 	<p>Allows you to define a custom popup menu that you can clone to the worksheet and also use as a numeric input to your model. Outputs the value of the popup defined in the dialog, which is usually cloned to the worksheet.</p>
<p>Run Model</p> 	<p>Primarily used to clone out the Run Simulation Now button onto the worksheet or notebook. This can also be done with the Buttons block. Runs the simulation when the “Run Simulation Now” button is pressed.</p>
<p>Switch</p> 	<p>Defines a switch for use on a model worksheet. Differs from the use of the switch control defined in the model menu in that it sends out a message if it is in a discrete event model. This will have the effect of making the other blocks in the model react immediately to the switch click.</p>

## Time

The blocks in this category work with time functions in the model.

Block	Function
<p>RealTimer</p> 	<p>Shows the duration of a simulation in real time. It should be placed at the far right side of the model worksheet.</p>
<p>TimeSync</p> 	<p>Synchronizes the model to run in real time. It does this by pausing on each simulation step until the amount of simulation time that has passed equals the amount of real time that has passed. This is only effective if the model is running faster than real time.</p>

# **Reference**

## **Upper Limits**

A list of the maximum numbers of things  
that you can do at one time

Like every program, ExtendSim has its limits. It is unlikely that you will find them in your normal work, but it is good to know what they are. Note: Some limits depend on available memory.

Steps in a simulation run	2 billion
Number of simulations in a multiple run	2 billion
Block name or label length, characters	31
Blocks in a model	2 billion
Blocks in a library	200
Libraries open at one time	40
Text files open at one time	200
Databases per model	2 billion
Tables per database/fields per table/records per table	2 billion/1,000/2 billion
Output connectors in a model (nodes)	2 billion
Connectors per block	255
Length of a block's ModL code (characters)	4 megabytes
Dialog items in a block	1024
Dynamic arrays (each array)	2 gigabytes
Number of array dimensions	5
Maximum index for array dimensions	2 billion elements total
Dynamic arrays per block	255
Columns in a table	255 (data table); 255 (text table)
Total table size (cells) per block for static data tables	3260 (data table); 2030 (text table)
Total table size (cells) each, for dynamic data tables	2 billion
Variable name length, characters	63
Maximum popup menu length	5100 characters or 20 strings
User defined function arguments	127
Nested loops	32
Maximum, minimum of real numbers	$\pm 1E\pm 308$
Maximum, minimum of integer numbers	$\pm 2,147,483,647$
Significant figures in real calculations	16 (double)
Number of attributes for discrete event item	500

# Reference

## Cross-Platform Considerations

File conversion, file name comparisons, and keyboard shortcuts  
for the Windows and Macintosh operating systems

## Libraries

Libraries are located in the ExtendSim7\Libraries folder, but can be in subfolders within that folder. Libraries needed by a model can also be at the model file location.

**Windows:** ExtendSim supports library file names up to 64 characters. Library names must end in the “.LIX” extension.

**Mac OS:** Library names can be up to 31 characters long and usually end in “Lib”, although this is not required.

## Models

For Windows, ExtendSim supports long file names. All Windows models must end in the “.MOX” extension. Mac OS model names can be up to 31 characters.

## Menu and keyboard equivalents

The following table lists some common actions and keyboard shortcuts under the Windows and Mac OS systems. “Appendix A: Menu Command Reference” starting on page 680 contains pictures of the ExtendSim menus, including the keyboard equivalents for the menu commands.

Action	Command	Windows keyboard	Mac OS keyboard
Open a model	File > Open	CTRL+O	CMND+O
Open a library	Library > Open Library	CTRL+L	CMND+L
Save a model	File > Save Model	CTRL+S	CMND+S
Close the active window	File > Close	CTRL+W	CMND+W
Run a simulation	Run > Run	CTRL+R	CMND+R
Stop a simulation	Run > Stop	CTRL+period (.)	CMND+period (.)
Stop a library search		CTRL+period (.)	CMND+period (.)
Select a parameter for sensitivity analysis	Edit > Sensitize Parameter (Select parameter first)	Hold down the CTRL key and click once on the parameter	Hold down the CMND key and click once on the parameter
Open a hierarchical block's structure to edit the icon, etc.	Develop > Open Block Structure (Select block icon first)	Hold down the Alt key while double-clicking on the block's icon	Hold down the Option key while double-clicking on the block's icon
Open a library block's structure to edit the Modl code or icon	Develop > Open Block Structure (Select block icon first)	Hold down the Alt key while double-clicking on the block's icon	Hold down the Option key while double-clicking on the block's icon
Remove the grid when aligning drawing objects, connectors, etc.		Hold down the Alt key while moving the object	Hold down the Option key while moving the object

Action	Command	Windows keyboard	Mac OS keyboard
Proportionately scale drawing object		Hold down the Shift key while resizing the object	Hold down the Shift key while resizing the object

### Transferring files between operating systems

The Windows and Mac OS versions of ExtendSim are cross-platform compatible. For example, if you build a model or a library on your Windows computer, you can move it to a Mac OS computer and ExtendSim will read it, and vice versa.

There are three considerations when transferring ExtendSim files between Windows and Mac OS systems: file name adjustments, physical transfer, and file conversion.

- ☞ Models and libraries developed in an older version and different platform may not transfer successfully. It is strongly recommended that you upgrade to the latest version on the source platform, re-save the structure of any hierarchical blocks stored in libraries (see “Saving hierarchical blocks to a library” on page 547) and recompile any libraries that you have created. Then re-save your models before transferring your files.

#### File name adjustments

- **Windows to Mac OS:** If you are transferring files from a Windows computer to a Mac OS, you do not need to change the file name or delete the extension; the Mac OS system can read names up to 31 characters long.

- ☞ It is important that you do not delete the MOX extension from a Windows model file name before transferring the model to your Mac OS system. The MOX extension is required so ExtendSim can identify the file as a Windows model. After ExtendSim has converted the Windows model to Mac OS format, save the model under the same name or a new name.

- **Mac OS to Windows:** If you transfer files from a Mac OS to a Windows computer, you may need to change the name of your file before you transfer it. ExtendSim supports file names of up to 64 characters for Windows. File names must end in a three-character extension (the extensions are “.MOX” for ExtendSim model names, “.LIX” for library names, and “.TXT” for text file names). To change your file names to Windows format, change the name of the file *on the Mac OS computer* using ExtendSim’s Save As command (if the file is open) or using the Finder (if the file is not open).

#### Physically transferring files

Once you have made any necessary file name adjustments, physically transfer the files between Windows and Mac OS computers. This process depends on your system resources and is independent of ExtendSim. For example, you might copy the file onto a memory stick. Or you could send the files directly from one computer to the other if the computers are networked.

- ☞ Libraries and extensions created on a Mac OS computer need to be converted *before* being physically transferred to the Windows system, as discussed below.

#### File conversion

Depending on the type of file, file conversion may be handled automatically by ExtendSim or may involve using a conversion application. As discussed below, ExtendSim automatically converts model files when they are opened on a different platform. If you program your own blocks, the

ExtendSim for Mac OS package includes a file conversion utility which you can use to convert libraries and picture resource extensions from one operating system format to another. Other extensions that you build, such as QuickTime movies, DLLs, and Shared Libraries require more extensive conversion. Include files are, of course, already cross-platform compatible.

### **Model files**

The Windows version of ExtendSim can read ExtendSim model files created on the Mac OS as long as the name format is correct, as discussed above. The Mac OS version of ExtendSim can read ExtendSim model files created under Windows without file name modification (in fact, as discussed in the Note above, the MOX extension should not be removed.) When you open a model file that was created on another operating system, ExtendSim will notify you that it is converting the file from that system to the current one. Once you save the model file, it will be in the format of your current operating system.

If your models (including hierarchical blocks) use libraries that you have created yourself, and you have changed the name of those libraries, ExtendSim will not be able to locate the library. In this case, ExtendSim will ask you to find and select the correct library as described in “Searching for libraries and blocks” on page 491. Keeping all your libraries in the Libraries folder will make this search process easier. Saving the model will cause the new libraries to be used from then on.

For model files that have blocks that access text files (such as the Read block from the Value library), you may need to change the name of the text file that is being read to conform to platform requirements, as discussed above. Be sure to also change the name of the file in the Read block’s dialog to correspond to the new file name.

- ☞ The first time you run a model that has been transferred from one operating system to another, any Equation blocks in the model will recompile to the format of the new system at the beginning of the simulation run. Messages that report this process may appear too quickly for you to read.

### **Hierarchical blocks in libraries**

If you have a hierarchical block saved in a library *and* you have renamed any of the libraries of the blocks *inside* the hierarchical block (for example, to comply with Windows format), you need to update the hierarchical block’s information so that it can locate the renamed libraries. The easiest way to do this is to drag hierarchical blocks from their libraries, place them on a worksheet, and update their structure, as discussed below.

- ☞ This is only required for hierarchical blocks saved in libraries; hierarchical blocks saved only in a model get updated with the model.

When you add a hierarchical block from a library to a model worksheet, the hierarchical block causes ExtendSim to open the libraries of the blocks inside it. Since you have renamed those libraries, ExtendSim will not be able to locate them. In this case, ExtendSim will ask you to find and open the correct libraries. Note: keeping all your libraries in the Libraries folder will make this search process easier.

If you save the model worksheet that contains the hierarchical block, the location of the renamed libraries is saved for the model only. Before you close the model worksheet, you also need to update the hierarchical block’s library information. To do this, open the hierarchical block’s structure window and then close it, causing the hierarchical block’s Save dialog to appear. In the dialog, choose **Also save to library**. This process is described in “Summary of results of modifying hierarchical blocks” on page 550.



### Libraries

The libraries that come with your ExtendSim package are already formatted correctly for your operating system. However, if you build your own libraries, and want to transfer them to a computer running a different operating system, you must convert them to the appropriate operating system format. You do this *on the Mac OS computer* using the Libraries > Tools > MacWin Conversion command (See “MacWin Conversion (Mac OS only)” on page 697). This converts libraries to the specified operating system format, ensures that the file names are properly formatted, and so forth.

After the libraries have been converted to Windows or Mac OS format, physically transfer them to the target computer (that is, keep them on the Mac OS or transfer them to a Windows computer). Then recompile the libraries under the target computer’s operating system using the Library > Tools > Compile Open Library Windows command.

The MacWin Conversion converts libraries to either Windows or Mac OS format. After conversion, you must recompile the library on the target computer. When you do this, the library will compile to native code for the target system. For example, if you compile on a Windows computer, the library will be in native Windows mode. If you compile on a Power Mac OS computer, the library will be in native Power Mac OS mode.

### Blocks that use the equation functions


If you build blocks that use the equation functions, your code needs to detect if the model is being opened on a different platform. See the ExtendSim Developer Reference for more information.

### Extensions

Extensions are files (such as pictures and DLLs) that can be accessed by ExtendSim to fulfill specialized tasks. Like libraries, the extensions that come with your ExtendSim package are formatted correctly for your operating system. However, if you build your own extensions, and you want to transfer your extensions or blocks to a computer running a different operating system, you will need to do some conversion:

- **Pictures:** As discussed in the Developer Reference, ExtendSim for Windows accepts three kinds of pictures: WMF (Windows MetaFiles), BMP (Bitmap), and a Mac OS picture resource file that has been converted to Windows format. ExtendSim for the Mac OS accepts only picture resources. To convert Mac OS picture resource files to Windows format, use ExtendSim’s MacWin Converter utility on the Mac OS. To convert Windows pictures to Mac OS format, use a graphics conversion application.
- **Sounds:** Shareware utilities are available to convert Mac OS sound resources (SNDs) to Windows sound files (.WAV) and vice versa.
- **DLLs and Shared Libraries:** On Windows the DLL functions will search the ExtendSim Extensions folder for a DLL file. For the Mac OS those same calls will search for a Shared Library file.

For more information, see the Developer Reference.

 The following ModL constants return TRUE or FALSE depending on the platform: PLATFORMMAC; PLATFORMWINDOWS; PLATFORMPOWERPC. You can use these constants in an if statement to make your code be cross-platform capable. For example:

746 | **Cross-Platform Considerations**  
Transferring files between operating systems

```
if (PLATFORMWINDOWS)
    windows specific code
else if (PLATFORMMAC)
    macintosh specific code
```

# Index

## Symbols

- \_3D objectID property 126
  - ObjectID 463
- \_Animation property 200
- \_cost attribute 126, 227, 231
- \_Item priority property 122
- \_Item quantity property 125, 200
- \_rate attribute 126, 227, 231
- 3D Animation tab (Simulation Setup) 477, 521
- 3D Bank Line Advanced model 429
- 3D Bank Line Final model 416
- 3D Bank Line Start model 417
- 3D camera 397
- 3D Controller block 411, 449, 482
- 3D Options tab of 3D Controller block 482
- 3D Position (X, Y, Z) option 480
- 3D Scenery block 447, 483
- 3D tab (Options command) 449, 476, 692
- 3D Text block 447, 483
- 3D window outside application option 399, 476, 693

## A

- ABC (activity based costing) 224
- About ExtendSim command 714
- accumulate data 241, 565
  - with Holding Tank block 241
- accumulating conveyor
  - Convey Flow block 344
  - Convey Item block 188
- accumulation point 345
- Action menu (E3D Editor) 470, 485
- ActiveX 665
  - automation 665
  - COM (Component Object Model) 666
- activities
  - definition 91
  - scheduling 173
- activity based costing 224–236
- Activity block 724
  - Item Animation tab 479
  - multitasking 183
  - Preempt tab 178
  - preemption options 178

- processing items 164
- processing options 167
- Shutdown tab 179
- tutorial 101
- ad hoc experiments 570
- Add All To Report command 597, 710
- Add All To Trace command 620, 712
- Add Connection Line Animation command 709
- Add Debug Code to Open Library... command 696
- Add External Code in Open Library... command 696
- Add Named Connection Animation command 709
- Add Selected To Report command 597, 710
- Add Selected To Trace command 620, 712
- address (of a data structure) 660
- agent-based modeling 51
- Airline Security model 401
- Align command 562, 698
- allocate item availability 255
- Allow data table titles copying option 694
- Alt key 698, 707
- Alternate path option 491, 690
- anchor point 33
- Animate 3D block 483
- Animate Item block 554
- Animate Value block 554
- Animating Queue Contents model 141
- animation
  - 2D 551–557
  - 3D 390–486
    - Block Animation tab 410, 480
    - Change all items to option 553
    - Change item animation using property option 553
    - debugging with 557, 616, 618
    - Do not change item animation option 553
    - functions 556
    - Item Animation tab 478, 552
    - pictures (adding) 556
- animation (2D) 551–557
  - Animate Item block 554
  - Animate Value block 554
  - animating item movement 552
  - faster button 551
  - for Item library blocks 552
  - for Rate library blocks 370
  - hierarchical block's icon 554

- Item Animation tab 552
- object 555
- pictures (selecting) 552
- Show 2D Animation command 709
- slower button 551
- animation (3D) 390–486
  - 3D Animation tab (Simulation Setup command) 521
  - and hierarchical blocks 474
  - Block Animation tab 480
  - Buffered mode 477
  - changing the resolution of the window 396
  - collidable objects 479
  - Concurrent mode 409, 477
  - controlling 482
  - conversion ratios 521
  - custom paths 467
  - displaying text 483
  - environmental effects 448
  - executing actions 483
  - features 391
  - internal animation 412
  - introduction 390
  - Item Animation tab 478, 552
  - linking objects to block positions 413
  - mode selection 521
  - modes 400
  - mounting objects 412
  - objects (creating and using) 442–463
  - objects to represent blocks 410
  - objects to represent items 409
  - opening the E3D window 396
  - performance considerations 475
  - prerequisites for 393
  - QuickView mode 477
  - QuickView mode vs Concurrent/Buffered mode 400
  - rotation of objects 412
  - running a model with 402
  - saving changes after modifying objects 459
  - scenery 411, 446, 483
  - Show 3D Animation command 709
  - speed controls 397
  - Terrain modes 438
  - Torque Game Engine 392
  - Transport Animation tab 481
  - tutorial 406
  - using an equation-type block 474
  - World modes 436
- Animation 2D-3D library 488, 554
- Animation Faster button 533, 551
- Animation library (legacy) 490
- animation object for 2D animation 555
- Animation Slower button 533, 551, 618
- Antithetic random variates option 520
- Append New Field command 642, 703
- Append New Records command 703
- application areas
  - continuous modeling 61, 72
  - discrete event modeling 96
  - discrete rate modeling 266
- application messages 534
  - link alerts 534
- arguments for distributions 606
- Array connector 498, 545
- arrays
  - dynamic 654
  - global 652–654
- arrival times 111–115
- Arrivals and Activity model 222
- Arrows option (connection lines) 558
- ASCII files
  - text files 663
- assumptions, changing 19
- attributes 115–122
  - \_cost 227, 231
  - \_rate 227, 231
  - arrays 121
  - costing 231
  - DB address 116, 119
  - deleting 255
  - example of use 106
  - for setting processing time 170
  - introduction to 94
  - managing 255
  - removing 214
  - renaming 255
  - resource blocks 214
  - string 116, 119, 255
  - string, creating a 106
  - stripping 214
  - system 231
  - to hold cumulative values 241
  - types 116
  - using 117
  - value 116, 119
- Attributes for Routing model 156
- attribute-sorted queue 129
- AutoCAD files 682
- Automatic search option 491, 495, 690
- automation (ActiveX) 665

Autoscale tools 591  
 autoscaling 35  
 Autostep options 518  
 axis (changing) 35

## B

backup files 681, 694  
 BAK files 681, 694  
 balking 131  
 Balking model 131  
 Bar Chart block 594  
 Batch and Unbatch Variable model 203  
 Batch block 195, 725
 

- batching items 194
- Item Animation tab 479
- Mount objects option 479
- options for 3D animation 479

 batch into one item 195  
 Batch means 565  
 Batch Mode Merge model 321  
 Batch on Demand model 199  
 batch size 203  
 batch/unbatch mode 321  
 batched value 204  
 batching 194–201
 

- \_Animation property 200
- \_Item quantity property 200
- attribute options 200
- batch size 195
- Batch tab 195
- delay kit at... 201
- item properties 199
- matching items 197
- on demand 199
- Options tab 195
- priority options 200
- Properties tab 200
- resources with items 213
- simple 197
- using attributes 197
- variable number of items 198, 203

 Batching and Unbatching model 202  
 Batching Variable model 198  
 beta distribution 607  
 Bias block 361, 732  
 bias order 327, 361
 

- displaying 366
- effective rate calculations 361
- in Merge and Diverge blocks 362
- setting 363

 biasing flow 360  
 binomial distribution 607  
 bitmap pictures 745  
 Black Connections command 558  
 block 496–502
 

- adding to a model 20, 26
- adding to a model using the Library Window 671
- arranging in libraries 495
- bad 502
- categories (Value library) 716
- category 690
- category (finding) 617, 685
- code management (CM) 672
- compiling 494, 696, 705
- connecting blocks 28
- connection methods 32
- connectors 15
- continuous 60
- copying to another library 502
- corrupted 502
- custom block models 79
- custom continuous blocks 60
- customizing 496
- debugging code 672
- decision type 256
- deleting 20
- dialog items (new) 706
- dialogs 16
- duplicating 39
- duplicating in libraries 502
- equation-based 601–604
- for data management and exchange 659
- help 8
- Help block (creating) 514
- Help button 16
- help text 496
- icons 15, 496
- information about 683
- labels 15, 740
- labels (finding) 617, 685
- labels (showing) 699
- linked to 3D object 413
- messages 260, 387, 535
- modifying 704
- moving 27
- MYO (make your own) 92
- names 15, 617, 740
- names (finding) 685

- new 704
- number (global) 685
- numbers (finding) 617, 685
- numbers (showing) 699
- passing type 256
- profiling code 712
- programming 60
- property aware 111
- random number generator 605
- red border around icon 672
- removing block from model 20
- removing from libraries 502
- renaming 502, 704
- residence type 256
- searches 492
- Sensor 373
- statistics 564
- storing in libraries 26
- structure 704
- structure window opens in front 691
- submenus for Item library 724
- submenus for Value library 716
- substituting one for another 492
- types 256
- types (mixing) 87
- Block Animation tab 480
  - selecting an object to represent the block 410
  - waypoints to represent blocks 410
- block messages
  - in discrete event models 260
  - in discrete rate models 387
- block number
  - global 617
  - local 501
  - showing 699
- block type table 256
- block units 297
  - defining 299
  - introduction 271
- blocking 154, 166, 183
  - definition 131
  - due to shutdown 183
  - in a Valve 365
  - in the Select Item Out block 150
  - using buffers to prevent 162
- blocking of items 166
- BlockNumber property of 3D objects 463
- blue frame around parameter field 632
- BMP (Bitmap) 556

- Boids model 54
- Boolean checkbox option 649
- Border command 695
- Border Thickness command 562
- BPR library (legacy) 490
- Breakout model 54
- Brush menu (E3D Editor) 470, 485
- Bucket Elevator 1 model 356
- Bucket Elevator 2 model 357
- Buffered mode 477, 521
- Buffering Operations model 159
- building a model 25
- buttons
  - Database (ExtendSim) 714
  - Open Notebook command 19
  - Pause/Resume 20
  - Run Simulation 18, 32
- Buttons block 507, 737

## C

- CAD files (importing) 682
- Calendar command 713
- Calendar date definition option 517
- calendar dates 217
  - defining non-calendar date 517
  - enabling 528
- calendar format example 217
- calendar of events 259
- camera 397
- Camera menu (E3D Editor) 485
- capacity
  - full and not-full 291
  - in a Tank block 291
  - in Convey Flow block 293
  - in Interchange block 292
  - maximized in Convey Flow block 293
  - Rate library blocks 291
- Car Wash model 100
- Catch Flow and Diverge model 332
- Catch Flow and Throw Flow model 331
- Catch Flow block 329, 732
- Catch Item block 148, 156, 728
  - groups 149
  - routing items 144
- Catch Value block 720
- categories of blocks 690
  - Item library 724

- Utilities library 736
- Value library 716
- Cauchy distribution 607
- central tendency of distribution 606
- Change all items to option 478, 553
- Change item animation using property option 478, 553
- Change Model Style command 561, 699
- Change Rate model 232
- Change Units block 281, 283, 732
  - changing flow units 300
  - relational constraints 307
- Changeover Quantity Goal model 336
- Changeover With Only Goals model 339
- changing
  - assumptions 19
  - libraries 690
  - parameters 20
  - text 686
- Check blocks for duplicate random number seeds option 520
- Check In License command 683
- Check Out License command 683
- Chi Square distribution 607
- Child popup selector (for database) 644
- City Planning model 77
- Clear command 676, 684
- Clear Database command 647
- Clear Statistics block 239, 566, 721
- clearing data from plotters 596
- Clearing Statistics model 239
- client application 658, 665
  - for DDE link 636
- Clipboard 674, 684, 688
- Clone layer tool 39
- Clone Selected Tables to Tab command 647, 702
- clones 504–506
  - deleting 505
  - finding original dialog item 505
  - introduction 38
  - moving 505
  - resizing 505
  - unlinked 506
- cloning dialog items 38, 504
- Close command 680
- Close Library command 695
- closed systems 95, 216
  - partially closed 96
- CM (code management) 672, 705
- code management (CM) 672, 705
- Collidable option 479, 480
- collision of 3D objects 462
- color (HSV) 562
- Color Connections command 558
- Color popup menu 562
- column index 662
- column separators 681
- COM (Component Object Model) 666
- Combine Priority Sensing model 385
- Combine Sensing Priority model 386
- Combined Rule model 138
- combining resources with cost accumulators 234
- Command block 661, 720
- Competing Requests for Flow model 327
- Compile Block command 494, 705
- Compile Open Library Windows command 494, 696
- Compile Selected Blocks command 494, 696
- compiling
  - blocks 494, 705
  - libraries 696
- Concurrent mode 409, 477, 521
- concurrent users 683
- conditional routing 157
- Conditional Routing model 158
- confidence interval 567
- confidence level 567
- connection lines 557–559
  - changing formats 559
  - Flow connection line types 559
  - Item line types 559
  - making a multi-segment connection 33
  - making a right angle connection 28
  - making a straight connection 32
  - selecting 559
  - types of 28, 558
  - Value line types 559
  - View Using Defaults option 558
- Connection Lines command 557, 698
- connections 557–560
  - anchor points 33
  - arrows option 558
  - checking 618
  - color options 558
  - connection lines 557–559
  - daisy-chaining 87
  - dashed option 558

- default types of lines 689
- definition 15
- deleting 670
- flow type 270
- hiding 560, 698
- incomplete 618
- line types 558
- lines 16
- multi-segment 33
- named 33, 560
- right angle as default style 689
- right angle option 558
- showing 560
- straight 32
- styles 558
- to multiple item inputs 248
- to multiple value inputs 87
- types 28
- connector
  - Array 498
  - arrays of connectors 498
  - common for Rate library blocks 368
  - common Item library connectors 257
  - compatible 500
  - connecting 28
  - definition 15
  - description 28
  - diamond 545
  - Flow 270, 498
  - hiding 698
  - Item 95, 498
  - item index 95
  - messages of item connectors 260, 262
  - messages of value connectors 261
  - names 546
  - text object 546
  - tool tips 8, 689
  - types 498, 545
  - Universal 498
  - User Defined 498, 545
  - Value 87, 498
  - variable 498
- connectors 497–500
  - BatchQuantityIn 195
  - D input 173
  - demand 158, 174, 176, 196, 199, 247
  - down 182
  - F 183
  - G 338
  - GS 337
  - hiding 560
  - PE 178
  - preempt 353
  - PT 241
  - R 309
  - S (sensor) 345
  - S (status) 365
  - SD input 179
  - SD output 180
  - select 250
  - sensor 247
  - showing 560
  - start 176
  - start (on Valve) 341
  - StatusIn 219
  - StatusOut 219
  - TR 216
- conserving resources 201
- Constant block 718
- constant distribution 607
- constant values
  - definition 57
- constraining resources 209
- constraints (discrete rate)
  - critical 307
  - impact on effective rates 315
  - relational 307
  - varying over time 278
- constraints (optimization) 585
  - adding 580
  - equations for 582
  - global 581, 586
  - individual 581, 585
- contents
  - empty and not-empty 294
  - Rate library blocks 293
- context-sensitive help 8
- Continue command 708
- Continue sequence of random numbers option 519
- Continue Simulation command 708
- continuous
  - application areas 61
  - blocks 60
  - libraries 60
  - models 60
  - programming 60
  - tab for setting steps 518
- continuous blocks
  - in discrete event models 250



- continuous modeling 60
  - definition 44
  - feedback 85
  - messaging 535
  - step size 83
  - timing 82, 526
- Continuous tab 518
- control panel 508
- Controlling Shifts model 222
- controls 509
  - Meter 510
  - Slider 509
  - Switch 510
- Controls command 509, 698
- conversion ratios (3D Animation tab) 478
- Convert Library to RunTime Format command 697
- Convey Flow block 342–346, 732
  - accumulate-fill empty segments 344
  - accumulate-maximum density 344
  - accumulation point 345
  - animation 373
  - behavior 342
  - capacity 293
  - critical constraints 311
  - dialog settings 343
  - Indicators tab 346
  - initial contents 295
  - Initialize tab 295
  - length units 298
  - modeling with 282
  - non-accumulating 344
  - Sensors tab 345
  - when to avoid using 346
- Convey Item block 188, 724
  - 3D animation 481
  - accumulating 188
  - capacity 414
  - distance ratio 188
  - from and to locations 187, 481
  - item length 413
  - move time 186
  - non-accumulating 188
  - processing items 164
  - speed and calculated distance 186
  - speed and distance 186
  - Stretch 3D object to conveyor's length option 480
  - using for 3D animation 413
- Copy (or Duplicate) Database command 646
- Copy command 675, 684
  - for DDE linking 636
- Copy Plot command 596
- Copy Tables command 647
- copy/paste 625
- copying 674–676
  - data 625
  - dialogs 675
  - from ExtendSim to other applications 675
  - notebooks 675
  - plotter data 596
  - plotters 675
  - text to Notebooks 538
  - within ExtendSim 674
- cost accumulator 225
- cost array 234
- Cost By Item block 233, 726
- cost equation 584
- cost rates 226
- Cost Stats block 234, 567, 726
- costs 224–236
  - assigning 228
  - calculating 235
  - direct materials 227
  - drivers 224
  - fixed 224, 234
  - per time unit 228
  - per use 228
  - storage 227
  - variable 224
  - variable cost 234
  - waiting 227
- Count Blocks block 737
- Create block 101, 125, 728
  - arguments of the distributions 251
  - generating items 110
  - Item Animation tab 479
  - options for 3D animation 479
  - setting processing time 164
- Create Database dialog 641
- Create new 3D animation object option 479
- Create/Edit Dynamic Link command 633, 686
- Creator Tree pane of WEC 438
- critical constraints 307, 379
  - Convey Flow block 311
  - defining 308
  - determining 315
  - Diverge block 311
  - meeting requirements 312
  - Merge block 311

- cross-platform compatibility 743
- Cumulative Time model 172
- cursor (drawing pen) 28
- Custom Blocks library 489
- Custom order 86
- Custom statistical method 565
- Custom Time model 170
- Cut command 684
- Cut Databases command 646
- Cut Tables command 647
- cycle time 254
  - tracking from other than origin 255
  - tracking item from origin 254
- Cycle Time 1 model 254
- Cycle Time 2 model 255
- cycling
  - fixed number of items 175
  - fixed period of time 177

## D

- D input connector 173
- daisy-chaining 87
- dashboard 506
- data
  - blocks for data access 660
  - clearing from plotters 596
  - copying/pasting 625
  - databases for storing 638
  - editing in database 648
  - exchanging (ActiveX) 665
  - exchanging with external applications 657
  - global arrays for storing 652
  - import/export methods 626
  - importing/exporting 626
  - management 624–668
  - management and exchange using blocks 659
  - managing 638
  - raw historical 586
  - reading 628
  - repository 638
  - sharing with external applications 636
  - source (organizing) 661
  - source indexing 661
  - structures (addressing) 660
  - structures (communicating with) 659
  - writing 628
- data accumulation 241
- Data address option 649

- data exchange
  - copy/paste 625
  - user interfaces for 624
- Data Fitter block 737
- data fitting 607
- Data Import Export block 661, 716
- Data Init block 654, 661, 716
- data organization 19
- Data Source Create block 653, 661, 716
- Data Specs block 654, 661, 717
- data structures 659
  - addressing 660
  - internal (linking to) 686
- data tables
  - linking to an ExtendSim database table 633
  - linking to internal structures 632
  - printing 673, 683
  - row and column titles (printing) 694
- database
  - external (exchanging data with) 658
  - external (opening links to) 687
  - Industry 647
  - internal (ExtendSim) 686
- Database (ExtendSim) 638–651
  - accessing with Read and Write blocks 645
  - advantages of using internal databases 639
  - commands 700
  - copying, renaming, or deleting 646
  - creating a new database 641
  - creation methods (overview) 640
  - database management 646
  - Database Random Distribution dialog 650
  - database window 700
  - DB address attribute 116
  - editing data 648
  - exporting 647
  - ExtendSim DB Add-In 650
  - Field Properties dialog 648
  - field type popup menu 649
  - formatting options 648
  - importing 647
  - importing an Industry database 647
  - index number 641
  - linking to 686
  - names 641
  - new database 641
  - number of databases per model 740
  - opening linked blocks 687
  - parent/child relationships 643

- random number seed 520
- random numbers for cells 650
- toolbar buttons 714
- database list
  - accessing 646
  - mode in Navigator 671
- Database List command 646, 713
- Database List mode 671
- Database menu 700
- Database Random Distribution dialog 650
- Database table
  - linking to 630
- database window
  - opening 646, 700
  - opening with Navigator 671
  - structure mode 641
  - viewer mode 642
- Date/Time option 649
- Day Shift Capacity Change model 220
- DB address attribute 116
  - defining 118
- DDE (Dynamic Data Exchange) 636–638
- DDE (dynamic data exchange) 687
  - updating Excel's remote references 638
- DDE linking 667
- DDL (dynamic data linking) 629–636
- debugger
  - Add Debug Code... 696
  - Continue 708
  - Generate Debugging Info 705
  - open breakpoints window 707
  - open debugger window 707
  - set breakpoints 707
  - Step Into 708
  - Step Out 708
  - Step Over 708
- debugging 614
  - blocks for 615
  - hints for debugging 614
  - red border around block icon 672
  - source code debugger 618
  - using animation 557
- Debugging command 523, 711
- Decimals option 649
- Decision block 158, 718
  - controlling the flow of items 144
- decision type blocks 256
- Default connection line style is right angle option 689
- Default model path option 689
- default view 16
- default view reverse 16
- Define conversion ratios option 478, 521
- delay kit... 201
- Delete DDE Link command 687
- Delete Include File command 707
- Delete Selected Records command 685
- delimiters 681
- delta connector 173
- delta time 83
  - definition 82
  - determining 83
  - other than 1 83
  - setting 83, 518
- DeltaTime variable 518
- demand connector 158, 174, 176, 247
- demand scarcity 324
- Demand Sensing Mode Diverge model 326
- density (maximum) 343
- Design Mode command 688
- deterministic models 58
- Develop menu 704
- dialog items
  - cloning 504
  - finding clones 505
- dialog window
  - printing 674
- dialogs 501
  - copying 675
  - open when running 501
  - opening 16, 501
  - printing 672, 673
  - title bar 501
- Dialogs report 596, 710
- discrete event
  - processes 91
  - systems 91
- Discrete Event library (legacy) 490
- discrete event modeling 94, 224
  - activity-based costing 224–236
  - animation (2D) 552
  - application areas 96
  - architecture 93
  - batching items 194
  - blocks for 92
  - closed and open systems 95
  - continuous blocks in 87, 250

- cycle time 254
- definition 44
- event scheduling 258
- events 94
- Executive block 255
- item generation 111–115
- Item library 92
- item movement 246
- items 93
- layout 93
- messaging 260, 535
- overview 93
- posting events 259
- preprocessing 249
- processing items 164
- queueing 128
- resources 105, 208
- restricting items 249
- routing items 144
- running 100
- statistics 238
- terminology 93
- time-based parameters 251
- timing 526
- tips 246
- travel time 248
- tutorial 100
- values 94
- zero time events 259
- discrete rate modeling 297
  - 2D animation of blocks 370
  - application areas 266
  - bias order 327, 361
  - bias order when merging/diverging 362
  - biasing flow 360
  - building a model 274
  - capacity 291
  - compared to other methods 267
  - competing requests for flow 327
  - contents 294
  - declaring flow units 277
  - defining bias order 328
  - definition 45
  - downstream demand 322, 383
  - dynamic constraint 278
  - event messages 387
  - Executive block flow messages 388
  - fixed flow rules 319
  - flow connector messages 388
  - flow rules 306
    - heterogeneous flows 267
    - homogeneous flows 267
    - hysteresis 341
    - indicators of flow 295
    - inflow branches 319
    - introduction 266
    - item connector messages 388
    - LP area 306, 377
    - LP technology 376
    - messages 387
    - messaging 386
    - outflow branches 319
    - overview 268
    - rate block flow messages 388
    - rate sections 305
    - rates (introduction) 271
    - throw and catch 329
    - time units 298
    - units and unit groups 271
    - upstream supply 322, 383
    - value connector messages 387
- Display Value block 532, 615, 720
- distance
  - metric 689
- Distance (m) option 479
- distance ratio (3D animation) 521
- distribute properties option 204
- distribution
  - arguments 251
  - central tendency 606
  - list of distributions 607–610
  - location argument 606
  - shape argument 606
  - shape of exponential 112
  - skewness 606
  - spread 606
  - theoretical 606
  - varying the arguments 251
- distributional mode 324
  - bias order determination 366
- Distributional Mode Diverge model 324
- Distributional Mode Merge model 325
- distributions
  - beta 607
  - binomial 607
  - Cauchy 607
  - Chi Square 607
  - constant 607
  - empirical 606, 607
  - Erlang 607

- exponential 112, 607
  - Extreme Value Type 1A 608
  - Extreme Value Type 1B 608
  - gamma 608
  - geometric 608
  - hyperexponential 608
  - Hypergeometric 608
  - inverse gaussian 608
  - Inverse Weibull 608
  - Johnson SB 608
  - Johnson SU 608
  - Laplace 608
  - Logarithmic 608
  - Logistic 609
  - loglogistic 609
  - lognormal 609
  - negative binomial 609
  - normal 609
  - Pareto 609
  - Pearson type V 609
  - Pearson type VI 609
  - Poisson 609
  - Power Function 609
  - Rayleigh 609
  - triangular 609
  - uniform integer 610
  - uniform real 610
  - user-defined 606
  - Weibull 610
  - Diverge block 732
    - bias order 327, 362
    - bias order table 364
    - critical constraints 311
    - displaying bias order 366
    - features 327
    - fixed rule modes 362
    - modes (mixed in model) 385
    - modes (summarized) 319
    - non-fixed rule modes 363
    - requirements for critical constraint 313
    - setting bias order 363
    - throwing flow 329
  - DLLs 513, 668, 745
  - Do not change item animation option 478, 553
  - down connector 182
  - Downloads and Updates command 713
  - downstream demand 322
    - cautions when determining 383
    - definition 383
    - discrepancy with upstream supply 324
    - scarcity of demand 324
  - drag and drop editing 539
  - Draw Right Angle Line tool 561
  - drawing pen cursor 28
  - drawing tools 561
    - colors 562
    - patterns 562
  - drivers (cross-platform) 745
  - drop shadows in hierarchical blocks 689
  - Drug Ingestion model 73
  - dt (delta time) 82, 526
    - stepsize 83
  - Duplicate command 34, 39, 685
  - duration goal 338
  - Duration Goal model 339
  - duration of simulation 82, 526
  - DXF files 682
  - dynamic arrays 654
    - linking to 687
    - number per block 740
  - Dynamic Data Exchange (DDE) 636–638, 666, 687
    - Show DDE Links command 637
  - dynamic data linking (DDL) 629–636
    - dialog item to database table 645
    - finding linked dialogs 635
    - link dialog 630
    - link dialog (description) 634
    - linking a parameter to a global array 631
    - linking parameter to an ExtendSim database 630
    - linking to a database table 633
    - linking to a global array 633
    - to a global array 633
  - Dynamic Link Libraries (DLLs) 513, 662, 668, 745
  - dynamic links to internal data structures 686
  - dynamic values 676
  - dynamically changing parameters 676
- ## E
- E3D Editor 433–440
    - creating scenery with 447
    - Gizmo 435
    - menu commands 484
    - mode categories 435
    - Terrain modes 438
    - World Editor Inspector (WEI) mode 434
    - World modes 436
  - E3D environment
    - controlling 393

- opening the E3D window 396
- overview 391
- selecting the environment file 521
- E3D window
  - 3D Animation tab 477
  - 3D window outside application 693
  - associated model 398
  - camera for E3D window 397
  - closing 399
  - conversion ratios 478
  - creating paths 467
  - definition 391
  - E3D Editor 433
  - environment file 432, 478, 521
  - exploring 396–400
  - ExtendSim icon 397
  - foot prints 693
  - interface controls 396
  - level of detail 693
  - MiniMap 397
  - MiniMap option 693
  - mouse-look toggle 399
  - navigation within the window 398
  - opening 396, 399, 412, 475
  - outside the application window 399
  - scenery 411
  - shadows 692
  - sounds in 462, 692
  - title bar 397
  - vehicle trails 693
- E3D Window command 399, 476, 713
- Each block defines its own bias order 328
- Each value unique option 649
- Edit menu 684
- Edit menu (E3D Editor) 485
- Edit or Delete Tab command 706
- Editor command 397
- effective rate 303
  - defining a zero effective rate 365
  - impacted by constraints 315
- Electronics library 488
- embedded objects
  - behavior 688
  - commands 688
- empirical distribution 607
- empty and not-empty 294
- Enable animation of 3D object option 412, 480
- End time 25, 530
  - manually controlling 255
- End time option 517
- ENDIF 707
- Enter Selection command 686
- environment file 432–433
  - Extend3Dl.mis 432
  - mis (mission) file 432
  - saving 459
  - selecting 521
  - ter (terrain) file 432
- Environment file option 478, 521
- Environment objects 438
- Equation block 512, 601, 718, 744
  - example of use 68
  - input variables 602
  - output variables 603
- Equation(I) block 241, 512, 601, 726
  - example of use 253
  - input variables 602
  - output variables 603
  - properties of items 110
- equation-based blocks 601–604
- equations
  - blocks for 600
  - in 3D animation 474
- Erlang distribution 607
- Euler integration 85, 611
- event calendar 259
- event scheduling 258
- Event Scheduling model 259
- events 16, 91, 94, 258, 387
  - calendar 259
  - current 260
  - future 260
  - generating 258
  - messages in discrete event models 260
  - posting 259
  - zero time 259
- Excel 663
  - and DDE linking with ExtendSim 637
  - exchanging data with 658
  - ExtendSim DB Add-In 650
  - ExtendSim DB command in 651
  - updating remote references for DDE linking 638
- Excel Add-In 701
- exchanging data
  - dynamic data linking (DDL) 629–636
  - piece-by-piece 629
- Executive block 729
  - advanced options 366

- Attributes tab 106
  - bias information for calculation 380
  - bias order definitions 363
  - bias order determination 366
  - description 255
  - Discrete Rate tab 364
  - global options 364
  - in discrete rate model 270
  - infinite rate 364
  - infinite rate setting 304
  - introduction 93
  - properties of items 110
  - update flow status 365
  - Valve options 365
  - zero effective rate 365
  - Executive block flow messages
    - in discrete rate models 388
  - Exit block 102, 728
    - removing items 110
  - Exit command 684
  - Explicit Ordering model 155
  - explicit shutdown 183
  - Explicit Shutdown model 183
  - exponential distribution 112, 607
  - Export Data command 628, 664
  - Export Data Table command 681
  - Export Database command 647, 701
  - Export Selected Tables command 647, 701
  - exporting data 626–628
    - methods 626
  - Extend3D.mis file 432, 478
  - ExtendSim
    - architecture 5
    - capabilities for modeling 4
    - databases 638–651
    - Excel Add-In 701
    - Help 8, 713
    - icon in E3D window 397
    - launching an alternate Extend application 681
    - legacy libraries 488, 489
    - levels of use 5
    - libraries 488
    - LT-RunTime version 678
    - messaging in models 533
    - network licenses 683
    - Options command 688
    - source code debugger 618
    - support 713
    - updates 713
    - upper limits 740
    - what's new 713
  - ExtendSim DB Add-In 650
  - ExtendSim Help command 713
  - ExtendSim menu 680
  - ExtendSim Product Line command 713
  - extensions (converting) 745
  - External Source Code command 705
  - Extreme Value Type 1A distribution 608
  - Extreme Value Type 1B distribution 608
- ## F
- F connector 183
  - feedback 85
    - delays 85
    - loops 85
  - Feedback block 738
  - Field name option 648
  - Field Name popup menu 631
  - Field Properties dialog 648
  - Field type option of Field Properties dialog 648
  - Field type popup menu (for database) 649
  - fields (database) 632
    - adding to a table 642
    - field types 642
    - for DB address attribute 116
    - in database table 639
    - managing 648
    - parent/child 643
  - fields (parameter)
    - blue frame 632
    - linking to a database or global array 630
    - outlined in green 630
    - outlined in red 630
    - outlined in yellow 630
  - FIFO queue 129
  - file conversion 743
  - file format
    - of original graphic 562
    - pictures 556, 745
  - File menu 680
  - File menu (E3D Editor) 484
  - file names 742
  - File Open command 15
  - files
    - backup 681, 694
    - exporting data 681
    - importing CAD 682

- importing data 681
  - importing DXF 682
  - opening most recent 684
  - sharing 677
  - transferring 743
  - filter options
    - block type filter 330
    - group filter 330
    - only connected blocks filter 330
  - Final messages, in Link dialog 635
  - Find Again command 686
  - Find and Replace block 616, 737
  - Find command 617, 685
  - finding 617
  - First run...Every run option 649
  - Fish Pond model 80
  - fixed costs 224, 234
  - fixed flow rule 319, 322
  - Fixed Items model 175
  - fixed rule modes 362
  - Fixed Time model 177
  - Flip Horizontally command 562, 699
  - Flip Vertically command 562, 699
  - flow
    - accumulation point 345
    - biasing 360
    - catching 329
    - competing requests for 327
    - connectors 270
    - controlled by items 348
    - controlling flow 334
    - controlling items 349
    - definition 268
    - delaying 334
    - indicators 295
    - Interchange block 352
    - levels 295
    - LP technology 376
    - managing units 366
    - maximized 268
    - options when goal is off 335
    - preference 361
    - ranges 295
    - rate 268
    - rules 306
    - status 365
    - streams (merging and diverging) 318
    - throwing 329
    - units 271, 297
    - flow connector messages
      - in discrete rate models 388
    - Flow connectors 498, 545
    - Flow Control tab 334
      - goals 335
      - hysteresis 341
    - Flow Controls Item model 350
    - Flow library (legacy) 490
    - Flow order 86
    - flow rates
      - effective 303
      - infinite 304
      - maximum 303
      - overview 303
      - types 303
    - flow rules
      - critical constraints 307
      - fixed 319, 322
      - information to Executive 379
      - relational constraints 307
    - flow units 271
      - changing 300
      - declaring 277, 299
      - group selector button 299
      - managing 299, 366
    - fonts for ModL code 691
    - Foot prints & vehicle trails option 449, 476, 693
    - footers 674
    - formats
      - data sources 661
      - for database fields 642, 648
      - of embedded objects 655
      - of scaled objects 458
    - From and To locations options 481
    - FTP (File Transfer Protocol) 667
    - full and not-full 291
    - functions (calling from blocks) 600
- ## G
- G (goal) connector 338
  - Game of Life model 52
  - gamma distribution 608
  - Gantt Chart block 594
  - Gate block 174, 249, 728
    - controlling the flow of items 144
  - Generate Debugging Info command 705
  - Generate Report command 664, 710
  - Generate Trace command 620, 712



generating events 258  
 generating items 111–115  
 Generic library (legacy) 490  
 Generic time unit 526  
 geometric distribution 608  
 Get block 120, 726  
     properties of items 110  
 Get distance from 3D path length button 429  
 Get Info command 683  
 Gizmo 435, 452  
 global arrays 652–654  
     advantages of using 652  
     creating and using 653  
     data types 652  
     interacting with 654  
     linking a parameter to a global array 631  
     linking to 686  
     linking to a data table 633  
     methods for creating 652  
     opening links to 687  
     populating with data 654  
 global block numbers 617  
 global time units  
     converting from local time units 517  
     definition 526  
     setting 25  
 Global time units option 517  
 Go To Function/Message Handler command 707  
 Go To Line command 707  
 goal  
     duration 338  
     options when off 335  
     quantity 335  
     starting 337  
     status 337  
 goal seeking 573  
 graphics 561  
     file formats 562  
 graphs 18  
 green background on database fields 643  
 green parameter fields 630  
 grid  
     icon 699  
     model worksheet 699  
     notebooks 699  
 Grid density  
     tool for plotters 591  
 groups

    for Catch Item block 149  
 GS connector 337

## H

headers 674  
 help 713  
     blocks 8  
     getting technical support 8  
     printing the text 672  
     technical support 8  
     tool tips for connectors 8  
     user forums 8  
 Help block 514  
 Help button 16  
 Help menu 8, 713  
 Hide Connections command 698  
 Hide Connectors command 698  
 hierarchical blocks 540–551  
     animation of icon 554  
     cautions when using 550  
     changing 548  
     characteristics 541  
     commands for 698  
     connecting 547  
     connectors 545  
     creating 541  
     creating a new 543  
     drop shadows 689  
     icon, modifying 545  
     introduction 36  
     library (converting) 744  
     library (saving in) 547  
     locking 678  
     Make Selection Hierarchical command 543  
     making a selection into a 542  
     modifying 548  
     New Hierarchical Block command 543  
     password protecting 678  
     pictures, adding 496  
     printing 672  
     printing by level 673  
     Rename Block command 549  
     renaming 704  
     Save Block to Library As command 548  
     saving 547  
     structure window 541, 548  
     submodel pane 541  
     submodels 545  
 hierarchy 540–551

- commands 698
    - introduction 36
    - physical 542
    - pure 542
    - uses 540
  - Histogram block 594
  - History block 239, 615, 726
  - History model 240
  - holding items 246
  - Holding Tank block 21, 717
    - accumulating values 252
    - integration in 610
    - summation in 610
  - HSV 562
  - hyperexponential distribution 608
  - Hypergeometric distribution 608
  - hysteresis 176
    - in discrete event models 175
    - in discrete rate models 341
  - Hysteresis model 341
- I**
- icon
    - changing the appearance (hierarchical block) 549
    - custom 496
    - grid 699
    - views 16, 496
  - IFDEF 707
  - IFNDEF 707
  - Imagine That Home Page command 714
  - Imagine That, Inc. 8
  - Import Data command 627, 664
  - Import Data Table command 681
  - Import DXF File command 682
  - Import New Database command 647, 700
  - Import Tables command 647, 701
  - importing data 626–628
    - methods 626
  - Include additional block information option 689
  - include files 706
  - indexing
    - table by data source type 662
  - indexing (data source) 661
  - indicators 295, 346
    - in Bucket Elevator 2 model 357
  - Indicators tab 296, 346
  - Industry database 647
  - infinite checkbox in Tank block 277
  - infinite rate 304, 364
  - inflow branch 319
  - Information block 254, 567, 615, 726
  - Init messages, in Link dialog 635
  - initial bias 531
  - initial conditions 530
  - initial contents
    - Rate library blocks 293
  - initial maximum rate 309
  - Initialize every record in this field to option 649
  - Initializing and Viewing model 140
  - Input Line Balancing model 147
  - input variables in equation-based blocks 602
  - Insert New Field command 703
  - Insert New Records command 704
  - Insert Object command 688
  - Inspector pane of the WEI 437
  - Integrate block 719
  - integration
    - definition 85
    - in the Holding Tank block 610
  - interactive simulation 16, 501, 509, 510
    - importing/exporting data 626
  - interarrival time 111
  - Interchange block 284, 352, 732
    - animation 371
    - behavioral rules 352
    - block units 297
    - capacity 292
    - critical constraints 310
    - initial contents 294
    - maximum rate 303
    - modes 354
    - preemption 353
    - release options 353
    - Tank is separate... 292, 295, 355
    - Tank only exists... 292, 294, 354
  - interface 506
    - buttons 507
    - On/Off Switch block 508
  - interface controls (in E3D window) 396
  - Interiors category of WEC 438
  - internet access 659
  - Interprocess communication (IPC)
    - definition 657
  - Inventory Management model 76
  - Inverse Gaussian distribution 608

- Inverse Weibull distribution 608
  - Item Animation tab 478, 552
  - item connector messages
    - in discrete event models 262
    - in discrete rate models 388
  - Item connectors 498, 545
  - Item Controls Flow model 349
  - item index 95
  - Item library 488, 724–729
    - 3D animation in 390
    - and continuous models 500
    - connectors 257
    - description 92
    - moving items 246
    - pitfalls to avoid 247
  - Item Messages block 616, 617, 736
  - item properties 94
  - Item Templates library 489
  - items
    - allocate availability 255
    - as cost accumulators 225
    - attributes 115–122
    - attributes (overview) 94
    - attributes example 106
    - balking 131
    - batching 145, 194
    - batching properties when unbatching 204
    - blocking 131, 154, 166
    - connector 95
    - controlled by flow 349
    - controlling flow 348
    - cost accumulators 225
    - costing 225
    - definition 93
    - delay time 167
    - distribute properties when unbatching 204
    - generating 111–115
    - holding 246
    - index 95, 126
    - Interchange block 352
    - jockeying 132
    - joining 145, 194
    - matching 197
    - merging streams 145
    - moving 246
    - parallel processing 149
    - predicting the path of 151
    - preempting 178
    - preserving properties when unbatching 204
    - priority (overview) 94
    - processing 164–191
    - processing by type 161
    - processing time 167
    - properties 94, 115–126
    - properties when batched 199
    - properties when unbatched 204
    - pulling 247, 263
    - pushing 246, 262
    - quantities (overview) 94
    - quantity, using 125
    - renegeing 131
    - resource constraints 209
    - resource vs cost accumulators 225
    - routing 149
    - scaling a large number 249
    - selecting 145
    - travel time 248
    - unbatching 149
    - values (informational) 94
    - viewing 247
  - Items (DB) library (legacy) 490
- ## J
- JIT 185
  - Jockey model 133
  - jockeying 132
  - Johnson SB distribution 608
  - Johnson SU distribution 608
  - just-in-time system 185
- ## K
- Kanban model 185
  - kanban system 185
  - keyboard shortcuts 742
- ## L
- labels for blocks 15
  - Laplace distribution 608
  - Launch Proof command 709
  - Launch StatFit command 710
  - layout
    - of a discrete event model 93
    - of a discrete rate model 270
  - least dynamic slack 135
  - Least Dynamic Slack model 135
  - Left to right order 86

- legacy libraries 488, 489
  - Animation 490
  - BPR 490
  - Discrete Event 490
  - Flow 490
  - Generic 490
  - Items (DB) 490
  - Mfg 490
  - Quick Blocks 490
  - SDI Tools 490
- length units 271, 298
- Level of detail option 476, 693
- levels of use 5
- Lib extension 742
- libraries 488–496
  - automatic search for 491
  - block categories 690
  - block storage 26
  - changing 690
  - closing 491, 695
  - compiling 696
  - compiling with debugging code 696
  - conversion to Run Time format 697
  - converting 745
  - copying blocks between 502
  - corrupted blocks 502
  - discontinued 489
  - example libraries 489
  - ExtendSim 488
  - file name size 742
  - legacy 488, 489
  - MacWin conversion 697
  - maintaining 695
  - new 493, 695
  - number of blocks per 740
  - open when ExtendSim launches 690
  - opening 26, 490, 695
  - preload 690
  - protecting block code 495, 696
  - removing blocks from 502
  - RunTime format 496
  - saving blocks in 494
  - saving hierarchical blocks in 547
  - searching for 491, 690
  - size 740
  - substituting 491, 495
  - transferring between platforms 697, 745
  - types 488
  - uses 26
  - version strings 697
  - window 493, 671
- Libraries tab of Options command 690
- library
  - Animation 2D-3D 488
  - Custom Blocks library 489
  - Electronics 488
  - Item library 92, 488, 724–729
  - Item Templates library 489
  - ModL Tips library 489
  - Plotter library 489
  - Rate library 269, 489, 732
  - third-party libraries 61
  - Tutorial library 489
  - Utilities library 489, 736
  - Value library 60, 489, 716–721
- Library menu 695
- library window 493
  - blocks in categories 705
  - dates option 690
  - mode in Navigator 671
  - opening 671, 696
  - opening at startup 690
  - printing 672
  - to copy blocks 502
- Library Window mode 671
- License Info command 683
- LIFO queue 129
- Lighting Tools menu (E3D Editor) 485
- limiters 665
- limits of ExtendSim 740
- Line Balance with Activities model 160
- line balancing 147, 158
- Line tool 561
- linear programming 376
- Link 2D/3D positions 413, 425, 451, 480
- link alerts 534
- Link button 631, 632, 633
- Link dialog 630
  - checkboxes 634
  - Find Link command 635
  - Link To popup menu 634
  - No User-Defined Link setting 634
  - popup menu 630
- Link to enclosing H-block option 480
- linking 636–638
- List blocks by category in menu option 690
- List of tables option 649
- LIX extension 742

- local block number 501
  - local time unit 527
  - location of distribution 606
  - Lock H-Blocks checkbox 678
  - Lock Model command 677, 699
  - locking a model 677
  - LOD (level of detail) 476
  - Logarithmic distribution 608
  - Logistic distribution 609
  - loglogistic distribution 609
  - lognormal distribution 609
  - Lookup Table block 20, 252, 719
    - time units 169
    - time-based parameters 251
  - Lookup Table model 252
  - loop, empty 367
  - LP area 306, 377
    - flow rules 306
    - precision 360
  - LP calculation 382
    - bias information 380
    - flow rules 379
    - sequence of events 377
    - table summarizing information 381
  - LP Solver 376
  - LP technology 376
    - introduction 269
    - LP area 306
    - LP Solver 376
- ## M
- M/M/1 queue 130
  - Macintosh
    - file conversion 743
    - keyboard shortcuts 742
  - Macintosh to Windows 743
  - MacWin Conversion command 697, 745
  - mailslots 668
  - mainframes 663
  - Make Selection Hierarchical command 543, 698
  - Make Your Own block 92
  - Manufacturing library (legacy) 490
  - markers in the E3D window 467
  - Markov chain 50
  - Markov Chain Weather model 50
  - Match Braces command 707
  - Match IFDEF/ENDIF command 707
  - match into one item 195, 197
  - Matching Item model 197
  - material handling 185
  - Math block 21, 719
    - controlling the flow of items 145
  - Max & Min block 147, 160, 719
    - controlling the flow of items 145
  - Maximize Service Level model 137
  - maximized capacity in Convey Flow block 293
  - maximizing service levels 136
  - maximum density 343
  - maximum rate 303
    - dynamically changing 309
    - Interchange block 310
    - Tank block 310
  - MDI interface option 694
  - Mean & Variance block 242, 566, 721
  - menu command shortcuts 742
  - Merge block 279, 733
    - bias order 327, 362
    - bias order table 364
    - catching flow 329
    - critical constraints 311
    - displaying bias order 366
    - features 327
    - fixed rule modes 362
    - modes (mixed in model) 385
    - modes (summarized) 319
    - non-fixed rule modes 363
    - proportional mode and empty loop 367
    - requirements for critical constraint 313
    - setting bias order 363
  - Merge Proportion Setting model 367
  - Merging Inputs model 147
  - messages 262
    - application 534
    - block 260, 535
    - block-to-block 264
    - during simulation run 533
    - in discrete rate models 386, 387
    - item connector 262
    - link alerts 534
    - needs 262
    - rejects 262
    - value connector 261
  - Meter control 510
  - Metric distance units option 689
  - Mfg library (legacy) 490
  - minicomputers 663

- MiniMap 397
- MiniMap option 476, 693
- Minimize Setup model 136
- minimizing setup 136
- Minimum Value model 314
- Miscellaneous tab of Options command 694
- Mission objects 438
- Mission Objects category of WEC 438
  - Environment objects 438
  - Mission objects 438
  - System objects 438
- Mixed type (clone drop) statistical method 565
- MM1 model 130
- model 166
  - adding blocks 26
  - agent-based 51
  - associated with E3D window 398
  - backup files 681, 694
  - balking of items 131
  - basics 15
  - before you start 55
  - block definition 15
  - blocking of items 131
  - blocks that represent functions 600
  - building 25
  - closing 680
  - comparison of types 44
  - connection lines 16
  - connectors 15
  - continuous 60
  - continuous (definition) 44
  - converting for cross platform use 744
  - copying 675
  - cross-platform 742
  - debugging hints 614
  - definition 42
  - deterministic 58
  - differential equation 83
  - discrete event (definition) 44
  - discrete rate (definition) 45
  - distributions 606–610
  - file names 742
  - goals 54
  - grid 699
  - icons 15
  - initial conditions 530
  - integration methods 85, 611
  - item quantities 125
  - jockeying of items 132
  - layout for discrete event 93
  - locking 677, 699
  - logical models 43
  - Macintosh file name 742
  - measuring model performance 616
  - messages 533
  - Monte Carlo 47, 522
  - multiple runs 522, 531
  - multiple windows 533
  - names 742
  - non-terminating systems 530
  - notebook 508
  - number of runs 530
  - open a new model 25
  - opening 15, 680
  - order of calculation for blocks 86
  - parts of a model 15
  - password protection 677
  - path options 689
  - pictures in 562
  - preemption 178
  - printing 673
  - process of creating models 54
  - profiling 532
  - random number generator 605
  - random number seed 519
  - refining 56
  - reneging of items 131
  - reporting 596–598
  - resources 209
  - reverting 681
  - run length 530
  - running 18
  - running multiple models simultaneously 533
  - saving 27, 681
  - sharing models 677
  - size 740
  - sound at end of run 689
  - starting 680
  - state chart 49
  - State/Action 49
  - statistical analysis 564
  - status bar 524
  - stochastic 58
  - styles 692, 699
  - systems 42
  - table of different types 45
  - terminating systems 530
  - Tool Tips 689
  - tracing 620–621

- uncluttering 88
- validation 57
- verification 56, 615
- warm-up period 531
- Windows file name 742
- Model menu 698
- Model Navigator mode 38, 671
- Model Style tab for Options command 692
- Model tab of Options command 689
- model worksheet
  - printing 672
- modeling
  - continuous 60
  - discrete event 90
  - discrete rate 266
- modes (3D animation) 400, 521
  - Buffered 477
  - Concurrent 477
  - QuickView 477
- modes (for database)
  - structure 641
  - viewer 642
- modes (Merge/Diverge) 319
  - mixed mode situations 385
  - sensing 383
- modes (Navigator)
  - Database List mode 671
  - Library Window mode 671
  - Model Navigator mode 38, 671
- ModL
  - Develop menu 704
  - protecting block code 495
  - saving code as text file 696
- ModL font option 691
- ModL Tips library 489
- Monte Carlo model 47
- Monte Carlo simulation 47, 522
- most recent files 684
- Mount item while activity is ongoing option 419, 479
- Mount objects option (Batch block) 479
- mounting 3D objects 460
  - mount object (definition) 460
  - mount point 460
  - mounting node 460
  - rider object 460
  - steps 423
- mouse-look toggle 399
- Move Selected Items to Tab command 706

- moving blocks 27
- MOX extension 742
- multi-dimensional analysis 571
- Multiple Cost Accumulators model 236
- Multiple Document Interface option 694
- Multiple Pools model 211
- Multiple Resources model 230
- multiple scenarios 522
- Multirun analysis 565
- multitasking 183

## N

- named connections 33, 560
  - Show Named Connections command 560
- names of blocks 15
  - length of name 740
- Navigator 670–672
  - Database List mode 671
  - introduction to 37
  - Library Window mode 671
  - Model Navigator mode 38, 671
  - printing 672
- Navigator command 713
- negative binomial distribution 609
- Network License command 683
- neutral mode 326
- New Block command 704
- New Database command 641, 700
- New Database Tab command 647, 702
- New Dialog Item command 706
- New Hierarchical Block command 543, 698
- New Include File command 706
- New Library command 494, 695
- New Model command 25, 680
- New Tab command 706
- New Table command 701
- New Text File command 664, 680
- Noisy FM model 75
- non-accumulating conveyor
  - Convey Flow block 344
  - Convey Item block 188
- Non-Calendar date definitions 517
- non-fixed rule modes 363
- Non-Processing model 241
- non-terminating system 530
- normal distribution 609
- Notebook command 712

- has data (after command) 712
- notebooks 19
  - as control panel 508
  - copying 675
  - copying text to 538
  - grid 699
  - printing 672
- Notify block 511, 615, 720
- Number of steps option 518
- Number option for field type 649
- Number shift 218

## O

- Object command 688
- Object Mapper block 666, 736
- object model 665
- ObjectID 126, 424, 463
- objective function 584
- objects (3D) 442–463
  - actions table 443
  - changing property values 452
  - collision 462
  - creating 444–449
  - deleting 449
  - gravity, friction, and momentum 462
  - inspecting, using the E3D Editor 434
  - mounting 412, 460
  - moving 451
  - properties 443
  - rotation 412, 455
  - scaling 457
  - scenery 446
  - showing and hiding 453
  - skins 450
  - unlinking 2D/3D positions 425
  - waypoints 410, 459
- objects (embedded)
  - using COM/OLE 666
- objects (graphic)
  - aligning 698
  - colors 562
  - copying 674
  - drawing 561
  - drawing tools 562
  - patterns 562
  - rotating 699
- observed statistic 242
- ODBC (Open DataBase Connectivity) 667
- OLE (Object Linking and Embedding) 666
- On/Off shift 218
- Only Simulate Messages command 711
- Open All Library Windows command 696
- Open Block Structure command 704
- Open Breakpoints Window command 707
- Open command 680
- Open Debugger Window command 707
- Open Dialog tool 591
- Open Dynamic Linked Blocks command 635, 687
- Open E3D Window button 402
- Open E3D Window tool 476
- Open Hierarchical Block Structure command 698
- Open Include File command 706
- Open Library command 26, 695
- Open Library Window command 493, 695
- Open library window option (Libraries tab) 690
- Open Navigator tool 670
- Open Sensitized Blocks command 687
- open systems 95, 216
- opening a model 15
- opening libraries 26, 490
- optimization 572–586
  - adding variables 575
  - algorithms 573
  - clone-drop 575
  - constraint equations 582
  - constraints 580, 585
  - cost equation 584
  - determining the form of the function 575
  - entering the objective function 577
  - maximum samples 585
  - objective function 584
  - overview 573
  - profit equation 584
  - setting limits for the variables 577
  - steps for using 573
  - terminate if best and worst 585
  - tutorial 573
  - variables table 583
- Optimize 1 model 574
- Optimizer block 575, 583, 719
  - Results tab 586
  - Run Parameters tab 585
- Option key 698, 707
- Options command 688
  - 3D tab 692
  - Libraries tab 690



- Miscellaneous tab 694
- Model Style tab 692
- Model tab 689
- Programming tab 691
- order
  - bias 327
  - of simulation 86
- outflow branch 319
- Output Line Balancing model 159
- output variables in equation-based blocks 603
- Oval tool 561

## P

- page breaks 682
- page numbers 682
- Page Setup command 682
- parallel processing 149, 166
  - buffering 158
  - examples 103
  - explicit ordering 155
  - line balancing 158
  - simple parallel connections 166
  - successive ordering 154
  - using one Activity block 166
- parameters
  - arguments 606
  - blue frame 632
  - changing 20
  - changing dynamically (methods for) 676
  - definition 57
  - green 630
  - linking to a global array 631
  - linking to an ExtendSim database 630
  - red 630
  - yellow 630
- parent/child relationships (for database) 643
  - green background on field 643
  - red background on field 643
  - relationship dialog 644
- Pareto distribution 609
- passing blocks 256
- password protection 677
- Paste command 684
- Paste DDE Link command 636, 687
- Paste Picture command 562
- paths for 3D animation 467
  - creating a custom path 426
  - creating markers 428
  - looping paths 469
  - selecting the path in the Transport block 428
  - visible or colored paths 469
- Pattern popup menu 562
- Pause at Beginning command 619, 711
- Pause command 20, 523, 525, 710
- Pause Sim block 523, 525, 616, 738
- Pause Simulation button 523
- Pause/Resume button 20
- PE input connector 178
- Pearson type V distribution 609
- Pearson type VI distribution 609
- PICT (picture) resource 557
- pictures 745
  - bitmap 745
  - copying 675
  - file formats 556, 562
  - from other applications 676
  - Windows MetaFiles 745
  - working with 562
- Planet Dance model 79
- Play a sound 511
- Play sound at end of run option 689
- plot tools 589
- plotter 588–596
  - autoscale (manual) 591
  - Autoscaling option 593
  - axis (Y1/Y2) 590
  - clearing data 596
  - closed during simulation 591
  - color, pattern, and width 590
  - copying 675
  - copying data 596
  - data pane 589
  - Data storage tab 593
  - dialogs 592
  - displaying results 18
  - Do not show plot option 592
  - Don't continue line to Endsim option 593
  - for debugging 616
  - Grid Density tool 591
  - Insert plotter background option 593
  - Key on-off tool 591
  - lines as reference 88
  - log tool 591
  - multiple axes 35
  - name 590
  - number format 590
  - Open Dialog tool 591

- open during simulation 591
- options in dialog of 592
- Plot every nth point option 593
- plot pane 588
- point style 590
- printing 672, 673, 683
- printing top plot 673
- Push Plot tool 592
- Redraw trace tool 592
- saving plots 592
- scaling axes 591
- second plotter 39
- Show instantaneous queue length option 593
- Show plot at end of simulation option 592
- Show plot button option 593
- Show plot during simulation option 592
- Show Trace tool 591
- split bar 588
- symbols 590
- tools 35, 589
- Trace properties tool 590
- traces default to patterns 694
- types of 593
- X axis shows calendar dates option 593
- Zoom in tool 592
- Zoom out tool 592
- Plotter library 489
- Plotter traces default to patterns option 694
- Plotter, DE Error Bars block 594
- Plotter, DE MultiSim block 594
- Plotter, Discrete Event block 102, 594
- Plotter, Error Bars block 595
- Plotter, FFT block 595
- Plotter, I/O block 88, 595
- Plotter, MultiSim block 88, 569, 595
- Plotter, Scatter (4) block 595
- Plotter, Scatter block 595
- Plotter, Strip block 595
- Plotter, Worm block 596
- Poisson distribution 609
- Poll constraint every... 309
- Polygon tool 561
- Popups block 738
- post versions of E3D functions 474
- Power Function distribution 609
- precision in LP area 360
- Predator/Prey model 72
- Predict the path of the item 151
- preempt connector 353
- Preempt tab 178
- Preempting model 178
- preemption 178
  - definition 177
  - options in Activity block 178
- preferences for ExtendSim 688
- Preload libraries option 690
- preprocessing 249
- preserve uniqueness 204
  - in both Batch and Unbatch 204
  - in either Batch or Unbatch 205
- preserved value 204
- Print and Page Setup hints 674, 682
- Print command 672, 682
- print dialog 673, 682
- Print header/footer option 694
- Print Setup command 682
- printing 672–674, 682
  - add frame 673
  - header and footer options 694
  - headers and footers 674
  - hierarchical blocks 673
  - page breaks 682
  - page numbers 682
  - Print and Page Setup hints 674
  - Print Setup settings (saving) 694
  - selecting what to print 672
- priorities 122–124
  - overview 94
  - used for routing 155
- Prioritize Front Model command 533, 709
- Prioritize With Bias Blocks model 362
- priority mode 322
  - bias order determination 366
- Priority Mode Diverge model 323
- Priority Mode Merge model 324
- Priority model 130
- priority-sorted queue 129, 130
- probability distributions 606–610
- processes
  - costs 229
  - discrete event 164
  - examples 164
  - interrupted 177
  - parallel 166
  - serial 165
- Processing by Type model 161

- processing items 164–191
  - fixed number of items 175
  - for a fixed period of time 177
  - hysteresis 175
  - interrupting 177
  - just-in-time (JIT) 185
  - Kanban system 185
  - multitasking 183
  - scheduling 173
  - setting the time 167
  - shutting down the process 179
  - time sharing 171
- processing time
  - custom 170
  - fixed 168
  - implied 171
  - random 169
  - scheduled 168
  - setup time 172
- Production Line Final model 412
- Production Line Start model 407
- Profile Block Code command 712
- profiling 532, 712
- profit equation 584
- programming
  - profiling 712
  - protecting block code 696
- Programming tab of Options command 691
- Prompt for output value 511
- Proof Animation 709
- properties 115–126
  - \_3D objectID 126
  - \_Animation 200
  - \_Cost 126
  - \_item index 126
  - \_Item quantity 125, 200
  - \_Rate 126
  - attributes 115–122
  - batched value when unbatching 204
  - distribute when batching 204
  - item index 126
  - of items 94
  - of items when batched 199
  - of items when unbatched 204
  - options when batching 200
  - preserved value when unbatching 204
  - priority 122–124
  - Properties tab in Batch/Unbatch blocks 200
  - quantity 124

- Properties command 683
- Properties tab of Batch/Unbatch blocks 200
- proportional mode 321
- Proportional Mode Diverge model 322
- Proportional Mode Merge model 322
- Protect Library command 495, 696
- PT connector 241
- pulling items 247, 263
- Pulse block 718
- pushing items 246, 262

## Q

- quantity goal 335
- Quantity Goal model 336
- quantity of items 125, 200
  - overview 94
- queue
  - animating contents 140
  - attribute-sorted 129
  - FIFO (first in, first out) 129
  - initializing contents 139
  - jockeying 132
  - least dynamic slack 135
  - LIFO (last in, first out) 129
  - M/M/1 queue 130
  - manipulating contents 139
  - matching by attributes 138
  - maximizing service levels 136
  - minimize setup 136
  - priority-sorted 129, 130
  - reneging 131
  - resource pool queue 212
  - server systems 129
  - sorting mechanisms 128
  - user-defined sorting 129
  - viewing contents 139
- Queue block 101, 727
  - in resource pool queue mode 208
  - queuing 128
  - resource pool queue mode 209
- Queue Equation block 133, 601, 727
  - input variables 602
  - output variables 603
  - queuing 128
  - tie breaking capabilities 137
- Queue Matching block 138, 727
  - queuing 128
- Queue Matching model 138

Queue Statistics model 48, 566  
 Queue Tools block 139, 736  
   queuing 128  
 queue/server systems 129  
 queueing disciplines 128  
 Quick Blocks library (legacy) 490  
 QuickView mode 409, 477, 521  
 Quit command 684

## R

R connector 309  
 Random Activity model 169  
 random distributions 606–610  
 Random Intervals model 112  
 Random Number block 21, 718  
   setting time-based parameters 251  
 random number generator 519, 605  
   optional 605  
   recommended 520, 605  
 random number stream 605  
 random numbers 604–606  
   for database cells 650  
   resetting 606  
 Random Numbers tab 519  
   and confidence intervals 567  
 random seed 519, 605  
 Random seed option 519  
 Random Shutdown model 182  
 random variables (definition) 57  
 randomness 58  
 ranking rules 135  
   least dynamic slack 135  
   maximizing service levels 136  
   minimize setup 136  
   tie-breaking 137  
 rate block flow messages  
   in discrete rate models 388  
 Rate library 489, 732  
   animation 370  
   common connectors 368  
   overview 269  
 rate sections 305  
   boundaries 305  
   determining 315  
 rates  
   effective 303, 365  
   flow 268  
   goal 335

  in discrete rate models 271  
   infinite 304  
   maximum 303  
   precision 306  
   sections 305  
   types in discrete rate model 303  
 rates of flow 303  
 Rayleigh distribution 609  
 Read block 628, 659, 664, 717  
   addressing the data structure 660  
   communicating with data structures 659  
   for accessing a database 645  
 Read only option 649  
 Read(I) block 628, 659, 725  
   addressing the data structure 660  
   communicating with data structures 659  
 Read-Only link, in Link dialog 635  
 RealTimer block 525, 738  
 Receive Inventory model 229  
 Record ID field option 649  
 Record Message block 615, 617, 737  
 records  
   creating (for database) 642  
 Rectangle tool 561  
 red background on database fields 643  
 red parameter fields 630  
 Redraw trace tool 592  
 reductio-ad-absurdum 56  
 reference line 88, 589  
 Refresh DDE Links command 688  
 relational constraints  
   definition 307  
   determining 315  
   permanent 380  
   state sensitive 380  
 Remove All From Report command 710  
 Remove All From Trace command 620, 712  
 Remove Debug Code in Open Lib ... command 696  
 Remove External Code in Open Lib... command 696  
 Remove License command 683  
 Remove Selected From Report command 597, 710  
 Remove Selected From Trace command 620, 712  
 Rename Block command 502, 704  
 Rename Database command 647, 701  
 Rename or Delete Database Tab command 702  
 Rename Table command 701  
 renegeing 131  
 Reneging model 132

- Replace All command 686
  - Replace command 686
  - replace with 685
  - Replace, Find Again command 686
  - Report Type command 597, 710
  - reporting 596–598
    - commands 710
    - Dialogs report 596
    - report types 710
    - Statistics report 596
    - steps 597
  - Reservoir 1 model 14
  - Reservoir 2 model 68
  - Reservoir 3 model 70
  - Reset random numbers for every run option 519
  - residence blocks 256
  - resolution popup menu (in E3D window) 396
  - Resource Item block 213, 228, 727
    - advantages and disadvantages 213
    - creating resources as items 209
  - Resource Pool block 209, 727
    - advantages and disadvantages 210
    - creating resources 208
  - Resource Pool Release block 209, 727
    - releasing resources 208
  - resources 91, 95
    - attributes for tracking information 214
    - batching method 213
    - combining with cost accumulators 234
    - conceptual 215
    - conserving 201
    - constraining flow of items 209
    - costing 225
    - for costing 229
    - from different resource pools 211
    - implicit 215
    - limited 209, 213
    - modeling methods 209
    - number available 209, 213
    - Resource Pool method 209
    - scheduling 216
    - scheduling using other methods 217
    - scheduling using Shift block 218
    - scheduling using TR connector 216
    - used in multiple places 212
  - Resources model 221
  - restraints on resources 209
  - results (displaying on plotters) 18
  - Resume command 20, 711
  - Revert Model command 681
  - rider object 460
  - Rotate Shape command 562, 699
  - rotating a 3D object 455
  - Rotation option 480
  - Rounded Rectangle tool 561
  - routing
    - based on priority 155
    - conditional 157
    - explicit 104, 155
    - implicit 151
    - implied 104
    - remotely 156
    - sequential 154
    - using Throw Item and Catch Item blocks 148, 156
  - row index 662
  - run length 530
    - determining 531
  - Run menu 522, 708
  - Run Model block 525, 738
  - Run Optimization command 579, 708
  - Run Optimization tool 579
  - run parameters 25, 516
  - Run Parameters tab 585
  - Run Simulation button 18, 32
  - Run Simulation command 18, 708
  - running a model 18, 522–526
  - runs
    - multiple (resetting random numbers) 606
    - setting the number of 517
  - Runs option 517
  - RunTime format 678, 695, 697
  - RunTime Startup Screen Editor command 697
- ## S
- S (sensor) output connector 345
  - S (status) connector 365
  - Save backup model files option 694
  - Save Block to Library As command 548
  - Save Model As command 681
  - Save Model command 681
  - Save Print Setup settings option 694
  - Save Text File As command 681
  - Save Text File command 681
  - saving
    - a hierarchical block 547
    - a library 494

- a model 27, 681
- an environment file 459
- Scale option 480
- scaling
  - number of items 249
- scaling a 3D object 457
  - format of scale property 458
- scattergram 595
- scenarios 522
- scenery for 3D animation 411, 446
- Scenery tab of 3D Controller block 482
- Scheduled Intervals model 114
- Scheduled Shutdown model 181
- Scheduled Time model 169
- scheduling
  - activities 173
  - events 258
  - labor 218
  - resources 218
  - shutdown 181
- Scheduling Activities 1 model 174
- Scheduling Activities 2 model 175
- scheduling algorithms 128
- Scheduling Resources model 217
- scrap generation 153
- Scrap Generation model 153
- Scroll To Messages command 712
- SD input connector 179
- SD output connector 180
- SDI Tools library (legacy) 490
- search for 685
- seed 519, 605
- Select All command 685
- select connector 250
- Select Item In block 145, 728
  - routing items 144
  - selection options 146
  - starving conditions 146
- Select Item Out block 729
  - blocking conditions 150
  - routing items 144
  - selection options 149
- Select mode (in Merge/Diverge block) 319
- Select Mode Diverge model 320
- Select Mode Merge model 320
- Select mode option (3D Animation tab) 477, 521
- Select Value In block 720
- Select Value Out block 720
- selection options for Select Item Out block 149
- selection options in Select Item In block 146
- sensing mode 325, 383
  - bias order determination 366
- sensitivity analysis 568–572
  - disabling 570
  - enabling 570
  - multiple dimensions 571
  - opening sensitivity blocks 687
  - overview 568
  - Sensitize Parameter command 569, 687
  - settings 570
  - steps 568
- Sensitivity Setup dialog 570
- Sensitize Parameter command 687
- Sensor block 383, 733
  - animation 373
- sensor connector 247
- Sensors tab 345
- sequential routing 154
- serial port 662
- serial processing 165
- Serial Processing model 165
- server application 658, 665
- server application for DDE link 636
- Set block 118, 726
  - properties of items 110
- Set Block Category command 704
- Set Breakpoints command 707
- Set Library Version command 697
- Set Simulation Order command 700
- Setup tab 25, 517
- setup time 172
- Setup Time 2 model 173
- Shadows option (3D tab) 449, 476, 692
- Shape Fill/Border command 562
- shape of distribution 606
- Shapes category of WEC 438
- Shared Libraries 513, 662, 668
- sharing files
  - model locking 677
- Sheep and Wolves model 54
- Shift block 175, 218, 728
  - adding to a model 342
  - in discrete rate model 342
  - managing resources 209
  - Number shift 218

- On/Off shift 218
  - shift types 218
  - status connectors 219
- Shift On and Off model 220
- Shift Selected Code Left command 707
- Shift Selected Code Right command 707
- Shipping model 354
- Show 2D Animation command 551, 709
- Show 3D Animation command 709
- Show 3D animation during simulation option 521
- Show block in 3D window as option 480
- Show Block Labels command 699
- Show Block Messages command 711
- Show Block Numbers command 674, 699
- Show Clipboard command 674, 688
- Show DDE Links command 637, 688
- Show Debug Messages command 712
- Show library window dates option 690
- Show Movies command 709
- Show Named Connections command 34, 560, 698
- Show Page Breaks command 674, 682
- Show Reporting Blocks command 597, 710
- Show Simulation Order command 619, 700
- Show Tracing Blocks command 620, 712
- Show/Hide Connections command 560
- Show/Hide Connectors command 560
- Shuffle Graphics tool 561
- shutdown 179
  - definition 177
  - explicit 183
  - options for items 180
  - options in Activity block 180
  - scheduled 181
- Shutdown block 728
  - controlling item processing 164
  - time between failures (TBF) 182
  - time to repair (TTR) 182
- Shutdown tab 179
- Sim messages, in Link dialog 635
- Simple Batching model 197
- Simple Connections model 151, 166
- Simple Resource Pool model 210
- Simple Routing model 152
- Simple Routing One Queue model 152
- Simulate Multitasking Activity model 184
- simulation
  - agent-based 51
  - beep at end of run 689
  - benefits 4
  - continuous modeling 60
  - controlling the run 525
  - Custom order 86
  - definition 43
  - deterministic 58
  - discrete event modeling 93
  - discrete rate modeling 266
  - duration 82, 526, 531
  - events 16
  - Flow order 86
  - importance 4
  - integration methods 85
  - interactive 16
  - Left to right order 86
  - messages 533
  - monitoring the run 525
  - Monte Carlo 47
  - multiple 522, 531
  - number of runs 517
  - number of runs (maximum) 740
  - order 86
  - order of execution 619
  - parameters 25
  - pausing 522, 525, 710
  - process 55
  - prompt for user input 511
  - results 18
  - running 18, 32, 522–526
  - running multiple models simultaneously 533
  - running with Equation/Equation(I) block 744
  - runs 25
  - runs (determining) 531
  - saving intermediate results 525
  - slowing down a 533
  - sound 511
  - speeding up a 531
  - state chart 49
  - State/Action 49
  - status bar 524
  - stepping through 522
  - steps 16
  - stochastic 58
  - stopping 511, 710
  - time 82
  - timer inconsistencies in discrete event models 524
  - timing 526
- simulation order 86, 700
- Simulation Setup command 25, 82, 516, 708

- 3D Animation tab 477, 521
- Continuous tab 518
- Random Numbers tab 519
- Setup tab 517
- Simulation Variable block 718
- skewness of distribution 606
- skins of 3D objects 409, 450
  - skin types 450
- Slider control 509
- Sort By Type model 233
- sounds 462, 511, 689, 745
- Sounds option (3D tab) 449, 476, 692
- source code
  - control 696
  - external 705
  - protection 696
- speed and calculated distance 186
- speed and distance
  - in a Convey Flow block 343
- spread of distribution 606
- spreadsheets 658
- SQL (Structured Query Language) 667
- standard line 88
- start connector 114, 176
- start connector (on Valve) 341
- start time
  - absolute vs. relative 114
- Start time option 517
- Starting seed used in last model run option 519
- starving
  - in a Valve block 365
  - in Select Item In block 146
- State Action model 49
- state chart modeling 49
- State/Action modeling 49
- StatFit 586–588, 710
  - tutorial 587
- StatFit Example model 587
- static values 676
- statistical analysis 564
- statistical bias 239, 566
- statistical data fitting 607
- statistical method 565
- statistics 564–567
  - Batch means method 565
  - bias 239
  - blocks for discrete event models 238
  - clearing 239
  - custom method 565
  - cycle time 254
  - multirun analysis 565
  - observed 242
  - sensitivity analysis 568–572
  - time weighted 242
  - warm-up period 239
- Statistics block 238, 564, 615, 721
- Statistics report 596, 710
- status bar 524
- steady-state systems 530
- Step command 710
- Step Each Block command 619, 711
- Step Entire Model command 619, 711
- Step Into command 708
- Step Next Animation command 619, 711
- Step Out command 708
- Step Over command 708
- Step The Flow Process model 350
- steps 16
  - number of 83, 518
  - number of (maximum) 740
  - size 83
- Stepsize calculations option 518
- stepwise refinement 4
- stochastic models 58
- Stop command 710
- Stop Message block 254, 737
- Stop the simulation 511
- storage costs 227, 235
- string attributes 106, 116
  - declaring 255
  - defining 118
- String option 649
- structure mode (in database) 641
- structure window 691
  - hierarchical blocks 548
  - printing 672, 674
- structure window of hierarchical blocks 541
- Structure window opens in front option 691
- Styles option (connection lines) 558
- submodels 36, 545
- Supply & Demand Warning model 384
- supply scarcity 323
- Supply Sensing Mode Merge model 326
- Support Resource Center command 713
- Switch block 508, 738



- Switch control 510
  - system attributes 231
  - System objects (E3D animation) 438
  - systems
    - continuous 61
    - definition 42
    - discrete event 164
    - discrete rate 267
- T**
- tab delimited files 665, 681
  - tabs
    - deleting dialog 706
    - moving dialog items to 706
    - new database 702
    - new dialog 706
    - renaming dialog 706
  - Tank block 276, 733
    - animation 370
    - block units 297
    - capacity 291
    - critical constraints 310
    - direction animation 371
    - infinite checkbox 277
    - initial contents 294
    - level animation 370
    - maximum rate 303
  - Tank Constraint model 311, 315
  - Tank Flow Unit model 300
  - technical support 713
    - documentation 7
    - help 8
    - how to get 8
    - information to provide 8
    - resources 7
  - terminating systems 530
  - Terrain Editor mode 438
  - terrain height in E3D window 439
  - Terrain modes of the E3D Editor 438
  - Terrain Texture Painter mode 439
  - terrains for the E3D window 470–471
  - text 538–539
    - as a connector in hierarchical blocks 546
    - as a named connection 34
    - border 539, 695
    - box 538
    - copied as picture to Notebook 538
    - copying 538
    - drag and drop 539
    - duplicating 34
    - entering 538
    - finding 685
    - formatting 539, 695
    - in E3D window 483
    - labels 34
    - transparent 539, 695
  - Text file font option 665, 689
  - text files
    - changing the font 665
    - closing 680
    - creating 664, 680
    - exporting 681
    - font option 689
    - importing 681
    - internet access 659
    - opening 664, 680
    - printing 672
    - reverting 681
    - saving 681
  - text label 34
  - Text menu 695
  - Texture pane of Terrain Texture Painter mode 440
  - Throw & Catch model 148
  - Throw Flow block 329, 733
  - Throw Item block 148, 156, 729
    - popup list of Catch Item blocks 148
    - routing items 144
  - Throw Value block 720
  - throw/catch
    - filters in discrete rate models 330
    - in discrete event models 148
    - in discrete rate models 329
  - throw/catch connection 330
  - time
    - between arrivals 111
    - between failures (TBF) 182
    - to repair (TTR) 182
  - Time per step option 518
  - time ratio (3D animation) 521
  - time sharing 171
  - Time Sync block 525
  - Time Unit block 719
  - time units 298, 526–529
    - generic 526
    - global 25, 517
    - in discrete rate model 300
    - local 527

- Time Weighted Statistic model 242
  - time weighted statistics 242
  - time-based parameters 251
  - timer 524
  - TimeSync block 738
  - Tool Tips 8
    - additional block information 689
    - on block dialogs option 691
    - on dialog editor 691
    - setting the option 689
    - uses 676
  - toolbar
    - buttons (list) 714
    - database tools 714
    - plotter 589
  - tools
    - autoscale plotter manually 591
    - color 562
    - drawing 561
    - Grid density (plotter) 591
    - Key on-off (plotter) 591
    - log (plotter) 591
    - open dialog (plotter) 591
    - patterns 562
    - plotter 589
    - Push Plot tool (plotter) 592
    - Redraw trace (plotter) 592
    - shuffling graphics 561
    - Trace properties (plotter) 590
    - Zoom in (plotter) 592
    - Zoom out (plotter) 592
  - Tools command 695
  - Torque Game Engine (TGE) 392
  - total cost 235
  - TR connector 216
  - tracing 620–621
    - steps 620
  - Transparent command 695
  - Transport Animation tab 481
  - Transport block 724
    - 3D animation 481
    - adding to a model 419
    - distance ratio 188
    - from and to locations 187, 481
    - Get distance from 3D path length 429
    - move time 186
    - processing items 164
    - selecting a path 428
    - speed and calculated distance 186
      - speed and distance 186
    - transportation 185
  - Transportation 1 model 190
  - Transportation 2 model 191
  - travel time 248, 419
    - move time setting 420
    - options 186, 421
    - speed and calculated distance setting 422
    - speed and distance setting 426
  - Tree pane of the WEI 437
  - Tree pane of WEC 438
  - triangular distribution 609
  - Tutorial library 489
- ## U
- Unbatch block 201, 725
    - unbatching items 194
  - Unbatch Mode Diverge model 321
  - unbatch/batch mode 321
  - unbatching 149, 201–204
    - batch size 203
    - batched value 204
    - create multiple items 202
    - distribute properties 204
    - item properties 204
    - preserve uniqueness 204
    - preserved value 204
    - release cost resources 202
    - variable number of items 203
  - Undo command 684
  - Uniform Integer distribution 610
  - Uniform Real distribution 610
  - unit groups 271, 298
  - units
    - block 271, 297
    - group 271, 298
    - length 271, 298
    - managing flow units 366
    - metric 689
    - time 298, 300
  - Universal connector 498, 545
  - Unmount option in Activity and Workstation blocks 479
  - unmounting 419
  - update flow status 365
  - Update Launch Control command 681
  - updates to ExtendSim 713
  - upper limits 740

- upstream supply 322
  - cautions when determining 383
  - definition 383
  - discrepancy with downstream demand 324
  - supply scarcity 323
- Use database table `_Seed...` option 520
- Use default connection line types option 689
- Use Grid command 699
- Use recommended random number...option 520
- Use Sensitivity Analysis command 709
- Use separators option 649
- User Defined connector 498, 545
- User Forum command 713
- user forums 8, 713
- user interface 504–514
  - for data exchange 624
- user-defined queue 129
- Utilities library 489, 736

## V

- validation 57, 66
- value attributes 116
  - defining 118
- value connector messages 250, 261
  - in discrete rate models 387
- Value connectors 87, 498, 545
- Value library 60, 489, 716–721
  - blocks in non-continuous models 500
- values
  - definition 57
  - dynamic 676
  - informational 94
  - static 676
- Valve block 276, 733
  - animating blocking/starving status 372
  - animating goal 373
  - animating hysteresis 373
  - animating limiting status 371
  - animation 371
  - critical constraints 309
  - duration goal 338
  - Flow Control tab 334
  - goal for rate 335
  - GS (goal status) connector 337
  - hysteresis 341
  - maximum rate 303
  - maximum rate initialization 309
  - polling constraints 309

- quantity goal 335
  - status options 365
- variable branches 319
- variable connectors 498
  - collapsing 500
  - contracting 499
  - expanding 499
- variable cost rate 234
- variable costs 224
- variables
  - input, in equation-based blocks 602
  - output, in equation-based blocks 603
- variables (definition) 57
- variables table 583
- verification 56, 615
- Verifying Information model 240
- View Using Defaults option 558
- viewer mode (in database) 642
- viewing items 247
- views (icon) 496

## W

- Wait Time block 717
- waiting costs 227, 235
- wants 262
- warm-up bias 531
- warm-up period 239, 531, 566
- waypoints 410
- WEC (World Editor Creator) mode 437
- Weekly and Daily Shifts model 221
- WEI (World Editor Inspector) mode 434, 436
- Weibull distribution 610
- What's New command 713
- window
  - database 700
  - E3D Window 399
  - library 695
  - Navigator 670
- Window menu 712
- Window menu (E3D Editor) 485
- Windows
  - file conversion 743
  - keyboard shortcuts 742
- Windows MetaFiles 745
- Windows to Macintosh 743
- WMF (Windows MetaFiles) 556
- worksheet

- grid 699
- Workstation block 725
  - Item Animation tab 479
  - processing items 164
- World Editor Creator (WEC)
  - mode 437
  - panes 438
- World Editor Inspector (WEI)
  - mode 434, 436
  - moving objects with 451
  - panes 437
- World Editor mode of the E3D Editor 436
- World menu (E3D Editor) 485
- World modes of the E3D Editor 436
- Write block 628, 659, 664, 717
  - addressing the data structure 660
  - communicating with data structures 659
  - for accessing a database 645
- Write(I) block 628, 659, 725
  - addressing the data structure 660
  - communicating with data structures 659

## Y

- yellow outline around field 637
- yellow parameter fields 630
- Yogurt Changeover model 355
- Yogurt Production model 274, 355

## Z

- Z is ground level option 480
- zero effective rate 365
- zero time events 259
- Zoom in tool 592
- Zoom out tool 592