

ReThink User's Guide

Version 5.1 Rev. 1



ReThink User's Guide, Version 5.1 Rev. 1

September 2014

The information in this publication is subject to change without notice and does not represent a commitment by Gensym Corporation.

Although this software has been extensively tested, Gensym cannot guarantee error-free performance in all applications. Accordingly, use of the software is at the customer's sole risk.

Copyright (c) 1985-2014 Gensym Corporation

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Gensym Corporation.

Gensym®, G2®, Optegrity®, and ReThink® are registered trademarks of Gensym Corporation.

NeurOn-Line™, Dynamic Scheduling™, G2 Real-Time Expert System™, G2 ActiveXLink™, G2 BeanBuilder™, G2 CORBALink™, G2 Diagnostic Assistant™, G2 Gateway™, G2 GUIDE™, G2GL™, G2 JavaLink™, G2 ProTools™, GDA™, GFI™, GSI™, ICP™, Integrity™, and SymCure™ are trademarks of Gensym Corporation.

Telewindows is a trademark or registered trademark of Microsoft Corporation in the United States and/or other countries. Telewindows is used by Gensym Corporation under license from owner.

This software is based in part on the work of the Independent JPEG Group.

Copyright (c) 1998-2002 Daniel Veillard. All Rights Reserved.

SCOR® is a registered trademark of PRTM.

License for Scintilla and SciTE, Copyright 1998-2003 by Neil Hodgson, All Rights Reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations, and Gensym Corporation disclaims any responsibility for specifying which marks are owned by which companies or organizations.

Gensym Corporation
52 Second Avenue
Burlington, MA 01803 USA
Telephone: (781) 265-7100
Fax: (781) 265-7101

Part Number: DOC075-510

Contents Summary

Preface xxv

Part I Modeling Using ReThink 1

Chapter 1 Running ReThink 3

Chapter 2 Organizing Models and Controlling Simulations 9

Chapter 3 Working with Models 39

Chapter 4 Using Blocks 105

Chapter 5 Using Instruments 205

Chapter 6 Using Resources 261

Chapter 7 Using Work Objects 345

Chapter 8 Using Reports 361

Chapter 9 Accessing External Databases 417

Chapter 10 Using Batch Simulation 445

Chapter 11 Using ReThink in Online Mode 455

Part II ReThink Reference 473

Chapter 12 Blocks Reference 475

Chapter 13 Instruments Reference 639

Glossary 775

Index 783

Contents

Preface xxv

About this Guide xxv

Audience xxvi

Conventions xxvi

Related Documentation xxvii

Customer Support Services xxix

Part I Modeling Using ReThink 1

Chapter 1 Running ReThink 3

Introduction 3

Starting the Server and Connecting the Client 4

Connecting to a Specific Server at Startup 5

 Connecting the Client to the Default Server 6

 Starting the Server on a Specific Port 6

 Connecting the Client to a Specific Server 6

Starting the Server with Your Application Loaded 7

Exiting ReThink 7

Chapter 2 Organizing Models and Controlling Simulations 9

Introduction 9

Working with Projects 11

 Creating a New Project 11

 Saving a Project 12

 Opening a Project 12

Configuring the Model Environment 13

 Creating a Model 13

 Creating an Organizer 16

Controlling the Simulation 17

 Activating and Deactivating the Scenario 18

	Starting and Stopping the Simulation	18
	Configuring the Scenario	20
	Configuring the Simulation Mode	20
	Running the Simulation in Jump Mode	21
	Running the Simulation in Step Mode	21
	Running the Simulation in Synch Mode	21
	Configuring the Duration of the Simulation	22
	Configuring the Simulation Version	23
	Configuring the Start Time of the Simulation	23
	Configuring Simulation Speed	23
	Configuring Animation	24
	Configuring Object Tracking	25
	Configuring the Behavior of Indicator Arrows	25
	Configuring the Computation Behavior	26
	Configuring the Scenario to Generate Identical Random Numbers	27
	Performing “What-if” Analysis on a Model	28
	Comparing Different Versions of the Same Model	28
	Using Different Scenarios to Compare the Same Model	30
	Using a Single Scenario to Control Multiple Models	31
	Working with Large Models	32
	Associating Existing Connectors on Task Block Details	32
	Associating Connectors on Other Types of Details	35
	Replacing Default Details of Model and Organizer Tools	36
	Viewing Demo Models	37
	Customizing Scenarios	38
Chapter 3	Working with Models	39
	Introduction	41
	Summary of Common Tasks	41
	Using the Project Menu	42
	Using the Project Menu	42
	Using the Manage Dialog	43
	Using the Project Submenus	44
	Navigating Applications	44
	Using the Navigator	45
	Searching for Objects	46
	Interacting with Workspaces	46
	Displaying a Detail Workspace	47
	Hiding a Workspace	47
	Deleting a Workspace	47
	Creating a Detail Workspace	48

Editing Workspace Properties	48
Scaling a Workspace	49
Shrink Wrapping a Workspace	49
Showing the Superior Object of a Detail Workspace	51
Printing a Workspace	51
Saving a Workspace to a JPEG File	51
Loading Background Images	52
Creating and Accessing Top-Level Workspaces	52
Using the Menus	53
Using the File Menu	54
Using the Edit Menu	54
Using the View Menu	55
Using the Layout Menu	56
Using the Go Menu	57
Using the Project Menu	58
Using the Workspace Menu	60
Using the Simulation Menu	60
Using the Tools Menu	61
Using the Help Menu	62
Using the ReThink Toolbox	63
Basic Activities	66
Constraints	67
Displays	67
Export Tools	67
Instruments	68
Online Activities	69
Reports	70
Resources	70
Tools	71
Using the G2 Toolbox	72
Interacting with Objects	72
Selecting Objects	73
Cutting, Copying, Pasting, and Deleting Objects	73
Controlling the Layout of Objects	74
Displaying the Properties Dialog for an Object	74
Resizing an Object	75
Editing Icon Color Regions	75
Using the Toolbars	75
Standard Toolbar	76
Simulation Toolbar	77
Web Toolbar	78
Layout Toolbar	79
Status Bar	79
Annotating Models	80

Using an Annotation Tool 80

Using Free Text 81

Using Readout Tables 81

Using Attribute Displays 83

Setting and Clearing Breakpoints and Indicators 84

Switching User Modes 85

Viewing Messages 86

Configuring User Preferences 87

Specifying User Preferences for Different Types of Users 88

Configuring User Preferences 90

Delivering Messages by Email 93

Starting the G2 JMail Bridge Process 94

Creating, Configuring, and Connecting the JMail Interface Object
94

Configuring ReThink to Send Email Messages 97

Examples: Sending Email Messages 99

Configuring Startup Parameter for Sending Email Messages 101

Configuring Network Interfaces 102

Configuring Message Browsers 102

Configuring Module Settings 102

103

Chapter 4 Using Blocks 105

Introduction 105

Creating Blocks 107

Creating Blocks 108

Source Block 109

Task Block 109

Sink Block 109

Copy Block 109

Merge Block 109

Branch Block 110

Batch Block 110

Associate and Reconcile Blocks 110

Store and Retrieve Blocks 110

Insert and Remove Blocks 111

Copy Attributes Block 111

Yield Block 111

BRMS Task Block 111

Connecting Blocks 112

Using Stubs to Connect Two Blocks 112

Inserting a Block Between Two Connected Blocks	112
Redisplaying the Paths of Connected Blocks	113
Disabling Path Redrawing	113
Creating and Deleting Stubs	114
Deleting a Stub	114
Creating a New Stub	115
Creating Loops in a Diagram	116
Replacing Blocks	117
Configuring the Type of Work that Blocks Process	118
Configuring the Path Type	118
Using the Default Path Type	119
Creating Work During Processing	120
Configuring the Path Types of Particular Blocks	121
Creating Class Definitions for Work Objects	121
Determining the Output Path Based on Its Type	122
Configuring the Animation of Paths	122
Configuring Path Types of Specific Blocks	123
Configuring Blocks	126
Configuring General Block Parameters	127
Configuring Specific Block Attributes and Features	128
Configuring Path Identity of Specific Blocks	132
Configuring the Duration of Blocks	133
Configuring the Cost of Blocks	135
Configuring the Animation of Blocks	137
Configuring Specific Blocks	138
Source Block	138
Task Block	139
Branch Block	141
Merge Block	141
Insert and Remove Blocks	142
Sink Block	143
Custom Blocks	144
Creating Hierarchical Views	144
Modeling the Detail of a Task	145
Interacting with the Detail	146
Understanding the Activities of Blocks	147
Determining the Current Activities	148
Understanding the Attributes of Activities	150
Customizing the Time Delay of Activities	152
Using Resources to Constrain Concurrent Activities	152
Limiting the Number of Concurrent Activities	153
Showing Work Backups on an Input Path	154
Analyzing the Wait Time Due to Work Backups	155
Showing Work Backups Interactively	157

Working with the Duration of Blocks	157
Specifying a Fixed Duration	158
Specifying a Random Duration	159
Fixed Distribution	162
Random Exponential	162
Random Normal	163
Random Uniform	163
Random Triangular	164
Random Erlang	164
Random Weibull	165
Random Lognormal	165
Random Gamma	166
Random Beta	166
Specifying Duration from a File	167
Specifying Duration Based on an Indexed Report Lookup	168
Specifying Duration Based on an Attribute of a Work Object	173
Specifying Duration Based on an Attribute Report Lookup	176
Using a Graph to Specify Duration	183
Creating an Arrival Rate Input Graph	183
Configuring the Arrival Rate Input Graph	184
Editing the Shape of the Arrival Rate Input Graph	188
Configuring the Block to Use the Graph	189
Specifying a Custom Duration	189
Understanding Total Work Time and Total Elapsed Time	190
Total Work Time	191
Total Elapsed Time	192
Relating Work Time and Elapsed Time of Activities and Blocks	192
How the Block Uses Total Work Time and Total Elapsed Time	193
Updating Duration Metrics for Blocks	194
Computing Duration for Multiple Units of Work	194
Working with Block Costs	196
Configuring the Cost of a Block	196
Specifying a Fixed Cost	196
Specifying a Variable Cost	197
Computing the Total Cost of a Block	198
Debugging Blocks	200
Viewing and Resetting Errors	200
Verifying Model Metrics	203
Testing Every Possible Outcome	203
Customizing Blocks	204
Chapter 5	Using Instruments 205
Introduction	205
Creating Instruments	207

Creating Instruments	207
Timestamp Feed	208
Accumulate Feed	208
Increment Feed	208
Change Feed	209
Parameter Feed	209
Attribute Feed	209
Copy Attributes Feed	209
Delta Time Probe	209
Sample Probe	209
Average Probe	210
Moving Average Probe	210
Interval Sample Probe	210
Parameter Probe	210
Copy Attributes Probe	210
Statistics Probe	210
Criteria Probe	211
Update Trigger Probe	211
N-Dimensional Sample Probe	211
Message Probe	211
Acknowledge Message Probe	211
Delete Message Probe	212
Connecting Instruments	212
Connecting Instruments to Objects	212
Replacing Instruments	214
Configuring the Animation of Instruments	214
Probing the Performance of Your Model	215
Configuring the Probe	216
Showing the Current Value of the Probe	220
Probing the Performance of Blocks	222
Probing the Performance of Work Objects	223
Probing the Performance of Resources	225
Three Techniques for Probing Resources	226
Probing the Average Utilization of the Current Resource	226
Probing the Average Utilization of the Top-Level Resource	227
Probing the Average Utilization of a Resource in a Pool	228
Charting Performance Metrics	228
Creating a Remote Chart	229
Exporting Probed Data to a CSV File	230
Exporting Probed Data Based on Model Events	231
Exporting Probed Data at Regular Time Intervals	233
Exporting Historical Data	235
Feeding Values into the Model	236
Configuring the Feed	237

Updating User-Defined Attributes of a Work Object 238
Updating System-Defined Attributes of the Model 240

Creating User Interface Objects for Feeding Values 243

Creating a Slider 243

Creating a Type-in Box 244

Creating a Chart Directly from a Probe 246

Creating a Chart 246

Updating Charts 247

Updating Charts Manually 248

Using an Action Button to Update Charts 248

Using a Rule to Update Charts 249

Configuring the Colors and Data Points of the Chart 251

Configuring the Axes of the Chart 252

Plotting Multiple Values on the Same Chart 254

Scaling the Current Value of a Remote 256

Offsetting the Current Value of a Remote 258

Showing Metrics for a Remote 258

Customizing Instruments 259

Chapter 6 Using Resources 261

Introduction 262

Using Resources to Constrain the Model 264

Creating a Resource 264

Allocating a Resource to a Task 266

Identifying the Associated Resource 268

Associating the Manager with a Different Resource 268

Replacing Resources 268

Showing Work Backups Due to Resource Constraints 268

Showing Currently Allocated Resources 270

Disabling a Resource 271

Creating a Pool of Resources 271

Creating a Pool for Any Resource 272

Creating a Generic Pool 274

Showing Pool Details 274

Deleting Pool Details 274

Computing Utilization and Duration Metrics 275

Computing Utilization Metrics 275

Computing Duration Metrics 276

Computing Metrics for Individual Resources 277

Computing Metrics for the Resource Pool 277

Keeping a History of Resource Utilization 279

Charting Resource Utilization 280

Working with Resource Costs	281
Assigning Costs to Resources in a Model	281
Computing the Cost of Individual Activities	283
Computing Total Costs Based on Resource Costs	283
Allocating Multiple Resources to a Task	285
Allocating the Same Pool to Multiple Tasks	286
Sharing the Same Resource in Multiple Pools	287
Allocating Partial and Multiple Resources	289
Specifying the Utilization of the Resource Manager	289
Specifying the Number of Available Resources	290
Determining the Maximum Number of Activities	291
Determining Whether to Use a Pool or an Individual Resource	291
Example of Allocating Partial Resources from a Pool	292
Computing Metrics for Individual Resources in a Pool	294
Computing Metrics for the Resource Pool	295
Example of Allocating Multiple Resources from a Pool	296
Computing Metrics for Individual Resources in a Pool	298
Computing Metrics for the Resource Pool	299
Allocating the Same Resource for Multiple Sequential Steps	300
Choosing Particular Resources from a Pool	303
Choosing the Lowest Cost Resource	303
Choosing the Resource with the Lowest Utilization	306
Choosing Resources Based on Priority	309
Allocating Resources Associated to Work Objects	312
Allocating the Same Resource to Different Blocks Based on Priority	317
Creating Resources with Different Efficiency Factors	319
Showing the Metrics of Resources	322
Displaying Resource Metrics	322
Example of Allocating Resources	323
Displaying Attributes with a Resource	324
Constraining the Availability of Resources	325
Allocating Resources With Constraints	325
Displaying Constraints	328
Constraining a Resource to Normal Business Hours	328
Configuring the Availability of the Resource	330
Temporal Scheduler Detail	331
Default Configuration of the Temporal Constraint Detail	331
Determining the Availability of Each Type of Constraint Visually	331
Displaying the Temporal Scheduler Detail	331
Configuring the Monthly Availability	332

	Configuring the Weekly Availability	333
	Configuring the Hourly Availability	334
	Configuring the Date Availability	336
	Using Constraints with Timing Resources	339
	Configuring the Animation of Resources	340
	Probing the Performance of Resources	341
	Populating Resource Pools Dynamically	342
	Customizing Resources	343
Chapter 7	Using Work Objects	345
	Introduction	345
	Configuring Path Types	347
	Using the Default Path Type	347
	Specifying a Container as the Path Type	347
	Specifying a User-Defined Object as the Path Type	348
	Automatically Generating the Work Object Class Definition	349
	Creating a New Class of Work Object	349
	Viewing User-Defined Attributes of Work Objects	351
	Comparing Work Objects and Resources	352
	Understanding the Activities of Work Objects	353
	Computing Utilization and Duration Metrics	354
	Computing Utilization Metrics	354
	Understanding the Duration Metrics of a Work Object	355
	Understanding the Utilization of a Work Object	356
	Example of Computing Utilization Metrics With No Constraints	357
	Example of Computing Utilization Metrics With Constraints	357
	Computing the Cycle Time of a Work Object	358
	Working with Work Object Costs	359
	Customizing Work Objects	360
Chapter 8	Using Reports	361
	Introduction	362
	Creating Reports	363
	Summary of Input and Output Reports	363
	Creating a Report	365
	Generating Output Report Data from the Model	368
	Applying Input Report Data to the Model	369
	Configuring the Time Unit	370

- Updating Output Reports at Regular Time Intervals 372
 - Configuring Output Reports to Update Regularly 372
 - Triggering Regular Updates for Multiple Reports 373
 - Triggering Updates Based on Model Events 377
 - Triggering Updates Manually 380
 - Configuring When Clients Refresh Their Data 380
- Keeping a History of Data Values 381
- Charting Report Data 383
- Configuring the Scope of the Report 384
- Filtering Report Data 385
- Configuring the Attributes to Appear in a Report 392
- Creating Reports in Excel 394
 - Creating a Report in Excel 395
 - Generating Output Report Data from the Model to Excel 398
 - Applying Input Report Data to the Model from Excel 400
 - Filtering Report Data in Excel 401
 - Controlling the Simulation from Excel 405
 - Connecting to and Disconnecting from the Server from Excel 405
- Writing to and Reading from CSV Files 406
 - Writing Output Report Data to CSV Files 406
 - Importing Input Report Data from CSV Files 407
- Writing to and Importing from Databases 408
- Creating Specialized Reports 408
 - Creating N-Dimensional Reports 408
 - Creating Indexed Lookup Reports 411
 - Creating Attribute Lookup Reports 411
 - Creating Attribute Change Event Reports 411

Chapter 9 Accessing External Databases 417

- Introduction 417
- Configuring ReThink for Database Access 418
 - Creating the Database 419
 - Configuring the ODBC Data Source 420
 - Starting the ODBC Bridge Process 421
 - Creating and Configuring the Database Interface Object 422
 - Connecting to the Database 424
- Creating a Work Object that Represents a Record 425
 - Creating a Class Definition for a Query Object 425
 - Using a Query Object in a Model 427
- Creating an SQL Query for Accessing the Data 427

Sourcing Records from a Database	428
Retrieving Records from a Database	432
Storing Work Objects to a Database Table	435
Storing New Objects in a Database	435
Updating Existing Records in a Database	437
Using Reports to Access External Databases	438
Configuring Report Objects for Database Access	439
Writing Output Report Data to a Database	442
Importing Input Report Data from a Database	444

Chapter 10 Using Batch Simulation 445

Introduction	445
Using the Batch Simulation Object to Run Simulations	446
Simulation Keywords	451
Report Keywords	452
Setting Attribute Values	453

Chapter 11 Using ReThink in Online Mode 455

Introduction	455
Using ReThink in Online Mode	456
How Online Mode Works	457
Using Interface Pools	458
Using Online Blocks	463
Handling Errors	463
Introduce Delays into the Process	464
Modeling Distributed Workflow Applications	464
Remote Process Source Block	464
Remote Process Task Block	465
Remote Process Sink Block	465
Interacting with Databases	466
DB Function Query Block	467
Database Stored Procedure Block	467
Database Update Object Block	467
Database SQL DML Block	468
Database Query Block	468
Database Commit Block	468
Database Rollback Block	468
Sending Email	468

Using JMS Messaging 470

Part II ReThink Reference 473

Chapter 12 Blocks Reference 475

Introduction 476

Common Attributes of Blocks 477

General Tab 478

Duration Tab 480

Cost Tab 483

Animation Tab 485

Common Menu Choices for Blocks 486

Common Attributes of Paths 487

General Tab 488

Branch Tab 489

Animation Tab 490

Common Menu Choices for Paths 491

Associate 492

Configuring the Association Mode 493

Creating New Associations 493

Adding Work Objects to Existing Associations 494

Showing Associated Work Objects 495

Specific Attributes 496

Specific Menu Choices 497

Batch 498

Configuring the Batch Mode 498

Batching Objects in a Group 499

Batching Objects By Summing an Attribute of a Work Object 499

Batching Objects Based on a Triggering Work Object 501

Batching Objects At Specified Time Intervals 503

Batching Objects into a Container 505

Showing Work Objects in the Container 506

Specific Attributes 507

Specific Menu Choices 509

Customization Attributes 510

Branch 511

Configuring the Branch Mode 511

Branching Based on Proportion 512

Branching Based on a Dynamic Proportion 513

Branching Based on Type 516

Interactively Selecting the Output Path 518

Branching Based on Attribute Value	518
Branching Based on a Range of Values	520
Branching Based on Rules that Set the Attribute Value	523
Path Attributes that Pertain Only to Branching	529
Specific Attributes	532
Specific Menu Choices	534
Customization Attributes	534
BRMS Task	535
Configuring the BRMS Rules to Invoke	535
Specific Attributes	536
Specific Menu Choices	536
Copy	537
Creating Copies of a Work Object	537
Identifying the Original Output Path	538
Adding Copies to Associations	539
Configuring the Number of Objects to Create	539
Specific Attributes	540
Specific Menu Choices	541
Customization Attributes	541
Copy Attributes	542
Copying Attributes from One Object to Another	542
Specific Attributes	546
Specific Menu Choices	547
Insert	548
Understanding the Paths of an Insert Block	548
Configuring the Insert Mode	548
Inserting a Single Object Into a Container	549
Inserting Objects into the Container By Looping	550
Inserting Objects Into the Container All at Once	551
Showing Work Objects in the Container	551
Specific Attributes	552
Specific Menu Choices	553
Merge	554
Merging Multiple Streams of Work	554
Merging Work That Loops Around a Process	555
Specific Attributes and Menu Choices	555
Reconcile	556
Reconciling Individual Associated Objects	556
Reconciling All Objects	557
Specific Attributes	560
Specific Menu Choices	561
Customization Attributes	561
Remove	562

Understanding the Paths of a Remove Block	562
Configuring the Remove Mode	563
Removing Objects from the Container By Looping	563
Removing Objects from the Container All at Once	565
Showing Work Objects in the Container	566
Specific Attributes	567
Specific Menu Choices	568
Customization Attributes	569
Retrieve	570
Configuring the Retrieve Mode	570
Retrieving Objects from a Pool	571
Retrieving Objects from a Pool at Random	571
Retrieving Associated Objects from a Pool	572
Retrieving Objects with a Particular Attribute Value from a Pool	573
Retrieving Based on a Range of Values	576
Retrieving All Work Object	577
Retrieving Copies of Work Objects from a Pool	577
Adding Retrieved Work Objects to Associations	578
Determining How the Block Handles Objects Not Found	578
Specific Attributes	579
Specific Menu Choices	583
Customization Attributes	584
Sink	585
Signalling the End of a Process	585
Specific Attributes and Menu Choices	586
Source	587
Configuring the Source Mode	588
Generating Work Objects Based on the Path Type	588
Configuring the Number of Objects to Generate for Each Output Path Type	589
Generating Work Objects from an External File	589
Format of Object File	589
Generating Work Objects Continuously	590
Stopping Generating Work Objects at the End of the File	592
Configuring Duration and Objects from an External File	592
Generating Work Objects	594
Configuring the Maximum Number of Objects	594
Configuring the Start and End Times	595
Specific Attributes	596
Specific Menu Choices	598
Customization Attributes	599
Store	600
Configuring the Store Mode	600
Storing Work Objects in a Pool	601
Storing Work Objects to a File	602

Storing Arrival Times to a File **603**
Specific Attributes **604**
Specific Menu Choices **606**
Customization Attributes **607**

Task 608

Processing Work and Sending It Downstream for Further Processing **608**
Processing Multiple Streams of Work Synchronously **609**
Processing Multiple Streams of Work Sequentially **610**
Generating Work in a Process **611**
Specifying the Number of Objects to Generate **612**
Deleting Work in a Process **613**
Modeling the Details of a Task **613**
Copying Attribute Values to the Output Object **614**
Specific Attributes **619**
Specific Menu Choices **620**

Yield 622

Configuring the Yield Mode **622**
Configuring a Random or Random Triangular Yield **623**
Configuring Yield Based on an Attribute of the Work Object **627**
Configuring a Proportional Yield **630**
Determining the Yield Value **634**
Specific Attributes **635**
Specific Menu Choices **637**
Customization Attributes **638**

Chapter 13 Instruments Reference 639

Introduction 640

Common Attributes of Instruments 642

General Tab for Feeds **643**
General Tab for Probes **644**
Animation Tab for Feeds and Probes **646**

Common Menu Choices for Instruments 648

Common Menu Choices for Feeds and Probes **648**
Common Menu Choices for Probes **649**

Acknowledge Message Probe 650

Specific Attributes **650**
Specific Menu Choices **651**

Average Probe 652

Determining the Value of the Probe **652**
Computing the Average of an Attribute Value **652**
Plotting the Minimum and Maximum Values **655**
Charting the Average of Quantitative Parameters **656**

Specific Attributes	657
Specific Menu Choices	658
Copy Attributes Probe	659
Rolling Up Metrics from the Detail to the Superior Task	659
Specific Attributes	665
Specific Menu Choices	666
Criteria Probe	667
Determining the Value of the Probe	667
Comparing Sampled Values Against a Criteria	667
Specific Attributes	669
Specific Menu Choices	670
Delete Message Probe	671
Specific Attributes	671
Specific Menu Choices	672
Delta Time Probe	673
Determining the Value of the Probe	673
Computing the Cycle Time	673
Computing a Partial Cycle Time	675
Specific Attributes	677
Specific Menu Choices	677
Interval Sample Probe	678
Determining the Value of the Probe	678
Sampling the Model at Regular Time Intervals	678
Specific Attributes	681
Specific Menu Choices	682
Message Probe	683
Generating Text Messages	683
Specific Attributes	685
Specific Menu Choices	687
Moving Average Probe	688
Determining the Value of the Probe	689
Computing a Moving Average of a Probed Value	689
Computing a Moving Average Directly	692
Computing a Moving Average of a Resource Directly	694
Specific Attributes	696
Specific Menu Choices	697
N-Dimensional Sample Probe	698
Collecting N-Dimensional Samples from the Model	698
Specific Attributes	701
Specific Menu Choices	701
Parameter Probe	702
Setting the Value of a Parameter	702

Specific Attributes	706
Specific Menu Choices	707
Sample Probe	708
Determining the Value of the Probe	708
Probing Attribute Values that the Model Computes	708
Probing Attribute Values of a Resource Directly	710
Charting Quantitative Parameters	711
Specific Attributes	714
Specific Menu Choices	715
Statistics Probe	716
Determining the Value of the Probe	716
Computing Statistics for a Probed Value	717
Specific Attributes	721
Specific Menu Choices	723
Update Trigger Probe	724
Specific Attributes	724
Specific Menu Choices	724
Accumulate Feed	725
Accumulating Values	725
Specific Attributes	729
Specific Menu Choices	730
Attribute Feed	731
Copying Attribute Values	731
Specific Attributes	735
Specific Menu Choices	737
Change Feed	738
Feeding New Values into Attributes of Blocks	738
Feeding Quantitative Values	739
Feeding Symbolic or Textual Values	744
Feeding New Values into Attributes of Resources	746
Generating Random Numbers Based on a Distribution	749
Specific Attributes	752
Specific Menu Choices	754
Customization Attributes	754
Copy Attributes Feed	755
Copying Attributes from a Block to a Work Object	755
Specific Attributes	759
Specific Menu Choices	760
Increment Feed	761
Incrementing a Counter	761
Specific Attributes	764
Specific Menu Choices	765

Parameter Feed	766
Getting the Value of a Parameter	766
Specific Attributes	769
Specific Menu Choices	770
Timestamp Feed	771
Feeding a Timestamp into a Work Object of the Model	771
Specific Attributes	773
Specific Menu Choices	773

Glossary 775

Index 783

Preface

Describes this guide and the conventions that it uses.

About this Guide **xxv**

Audience **xxvi**

Conventions **xxvi**

Related Documentation **xxvii**

Customer Support Services **xxix**



About this Guide

This guide explains how to use all the features of ReThink to create simulation models of your business process. It teaches you how to:

- [Start the server and connect the client.](#)
- [Organize and control your models.](#)
- [Work with models through the user interface.](#)
- [Create, connect, and configure blocks to describe your business process.](#)
- [Use instruments to obtain metrics about your model and feed key parameters into your model.](#)
- [Constrain your model, based on resource availability.](#)
- [Track performance based on each unit of work that flows through the model.](#)
- [Generate output reports of key metrics and input reports for feeding values into the model.](#)
- [Access external databases for generating, storing, and retrieving work, and for reporting.](#)

- [Run multiple simulation using a script by configuring a Scenario Manager.](#)
- [Communicate with external databases and other systems in real time for online transaction processing.](#)

It also provides two reference sections:

- [Block reference, which describes how to use each block.](#)
- [Instrument reference, which describes how to use each instrument.](#)

Audience

This manual is written for ReThink modelers to help them learn how to use ReThink objects to create models. It explains how each type of ReThink object works and shows examples of typical uses. This manual also contains a reference section that describes how to use each type of block and instrument.

For introductory information about modeling using ReThink, see *Getting Started with ReThink*.

This manual assumes that you have a basic understanding of the G2 environment in which ReThink runs. In particular, ReThink modelers interact with workspaces, objects, classes, and connections. However, this book explains how to perform most G2 operations that you will need to run ReThink.

If you are a ReThink developer who wants to customize the definition of ReThink objects, see the *Customizing ReThink User's Guide*.

Conventions

This tutorial uses the following typographic conventions:

Example	Description
true	Parameter and metric values
task	Glossary terms
<i>c:\Program Files\Gensym\ g2-2011\rethink\ kbs\rethink.kb</i>	Pathnames and filenames

Related Documentation

ReThink

- *Getting Started with ReThink*
- *ReThink User's Guide*
- *Customizing ReThink User's Guide*

G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*
- *G2 System Procedures Reference Manual*
- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*
- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

G2 Utilities

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*
- *G2 GUIDE User's Guide*
- *G2 GUIDE/UII Procedures Reference Manual*

G2 Developers' Utilities

- *Business Process Management System Users' Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*
- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

Bridges and External Systems

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*
- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2 OPCLink User's Guide*
- *G2-PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*
- *G2-HLA Bridge User's Guide*
- *G2 WebLink User's Guide*

G2 JavaLink

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

G2 Diagnostic Assistant

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

Customer Support Services

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone, by fax, and by email.

To obtain customer support online:

➔ Access G2 HelpLink at www.gensym-support.com

You will be asked to log in to an existing account or create a new account if necessary. G2 HelpLink allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.
- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

To obtain customer support by telephone, fax, or email:

➔ Use the following numbers and addresses:

	Americas	Europe, Middle-East, Africa (EMEA)
Phone	(781) 265-7301	+31-71-5682622
Fax	(781) 265-7255	+31-71-5682621
Email	service@gensym.com	service-ema@gensym.com

Modeling Using ReThink

Chapter 1: Running ReThink

Describes how to start the server and connect the client.

Chapter 2: Organizing Models and Controlling Simulations

Describes how to use models, scenarios, and organizers to organize and control models.

Chapter 3: Working with Models

Describes how to work with models through the menus and toolbars.

Chapter 4: Using Blocks

Describes how to clone, connect, and configure blocks to create a model, and how to specify the duration and cost of block activities.

Chapter 5: Using Instruments

Describes how to use ReThink instruments to probe and chart the performance of your model, and how to feed input parameters into your model.

Chapter 6: Using Resources

Describes how to use resources and temporal constraints to constrain your model and compute cost and utilization statistics.

Chapter 7: Using Work Objects

Describes how ReThink uses work objects, which are objects that blocks create, process, and delete when a model is running.

Chapter 8: Using Reports

Describes how to view metrics and enter parameter values through various types of reports.

Chapter 9: Accessing External Databases

Describes how to access databases.

Chapter 10: Using Batch Simulation

Describes how to use run multiple simulations from a script.

Chapter 11: Using ReThink in Online Mode

Describes how to use ReThink in online mode.

Running ReThink

Describes how to start the server and connect the client.

Introduction 3

Starting the Server and Connecting the Client 4

Connecting to a Specific Server at Startup 5

Starting the Server with Your Application Loaded 7

Exiting ReThink 7



Introduction

ReThink is a client/server application. ReThink provides a batch file that, by default, starts the G2 server as a hidden process on the local machine at a default port (1111).

To run ReThink, you must connect the Telewindows client to the server. By default, Telewindows automatically connects to the server running on the local machine on the default port.

You can run ReThink in a secure G2 environment, which means users must provide a password before ReThink grants access to a KB. User names and passwords are stored in the *g2.ok* file. For details on how to configure ReThink to run in a secure G2 environment, see Chapter 54 "Licensing and Authorization" in the *G2 Reference Manual*.

Starting the Server and Connecting the Client

You can start the server and connect the client by using the Start menu.

To start the server and connect the client:

- 1 Choose Start > Programs > Gensym G2 2011 > G2 G2 ReThink > Start G2 ReThink Server.

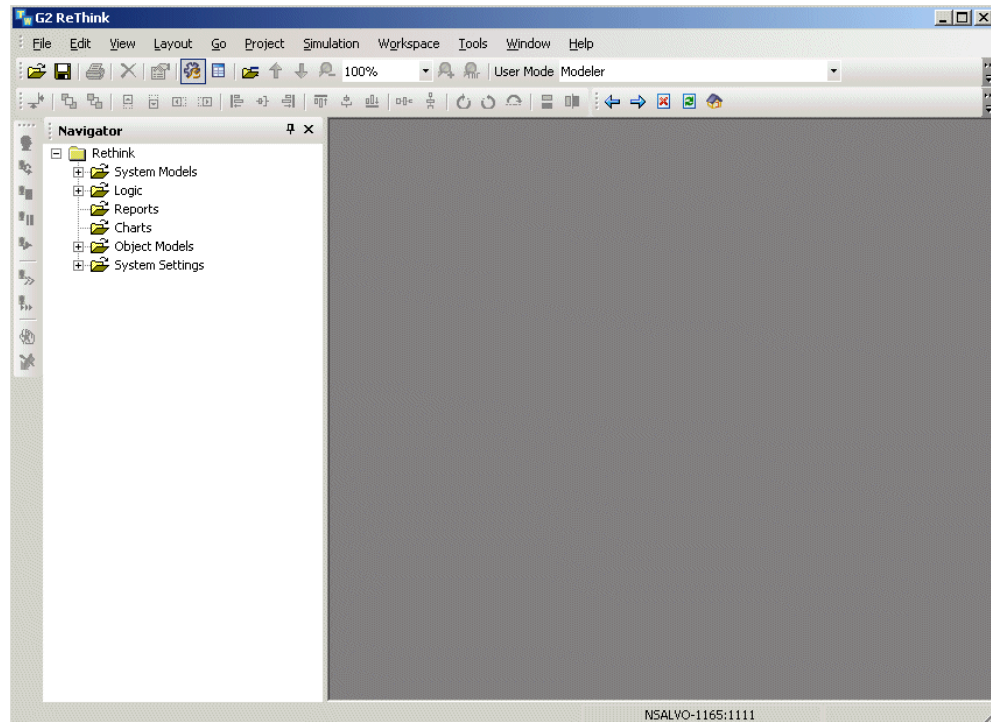
This menu choice starts the G2 server, using the *StartServer.bat* batch file, located in the *g2* directory of your ReThink installation directory. It starts the server on the local machine on TCP/IP port number 1111, and it automatically loads the KB named *rethink-online.kb*.

When the server has been started, the G2 icon appears in the system tray. When the server is running, the icon looks like this: 

- 2 Once the server is running, connect the client in one of two ways:
 - ➔ To connect Telewindows to the server running on the default host and port, choose Start > Programs > Gensym G2 2011 > Telewindows Next Generation.
 - or**
 - ➔ To connect Telewindows to the server running on the local host on the current port, right-click the G2 icon in the system tray and choose Connect Telewindows.

The Telewindows client is now connected to the G2 server.

When the client is connected and all files have finished loading, you will see this window:



Connecting to a Specific Server at Startup

You can run the ReThink client and server on different computers, or multiple ReThink servers on the same computer.

You can:

- [Connect the client directly to the server.](#)
- [Start the server on a specific port.](#)
- [Connect the client to a specific server.](#)

Connecting the Client to the Default Server

To connect the client to the default server:

- 1 Start the ReThink server from the Start menu.

By default, the server starts on the local host at port 1111. Each time you start a new server on the same machine, the port number increments by one. For example, if you start another server, the port number would be 1112.
- 2 To determine the host and port, hover the mouse over the G2 server icon in the system tray.

For example, *MY-HOST:1111* means the server is running on the machine named *MY-HOST* at port 1111.
- 3 Right-click the G2 server icon in the system tray and choose Connect Telewindows.

The Telewindows client connects to the specific host and port of that server.

Starting the Server on a Specific Port

To start the server on a specific port:

- 1 Right-click the Start G2 ReThink Server menu choice in the Start menu and choose Create Shortcut.
- 2 Rename the shortcut and/or drag it to your desktop, as needed.
- 3 Display the properties dialog for the shortcut and click the Shortcut tab.
- 4 Configure the Target property in the dialog to be the specific port on which to start the server, using the *-tcpport* command-line option.

For example, to start the server on port 1115, the shortcut would look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\StartServer.bat"  
-kb ..\rethink\kbs\rethink-online.kb -nowindow -tcpport 1115
```

Connecting the Client to a Specific Server

To connect the client to a specific server:

- 1 Create a shortcut to the *twng.exe* file located in the *g2* directory of your ReThink product installation, either directly or by creating a shortcut from the Telewindows Next Generation menu choice in the Start menu.
- 2 Display the properties dialog for the shortcut and click the Shortcut tab.

- 3 Configure the Target by appending the host and port of the ReThink server to which to connect, using this syntax: *host :port*.

For example, to connect to *my-host* at port *1115*, the shortcut would look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\twng.exe"  
my-host:1115
```

Starting the Server with Your Application Loaded

By default, the server starts up with the default ReThink application running, *rethink-online.kb*. Once you create an application, you might want to create a shortcut to the ReThink server that automatically loads your application at startup.

To start the server with your application loaded:

- 1 Copy the Start G2 ReThink Server shortcut from the Start menu.
You can rename the shortcut and drag it to your desktop, as needed.
- 2 Display the properties dialog for the shortcut and click the Shortcut tab.
- 3 Configure the application to load by editing the Target.

For example, to load the application named *aero.kb* located in the *\rethink\examples* directory, the Target should look like this:

```
"C:\Program Files\Gensym\g2-2011\g2\StartServer.bat"  
-kb "c:\Program Files\Gensym\g2-2011\rethink\examples\aero.kb"  
-nowindow
```

Exiting ReThink

To exit ReThink, you disconnect the client from the server, then shut down the server. By default, you can only exit the server directly from the client in Developer mode.

To disconnect the client from the server:

- ➔ Choose File > Close.

To exit the server:

➔ Right-click the G2 server icon in the system tray and choose Shut Down G2.

or

- 1** Choose Tools > User Mode > Developer.
- 2** Choose File > Exit.
- 3** Click Yes in the confirmation dialog.

Organizing Models and Controlling Simulations

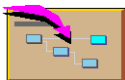
Describes how to use models, scenarios, and organizers to organize and control models.

Introduction	9
Working with Projects	11
Configuring the Model Environment	13
Controlling the Simulation	17
Configuring the Scenario	20
Performing “What-if” Analysis on a Model	28
Working with Large Models	32
Viewing Demo Models	37
Customizing Scenarios	38



Introduction

The first step in building a ReThink model is to create a new project. ReThink takes care of loading and saving all the required modules for you; therefore, we recommend that you always create a new project.



The next step is to create a **model**, which contains the blocks, resources, and instruments that describe your business process. You place these objects on the subworkspace of the model, which is called the **detail**.



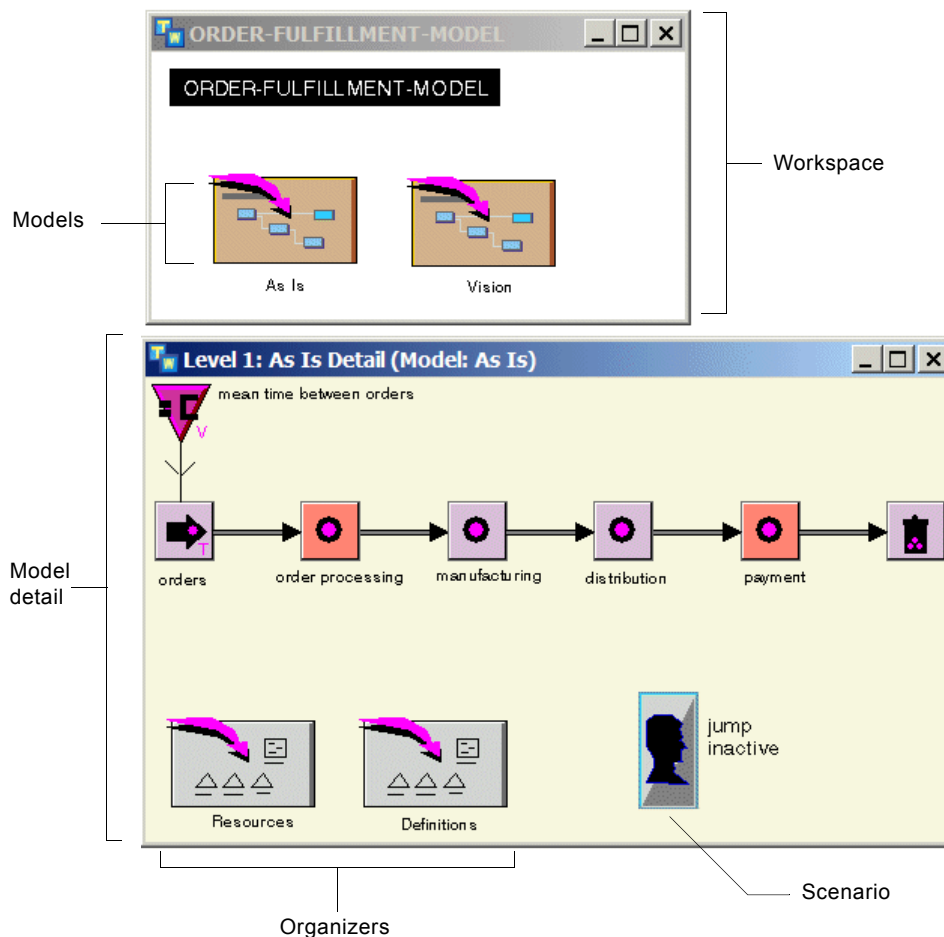
A model typically has one or more **organizers**, which contain various types of objects that the model requires but that are not directly connected to the model, such as work object definition classes or resources. You place these objects on the organizer's detail.



Each model must define a **scenario**, ReThink's discrete event simulation engine, which tracks events by using a simulation clock. A scenario controls various aspects of the model, including the mode, status, and **simulation time**, which represents the current time at which the simulation is running.

Once you have created a model of your process, you typically experiment with alternative scenarios and processes to test the performance of the model under different conditions. For example, you might test the model under a heavy load and a light load to see how it performs, or you might test the same set of input parameters against a model with fewer resource constraints. Thus, scenarios serve as control centers for a model, allowing you to perform **"what-if" analysis**.

This figure shows the top-level workspace for an order fulfillment process that defines two alternative models and the detail of one of the models:



This topic describes how to:

- [Work with projects.](#)
- [Configure the model environment.](#)
- [Control the simulation.](#)
- [Configure the scenario.](#)
- [Perform “what-if” analysis on a model.](#)
- [Work with large models.](#)
- [View demonstration models.](#)
- [Customize scenarios.](#)

Working with Projects

A ReThink project consists of a set of related files that form a knowledge base. Each file contains a stand-alone **module**, which together make up a **module hierarchy**. Each module is associated with its own `.kb` file, whose name typically corresponds to the module name. The module hierarchy consists of a top-level module and a number of lower-level modules. The top-level module requires the lower-level modules to run.

You can:

- [Create a new project.](#)
- [Save a project.](#)
- [Open a project.](#)

Creating a New Project

When creating a new project, ReThink creates a new, blank project with the name you enter. The new project is saved in the `projects` directory of your ReThink installation directory.

Note Creating a new project replaces the existing model in memory. Therefore, before you create a new project, be sure to save the existing project, as necessary.

To create a new project:

- 1 Choose File > New.
- 2 Enter the name of the project.
The project name cannot contain spaces.
- 3 Ensure that ReThink is chosen as the selected library.
- 4 Check or uncheck any additional libraries, depending on how your application needs to access external data.
- 5 Click OK.


ReThink displays the Operator Logbook as it creates a new project with the name you specify, then loads the new project and all required modules onto the server. When all modules have been successfully loaded, the menu bar updates. You must wait until the KB has finished loading in the server before you can access your project.

Saving a Project

Projects are stored in the *projects* directory of your ReThink installation directory.

ReThink saves the top-level module only; it does not save the required modules. Unless you are customizing ReThink, you do not generally need to save the required modules.

To save a project:


- ➔ To save a project to the project file that was loaded when you started the client, choose File > Save or click the equivalent toolbar button: 
- or
- ➔ To save the project to a different project file, choose File > Save As, enter a new project name, or choose an existing project name from the list of available projects on the server.

Opening a Project

To open a project, specify the project name associated with the top-level module in the module hierarchy.

Note Opening a new project replaces the existing application in memory. Therefore, before you open a new project, be sure to save the existing project, as necessary.

To open a project:

- 1 Choose File > Open or click the equivalent toolbar button () to display the Open Project dialog.
- 2 Enter or choose the project to open and click Open.
- 3 Click Yes in the confirmation dialog.

ReThink displays the Operator Logbook as it loads the project file and all required modules onto the server. When all modules have been successfully loaded, the menu bar updates. You must wait until the KB has finished loading in the server before you can access the application.

Configuring the Model Environment

When you first begin modeling your business process, you typically create a **stand-alone model**, which contains all aspects of the complete model: blocks, resources, instruments, charts, user interface elements, and class definitions.

You create your ReThink model by creating and configuring these objects within the model environment:

- **Model** – A container in which you build your ReThink model.
- **Scenario** – The discrete event simulation engine for controlling the simulation.
- **Organizer** – A container in which you place various objects in your model.


Note Every model must have a Scenario; otherwise, you cannot configure certain aspects of the model.

Creating a Model

You create and organize models through the Projects menu, in which case ReThink keeps track of where the model is located.

You can also create models directly from the ReThink toolbox and place them on a top-level workspace.

To create a model:


- 1 Do one of the following:
 - ➔ Choose Project > System Models > Business Processes > Manage, then click the New button () to display the properties dialog for creating a new model.

or

→ In the Navigator, expand System Models in the tree, mouse right on the Business Processes node, and choose New Instance.

- 2 Configure the Label to name your model.
- 3 Configure the Best Practice URL, as needed.


The Best Practice URL can reference any HTML file, either on the World Wide Web or on the file system, or any RTF file on the file system, which describes the model. When this attribute is configured, choosing Show URL or clicking the model displays the file in a browser window.

- 4 Configure the Model Version to be any number to uniquely identify the model.
- 5 Click OK to add the model to the Manage dialog and Navigator.
- 6 In the Manage dialog, select your model and click the Model button () to show the model detail, or in the Navigator, mouse right on the model and choose Show Detail.

You create your model on this detail.

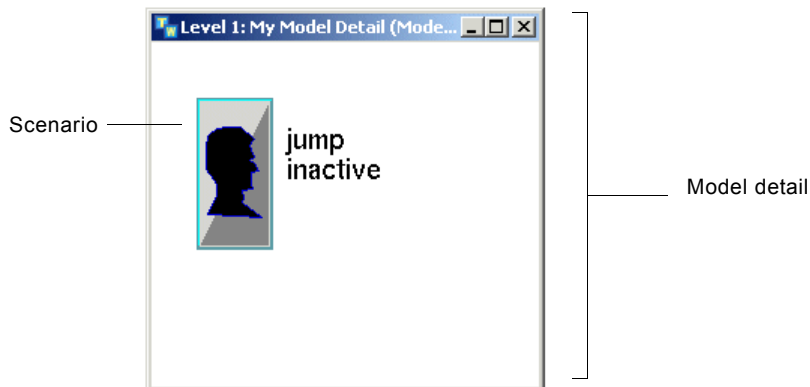
- 7 Choose View > Toolbox - ReThink and click the Tools button:



- 8 Select a Scenario and click to place it on the model detail.
- 9 Choose Layout > Shrink Wrap or click the equivalent toolbar button: 

The workspace and window size adjust to fit the model. Drag the Scenario to adjust the borders of the workspace.

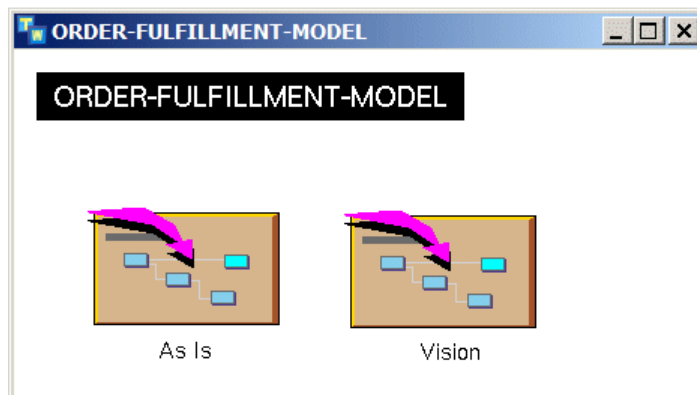
The model detail should look similar to this:



Alternatively, to create a model and place it on a top-level workspace:

- 1 Choose Workspace > New.
For more information, see [Creating and Accessing Top-Level Workspaces](#).
- 2 Display the ReThink toolbox and click the Tools button.
- 3 Create a Model object and place it on the workspace.
- 4 Choose Show Detail on the model to show its detail.

For example, here is a top-level workspace with two models:

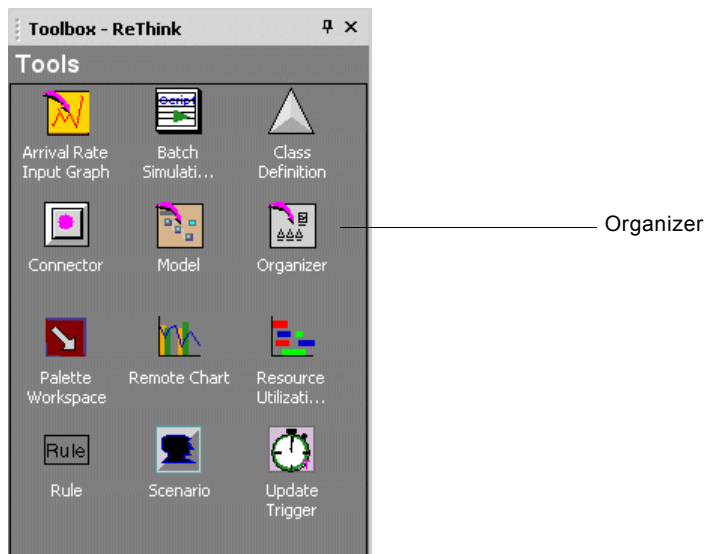


Creating an Organizer

When you run a model, ReThink automatically creates class definitions for the work objects that the model processes. Typically, you place these class definitions on the detail of an Organizer. You can also store other objects such as resources on the detail of an organizer. You use an organizer for any object that the model requires to run but that is not directly connected to the model.

To create an organizer:

- 1 Choose View > Toolbox - ReThink and click the Tools button:

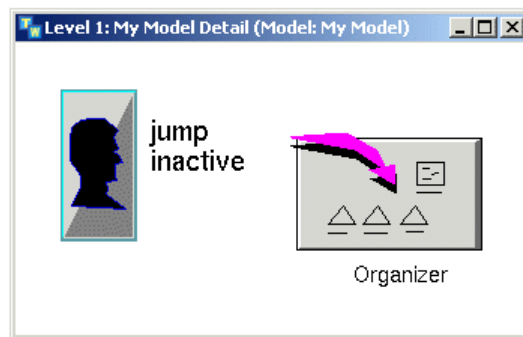


- 2 Select an Organizer and click to place it on the model detail.
- 3 Choose Properties on the organizer.
- 4 Edit the Label parameter to be any text, then drag the label to the desired location next to the organizer.
- 5 Choose Create Detail on the organizer.

The organizer detail appears in its own window. You place objects on this detail, as needed.

- 6 Choose Show Detail on the organizer to show the detail.

This figure shows a model with a scenario and an organizer on the model detail:



Note If an organizer's detail contains resources, the organizer must be associated with a scenario. In this case, we recommend that you place organizers on the model detail, rather than on the same workspace.

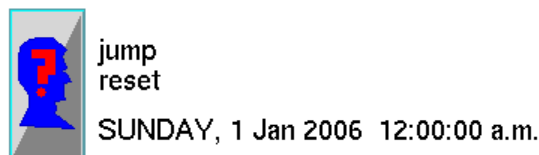
Controlling the Simulation

You control the simulation from the Simulation menu or toolbar. These menu choices and toolbar buttons affect the scenario, which is the control center for running the simulation. Every model must contain a scenario.

To control the simulation, you:

- [Activate and deactivate the scenario.](#)
- [Start and stop the simulation.](#)

The scenario shows the simulation mode, status, and current simulation time, for example:

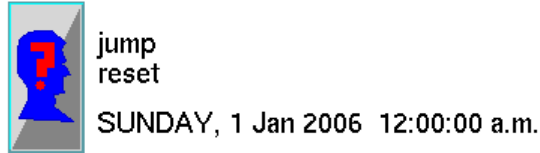


Activating and Deactivating the Scenario

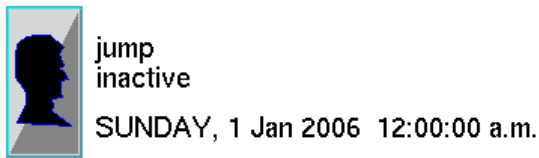
Before you can run a simulation, you must activate the Scenario. Certain menu choices, such as Show Scenario, are only available when the scenario is active.

The following figure shows an active and inactive scenario:

active




inactive





Note Starting the simulation automatically activates the scenario; therefore, typically, you do not need to activate the scenario explicitly.



To activate and deactivate a scenario:

➔ Display a model that contains a scenario, then choose Simulation > Activate or click the equivalent toolbar button () to toggle activation.

Starting and Stopping the Simulation

You control the status of the simulation by using menu choices in the Simulation menu, by using keyboard shortcuts, or by clicking the equivalent toolbar button, as follows:

Menu Choice/ Shortcut	Toolbar Button	Status	Description
Start All		running	Start the simulation running.
Reset		resetting	Reset the simulation.

Menu Choice/ Shortcut	Toolbar Button	Status	Description
Pause		paused	Pause the simulation.
Continue		running	Continue running the paused simulation.

By default, the simulation runs in Jump mode, which advances the simulation clock continuously with each discrete event. For alternative ways to run the simulation, see [Configuring the Simulation Mode](#).

To start and stop the simulation:

- 1 Start the simulation by choosing Simulation > Start All or by clicking the equivalent toolbar button.

Choosing Start All activates the scenario and resets the simulation before starting.

Resetting the simulation can take a period of time, depending on the size of the model and the speed of your computer. The status of the scenario is resetting.

Once the simulation is reset, you will begin to see objects flowing through the model. You can only see these objects as they flow through the model if animation is enabled. The model also begins computing metrics. The status of the scenario is running.

- 2 To pause the simulation, choose Simulation > Pause or click the equivalent toolbar button.

Pausing the simulation stops the simulation clock, causes all objects to stop flowing through the model, and stops computing metrics. The status of the scenario is paused.

- 3 To resume running the simulation after pausing, choose Simulation > Continue or click the equivalent toolbar button.

The simulation clock begins advancing again, objects begin flowing through the model, and the status of the scenario is running.

- 4 To reset the simulation, choose Simulation > Reset or click the equivalent toolbar button.

Resetting the simulation resets the simulation clock, deletes all objects that the model created, and resets all metrics to their initial values.

Configuring the Scenario




You can configure these features of the scenario:

- [Simulation mode](#), which determines whether the simulation runs continuously, step-by-step, in real time, or online.
- [Duration](#) of the simulation.
- [Version](#) of the simulation.
- [Start time](#) of the simulation.
- [Simulation speed](#).
- [Animation](#).
- [Speed at which objects flow along paths](#).
- [Indicator arrow](#) behavior.
- [Computational behavior](#) of the scenario.
- [Random number generation](#).

Configuring the Simulation Mode

You can run the simulation in one of four modes by using the menu choices in the Simulation menu or the equivalent toolbar button, as this table describes:

Menu Choice	Toolbar Button	Mode	Description
Jump Mode		jump	The normal discrete event simulation mode. Events occur in their normal time sequence while a simulation is running, but the real-time clock advances non-linearly relative to the simulation clock. After each discrete event, ReThink immediately advances the simulation clock to the start time of the next event. Objects flow through the model without stopping.
Step Mode		step	The mode you use for careful examination of the model. ReThink pauses after each event so you can walk through the simulation one step at a time. When you continue running the simulation, the clock immediately advances to the start time of the next event, then stops.

Menu Choice	Toolbar Button	Mode	Description
Synch Mode		synch	The mode you use to help visualize the relative times between events. ReThink scales the simulation time to real time. For example, you can use this mode to run the simulation at one hour per second of real time. Most of the time when you are running a simulations, however, you let the simulation clock keep track of the time by using either jump or step mode.
Online Mode	N/A	online	The mode you use to run ReThink for online transaction processing. For details, see Using ReThink in Online Mode .

You configure the simulation mode from the Simulation menu or toolbar, or on the properties dialog for the scenario.

Running the Simulation in Jump Mode

To run the simulation in jump mode:

- 1 Choose Simulation > Jump Mode, click the equivalent toolbar button, or display the properties dialog for the Scenario and configure the Mode to be Jump.
- 2 Choose Simulation > Start All or click the equivalent toolbar button to start the simulation

Running the Simulation in Step Mode

To run the simulation in step mode:

- 1 Choose Simulation > Step Mode, click the equivalent toolbar button, or display the properties dialog for the Scenario and configure the Mode to be Step.
- 2 Choose Simulation > Start All or click the equivalent toolbar button to take a single step in the simulation, then pause the simulation.
- 3 Choose Simulation > Continue or click the equivalent toolbar button to take a single step for each discrete event, then pause the simulation.

Running the Simulation in Synch Mode

When you run the model in synch mode, you must configure the proportion of simulation time to real time. The larger the number, the faster the overall execution time of the simulation. By default, the value is 1, which means the timing parameters contribute exactly the specified amount of simulation time to

the simulation clock. A value of 2 means they contribute half the amount of time. For example, if a timing parameter specifies 10 minutes to perform a task, and the Seconds per Tick is 2, the parameter contributes 5 minutes of simulation time.

Note If your model is complex, ReThink may not be able to keep up with the specified synch rate. For example, if you specify a synch rate such that one minute of simulation time is equivalent to one year of real time, ReThink might require more than one minute to process a year-long simulation. If this is the case, ReThink gives no indication that the synch rate is too slow; however, the metrics that the simulation computes are correct.

To run the simulation in synch mode:

- 1 Display the properties for the scenario, click the Scenario tab, and configure Seconds per Tick to be a number greater than one.
- 2 Choose Simulation > Synch Mode, click the equivalent toolbar button, or display the properties dialog for the Scenario and configure the Mode to be Synch.
- 3 Choose Simulation > Start All to run the simulation.
- 4 Choose Simulation > Pause to pause the simulation.
- 5 Choose Simulation > Continue to continue running the simulation in synch mode.

Configuring the Duration of the Simulation

Depending on your model, you typically configure the scenario to run for a specific duration, such as a month, a year, or ten years. You do this by specifying the ending time of the simulation. A duration of 0 means the simulation will run for the maximum allowable simulation time, which is 17 years.

Tip To ensure accurate reporting, configure the duration of the simulation to be slightly longer than the end of the last update period.

To configure the duration of the simulation:

- ➔ Display the properties dialog for the scenario, click the Scenario tab, and configure the Duration to be the amount of time the simulation should run, for example, 3 months or 2 years.

Configuring the Simulation Version

When performing “what-if” analysis, you often use different scenarios with the same or with different versions of the model. When using the Scenario Manager to run multiple simulations from a script, you typically output the data to a report. To identify which scenario was used to generate the data, you configure the version of the simulation, using a unique number.

To configure the simulation version:

- ➔ Display the properties dialog for the scenario, click the Scenario tab, and configure the Simulation Version to be a unique number.

For example, you might use 1.0, 1.1, and 1.2 for the scenario associated with three different versions of a model.

Configuring the Start Time of the Simulation

By default, ReThink uses January 1, 2006 as the start time of the simulation. You might want to start the simulation at a different time.

To configure the start time of the simulation:

- ➔ Display the properties dialog for the scenario, click the Start Time tab, and configure the Year, Month, Day, Hour, Minute, and/or Second to be the start time of the simulation.

When you reset the simulation, the new start time appears.

Configuring Simulation Speed

By default, the scenario is configured to run as fast as possible, which means that when you are running in jump or synch mode, the clock advances with each new event as fast as it can. This default configuration is desirable when:

- Running simulations from a script, using the Scenario Manager.
- You do not need to interact with the model while the simulation is running.
- You do not need to visualize the animation of objects as they flow through the model.

Depending on the speed of your computer, running the simulation as fast as possible often means that you will experience delays when interacting with the user interface while the simulation is running. For example, when you click the Pause button to pause the simulation, the simulation clock continues to advance until the processing has caught up with the user interaction. Similarly, you will experience delays when configuring values through the dialogs or reports while the simulation is running.

If you want to interact with the user interface while the simulation is running, for example, to configure parameter values, you should slow the simulation down to allow more time for the user interface to respond.

To slow the simulation down, you configure the simulation speed to be a larger number. Depending on the speed of your computer, you might try a simulation speed of 10 or 15.

Note Once you have configured your model, we recommend that you set the Simulation Speed to 0 for optimal performance, the default.

To configure the simulation speed:

- ➔ Display the properties dialog for the scenario, click the Options tab, and adjust the Simulation Speed, where the larger the number, the slower the speed.

Configuring Animation

ReThink animates running simulations by physically moving objects along paths and highlighting blocks as they become active. You can use your model as a communication tool to visualize work objects as they flow along paths.

Depending on the speed of your computer, you might need to adjust the animation speed. In particular, if you have a relatively fast computer, you might need to slow down the animation speed to better visualize the flow of work objects. However, keep in mind that performance degrades the slower the animation speed.

You configure animation speed, based on the number of milliseconds it takes to move an object from the beginning of a path to the end of a path. By default, it takes an object 5 milliseconds to move along a path. To slow down the animation, you might want to move an object along the path in 8 or 10 milliseconds.

Note Once you have configured your model, we recommend that you disable animation for optimal performance.

To disable animation for a simulation:

- ➔ Display the properties dialog for the scenario, click the Options tab, and click the Enable Animation option off.

To adjust animation speed:

- ➔ Display the properties dialog for the scenario, click the Options tab, and adjust the Animation Speed, where the larger the number, the slower the speed.

Configuring Object Tracking

You can configure the scenario so that objects that flow through the model keep track of the blocks through which they have passed since they were created. When object tracking is enabled, you can pause the simulation and show tracking for any object. Object tracking is a useful way of verifying the model to ensure objects flow along the correct paths.

When showing object tracking, ReThink animates all the blocks upstream of the current block, in order, starting at the current block and ending at the block that created the object. ReThink repeats this process four times, by default. You can configure the color used for animating blocks when showing object tracking.

To configure object tracking:

- 1 Display the properties dialog for the scenario, click the Options tab, and click the Enable Tracking option on.
- 2 Configure the Animation Repeat Counter to be the number of times to animate the blocks when showing object tracking.
- 3 Configure the Animation Color to be the highlight color to use.

To show object tracking for an object:

- ➔ Pause the simulation, then choose Show Flow History on a object in the model.

Configuring the Behavior of Indicator Arrows



ReThink places indicator arrows next to objects when you show various objects associated with other objects, such as when you show the scenario associated with a block.

By default, the indicator arrow remains on the workspace until you explicitly hide it. You can clear all indicators automatically, or you can configure the scenario to clear indicators automatically after a certain number of seconds.

Note To clear all indicators, the scenario must be active, as described in [Activating and Deactivating the Scenario](#).

To clear an individual indicator arrow:

- ➔ Left click the indicator arrow.

To clear all indicator arrows:

- ➔ Activate the scenario, then choose Simulation > Clear Indicators or click the equivalent toolbar button: 

To clear indicators associated with a scenario after a timeout period:

- 1 Display the properties dialog for the scenario, click the Options tab, and configure the Indicate Mode by choosing Timeout.
- 2 Configure the Timeout period after which indicators should automatically disappear, in seconds.

Configuring the Computation Behavior

By default, ReThink computes metrics for all blocks in the model, including Task blocks with detail and blocks on the detail itself. Sometimes, you are only interested in the metrics for blocks on the detail, not the Task block itself.

You can disable the computation of metrics for Task blocks with detail to improve the performance of a simulation. If your model contains numerous such tasks, you will notice a significant increase in performance simply by disabling metrics for these high-level blocks. You control this behavior by configuring the scenario, which affects all blocks associated with the scenario.

When you disable the computation of Task blocks with detail, ReThink no longer computes duration, cost, or block metrics for any top-level Task blocks associated with the scenario. Also, ReThink no longer animates top-level tasks, regardless of the specification of Enable Animation.

Note If you disable the computation of metrics while a model is running, ReThink stops computing metrics at that point. If you later resume the computation of metrics, the computed values will not be accurate for the current simulation. Therefore, we recommend that you enable and disable metrics before you run the simulation.

You can configure these parameters related to computation behavior:

Parameter	Description
Compute All Blocks	Enables the computation of Task blocks with detail. The default value is off, which means ReThink computes values for Task block details only; it does not compute values for the block itself.
Update Charts	Enables the updating of charts. The default value is off, which means that you must update charts manually or by using a button or a rule. For details, see Updating Charts .
Enable Metrics Toolbar Update	Disables the updating of the Metrics toolbars. The default value is on, which means that the Metrics toolbars update once every half second.

Note Once you have configured your model, we recommend that you disable updating of the Metrics toolbar for optimal performance.

To compute metrics for Task blocks with detail:

→ Display the properties dialog for the scenario, click the Options tab, and enable the Compute All Blocks option.

To update charts automatically:

→ Display the properties dialog for the scenario, click the Options tab, and enable the Update Charts option.

To disable metrics toolbar updating:

→ Display the properties dialog for the scenario, click the Options tab, and disable the Enable Metrics Toolbar Update option.

Configuring the Scenario to Generate Identical Random Numbers

By default, the scenario generates random numbers that vary with each simulation by generating a new seed number for each simulation. You might want to run a simulation with identical random numbers to test different model configurations, using the same set of random numbers.

ReThink generates random numbers to compute mathematical distributions such as Random Normal, Random Exponential, or Random Triangular. You use these random distributions to generate values for the Duration attribute of blocks.

To configure the scenario to generate identical random numbers:

- 1 Display the properties dialog for the Scenario and, on the Scenario tab, configure the Seed Value to be an integer that holds the seed value, which ReThink uses as the basis for randomly generated numbers.

By default, it generates a random number as the Seed Value.

- 2 Disable the Generate New Seed option to use the specified Seed Value for each simulation.

- 3 Reset the Scenario to use these new values.

Each time you run the simulation, using this scenario, ReThink uses the same set of randomly generated numbers.

Performing “What-if” Analysis on a Model

Once you have created a stand-alone model of your business process, you will typically want to perform “what-if” analysis on your model. You do this by creating and testing alternative scenarios to begin re-engineering your process.

ReThink lets you experiment with alternative scenarios by allowing you to:

- [Copy and modify an existing model and compare the results](#) against the existing model.
- [Associate different scenarios with the same model](#) to test different input parameters against a single model.
- [Use the same scenario with different models](#).

Comparing Different Versions of the Same Model

You might want to compare two versions of the same model to perform “what-if” analysis. For example, you might test a model by using different resource constraints, or you might compare the performance of two slightly different work flows, one that synchronizes work and the other that does not.

To do this, copy an existing model and edit the configuration of one of the models.

Note You should always reset and deactivate a model before you copy it; otherwise, the copied model will contain work objects that you cannot delete by resetting.

Caution When you copy a model that has resources or pools that have been assigned to a block, you must reassign those resources and pools to objects in the new model. For resources, you must choose the resource associated with the manager; otherwise, the resource will not be used in the new model. For pools, you must reassign the pool to the block; otherwise, the block will use the pool in the *original* model.

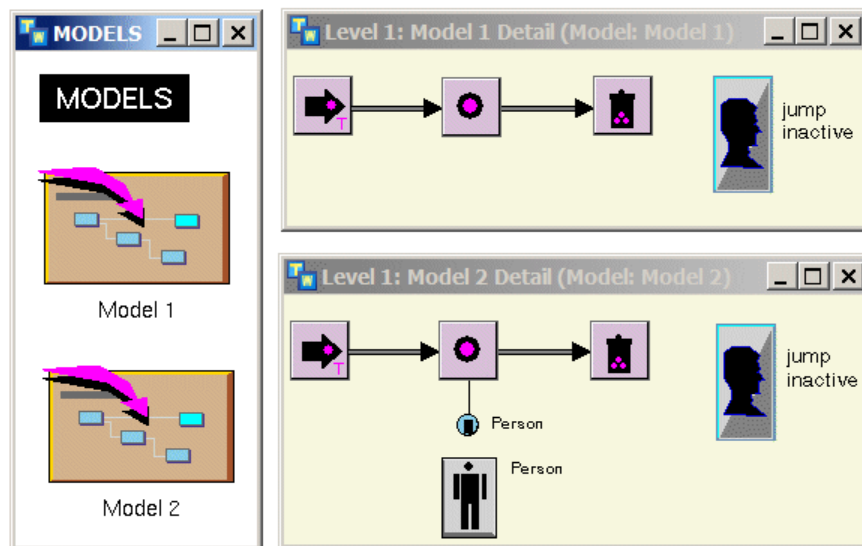
To compare different versions of the same model:

- 1 Reset the model and deactivate the scenario.
- 2 Select the model you want to copy and choose Edit > Clone or choose Clone from the popup menu, then click on a workspace to copy the model.
- 3 Name the new model to distinguish it from the original.

- 4 Display the model’s detail and reconfigure the model.
For example, you might add or remove resources, change the durations or costs of the blocks, or change the blocks the model uses to describe the process.
- 5 If the model contains resources, choose the Choose Resource menu choice on each Resource Manager in the model, then choose Select on the associated resource.
- 6 If the model contains blocks with associated pools, choose the Choose Pool menu choice on each block, then choose Select on the associated pool.
- 7 Run the two models independently of each other, either simultaneously or separately.
- 8 Compare the results through dialogs and reports.

Once you have copied a model, you can do a side-by-side comparison of the models. You can run the simulations simultaneously to compare the results while they are running, or you can run them independently to compare the results at the end of the simulation.

For example, here are two models, one of which is constrained by resources and the other of which is not:



Model 1 has no resource constraints, while Model 2 does.

Using Different Scenarios to Compare the Same Model

Another way of performing “what-if” analysis on a model is to compare the outputs of different sets of input parameters against the same model. For example, you might want to run the same model under light and heavy capacity and compare the performance results.

To do this, you associate multiple scenarios with a single model, only one of which can be active at a time:



Model

A model can have multiple associated scenarios.



Scenario A
jump
inactive

MONDAY, 1 Jan 2006 12:00:00 a.m.



Scenario B
jump
reset

MONDAY, 1 Jan 2006 12:00:00 a.m.

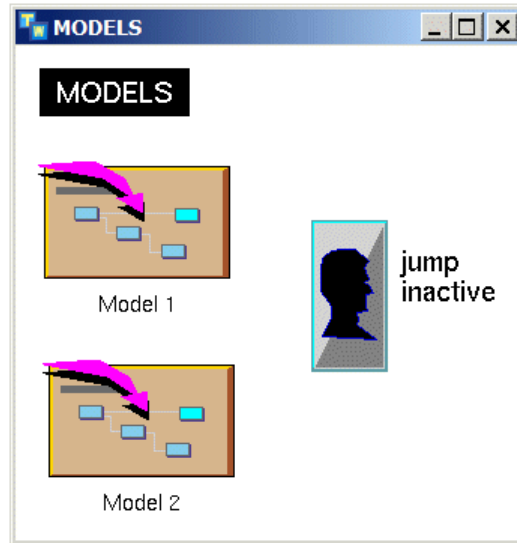
To use multiple scenarios with a single model:

- 1 Create two scenarios, either from the toolbox or by copying an existing scenario.
- 2 Activate one of the scenarios.
The icon and status value change.
- 3 While one scenario is active, activate the other scenario.

ReThink automatically deactivates the currently active scenario and activates the other scenario, because only one scenario can be active at any time.

Using a Single Scenario to Control Multiple Models

In the early stages of developing alternative models, when all you are doing is controlling the status and mode of each model, you can associate the same scenario with multiple models, as this figure shows:



When the scenario controls all the models in the same way, you can place it on the same workspace as the models.

If you use this configuration, the scenario now controls both models. For example, the Start All button starts the sources of *both* models.

Caution When you copy a model that has resources or pools that have been assigned to a block, you must reassign those resources and pools to objects in the new model. For resources, you must choose the resource associated with the manager; otherwise, the resource will not be used in the new model. For pools, you must reassign the pool to the block; otherwise, the block will use the pool in the *original* model.

Working with Large Models

If you have a large model, you might want to use one or both of the following techniques to break up your model into more manageable pieces:

- Associate two connectors on different details, for example, by:
 - [Associating connectors on Task block details](#) when the top-level tasks are not connected.
 - [Associating connectors on model details](#).
- [Replace the default detail associated with a model or organizer](#) with a top-level workspace that you create.

By using these features in conjunction with modules, you have a powerful way of dividing up large models into separate modules so that multiple people can develop and run different parts of the model simultaneously.

For more information, see *Customizing ReThink User's Guide*.

Associating Existing Connectors on Task Block Details

You might have a model in which connecting two top-level Task blocks with detail causes paths to overlap in a way that is distracting to the overall readability of the model. One solution is to delete the top-level connection between the tasks and associate the connectors on the detail. That way, work objects can flow between the two tasks even though the top-level Task blocks themselves are not connected. You can then move the top-level tasks to improve readability, or you can move one of them to an entirely different workspace.

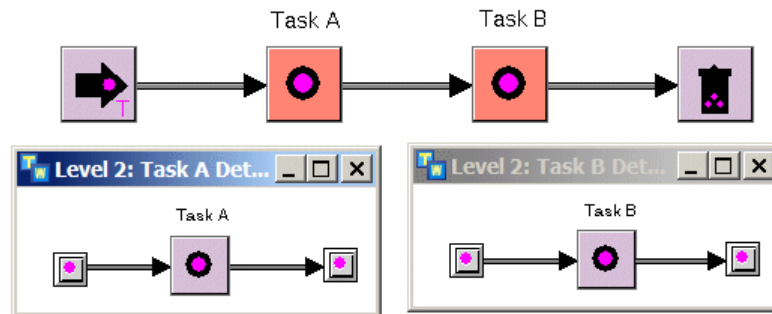
The following set of steps describe how to associate existing connectors on the details of two Task blocks.

To associate existing connectors on Task block details:

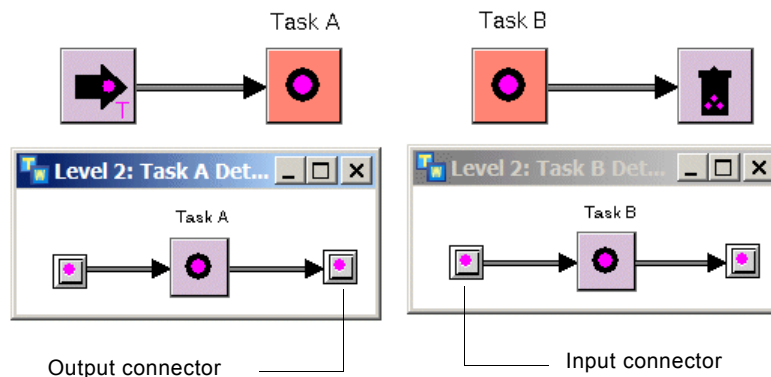
- 1 Start with a model in which two Task blocks with detail are connected.

In the following model, suppose you want the details to be connected, but you do not want the top-level tasks to be connected. For example, you might want to place each Task block on a different workspace, or you might have a

complex model on a single workspace in which you want to disconnect the top-level tasks to make the workspace easier to organize.

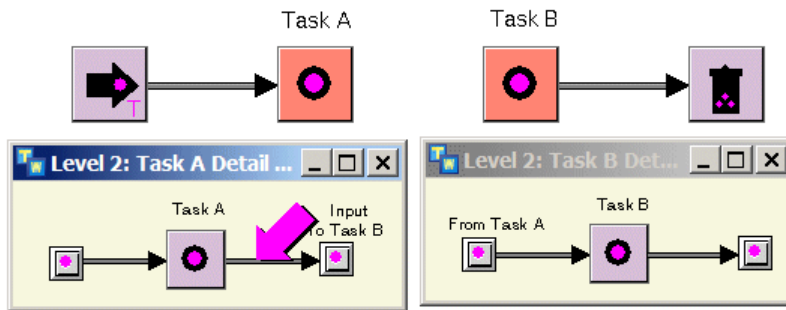


- 2 Delete the connection between the top-level Task blocks, then delete the stubs. The model should look something like this:

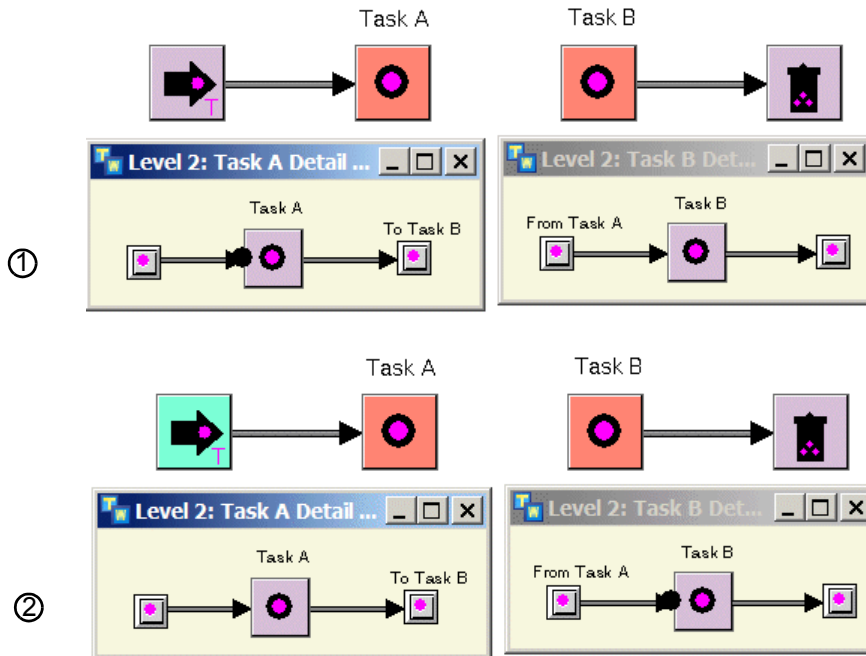


- 3 In the properties dialogs for the output connector on Task A's detail and the input connector on Task B's detail, configure the Label to identify each connector, for example, To Task B and To Task A.
- 4 Choose the Choose Connector menu choice on the output connector on Task A's detail.
- 5 Choose Select from the input connector's menu on Task B's detail to select it. ReThink displays an indicator arrow next to the selected connector. The details are now connected, even though the high-level tasks are not connected.
- 6 To verify that the connectors are connected, click one of the connectors.

An indicator arrow points to the connector to which the selected connector is associated:



When you run the simulation, work objects flow across the details of the two tasks:



Associating Connectors on Other Types of Details

Suppose you have a large model that you want to break up into two different models. You can do this by associating connectors on the details of two models. To do this, you use the Connector located on the Tools palette of the ReThink toolbox.

You can associate connectors on any type of detail with connectors on any other type of detail, for example:

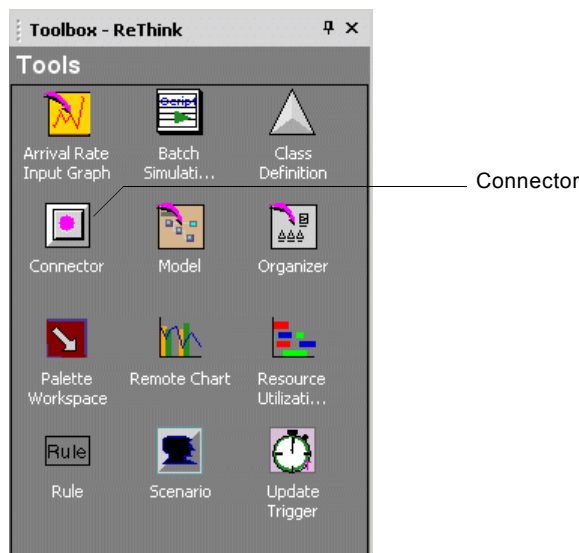
- A connector on a model detail with a connector on a Task block detail.
- A connector on a Task block detail that is one level deep in the hierarchy with a connector on a Task block detail that is three levels deep.

Note When adding new connectors to Task block details, always delete the unconnected paths on the top-level blocks first; otherwise, ReThink creates additional connectors for you when you show the detail.

The following set of steps and examples show how to associate connectors on two Model details.

To associate connectors on Model details:

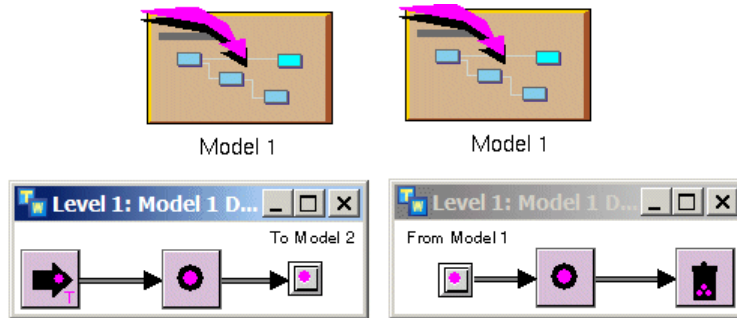
- 1 Create two models and show their details.
- 2 Create a model on each detail, such that work objects flow from the last block on the first model to the first block on the second model.
- 3 Display the Tools palette of the ReThink toolbox:



- 4 Create two connectors, connect one to the output path of the first model, and connect the other to the input path of the second model.
- 5 Label each connector to indicate the source and destination of the work objects.
- 6 Choose the Choose Connector menu choice on the first connector, then choose Select from the second connector's popup menu to select it.

The model details are now connected even though no top-level Task blocks are connected.

For example, here is a simple model that is split across two model details:



Replacing Default Details of Model and Organizer Tools

You might want to replace the default detail of a model or organizer with a named, top-level workspace. For example, if you have already developed a model on a workspace that you created, and you decide later that you want to associate the workspace with a model or an organizer, you can replace the default detail with the existing workspace.

You can also assign the workspace to its own module, which means you can split a model across multiple modules and save them independently. This means different people can work on different portions of the model simultaneously.

Once you have replaced the default detail of a model or organizer with a top-level workspace, you can associate connectors on different details so work objects flow across top-level workspaces.

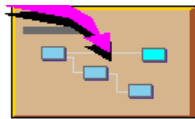
Finally, once you have replaced the default details with top-level workspaces, you can assign those workspaces to different modules in the hierarchy to support team development.

To replace the detail of a model or organizer with a top-level workspace:

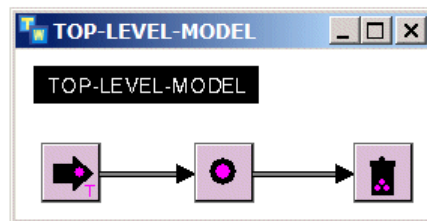
- 1 If you have already created a detail, choose Show Detail on a model or organizer to display the detail.
- 2 Select the detail and choose Edit > Delete.
- 3 Choose Workspace > New Workspace to create a new top-level workspace.
- 4 Choose Properties on the workspace background and configure the name.
- 5 Choose the Choose Detail menu choice on the model or organizer, then mouse right on the background of the new workspace to display its menu and choose Select.

ReThink highlights the border of the workspace indicating that it has been selected. The top-level workspace is now associated with the tool as its detail. You display the new detail normally, by using the Show Detail menu choice.

The following figure shows a model in which the detail is a top-level workspace:



Top-Level Model



Viewing Demo Models

ReThink provides a number of a tutorial models. For details, see the *Getting Started with ReThink*.

ReThink also provides demonstration models that you can view named *rethink-40-doc-examples.kb* and *rethink-40-online-examples.kb*.

These models are all located in the *rethink\examples* directory of your ReThink product directory.

To view ReThink demo models:

- ➔ Choose File > Open and choose the file to open.

Customizing Scenarios

You can customize the procedure that defines the reset behavior of a scenario. For detailed information, see the *Customizing ReThink User's Guide*.

Working with Models

Describes how to work with models through the menus and toolbars.

Introduction	41
Summary of Common Tasks	41
Using the Project Menu	42
Navigating Applications	44
Interacting with Workspaces	46
Using the Menus	53
Using the ReThink Toolbox	63
Using the G2 Toolbox	72
Interacting with Objects	72
Using the Toolbars	75
Annotating Models	80
Setting and Clearing Breakpoints and Indicators	84
Switching User Modes	85
Viewing Messages	86
Configuring User Preferences	87
Configuring Network Interfaces	102
Configuring Message Browsers	102
Configuring Module Settings	102
	103

Introduction


To work with ReThink models, you perform these tasks:



- [Use the Project menu.](#)
- [Navigate models.](#)
- [Interact with workspaces.](#)
- [Use the menus.](#)
- [Use the ReThink toolbox.](#)
- [Use the G2 Toolbox.](#)
- [Interact with objects in the model.](#)
- [Use toolbars.](#)
- [Annotate the model.](#)
- [Set and clear breakpoints and indicators.](#)
- [Switch user modes.](#)
- [View messages.](#)
- [Configure user preferences.](#)
- [Configure network interfaces.](#)
- [Configure message browsers.](#)
- [Configure module settings.](#)

You can also view a [summary of command tasks](#).

Summary of Common Tasks

This section summarizes how to perform common tasks in ReThink:

To...	Do this...
Display the popup menu for an object on a workspace	Click the right mouse button on the object.
Display the properties dialog for an object on a workspace	Double-click the object, select the object and press the F4 key, or choose Properties from the object's popup menu. You can also select the item, then choose Edit > Properties or click the equivalent toolbar button: 

To...	Do this...
Display the detail for an object, such as a model or a Task block with detail	Choose Show Detail from the popup menu for the object, choose View > Show Details, enter Ctrl + right click on the object, or click the equivalent toolbar button: 
Display the ReThink toolbox	Choose View > Toolbox - ReThink.
Adjust the size of a workspace and its associated window to fit the contents of the workspace	Choose Shrink Wrap on the workspace, choose Layout > Shrink Wrap, or click the equivalent toolbar button: 
Hide a workspace	Click the Minimize button on the window, choose Hide on the workspace, choose View > Hide, or enter Ctrl + right click on the workspace.

Using the Project Menu

You create, configure, and interact with ReThink objects to create a model by using the Project menu.

You can also create and interact with objects through the Navigator, and you can search for objects once they exist. For more information, see:

- [Using the Navigator.](#)
- [Searching for Objects.](#)

Using the Project Menu

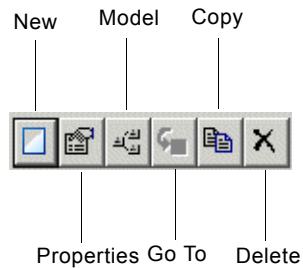
The Project menu allows you to create and manage the various objects you need to build a ReThink application.

For details, see [Using the Project Menu.](#)

Using the Manage Dialog

The Manage dialog allows you to create and configure new ReThink objects, show model details, copy and delete objects, and perform specific operations.

The Manage dialog provides these toolbar buttons:



The buttons in the Manage dialog are enabled or disabled, as appropriate, for the particular type of object.

The Go To button is disabled in Modeler mode because, typically, you interact with objects through properties dialogs and model details. You can go to objects directly through the Navigator or search, if desired.

For information about interacting with objects directly, see [Interacting with Objects in Developer Mode](#).

To use the Manage dialog:

- 1 Choose a submenu from the Project menu and choose Manage.
If the submenu has additional submenus, choose one of the submenus. The Manage dialog appears, which includes all objects in the submenu.
- 2 To create a new object, click the New button in the Manage dialog.
A properties dialog appears for configuring the object. The default name is a unique, system-generated name.
- 3 Configure the properties, depending on the type of object, and click OK.
For information on configuring the properties, see the various chapters in this guide.
The object now appears in the Manage dialog.
- 4 Select an object in the list to enable the toolbar buttons, as appropriate for the type of object.

- 5 To display the properties dialog for an object, click the Properties button.
Note that the only way to configure the properties of a container object once it has been created is through the Manage dialog.
- 6 To display the detail associated with a container object, click the Model button.
This button is only available if the object has detail.
- 7 To copy an existing object, select the object you want to copy, then click the Copy button.
A properties dialog appears for configuring the copy. The default name is the existing object name with -copy appended.
- 8 To delete an object, select the object you want to delete and click the Delete button.

Using the Project Submenus

ReThink provides access to the various objects in a model through submenus of the Project menu. Selecting the menu choice for a configuration object displays the properties dialog for the object. Selecting the menu choice for a container object displays its detail.

To use the project submenus:

- 1 Choose a submenu from the Project menu.
If the submenu has additional submenus, choose a submenu until you see a submenu that includes the names of all objects of that type.
- 2 Choose an object from the submenu.

Navigating Applications

To navigate applications, you can:

- [Use the Navigator.](#)
- [Search for objects.](#)


For information on creating and managing objects through the Project menu, see [Using the Project Menu.](#)

Using the Navigator

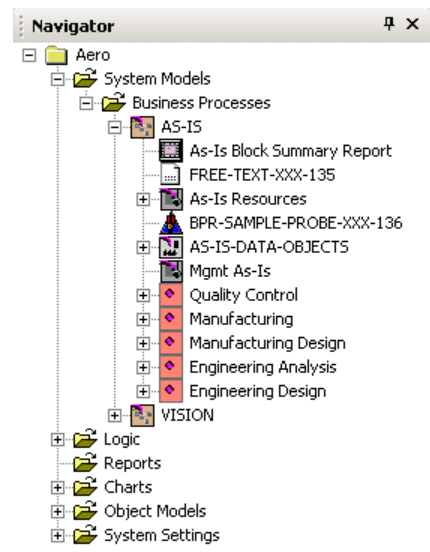
The Navigator displays all the elements of a project.

You can interact with objects in the Navigator, for example, showing its properties or going to the detail, depending on the type of object. You can also create new objects from the Navigator.

To display the Navigator:

- ➔ Choose View > Navigator or click the equivalent toolbar button () and expand the tree view to the desired level.

Here is the Navigator for the Aero model with the tree expanded to show the objects in the As Is model:



To interact with objects in the Navigator:

- ➔ Right-click a node in the Navigator and choose the desired menu choice.

In addition to the menu choices that you normally get when you right-click the object, you can choose Go To to show the selected object. Depending on the type of object, you might go to the object on a detail or you might go to the object in a repository.

You can also choose New Instance on the Business Processes folder to create a new model directly from the Navigator.

Searching for Objects

You can search for specific types of objects, by matching text in the label field and/or the target class, depending on the type of object. You can also go directly to named objects.

To search for objects:

- 1 Choose Tools > Search and choose a category of object to be found.
- 2 Enter the Keyword text to match and, depending on the type of object, optionally, the Target Class.
- 3 Configure Search By to search for the keyword only, class only, keyword or class, or keyword and class.
- 4 Click the Search button.

The search results include all objects whose label matches the specified text.

- 5 Select an object and click the Go To button.

An arrow appears next to the found object, if it exists; otherwise, the Search dialog display No Matches Found.

Depending on the type of object, you might go to the object on a detail or you might go to the object in a repository. You can interact with the object through its menu choices, for example, to go its detail or show its properties.

To go to a named object in the model:

- ➔ Enter the exact name of an object in the Go To type-in box on the toolbar:

A screenshot of a toolbar element. It consists of a small rectangular button with the text "Go To" on the left side, followed by a larger, empty text input field.

A red arrow points to the named object on a workspace.

Interacting with Workspaces

You place all model objects on detail workspaces, which appear their own window. You display and interact with workspaces in these ways:


- [Display a detail workspace.](#)
- [Hide a workspace.](#)
- [Delete a workspace.](#)
- [Create a detail workspace.](#)
- [Edit workspace properties.](#)
- [Scale a workspace.](#)

- [Shrink wrap a workspace](#) to fit the enclosed elements.
- [Show the superior object](#) for a workspace.
- [Print a workspace](#).
- [Save a workspace as a JPEG file](#).
- [Assign a background image to a workspace](#).
- [Create and access top-level workspaces](#).

Displaying a Detail Workspace

A number of objects define detail, which is a workspace associated with the object on which you place other objects. For example, models, organizers, and Task blocks all define detail.

To display detail for an object:

- Right-click the background of a workspace and choose Show Detail, choose View > Show Details, or click the equivalent toolbar button: ()

or

- Press Ctrl + right-click on the object.

Hiding a Workspace


To hide a workspace:

- Right-click the background of a workspace and choose Hide or press Ctrl + right-click on the workspace.

Deleting a Workspace

Deleting a workspace permanently deletes it from the server, including all objects on the workspace.

To delete a workspace:

- Select a workspace and choose Edit > Delete, right-click the background of a workspace and choose Delete, or click the equivalent toolbar button: ()

Creating a Detail Workspace

If you delete a detail workspace associated with a model, you can create a new detail for an existing model.

To create a detail workspace:

- 1 Show the detail of a model.
For details, see [Displaying a Detail Workspace](#).
- 2 Delete the detail workspace associated with the model.
For details, see [Deleting a Workspace](#).
- 3 Show the Navigator.
For details, see [Using the Navigator](#).
- 4 Right-click the model whose detail you deleted and choose Create Detail.


The model has a new detail workspace.

Editing Workspace Properties

You can edit the name of the workspace, as well as the background and foreground colors, and the margins around the objects at the edges of the workspace. By default, the background color is white and the foreground color is black.

For information about configuring the background image, see [Loading Background Images](#).

To edit workspace properties:



- 1 Select a workspace and choose Edit > Properties, right-click the background of a workspace and choose Properties, or click the equivalent toolbar button:
()
- 2 Configure the Names to be any text.
The text is converted to a symbol, with hyphens in place of spaces when you accept the dialog.
- 3 Configure the Workspace Margin by entering the number of pixels.
- 4 Configure the Foreground Color and Background Color by choosing a color.

The name appears at the top of the workspace when you accept the dialog.

Scaling a Workspace

You can scale a workspace to fit the current window, or zoom a workspace in, out, or to a specific scale.

To scale a workspace:

→ Choose View > Zoom In or Zoom Out, enter Ctrl + = to zoom in or Ctrl + - (minus) to zoom out, or click the equivalent toolbar buttons: ( )

or

→ Choose View > Zoom, then choose or enter a zoom scale, or enter a specific zoom scale in the zoom scale on the toolbar: (100% ▾)


or

→ Choose View > Zoom to Fit or click the equivalent toolbar button: ()

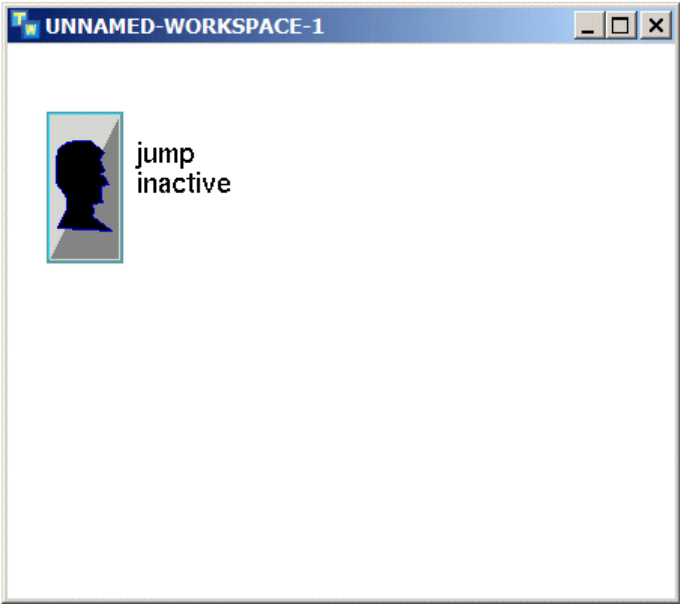
Shrink Wrapping a Workspace

When you move objects on a workspace beyond the visible borders, the borders adjust to fit the objects. When you move objects on a workspace such that the workspace contains extra space at its borders, you can adjust the borders by shrink wrapping the workspace. Shrink wrapping a workspace also adjusts the window size. You can resize the window to make it smaller to add scroll bars to the window.

To shrink wrap a workspace:

→ Select a workspace and choose Layout > Shrink Wrap or click the equivalent toolbar button: ()

This figure shows a workspace that has extra space around its borders:



This figure shows the result of shrink wrapping the workspace:




This figure shows the result of dragging the object on the workspace so it has extra space around its borders, then adjusting the window size to make it smaller, which adds scroll bars:



Showing the Superior Object of a Detail Workspace

You can show the superior object of a detail workspace, for example, the detail of or Task block with detail.

To show the superior object of a detail workspace:


- ➔ Right-click the background of a workspace and choose Go to Superior, or select a detail workspace and choose View > Go to Superior or click the equivalent toolbar button: ()

The workspace with the superior object is now on top with an indicator arrow next to the object.

Depending on the type of object, you might go to the object in a repository. You can interact with the object through its menu choices, for example, to show its properties.

Printing a Workspace

To print a workspace:

- ➔ Choose File > Print, or enter Ctrl + P or click the equivalent toolbar button (), and configure the Print dialog.

Saving a Workspace to a JPEG File

To save a workspace to a JPEG file:

- ➔ Choose File > Save as JPEG and specify a file name.

Loading Background Images

You can load one or more JPEG, XMB, or GIF files as the background for a workspace.

To load a single background image:

- ➔ Choose Workspace > Load Background Image, navigate to the image to use as the background, and click Open.

To load multiple background images:

- 1 Create one or more Image Definition objects from the Displays palette of the ReThink toolbox.
- 2 Configure the Image Name and File Name of Image, and optionally click Save Image Data with KB to save the image data as part of the KB.
- 3 Show the properties dialog for a workspace, click the Background tab, click the Add Background Image button to add one or more rows, choose an image definition from the list of existing definitions, and configure the X-Y position of each image.

To remove background images:

- ➔ Choose Workspace > Delete Background Image.

or

- ➔ Show the properties dialog for a workspace, click the Background tab, select a row, and click the Remove Background Image button.

Creating and Accessing Top-Level Workspaces

Typically, you create new workspaces when you create models through the Project menu. However, you can also create new workspaces directly through the Workspace menu, which are top-level workspaces that you can access by name.


To create a new top-level workspace:

- 1 Choose Workspace > New.

The workspace is assigned a unique number, which starts with unnamed-workspace.

- 2 Configure the workspace properties as described in [Editing Workspace Properties](#).

To access the top-level workspace:

- 1 Choose Workspace > Get or click the equivalent toolbar button: ()
A list of all top-level workspaces available in the current user mode appears.
- 2 Select a workspace and click OK.

Using the Menus

The top-level menu bar consists of these menus:

Menu	Description
File	Standard file operations, and print and export operations for workspaces.
Edit	Standard editing operations for objects on workspaces.
View	Display the various toolboxes and toolbars, display the Navigator, zoom workspaces, show details, and show superior objects.
Layout	Standard layout operations for objects on workspaces, including align, distribute, rotate, reflect, order, nudge, as well as shrink wrapping workspaces.
Go	Standard browser navigation operations and interaction with the server.
Project	Manage system models, object models, reports, charts, system settings, and user preferences.
Simulation	Activate and deactivate scenarios, control the simulation mode, start, reset, pause, and continue the simulation, and clear indicators and breaks.
Workspace	Create new and get existing workspaces, and edit background images for workspaces.
Tools	Find model objects, show users, and switch user modes.
Window	Control window positioning and choose the active window.
Help	Display online help.

The following sections summarize each of these menus.

For information about how to use specific menu choices, see the referenced sections.

For information about additional menu choices available in Developer mode, see the *Customizing ReThink User's Guide*.

Using the File Menu

The File menu allows you to perform basic file and module operations.

Menu Choice	Description
New	Creates a new project. See Working with Projects .
Open	Opens an existing project, replacing the one currently loaded.
Save	Saves the top-level module of the current project.
Save As	Saves the top-level module of the current project to a user filename. You save models to filenames with a <i>.kb</i> extension.
Save as JPEG	Exports the currently selected workspace as a <i>.jpg</i> file.
Print	Prints the currently selected workspace to a postscript printer.
Close	Exits the client.

Using the Edit Menu

The Edit menu allows you to perform basic edit operations for objects.

Menu Choice	Description
Delete	Deletes the selected object.
Transfer	Transfers the selected object to the mouse. Click on a workspace to transfer the object.
Clone	Transfers the selected object to the mouse. Click on a workspace to clone the object.

Menu Choice	Description
Select All	Selects all objects on a workspace.
Properties	Displays the properties dialog for the selected object.
Colors	Changes the colors of the icon regions of the selected objects.

Using the View Menu

The View menu allows you to show and hide toolboxes and toolbars, and to control the zoom scale.

For details about each of the toolboxes, see [Using the ReThink Toolbox](#).

The View menu contains the menu choices in the following table:

Menu Choice	Description
Toolbars > Standard	Toggles the Standard toolbar, which contains standard buttons for file and edit operations.
Toolbars > Layout	Toggles the Layout toolbar, which contains buttons for performing standard layout operations for objects on workspaces.
Toolbars > Web	Toggles the Web toolbar, which contains standard buttons for browsing HTML and text pages.
Toolbars > Simulation	Toggles the Simulation toolbar, which contains buttons for controlling the simulation run state and mode.
Status Bar	Toggles the status bar, which displays the connection status to the server.
Message Board	Displays the G2 Message Board, which displays text messages.
Message Browser	Displays a message browser of operator messages.
Navigator	Toggles the display of a tree view of all objects in the current project. See Navigating Applications .

Menu Choice	Description
Toolbox - ReThink	Toggles the display of the ReThink toolbox, which contains all the objects you need to create a model and run a simulation, including Tools, Basic Activities, Instruments, Resources, and Reports.
Zoom	Scales the selected workspace.
Zoom In	
Zoom Out	
Zoom to Fit	
Hide	Hides the currently selected workspace.
Go to Superior	Displays the superior object of the currently selected workspace.
Show Details	Shows the detail workspace of the currently selected object.

Using the Layout Menu

The Layout menu allows you to interact with objects on workspaces. For details, see [Interacting with Objects](#).

Menu Choice	Description
Order >	Controls the stacking order of selected objects on workspaces.
Bring to Front	
Send to Back	
Nudge >	Micro-adjusts the position of selected objects in each direction.
Nudge Up	
Nudge Down	
Nudge Right	
Nudge Left	

Menu Choice	Description
Align or Distribute >	Aligns two or more selected objects along various axes. Distributes three or more selected objects vertically or horizontally.
Align Left	
Align Center	
Align Right	
Align Top	
Align Middle	
Align Bottom	
Distribute Horizontally	
Distribute Vertically	
Rotate or Flip >	Rotates and reflects the selected objects.
Normal	
90 Clockwise	
90 Counterclockwise	
180	
Flip Horizontally	
Flip Vertically	
Shrink Wrap	Adjusts the borders of the selected workspace to just fit the contained objects.

Using the Go Menu

The Go menu allows you to perform standard browser navigation and interact with the server.

Menu Choice	Description
Back	Provides standard browser operations for HTML and text pages.
Forward	
Stop	
Refresh	
Home	

Using the Project Menu

The Project menu allows you to interact with all the objects in the current project, as follows:

Menu Choice	Description
Initialize Application	Clears the Message Browser.
Uninitialize Application	
My User Preferences	Configures user preferences for the current user. See Configuring User Preferences .
System Models > Business Processes	Creates and manages ReThink business process models.
Logic > Business Rules	Creates and manages business rules. See the <i>Business Rules Management System User's Guide</i> .
Reports	Creates and manages a variety of reports.
Charts	Creates and manages various types of charts.
Object Models > Business Objects Business Processes	Creates and manages business objects and processes for use with business rules.
System Settings	Creates and manages the various system settings described below.

Menu Choice	Description
System Settings > Interfaces > SQL SMTP JMS HTTP	Creates and manages network and database interface objects for communicating with various types of external systems.
System Settings > Interface Pools > SQL SMTP JMS	Creates and manages network and database interface pools for communicating with various types of external systems.
System Settings > Message Browsers > Queues Events Messages Access Tables Templates	Creates and manages custom message browsers and queues.
System Settings > Users	Creates and manages user preferences. See Configuring User Preferences .
System Settings > System Performance	Enables, disables, and configures system performance metrics.
System Settings > Event and Alarm Metrics	Enables and disables event and alarm metrics.

Using the Workspace Menu

The Workspace menu allows you to interact with workspaces. For details, see [Interacting with Workspaces](#).

Menu Choice	Description
New	Creates a new workspace.
Get	Displays a list of named workspaces, which you can display.
Load Background Image Delete Background Image	Loads and deletes background images for the selected workspace.

Using the Simulation Menu

The Simulation menu allows you to control the simulation.

Menu Choice	Description
Activate	Activates or deactivates the current scenario.
Start All	Starts all Source blocks associated with the current scenario.
Reset	Resets the simulation for the current scenario.
Pause	Pauses the simulation for the current scenario.
Continue	Continues running a paused simulation for the current scenario.
Jump Mode	Sets the scenario to jump mode, where the simulation clock advances continuously with each discrete event.
Step Mode	Sets the scenario to step mode, where the simulation clock pauses after each discrete event.
Synch Mode	Sets the scenario to synch mode, where the simulation clock is synchronized with real time, based on a scale factor.

Menu Choice	Description
Online Mode	Sets the scenario to online mode, where ReThink performs real-time transaction processing.
Clear Breaks	Clears all break points that have been set in the model.
Clear Indicators	Hides all indicator arrows in the model.

Using the Tools Menu

The Tools menu allows you to browse objects in the model.

Menu Choice	Description
Search	Allows you to search for objects in a model by name or label. See Searching for Objects .
Show Users	Shows the users currently logged into the server.
User Mode > Administrator System-Administrator Developer Modeler Operator	Changes the user mode. The default user mode is Modeler, which allows you to create models by copying, connecting, and configuring objects, and to run simulations. Operator mode allows end users to view models only. Developer mode allows developers to customize the application. Note: In general, you work in Modeler mode. Very occasionally, modelers need to switch to Developer, Administrator, or System Administrator mode to perform particular tasks. See Switching User Modes .

Using the Help Menu

The Help menu allows you to access online help that displays as a window within the client:

Menu Choice	Description
G2 Help Topics	Display the G2 online help.
ReThink Help Topics	Display the ReThink online help.
Server Information	Displays version information about the server.
About G2	Displays the G2 title block, which shows the current version.
About ReThink	Displays the ReThink title block, which shows the current version.

You can view PDF versions of the following guides:

- *ReThink User's Guide*
- *Customizing ReThink User's Guide*
- *Getting Started with ReThink*

To view the online manuals:

➔ Choose Start > Programs > Gensym G2 2011 > Documentation > G2 ReThink and choose the manual you want to view.

Using the ReThink Toolbox

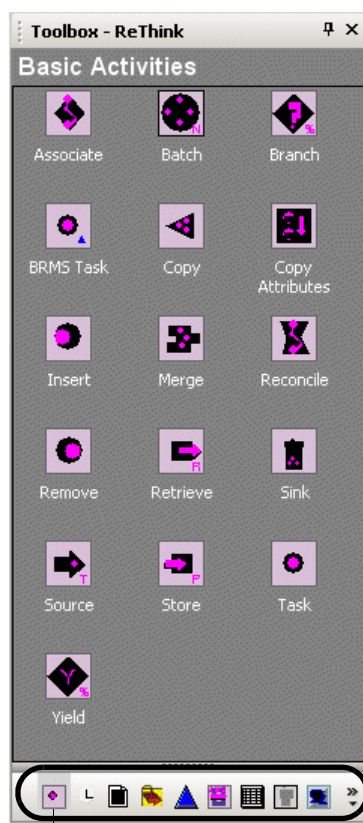
The ReThink toolbox contains all of the objects that you use to create a model.

To display and interact with the ReThink toolbox:

- 1 Choose a toolbox from the View menu.

The toolbox appears with the first palette in the toolbox visible. The palettes are organized alphabetically. You access the various palettes in the toolbox by clicking the buttons at the bottom of the toolbox.

Here is the Basic Activities palette of the ReThink toolbox:




Basic Activities

Click the buttons to display the various categories of objects in the toolbox.

- 2** To access the various palettes in the toolbox, hover the mouse over a button to display its tool tip, then click the button to display the palette.

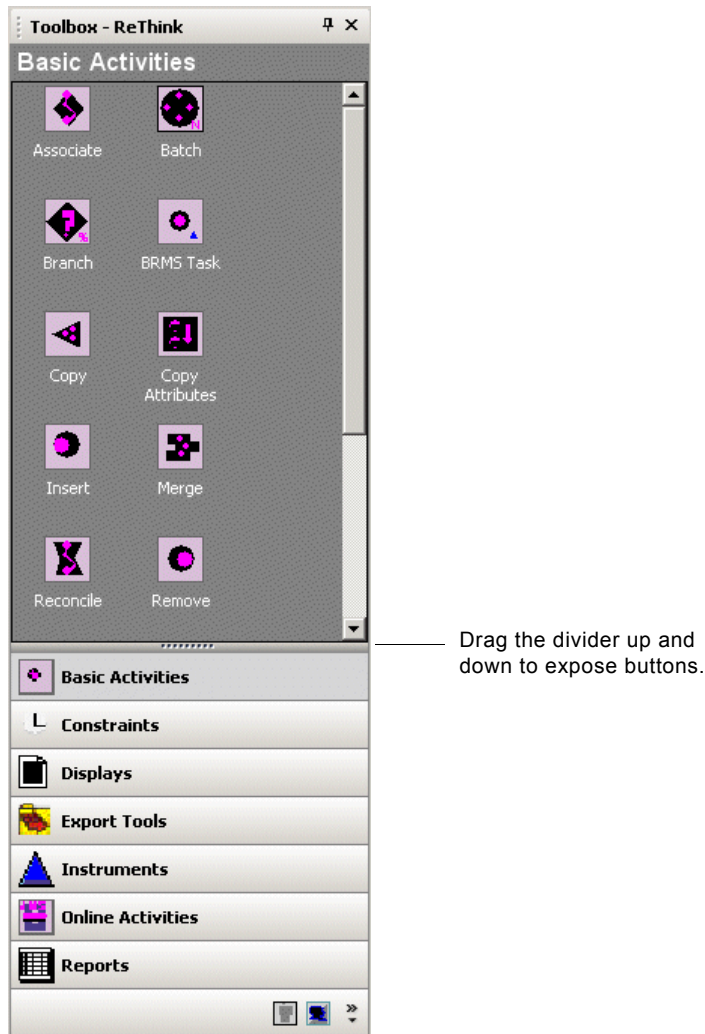
Depending on the size of toolbox, the toolbar at the bottom shows only a subset of the available buttons in the toolbox.

- 3** To display the additional buttons in the toolbox, click the configure button at the far right of the toolbar (), then choose a palette.
- 4** To configure the buttons that are visible in the toolbar and associated configuration menu, choose Add or Remove Buttons to display a list of all palettes, then choose a button to add or remove.

Once you have configured the buttons you want, you can expand the buttons to show their labels for some or all of the buttons.

- 5** To show button labels in the toolbox, drag the divider at the bottom of the toolbox up to expose the buttons with their labels.

For example, here is the ReThink toolbox with button labels showing:



Once you have configured the buttons you want to appear in the toolbox, you can auto hide the toolbox by clicking the pin in the upper right corner of the toolbox.

Note Do not close the toolbox or the toolbox reverts to the default set of buttons.

- 6 Click the pin to autohide the toolbox, and move the mouse over the tab to display the toolbox after it has been hidden.

You can display, configure, and autohide multiple toolboxes, as needed, each of which will have its own toolbox tab.

Choose a topic:

[Basic Activities](#)

[Constraints](#)

[Displays](#)

[Export Tools](#)

[Instruments](#)

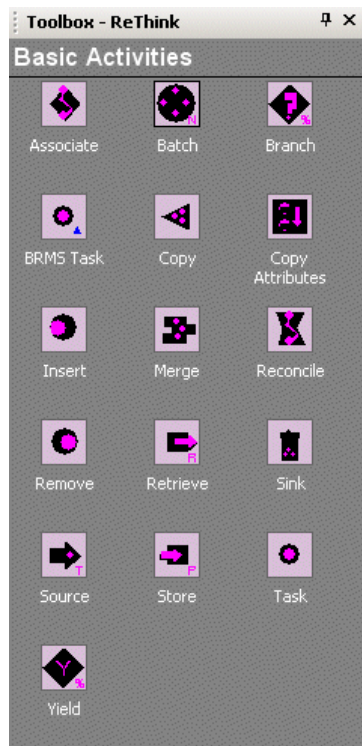
[Online Activities](#)

[Reports](#)

[Resources](#)

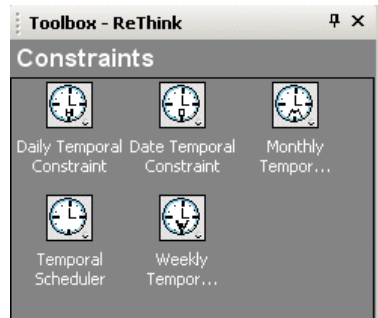
[Tools](#)

Basic Activities



See [Using Blocks](#) and [Blocks Reference](#).

Constraints



See [Constraining the Availability of Resources](#).

Displays



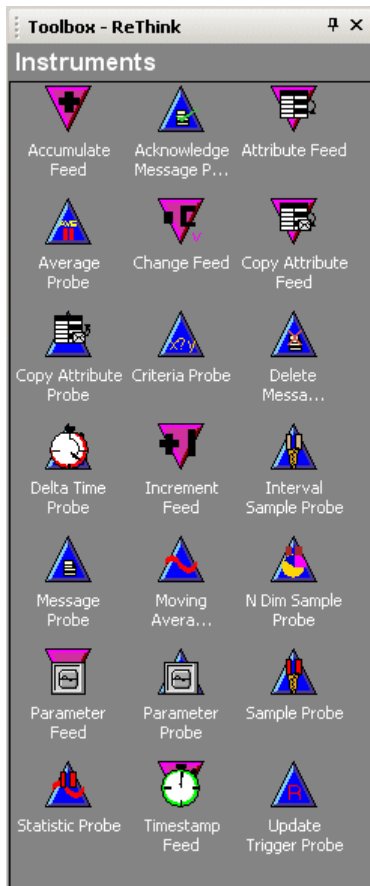
See [Annotating Models](#).

Export Tools



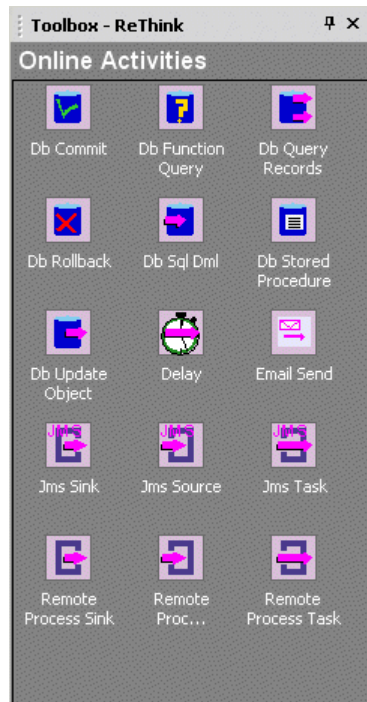
See [Exporting Probed Data to a CSV File](#).

Instruments



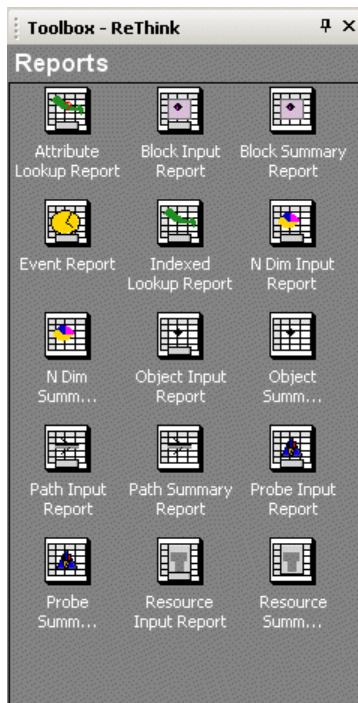
See [Using Instruments](#) and [Instruments Reference](#).

Online Activities



See [Using ReThink in Online Mode](#).

Reports



See [Using Reports](#).

Resources



See [Using Resources](#).

Tools



For information on...	See...
Arrival Rate Input Graph	Using a Graph to Specify Duration
Batch Simulation Object	Using Batch Simulation
Class Definition	Specifying a User-Defined Object as the Path Type.
Connector	Associating Connectors on Other Types of Details.
Model, Organizer, and Scenario	Organizing Models and Controlling Simulations.
Palette Workspace	<i>Customizing ReThink User's Guide.</i>
Remote Chart	Creating a Remote Chart.
Resource Utilization Chart	Charting Resource Utilization.

For information on...	See...
Rule	Branching Based on Rules that Set the Attribute Value.
Update Trigger	<ul style="list-style-type: none"> • Exporting Probed Data at Regular Time Intervals. • Triggering Regular Updates for Multiple Reports.

Using the G2 Toolbox

In general, you use the G2 toolbox when customizing models.

In addition, several ReThink features require that you access objects on the G2 toolbox. For details, see:

- [Using an Action Button to Update Charts.](#)
- [Using Interface Pools.](#)
- [Parameter Probe](#) and [Parameter Feed.](#)

For details, see the *Customizing ReThink User's Guide*.

Interacting with Objects

You can interact with objects in a model by using the Edit menu, the object's popup menu, and the Layout menu. Many of the menu choices have shortcuts and/or equivalent toolbar buttons.

When you create a model, we recommend that first, you place the blocks on the workspace, then you align and distribute them, using buttons on the Layout toolbar, then you connect them, as needed.

You configure attributes of objects through properties dialogs.

Selecting Objects

To select one or more objects:

→ Click an object to select it.

or

→ Click and drag a rectangular area to select all the objects in the rectangle.

or

→ Use Shift key and click on an object to add or remove it to or from an existing selection.

or

→ Use the Alt key and click on a connected network of objects to select all the connected objects.

To select all objects on a workspace:

→ Choose the Edit > Select All or enter Ctrl + A.

Cutting, Copying, Pasting, and Deleting Objects

When you copy an object, the new object has the same property values as the existing object. If the object has details, the new object has the same details. You can transfer objects from one workspace to another.


To copy and paste objects:

→ Select one or more objects to copy, choose Edit > Clone, then click on any workspace to paste the selected objects to the workspace.

To cut and paste objects:



→ Select one or more objects to cut, choose Edit > Transfer, then click on any workspace to paste the selected objects to the new workspace.

To delete objects:



→ Select an object, then choose Delete from the Edit menu or from the popup menu, press the Delete key, or click the equivalent toolbar button (), then click Yes to confirm the deletion.

Controlling the Layout of Objects







To adjust the order of objects:

- Select an object, then choose Layout > Order > Bring to Front or Send to Back or click the equivalent toolbar button: ( )



To rotate or flip objects:

- Select an object, choose Layout > Rotate or Flip, then choose the desired action from the submenu or click the equivalent toolbar button:  


To align objects:

- Select two or more objects, choose Layout > Align or Distribute, then choose the desired align action from the submenu or click the equivalent toolbar button: (   |   )

To distribute objects:

- Select three or more objects, choose Layout > Align or Distribute, then choose the desired distribute action from the submenu or click the equivalent toolbar button: ( )

To nudge an object up, down, right, or left:

- Select an object, choose Layout > Nudge, then choose the desired nudge action from the submenu; or hold down the Ctrl key while pressing the up, down, right, and left arrow keys to nudge the item in the desired direction; or click the equivalent toolbar button: 

For information on the Shrink Wrap toolbar button on the Layout toolbar, see [Shrink Wrapping a Workspace](#).

Displaying the Properties Dialog for an Object

To display the properties dialog for an object:

- Double-click the object.


or

- Select the object and press the F4 key.

or

- Choose Properties from the object's popup menu.

or

- Select the object, then choose Edit > Properties or click equivalent toolbar button: ()

Resizing an Object

You might need to resize an object.

To resize an object:

→ Click an object to select it, and drag the selection handles to resize the object.

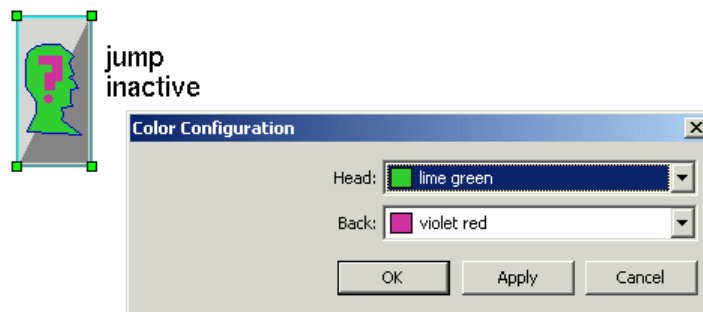
Editing Icon Color Regions

You can edit the color of any named region of any icon.

To edit icon colors:

- 1 Click an object to select it, and choose Edit > Colors.
- 2 Configure the color of the named icon region for the object, as desired.

For example:



Using the Toolbars

ReThink provides a number of toolbars that you can use to interact with models.

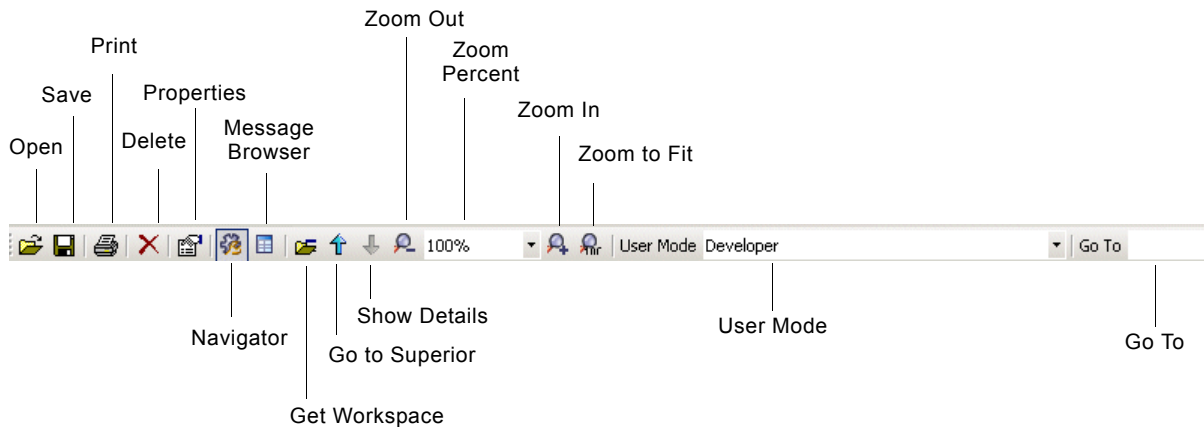
The toolbars are all docked, by default. You can drag the toolbar to a new location or off the toolbar to make it a floating toolbar.

The available toolbars are:

- [Standard toolbar](#)
- [Simulation toolbar](#)
- [Web toolbar](#)
- [Layout toolbar](#)
- [Status bar](#)

Standard Toolbar

The Standard toolbar contains many of the toolbar buttons that you need to work with the model:



To hide and show the Standard toolbar:

➔ Choose View > Toolbars > Standard.

For information on this button...

See...

Open

[Opening a Project.](#)

Save

[Saving a Project.](#)

Print

[Printing a Workspace.](#)

Delete

[Cutting, Copying, Pasting, and Deleting Objects.](#)

Properties

[Displaying the Properties Dialog for an Object.](#)

Navigator

[Using the Navigator.](#)

Message Browser

[Viewing Messages.](#)

Get Workspace

[Creating and Accessing Top-Level Workspaces.](#)

Go to Superior

[Showing the Superior Object of a Detail Workspace.](#)

Show Detail

[Displaying a Detail Workspace.](#)

Zoom In, Zoom Out,
Zoom Percent, and
Zoom to Fit

[Scaling a Workspace.](#)

**For information
on this button...****See...**

User Mode

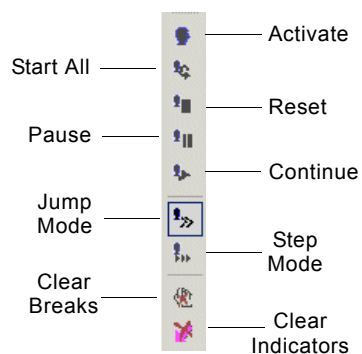
[Switching User Modes.](#)

Go To

[Searching for Objects.](#)

Simulation Toolbar

The Simulation toolbar contains toolbar buttons that you need to configure and run a simulation:

**To hide and show the Simulation toolbar:**

➔ Choose View > Toolbars > Simulation.

**For information
on this button...****See...**

Activate

“Activating and Deactivating the Scenario” on page 17.

Start All, Reset, Pause, Continue

“Starting and Stopping the Simulation” on page 18.

Jump Mode and Step Mode

“Configuring the Simulation Mode” on page 20.

Clear Breaks

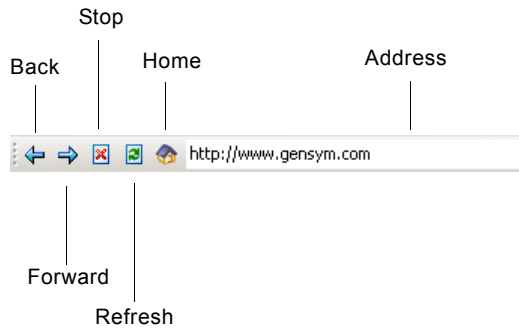
[Debugging Blocks.](#)

Clear Indicators

“Configuring the Behavior of Indicator Arrows” on page 25.

Web Toolbar

The Web toolbar provides the standard browser navigation buttons and commands for browsing HTML pages:



To hide and show the Web toolbar:

➔ Choose View > Toolbar > Web.

You can go to any URL, including any HTML file on the World Wide Web or on the file system, or any RTF file.

To go to an HTML file on the World Wide Web, you use the standard HTTP protocol, for example, `http://www.gensym.com`.

To go to an HTML or RTF file on the file system, you use this protocol:

```
file:\<drive>:\<directory>\<filename>
```

For example, to go to the readme file, you would use:

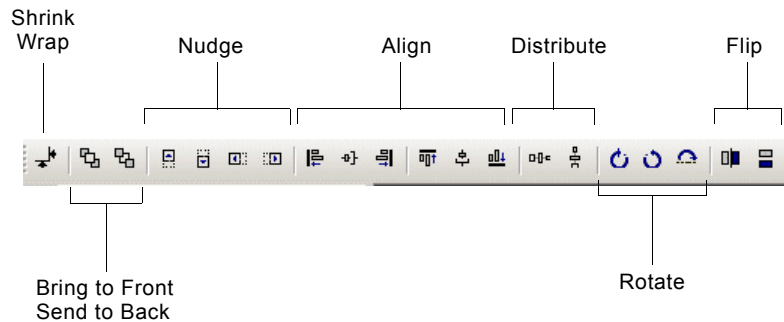
```
file:\C:\Program Files\Gensym\g2-2011\doc\rethink\rethink-readme.html
```

You navigate by using standard buttons in the Web toolbar or in the Go menu.

You configure the Home button URL in your user preferences. For more information, see [Configuring User Preferences](#).

Layout Toolbar

The Layout toolbar contains toolbar buttons that you need to control the visual layout of objects on a workspace:



To hide and show the Layout toolbar:

➔ Choose View > Toolbars > Layout.

For information on this button...

See...

Shrink Wrap

[Shrink Wrapping a Workspace.](#)

Send to Front, Send to Back, Nudge, Align, Distribute, Rotate, Flip

[Controlling the Layout of Objects.](#)
 “Aligning and Distributing Blocks” on page 89.

Status Bar

The status bar shows various status information, such as the host and port of the client, the current file being loaded, and the progress bar.

By default, the status bar also shows the current message in the operator Message Browser. For information on how to disable this feature, see [Configuring User Preferences.](#)

To hide and show the status bar:

➔ Choose View > Status Bar.

Annotating Models

You can annotate a model by:

- Placing an [annotation](#) next to the object and entering text on its detail.
- Adding [free text](#) anywhere on a workspace.
- Using a [readout table](#) to display the current value of metrics that update while the model is running, for example, the values of paths.
- Adding [attribute displays](#) for the various attributes of an object.

You find Annotation tools, free text, and readout tables on the Displays palette of the ReThink toolbox. Attribute displays appear next to most ReThink objects.

To show displays in the toolbox:

➔ Choose View > Toolbox - ReThink and show the Displays palette:



Using an Annotation Tool

You use an Annotation tool to create icons that display information about the model.

To annotate a model:

- 1 Select an Annotation tool from the Displays palette of the ReThink toolbox and place it next to the object you want to annotate.
- 2 Display the properties dialog for the Annotation tool and configure the Annotation Text.
- 3 Choose Show Annotation on the tool to show a detail workspace with the text, or simply click and hold the mouse button on the tool to show the annotation text.
- 4 To move the Annotation tool, drag it by using the right mouse button.

Here is an example of an Annotation tool:



Here is some information you want to appear

Using Free Text

You can label any part of your model by using free text or free text with a border.

To label a model, using free text:

- 1 Select a Free Text or Borderless Free Text from the Displays palette of the ReThink toolbox and place it anywhere on a workspace.
- 2 Double-click the ellipses (...), edit the text, and press Ctrl + Return, or choose Properties on the text and edit the Text.
- 3 Click the Appearance tab and configure the background color, foreground color, font size, and font scale, as needed.

Here is an example of a free text and a borderless free text:

Borderless Free Text

Free Text

Using Readout Tables

You can use a readout table to display the current value of an attribute of any ReThink object. You can specify how frequently the readout table updates its current value. You can also choose to format time values, using hours, minutes, and seconds.

You typically use readout tables to display path attributes, since you cannot probe the attributes of a path.

To use a readout table, you must name the object whose attributes you want to display, which you must do in Developer mode.

Caution Be careful when you copy models that contain readout tables because the readout in the copied model will be displaying data about the original model. Be sure to update the Expression to Display of the readout tables in the copied model to refer to objects in the copied model.

To create a readout table:

- 1 Name the object whose value you want to display in the readout table:
 - a Choose Tools > User Mode > Developer.
 - b Display the properties dialog for the object and click the Customize tab.
 - c Configure the Names to be a unique name.
 - d Choose Tools > User Mode > Modeler.
- 2 Select a Readout Table from the Displays palette of the ReThink toolbox and place it on a workspace near the objects whose value you want to display.
- 3 Display the attribute table for the readout table and edit the label-for-display to be the text label.
- 4 Configure the expression-to-display to specify a G2 expression that refers to the name of the attribute whose value you want to display.

For example, to display the value of the Current Waiting metric of a path whose name is **source-path-1**, you would use this expression:

the current-waiting of source-path-1
- 5 Configure the default-update-interval to be the frequency with which the readout table updates its value.
- 6 Configure the display-format to be interval to display time values using days, hours, minutes, and seconds, as opposed to seconds.

The readout table begins updating immediately according to the update interval you specify.

For example, this readout table shows the value of the Inventory Days of Supply metric of the Base Manufacture role's output path:

Inventory Days of Supply	4.614
--------------------------	-------

Base Manufacturer

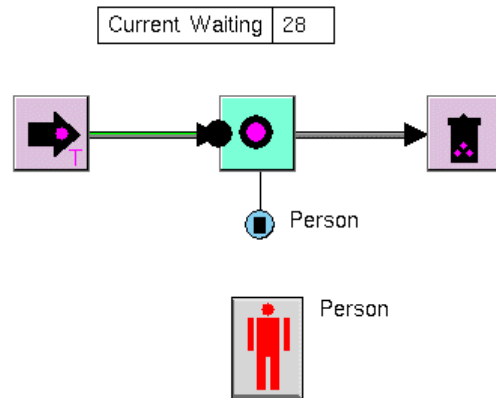


BM-1

Consumer



For example, this readout table shows the value of the Current Waiting metric of the Task block's input path:



Using Attribute Displays

By default, ReThink objects include attribute displays that show the Label of the item. When you edit an object's Label through the properties dialog, the attribute display updates. You can drag the attribute display to a new location next to the object.

You might want to add attribute displays to the layout to show the value of other parameters and/or metrics next to the object. You can also display the name of the parameter or metric with its value.

Note Many of the attributes that you want to display are available in Developer mode only. However, many of the attributes that you configure through dialogs are stored in subtables, for example, the cost-subtable and duration-subtable, which you cannot display as attribute displays.

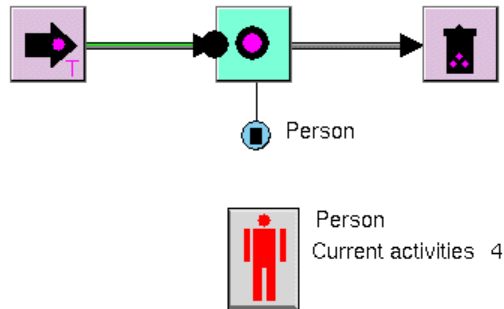
To edit attribute displays for an object:

- 1 Choose Tools > User Mode > Developer.
- 2 Choose Table on the item whose attribute you want to display.
- 3 In the attribute table, mouse right on the attribute you want to display and choose **show attribute display**.

The attribute value appears next to the object.

- 4 To display the name of the attribute with its value, mouse right on the attribute display and choose **add name of attribute**.

The following figure shows a resource with the Current Activities attribute visible as an attribute display:



Setting and Clearing Breakpoints and Indicators

You can set breakpoints to pause the simulation at various points in the model. When you run the simulation with a breakpoint set, the simulation pauses at that point and an indicator arrow appears at the breakpoint.

To set a breakpoint:

- Choose Set Break on a block.

When you run the simulation, an indicator arrow appears at the breakpoint you set, and the simulation pauses. Click the indicator to continue running the simulation until it reaches the breakpoint again.

For information on controlling the behavior of indicators, see [Configuring the Behavior of Indicator Arrows](#).

To clear breakpoints:

- Choose Simulation > Clear Breaks or click the equivalent toolbar button:



To clear all indicators:

- Choose Simulation > Clear Indicators or click the equivalent toolbar button:



Switching User Modes

You build and run applications in one of four built-in **user modes**, or you can define your own user mode. The user mode determines what you can and cannot do when you create your application and run it. For example, the user mode determines whether you can move, edit, and delete objects, and whether you can use the full set of G2 features in your model. For example, the user mode determines the parameters that you can configure.

ReThink supports the following user modes for these classes of users:

This type of user...	Works in this user mode...	Which allows you to...
Managers and end users	Operator	View pre-built applications without damaging them in any way. Operators cannot open, save, run, or configure applications.
Business analysts who create applications	Modeler	Create, connect, and configure blocks, instruments, resources, and reports to create models and run simulations. This is the default user mode.
ReThink experts and G2 programmers	Developer System-Administrator Administrator	Create customized versions of ReThink blocks, instruments, and resources to provide functionality that is not part of the basic tool set.

End users of fully developed applications generally work in Operator mode. Operator mode is restricted so that users may run a model but may not create, configure, or delete objects.

As a model developer, you will almost always be working in Modeler mode. This manual assumes you are working in Modeler mode, unless otherwise stated. Occasionally, as a model developer, you will also need to go into Developer mode to perform certain tasks.

If you are an expert who is customizing ReThink, you will be working mostly in Developer mode. The *Customizing ReThink User's Guide* assumes you are working in Developer mode.

The user mode that is available to you depends on your login privileges.

To switch to a different user mode:

➔ Choose Tools > User Mode or configure the User Mode on the toolbar.


Viewing Messages

Various messages might occur when running a simulation, which you can view in the Message Browser or Message Board, depending on the type of message. By default, operator messages appear in the Message Browser, whereas error messages appear in the Message Board.

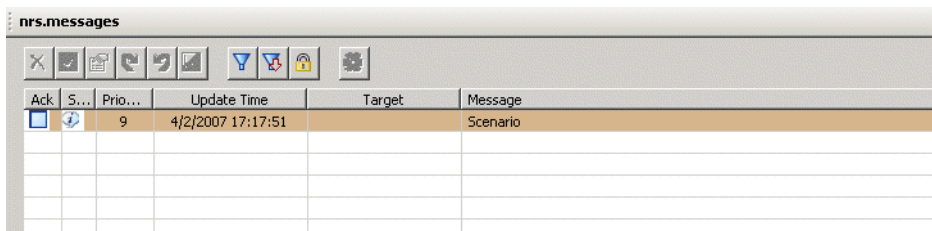
You can configure your ReThink model to generate operator messages in the Message Browser. For details, see [Message Probe](#), [Acknowledge Message Probe](#), and [Delete Message Probe](#).

You can configure how the browser handles messages by configuring the module settings. For more information, see [Configuring Module Settings](#).


To view the Message Browser:

➔ Choose View > Message Browser or click the equivalent toolbar button: ().

Here is the Message Browser with a message that occurs when you choose Show Scenario on an object:



The screenshot shows a window titled "nrs.messages" with a toolbar containing icons for delete, refresh, undo, redo, search, and lock. Below the toolbar is a table with the following data:

Ack	S...	Prio...	Update Time	Target	Message
<input type="checkbox"/>		9	4/2/2007 17:17:51		Scenario

To interact with messages in the Message Browser:

➔ Select the message and click a button in the toolbar.

You can delete and acknowledge the message, show properties, show the target and initiator of the message, configure filters, and lock the view.

To delete all messages in the Message Browser:

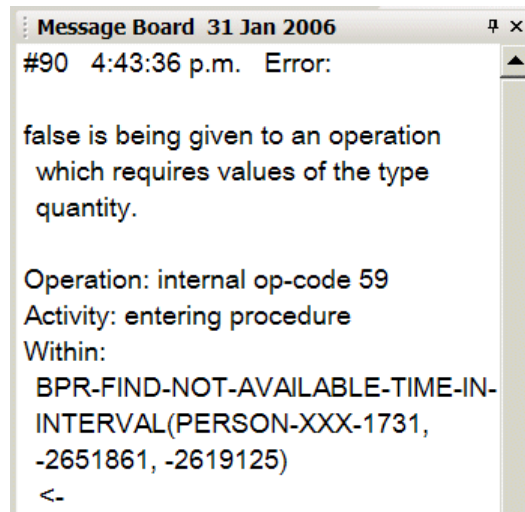
➔ Choose Project > Initialize Application.

To view the Message Board:

➔ Choose View > Message Board.

The Message Board remains open until you close it. You can also auto hide it by clicking the pin in the upper-right corner.

Here is the Message Board with an error message:



Configuring User Preferences

ReThink allows you to configure different levels of access and default behavior for different categories of users. When a particular user starts ReThink, the user preference associated with that user restricts the access and provides default behavior, as appropriate for the given user.

You can configure the following preferences:

- The default user mode, which determines the level of access to ReThink features.
- Subscription to queues.
- Message filter to subscribed queues, for filtering messages based on priority, process map, type, category, target, assigned to, age, and acknowledgement status.
- Acknowledgement and deletion permission and behavior in the Message Browser.
- Client disconnection, server shutdown, and modeling configuration permissions, and whether the user is an administrator.
- The default behavior for interacting with objects through menus and showing the logbook.
- Email and mobile email addresses for use with the JMail interface.

Specifying User Preferences for Different Types of Users

ReThink creates a default user preference for the ReThink server to determine the level of access and default behavior for all users that log into the server. Similarly, ReThink creates one user preference for each user associated with a G2 login account. The name of the user preference corresponds with the user name specified in the *g2.ok* file. For more information, see Chapter 62 “Licensing and Authorization” in the *G2 Reference Manual*.

If you are logged in as the user named **administrator**, you are automatically configured to be the Administrative User and can create and configure user preferences for all users. If you are logged in as any other user, you can only configure your own user preferences. You can be logged in either to your windowing system or to the ReThink server through a secure G2 as **administrator**.

We recommend that the user preference for the server provide access to all available features, and that it use either Modeler or Developer mode. The user preferences for the clients should provide appropriate levels of access and should use the appropriate user mode, depending on the type of user. For example, you might configure user preferences as follows for these types of users:

For this type of user...	Use this default user mode...	And provide these permissions and defaults...
Operators, who interact with messages only	operator	<ul style="list-style-type: none"> • Disconnect permission • Acknowledge message permission • Show message in operator mode by default • Subscribe to appropriate queues, depending on the model
Modelers, who create models	modeler	<ul style="list-style-type: none"> • Disconnect permission • Configuration permission • Acknowledge message permission • Delete message permission • Subscribe to Messages queue

For this type of user...	Use this default user mode...	And provide these permissions and defaults...
Developers, who use G2 to customize models	developer	<ul style="list-style-type: none"> • Indicate items upon menu selection • Disconnect permission • Shutdown permission • G2 Logbook • Acknowledge message permission • Delete message permission • Subscribe to all queues
Administrators, who configure user preferences for all users, using the ReThink user interface	system-administrator	The same as developers, plus Administrative User.
Administrators, who configure user preferences for all users, using G2's user interface	administrator	Note: You must log in as administrator to enable the Administrative User option.

Configuring User Preferences

In Modeler mode, you can configure these attributes for each user preference. For information about additional attributes that you can configure in administrator mode, see the *Customizing ReThink User's Guide*.

Attribute	Description
General	
User Name	The user name of the user that starts either the server or the client, which is read-only. If you are an administrative user, you can create new user preferences for specific users. For details, see Configuring User Preferences .
Default User Mode	The default user mode for the specified user, which is <code>modeler</code> , by default. The options are: <code>operator</code> , <code>modeler</code> , <code>developer</code> , <code>system-administrator</code> , and <code>administrator</code> .
User Interface Theme	The Windows user interface theme. The default value is <code>window-theme-2003</code> .
Email Address Mobile Email	E-mail and mobile e-mail address of the specified user for sending email when a message occurs. For more information, see Delivering Messages by Email .
Home Process Map	A process map to use as the background in the operator interface. The default process map is <code>default view</code> , which is associated with the process map named <code>guif-default-main-view</code> . Click Select to display a list of all process maps in the KB and choose a map to use as the default background.
Telnet Command	The command for launching a Telnet session.
Default Web Location	The default URL when clicking the Home button in the Web toolbar.
Set Default User Mode	Whether the default user mode should be set upon startup.

Attribute	Description
Indicate Items	<p>Configures the behavior when choosing items from the Project menu. By default, ReThink displays the properties dialog or the model detail, depending on the type of item.</p> <p>Developers who are familiar with G2 and prefer to work with the iconic representations of items might want to enable the Indicate Items option, in which case, choosing items from the Project menu goes directly to the item.</p>
Extended Menus	<p>Whether to display the complete list of objects in the Project submenus, the default. If your project has many domain models, for example, you might want to disable this option, in which case, selecting Project > System Models > Business Models displays the Manage dialog for interacting with object.</p>
Show Logbook	<p>Whether to show the G2 Logbook when errors occur. By default, the G2 Logbook does not appear. Modelers or developers who are familiar with G2 might want to enable the Show Logbook option. We recommend that you disable this option for operators and modelers who are not familiar with G2.</p>
Tabbed Mdi Mode	<p>Whether to display workspaces in tabs in the window.</p>
Restore Last Pane Settings	<p>Whether to restore the settings for panes upon connection.</p>
Message Browser	
Email Notification Mobile Email Notification	<p>The format when sending e-mail and mobile e-mail messages. By default, the value is never, which means email messages are not sent. For details, see Delivering Messages by Email.</p>
Modeler Browser	<p>The browser to use in Modeler mode. The default is <code>gevm-modeler-message-view-template</code>, which is the browser that appears when you choose View > Message Browser.</p>

Attribute	Description
Operator Browser	The browser to use in Operator mode. The default is <code>gevm-operator-message-view-template</code> , which is the browser that appears when you are in Operator mode.
Acknowledge Messages Upon Selection	Whether to acknowledge messages automatically when the operator selects a message in the Message Browser view of the operator interface. By default, messages are not automatically acknowledged. When Ack Msg Upon Selection is enabled, Ack Msg Permission must also be enabled.
Show Browser in Operator Mode	Whether to show the Message Browser by default view in the operator interface, or whether to show the process map view. By default, the Message Browser appears as the default view in the operator interface.
Enable Status Bar Message Browser	Whether to show the most recent message in the status bar.
Beep Enabled	Whether to enable beeping when new messages arrive in the Message Browser, as well as when they are acknowledged and deleted. By default, beeping is enabled.

To configure user preferences for yourself:

- ➔ Choose Project > My User Preferences and configure the user preferences, as needed.

For example, here is the default user preferences dialog appears for the user named nrs:

To configure user preferences for other users:

- ➔ Choose Project > System Settings > Users and choose the user whose preferences you want to configure.

For information on creating new user preferences, see *Customizing ReThink User's Guide*.

Delivering Messages by Email

You can configure the user preference for individual users to provide an email address and a mobile email address, then configure rules for when to send email messages when an event occurs.

You can configure ReThink to format the message as short plain text, suitable for cell phones, for example, plain text with full message contents, or as an HTML document. You can also configure when to send a message, based on when it was created or updated, whether the user is currently connected to the server, and the priority of the message.

To deliver messages by email, you:

- [Start the G2 JMail Bridge process.](#)
- [Create, configure, and connect a JMail Interface object.](#)
- [Configure ReThink to send email messages.](#)
- [View examples.](#)
- [Configure startup parameter for sending email.](#)

Starting the G2 JMail Bridge Process

To deliver messages by email, you must start the G2 JMail Bridge process. You identify the host and port to which the bridge is connected for configuring in the JMail Interface object.

To start the G2 JMail Bridge process:

➔ Choose Start > Programs > Gensym G2 2011 > Bridges > G2 JMail Bridge.

The G2 JMail Bridge process appears in the command window.

To determine the bridge port:

➔ Open the command window for the bridge process.

The last line indicates the TPC/IP host and port number, for example:

```
TCP_IP:NSALVO-1165:22080
```

Creating, Configuring, and Connecting the JMail Interface Object

To deliver messages by email, you must create and configure a JMail Interface object, which specifies:

- A name.
- The host and port of the machine running the G2 JMail Bridge.
- Information about the SMTP mail server, including the user name, password, incoming and outgoing SMTP mail host, and SMTP protocol.

If the bridge process is running on the local machine, the host is `localhost`. The default port number is 22080, 22081, 22082, etc., depending on the number of clients that are currently connected on that port.

Note To configure a JMail Interface object, you must be in Developer mode.

Once you have configured the JMail interface object, you can connect it to the G2 JMail bridge process.

To create, configure, and connect a JMail Interface object:

- 1 Choose Tools > User Mode > Developer.
- 2 Choose Project > System Settings > Interfaces > SMTP > Manage and click the New button to create a new JMail Interface object.

Alternatively, you can choose View > Toolbox - G2, click the Network Interfaces tab, and create a JMail Interface object.

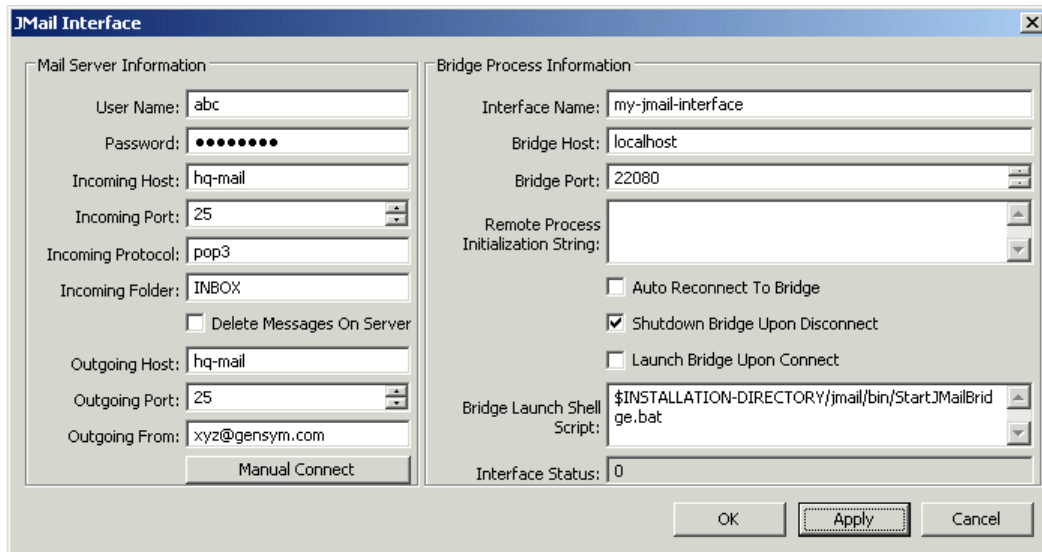
- 3 In the properties dialog for the JMail Interface object, configure the Interface Name attribute to be any symbol, for example, my-jmail-interface.
- 4 Configure the Bridge Host and Bridge Port to be the host and port of the machine on which you started the G2 JMail Bridge process.
- 5 Configure the following additional information:

Attribute	Description
User Name	The user name of the account to which email should be sent.
Password	The password of the user account to which email should be sent.
Incoming Host	The name of the host computer used for incoming email.
Incoming Port	The port number of the host computer used for incoming mail.
Incoming Protocol	The SMTP protocol that the incoming mail host uses. The default is pop3 .
Incoming Folder	The folder name of the user account to which to send email. The default is inbox .
Delete Messages on Server	Whether to delete the email message on the mail server after it is sent. By default, messages are not deleted.
Outgoing Host	The name of the host computer used for outgoing email.
Outgoing Port	The port number of the host computer used for outgoing mail.

Attribute	Description
Outgoing From	The name to use as the From address when the email message is sent, which cannot contain spaces.
Auto Reconnect to Bridge	Whether to automatically reconnect if the connection goes down.
Shutdown Bridge Upon Disconnect	Whether to shutdown the bridge when the connection is closed.
Launch Bridge Upon Connect	Whether to launch the bridge when a connection is made.
Bridge Launch Shell Script	Pathname to script for launching the bridge.

- 6 Click Apply to apply these values.
- 7 Click the Connect button in the dialog to connect the interface to the bridge.
- 8 Choose Tools > User Mode > Modeler to return to Modeler mode.

For example:



Configuring ReThink to Send Email Messages

You configure ReThink to send email messages through the user preferences dialog.

To configure ReThink to send email messages:

- 1 Choose Project > My User Preferences.
- 2 Configure Email Address and/or Mobile Email.
- 3 Choose the rule to use for each of the configured email addresses, as follows:
 - **never** – Do not send e-mail messages. This is the default rule.
 - **send-as-text** – Send the message text and details as plain text.
 - **send-as-short-text** – Send the message text only as plain text.
 - **send-as-html** – Send the message text and details as HTML.
 - **only-high-priority-as-text** – Send the message text and details as plain text only if the priority is 1.
 - **only-high-priority-as-short-text** – Send the message text as plain text only if the priority is 1.
 - **only-high-priority-as-html** – Send the message text and details as HTML only if the priority is 1.
 - **if-not-connected-send-short-text** – Send the message text as plain text only if the user is not connected to the server.
 - **if-not-connected-send-as-text** – Send the message text and details as plain text only if the user is not connected to the server.
 - **if-not-connected-send-as-html** – Send the message text and details as HTML only if the user is not connected to the server.
- 4 Configure the model to send a message, using the Message probe.
For details, see “Message Probe” on page 653.

When a message occurs, ReThink also sends an email to the specified addresses.

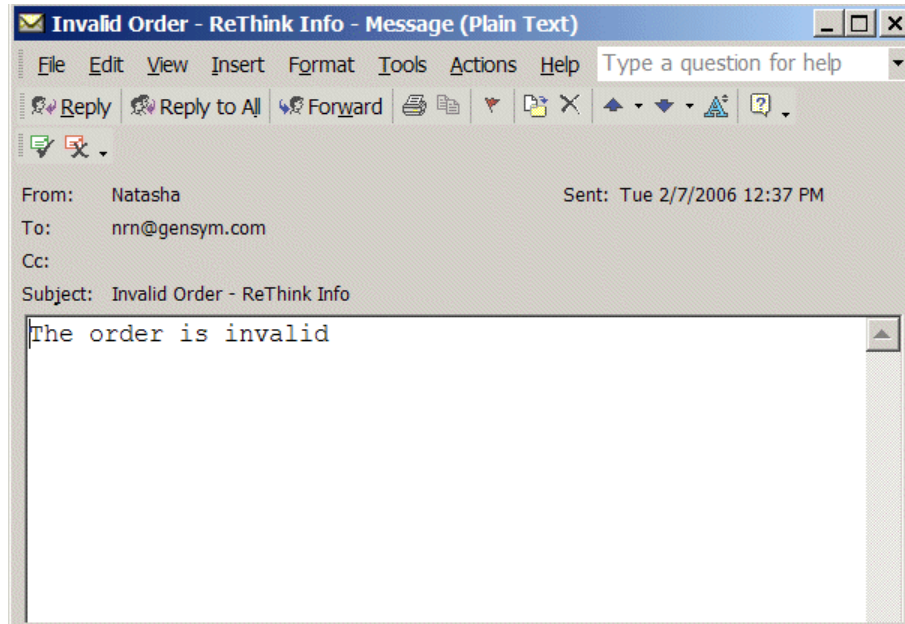
Here is the User Preferences dialog with both email addresses and rules configured:

The screenshot shows the 'User Preferences' dialog box with two sections: 'General' and 'Message Browser'. In the 'General' section, the 'Email Address' and 'Mobile Email' fields are circled in black. In the 'Message Browser' section, the 'Email Notification' and 'Mobile Email Notification' dropdown menus are also circled in black. The 'Apply' button is highlighted with a dashed border.

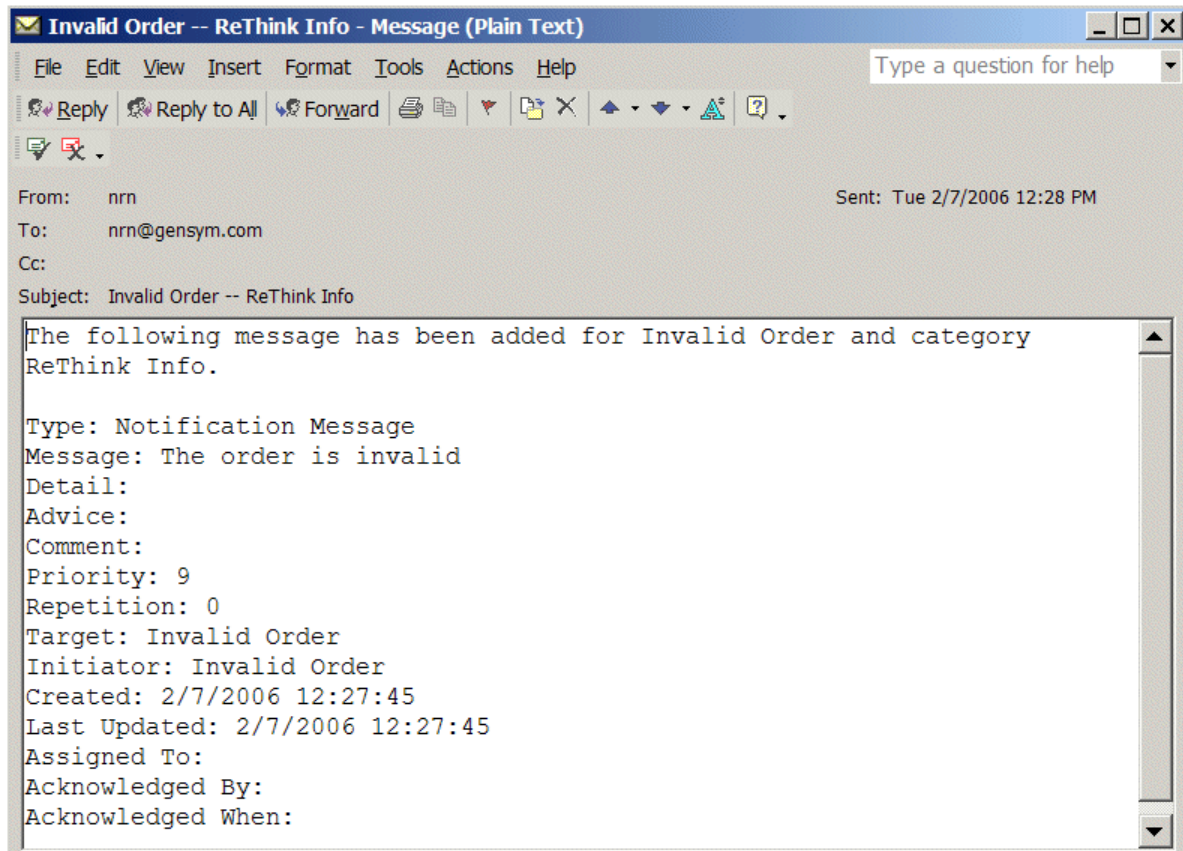
Section	Field	Value	Option	Checked
General	User Name	NRS	Set Default User Mode	<input type="checkbox"/>
	Default User Mode	MODELER	Indicate Items	<input type="checkbox"/>
	User Interface Theme	WINDOWS-THEME-2003	Extended Menus	<input checked="" type="checkbox"/>
	Email Address	abc@gensym.com	Show Logbook	<input type="checkbox"/>
	Mobile Email	xyz@gensym.com	Tabbed Mdi Mode	<input type="checkbox"/>
	Home Process Map	default view	Restore Last Pane Settings	<input type="checkbox"/>
	Telnet Command	"C:\Program Files\PUTTY\putty.exe"		
	Default Web Location	http://www.gensym.com		
Message Browser	Email Notification	SEND-AS-HTML	Ack Msg Upon Selection	<input type="checkbox"/>
	Mobile Email Notification	SEND-AS-SHORT-TEXT	Show Browser In Operator Mode	<input checked="" type="checkbox"/>
	Modeler Browser	GEVM-MODELER-MSG-VIEW-TEMPLATE	Enable Status Bar Message Browser	<input checked="" type="checkbox"/>
	Operator Browser	GEVM-OPERATOR-MSG-VIEW-TEMPLATE	Beep Enabled	<input checked="" type="checkbox"/>

Examples: Sending Email Messages

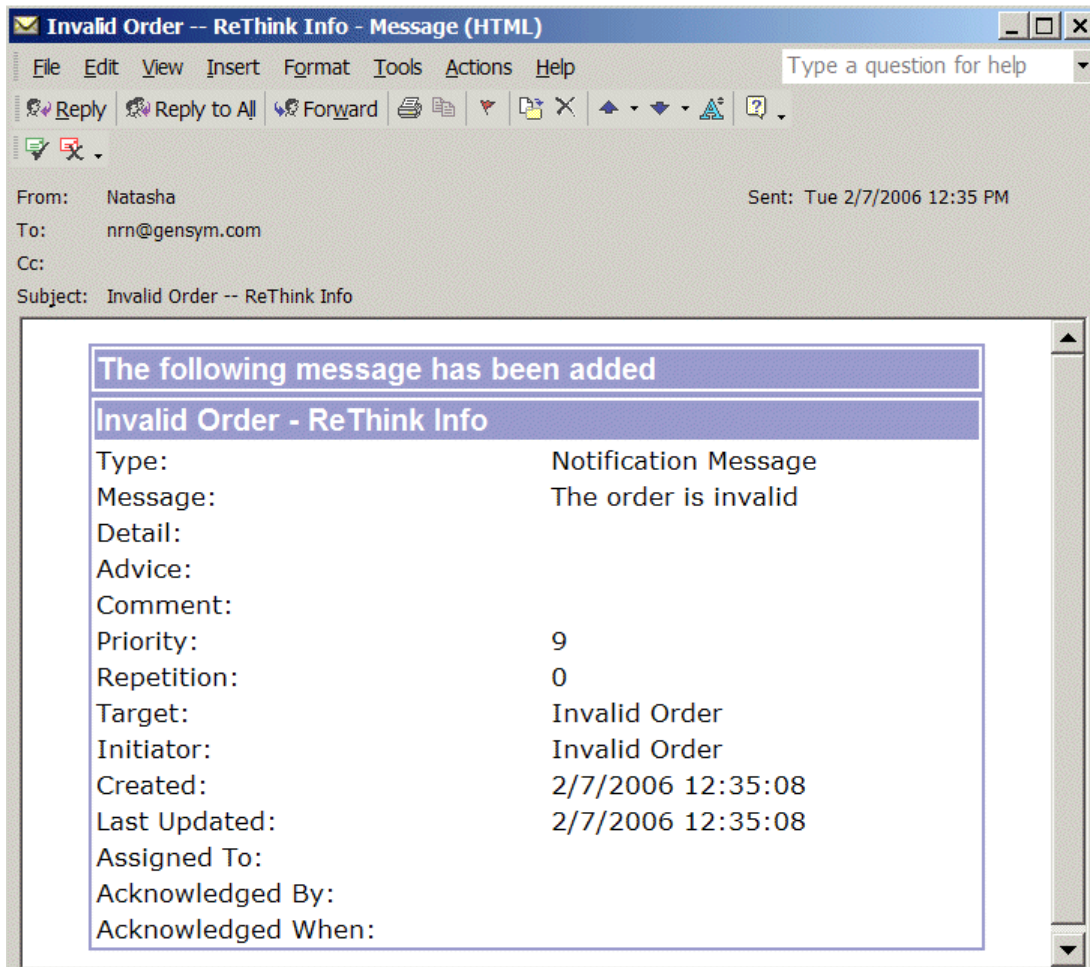
Here is an example of a message that includes the message text only in plain text:



Here is an example of a message that includes the message text and details in plain text:



Here is an example of a message that includes the message text and details in HTML format:



Configuring Startup Parameter for Sending Email Messages

You can configure the following startup parameter in the configuration file:

JMAIL-INTERFACE-NAME=none

Specifies the default JMail interface to use for sending email messages.

For details about using the configuration file, see the *G2 Run-Time Library User's Guide*.

Configuring Network Interfaces

ReThink allows you to configure various types of network interfaces for communicating with external systems, using the Project > System Settings > Interfaces menu.

Note To configure network interfaces, you must be in Developer mode.

SQL interfaces provide communication with databases. For details, see [Accessing External Databases](#).

You configure SMTP interfaces to send email when a message occurs. For details, see [Delivering Messages by Email](#).

ReThink also allows you to send email messages, using a JMail Connection pool and to send messages to JMS message servers, using a JMS Connection pool. For details, see [Sending Email](#) and [Using JMS Messaging](#).

Configuring Message Browsers

By default, all messages go to the Messages Browser, which you access by choosing View > Message Browser.

ReThink allows you to create and configure different message browsers and message queues. You do this by using the Project > System Settings > Message Browsers menu.

For details, see the *G2 Event Manager User's Guide*.

Configuring Module Settings

You can configure the default module settings object to customize various features of ReThink.

To configure BPR module settings:

- 1 Choose Tools > User Mode > Developer.
- 2 Choose Get Workspace > bpr-top-level.
- 3 Click the Settings button.
- 4 Clone the bpr-module-settings object and place it on a workspace in your top-level module.
- 5 Configure its attributes, as needed.

Here is the default BPR module settings objects:



a bpr-module-settings	
UUID	"e0819ef6c18a11d9b88500080266cd55"
Notes	OK
Item configuration	none
Names	none
Bpr indicate enabled	true
Bpr indicate add message to browser	true
Bpr indicate message type	gevm-notification-message
Bpr indicate message category	ReThink Info
Bpr indicate message priorities	9
Default bpr path line pattern	solid
Default bpr path arrows	very thin filled triangle
Bpr allow dot notation	true

Attribute	Description
bpr-indicate-enabled	Whether to show indicator arrows when errors occur.
bpr-indicate-and-message-to-browser	Whether to add indicator messages to the Message Browser.
bpr-indicate-message-type	The message type when adding indicator messages to the Message Browser.
bpr-indicate-message-category	The message category when adding indicator messages to the Message Browser.
bpr-indicate-message-priorities	The message priority when adding indicator messages to the Message Browser.
default-bpr-path-line-pattern	The default line pattern for paths between blocks.
default-bpr-path-arrows	The default arrow style for paths.
bpr-allow-dot-notation	Whether to allow dot notation for specifying attributes of subobjects in various blocks, for example, my-subobject.my-attr.

Using Blocks

Describes how to clone, connect, and configure blocks to create a model, and how to specify the duration and cost of block activities.

Introduction	105
Creating Blocks	107
Connecting Blocks	112
Configuring the Type of Work that Blocks Process	118
Configuring Blocks	126
Creating Hierarchical Views	144
Understanding the Activities of Blocks	147
Working with the Duration of Blocks	157
Working with Block Costs	196
Debugging Blocks	200
Customizing Blocks	204

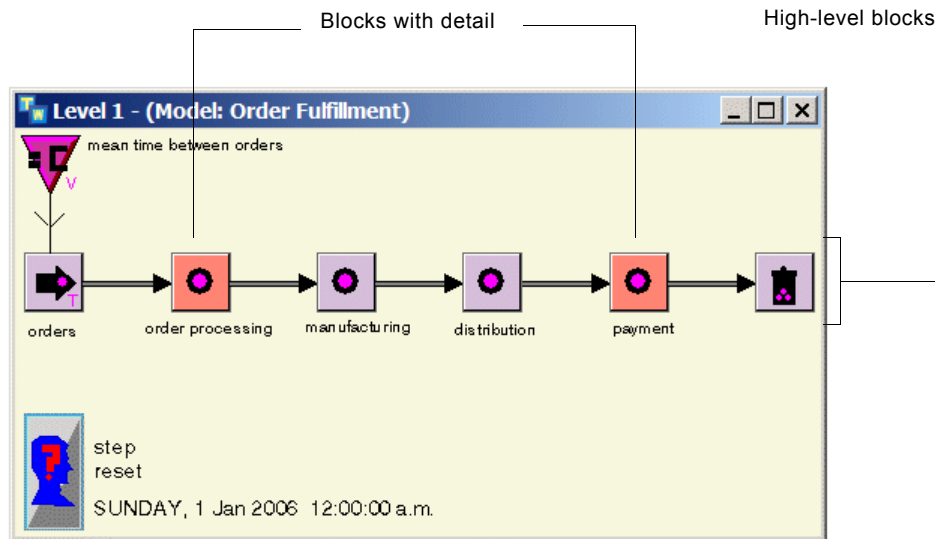


Introduction

You can use ReThink to model various types of business processes. For example, an order fulfillment process creates orders and invoices and ships products, and a manufacturing process manufactures widgets, places them in inventory, removes them from inventory, and replenishes the inventory.

You use a ReThink **block** to represent each step in the process, for example, billing or manufacturing. You use ReThink Task blocks to represent a high-level view of the process, which you can then break down into greater and greater levels of detail. The blocks at the lowest level in the process represent the individual **tasks**, which are value-added steps in the process that perform an action, contribute to costs, and/or have a duration.

This model shows a high-level view of an order fulfillment process, which consists of four basic tasks: Order Processing, Manufacturing, Distribution, and Payment. Two of the tasks have greater levels of detail: Order Processing and Payment.



The basic steps in creating a model are to:

- Create blocks from the ReThink toolbox.
- Connect the blocks together.
- Configure the parameters of the block.

The various tasks in the process operate on different work objects. They create them, delete them, copy them, establish associations between them, assemble them, disassemble them, and so on. Work objects flow along directed **paths**, which you use to connect blocks in the model, and represent the inputs and outputs of a business process.

When you run a model, work objects flow into and out of the blocks in the model by traveling on connection paths. Depending on the resource constraints you place on the model, work objects can flow freely through the model, or they can back up on an input path to a block.



When a block processes its work objects, it creates an activity object, which represents the amount of work required to process the work objects on the input

paths of a block. Each activity has an associated duration and cost, which you can view.

Each type of block computes summary metrics, including the number of activities, the total duration, and the total cost. You use ReThink instruments to do performance analysis on the model, based on these block metrics.

For more information on...	See...
Work Object	Using Work Objects.
Resources	Using Resources.
Instruments	Using Instruments.

Creating Blocks

The first step in creating a model is to create blocks from the Basic Activities tab of the ReThink toolbox and place them on a model detail.

Following is a summary of each ReThink block.

See also [Blocks Reference](#).

Creating Blocks

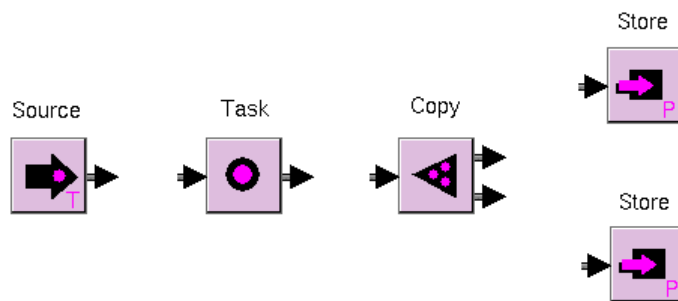
To create blocks:

- 1 Display the Basic Activities palette of the ReThink toolbox:



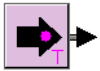
- 2 Select a block in the toolbox, then click on a workspace to place the block.
ReThink creates default stubs, depending on the type of block.
- 3 Display the properties dialog for the block and configure the Label as a text, then drag the label to the desired location next to the block.

The following figure shows a Source, Task, and Copy block, and two Store blocks:



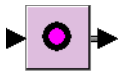
You connect various blocks together to form a model. For more information, see [Connecting Blocks](#).

Source Block



You use a Source block to create objects that are the inputs to a process. For example, you might use a Source block to create orders, service calls, or any other kind of external stimulus or impetus. You specify the frequency with which work flows into a model from a Source block, using a distribution, a file, or a database.

Task Block



You use a Task block to represent any activity that adds value in a process. It can have any number of inputs and outputs. It represents a primitive activity, a collection of activities, or a subprocess. You can decompose a Task block into multiple levels of detail, as needed. For example, you use a Task block to model each high-level stage in an order fulfillment process, such as order processing, manufacturing, and delivery, where each high-level task describes the details of each subtask.

Sink Block



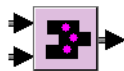
A Sink block is the counterpart to a Source block. You position it at the end of a process. ReThink does not require that you use a Sink block, but it is useful for indicating the end of a particular line of processing when you do not need to save the work objects.

Copy Block



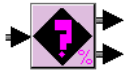
You use a Copy block to create multiple copies of an object. A Copy block can have any number of inputs or outputs. Whenever a Copy block receives an input, it outputs as many copies as it has output paths. For example, you use this block to make multiple copies of a document in a process.

Merge Block



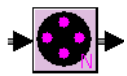
You use a Merge block to combine different types of objects onto a single output path. The block can have any number of inputs, but it typically has only one output. Whenever it receives an object, it sends the object onto its output path for sequential processing. You use this block, for example, to merge two separate streams of orders.

Branch Block



You use a Branch block to implement any kind of decision-making operation, for example, routing, sorting, collating, or any other kind of branching. A Branch block supports several kinds of decision-making. For example, you can branch work based on probability, based on the type of object that the Branch receives, based on the value of an attribute of an object, or based on the output path the user selects. You can also implement more complex types of branching by specifying proportions based on the number of times a work object loops around a process, or by specifying rules that test the values of multiple model attributes.

Batch Block

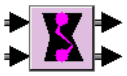


You use a Batch block to gather a batch of items before sending them downstream in a process. A Batch block supports several types of batching. You can batch work by waiting for a critical threshold of objects to arrive at the block before passing them downstream in the process, such as batching items into a box or boxes onto a truck. You can batch work by waiting until the sum of values of an attribute of an input work object exceeds a threshold, such as the weight of a box. You can batch work by waiting until a trigger work object arrives at the block, such as a truck. Finally, you can batch work objects according to a schedule, such as once an hour between the hours of 9:00 and 5:00 on weekdays.

Associate and Reconcile Blocks



You use an Associate block to associate two objects that the model processes at different rates. For example, you use an Associate block to associate an order and its invoice so you can match them again later in the process. You can create a new association between objects, add objects to an existing association, or remove objects from an association. You can also choose to associate all of the objects on the input path.



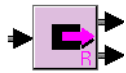
You use the Reconcile block to match together associated objects. For example, you use this block to match associated orders and invoices. Orders and invoices form input queues, waiting for their associates. When the associated invoice for an order arrives, the Reconcile block outputs the order and the invoice together, each on its own output path. If the input work object is associated with multiple work objects, you can require that the Reconcile block wait until all associated objects arrive at the block before reconciling them all.

Store and Retrieve Blocks



You use the Store block to model any kind of storage operation. This block can store its inputs in a resource pool, where they wait until a Retrieve block retrieves them. For example, you use a Store block to model a customer database, an order database, or a manufacturing inventory. It is a versatile block; you use it to model any place in the process that represents off-line storage. You can also store objects

to a file or to a database and then use a Retrieve or Source block to create objects from this file or database.



You use the Retrieve block to retrieve items from a pool or from a database, for example, to retrieve an object from inventory to fill an order. You can retrieve objects from a pool at random, based on an object being associated with another object, or based on an attribute value of the retrieved object. You can also retrieve objects one at a time or all at once, or you can retrieve a copy of the object, rather than the actual object.

Insert and Remove Blocks



You use an Insert block to insert objects into a container object, and you use a Remove block to remove the inserted objects from the container. For example,



you use the Insert block to add line item objects to an order, and you use the Remove block to remove those line items at a later time. You can insert and remove objects one at a time or all at once.

Copy Attributes Block



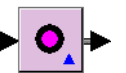
You use the Copy Attributes block to copy attribute values from one object to another object. For example, you use this block to copy tracking information from an existing invoice to a purchase order when the purchase order is created.

Yield Block



You use the Yield block splits the input work object into two objects and computes the yield, based on an attribute of the input work object. The block copies the computed yield into the attribute of the output work objects that represent manufactured products. It copies the balance into the attribute of the output work objects that represent defective products. You can configure the block to compute the yield, based on a random function, a specific proportion, or an attribute of the input work object.

BRMS Task Block



You use the BRMS Task block to invoke Business Rules Management System (BRMS) rules on the work objects it processes. BRMS rules provide a mechanism for easily editing, organizing, analyzing, and executing business rules.

Connecting Blocks

You have a number of ways to connect blocks together to form a model, depending on the current configuration of the blocks.

Using Stubs to Connect Two Blocks

All blocks on the ReThink toolbox have default **stubs**, which you use to create connections between blocks. The basic way of connecting blocks is to connect the default stub from one block to the default stub of another block.

To connect two blocks, using stubs:

- 1 Click the output stub of one block to attach it to the mouse.
- 2 Drag the stub into the input stub to which you want to connect.
- 3 Click with the mouse to connect the stubs.



Inserting a Block Between Two Connected Blocks

Sometimes, you have to insert a block between two blocks that are already connected. To do this, first you delete the connection between the blocks, then you connect the new block between them.

To insert a block between two connected blocks:

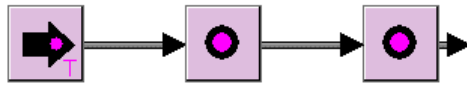
- 1 Choose Delete on the path between the two blocks and click OK to confirm the deletion.

ReThink disconnects the blocks:



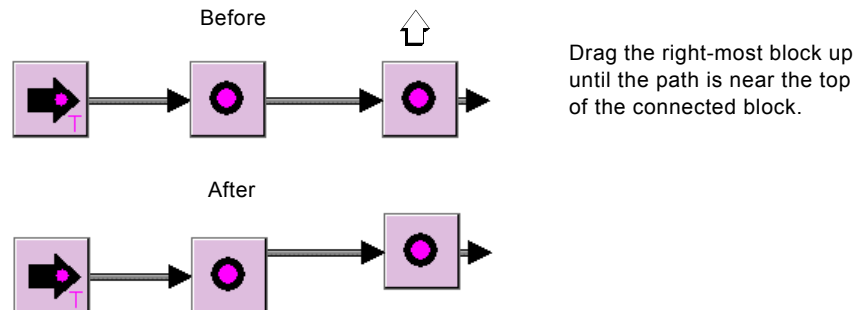
- 2 Move one block out of the way to make room for the new block.
- 3 Create a new block from the Basic Activities palette of the ReThink toolbox and place it between the two blocks.
- 4 Align and distribute the blocks, as needed.

- 5 Connect the stubs of all blocks:



Redisplaying the Paths of Connected Blocks

When you move connected blocks to a new location, ReThink logically redisplay the paths automatically for you along the existing side of the block. For example:

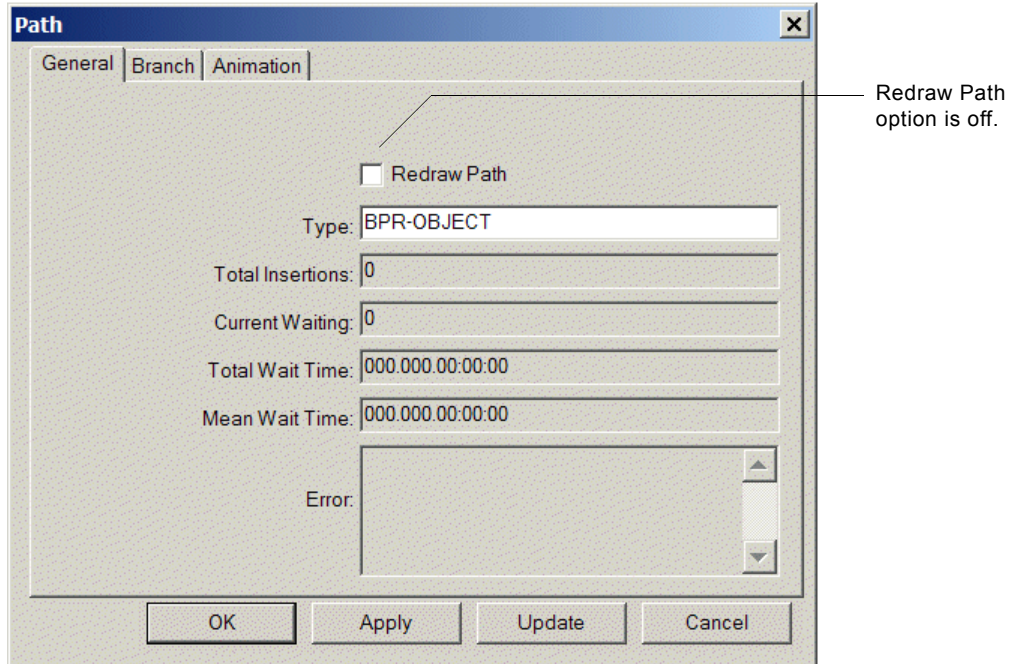


Disabling Path Redrawing

Sometimes, you might not want the path to adjust its location along the existing side of a block when you move the block. You can disable path redrawing to create a bend in the path rather than adjusting its location along the block.

To disable path redrawing:

- 1 Display the properties for the connection path between two blocks.
- 2 Click the Redraw Path option off.



Creating and Deleting Stubs

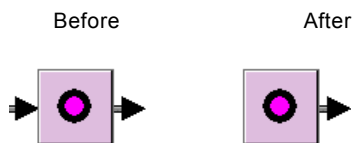
Often, you need to add input or output stubs to a block. For example, when a Task block processes multiple work objects, you need to add stubs to the block. Typically, you delete the existing stub first so you can space the stubs evenly along the side of the block.

Deleting a Stub

You might need to delete a stub on a block. For example, to evenly space a number of stubs that you create, you might first delete the existing stub.

To delete a default stub on a block:

- ➔ Click the stub to attach it to the mouse, drag the stub into the center of the block, and click to delete the stub.



Creating a New Stub

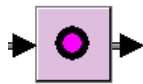
Sometimes, you need to create new input and output stubs on a block. Once you have created the stub, you can drag it to a new location on the block.

Alternatively, if you need to create a new stub in a different location on the block, you can simply drag a stub from another block into the block, then delete the connection.

To create a new input stub:

→ Choose Create Input on the block.

ReThink adds a stub to the left-hand side of the block:



To create a new output stub:

→ Choose Create Output on the block.

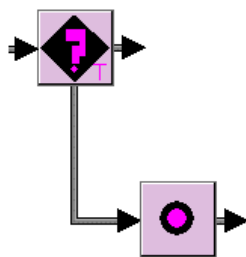
ReThink creates an output stub on the right-hand side of the block. It places the first new output stub above the existing stub and the next output stub below the existing stub.



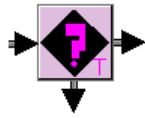
Sometimes, you need to create stubs that come out of the top or bottom of the block instead of out of the left or right side of the block. For example, when making loops in a diagram, it is often useful to have the paths leading out of the bottoms of the connected blocks.

To create an output stub coming out of the bottom of a block:

- 1 Clone a Task block from the Basic Activities palette of the ReThink toolbox and place it below the block on you want create an output stub.
- 2 Drag the *input* stub of the Task block into the bottom of the block.



- 3 Delete the connection between the blocks.



You now have an output stub coming out of the bottom of the block.

- 4 Delete the Task block.

You can use this same technique to create output stubs coming out of the top of the block, or input stubs coming out of the top or bottom of the block.

For an example, see [Creating Loops in a Diagram](#).

Creating Loops in a Diagram

Often you need to create loops in a diagram. For example, the output path of a Branch block might loop back into a Merge block based on some probability, or the output path of an Insert block might loop back into a Merge block to insert objects into a container object.

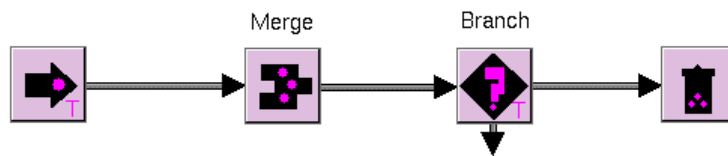
For example, suppose you wanted to make a loop from a downstream Branch block to an upstream Merge block.

To create a loop in a diagram:

- 1 Create an output stub leading out of the bottom (or top) of the downstream block.

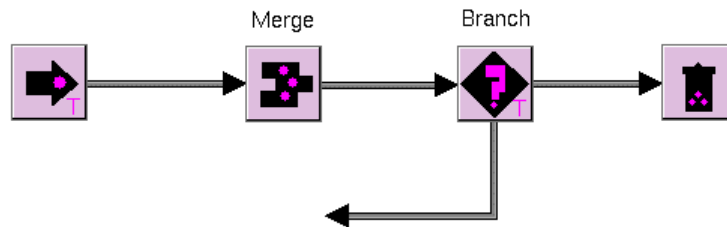
For details, see [Creating and Deleting Stubs](#).

The model should look like this:



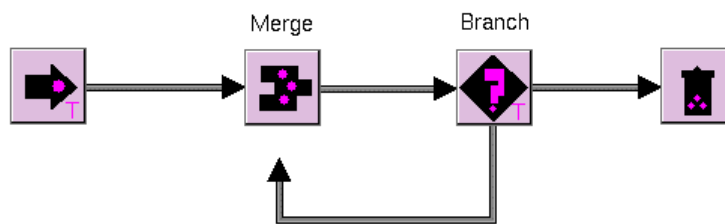
- 2 Click the output stub of the downstream block and drag it down (or up) and across to create the first bend in the loop.

The model should look like this:



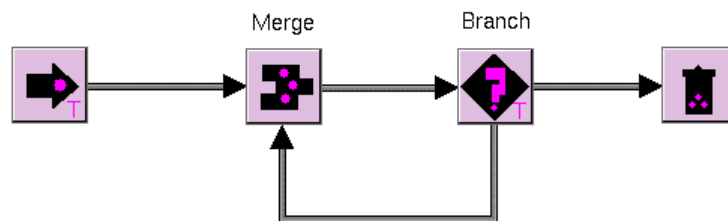
- 3 With the stub still connected to the mouse, click the mouse again and move the stub up (or down) to create the second bend in the loop.

The model should look like this:



- 4 Move the stub to the upstream block and click to connect.

The model should look like this:



Replacing Blocks

You can drop a new block on top of an existing connected block to replace the block. ReThink maintains all existing connections. The new block copies the configuration information from the existing block and uses it in the new block's configuration. For example, if the Mean of the original block is 1 hour, the Mean of the new block will also be 1 hour, even if they are different types of blocks.

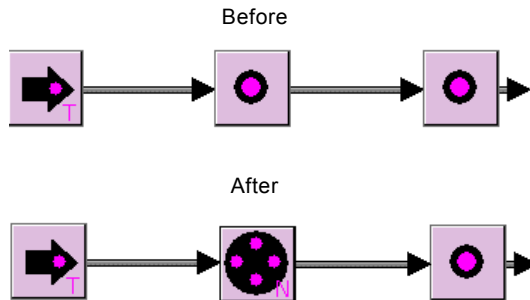
Note If you replace a Task block that has detail with a block other than another Task block, the new block no longer has detail; however, if you swap a task that has detail with another task, the new task still has detail.

To replace an existing connected block with a new block:

- ➔ Create a new block from the Basic Activities palette of the ReThink toolbox, place it exactly on top of the existing connected block, and click to replace.

Note You might need to adjust the blocks slightly to redisplay the paths.

The model might look like this:



Configuring the Type of Work that Blocks Process

ReThink blocks process work objects, which ReThink automatically creates when you run a simulation. To configure the type of work that a model processes, you specify the name of a work object class as the **path type** for the various blocks in the model, or you use the default. The type you specify depends on the type of block and the requirements of the model.

By default, ReThink work objects are a subclass of `bpr-object`, which is an internal class definition that ReThink provides. If the class of work object that you specify on a path does not already exist, ReThink automatically creates a class definition and places it on the current workspace.

You can also configure the color of the various paths in the model.

For general information on work objects, see [Using Work Objects](#).

Configuring the Path Type

You configure the path types of blocks by specifying a new or existing work object class as the value of the Type parameter of a path.

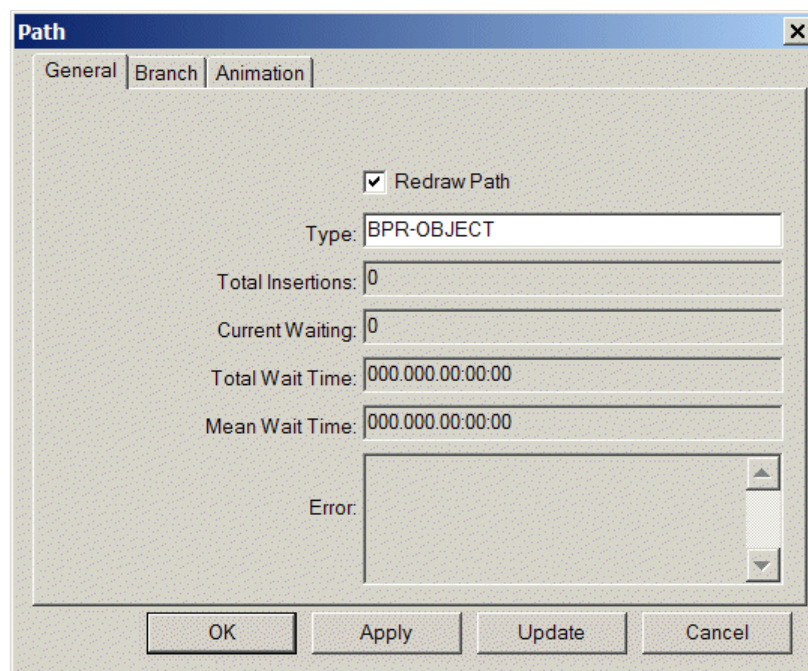
The default path type is `bpr-object`, which allows any work object that is an instance of `bpr-object` or its subclasses, to travel on the path. If you do not specify the path type for a path, ReThink uses the default path type when it determines how to process the object. This behavior varies depending on the type of block.

Tip In general, you should be as specific as possible when you specify the path type, depending on the requirements of the model.

If you are configuring the output path of a Branch block, the properties dialog contains additional parameters. For more information on path parameters of a Branch block, see [Path Attributes that Pertain Only to Branching](#).

To configure the type of work object that a block processes:

- 1 Display the General tab of the properties dialog for the path whose type you want to configure to display this dialog:



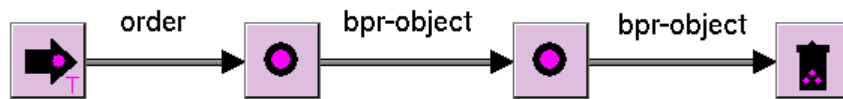
- 2 Configure the Type to be the class name of the work object that the block processes.

The following sections show examples of configuring path types.

Using the Default Path Type

Often, it is not necessary to configure the path types of a block. For example, when you configure a Task block to process a work object whose type has already been specified by an upstream block, you do not need to specify the input or output path types. The task passes the work object on its input and output paths, using the default path type.

The following simple model illustrates this concept, where the second Task block uses the default path types for both its input and output paths:

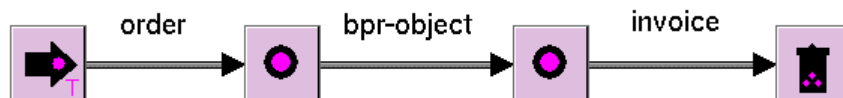


In this example, the Source block generates an order, as specified by its output path type. The order then flows to the two Task blocks, whose path types are both `bpr-object`. The order can flow on the output paths of the two Task blocks because `order` is a subclass of `bpr-object`.

Creating Work During Processing

Sometimes, you need to specify only the output path type. For example, when you configure a Task block that deletes the input object and generates a new output object, you only need to specify the output path type.

The following model is a slight variation on the previous model, where the second Task block deletes the order and generates an invoice:

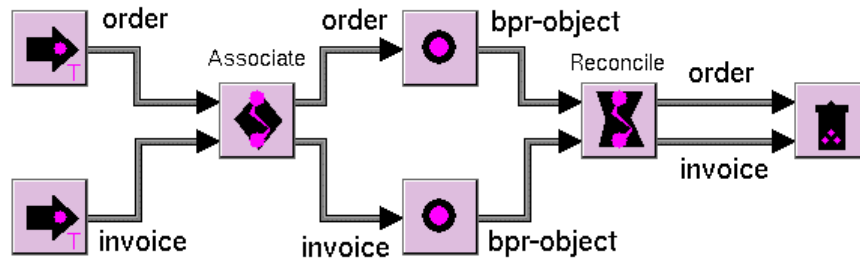


It is important to understand when ReThink creates a work object in a process and when it deletes the work object. Knowing this, you can interpret the metrics that ReThink computes for work objects, such as the creation time and the total amount of work applied to that object. In the example above, the **creation time** of the order is the start time of the simulation plus the deviation of the Source block, whereas the creation time of the invoice is the simulation time at the moment the second task creates the invoice.

Note that because the second task deletes the order, by default, the metrics associated with the order are not longer available to the model, and the metrics associated with the invoice are based on a different creation time from that of the order. However, you can configure the Task block to copy attribute values from the input to the output work object. For details, see [Copying Attribute Values to the Output Object](#).

Configuring the Path Types of Particular Blocks

For certain types of blocks, you need to configure both the input and output path types. For example, when you configure an Associate block, you must configure both the input and output path types to identify the two associated objects, as this model illustrates:



With the Associate block, as with other blocks with multiple input and output paths, it is important that you configure the output path types so that the inputs travel down the appropriate output path. In the example above, if you used the default output types for the two output paths of the Associate block, ReThink would process both the order and the invoice on the same path.

Certain blocks require input work objects that are a subclass of another built-in ReThink class, `bpr-container`.

When you configure the output paths of a Branch block, you also need to specify various path parameters relating to branching.

For specific information on how each block handles input and output path types, see the description of each block in [Blocks Reference](#).

For an example of specifying a container output path, see [Specifying a Container as the Path Type](#).

For information on specifying the output path types of a Branch block, see [Branch](#).

Creating Class Definitions for Work Objects

If the type of work object you configure does not exist, ReThink automatically creates a class definition for the object and places it on the model workspace. For certain blocks and instruments, you must pre-define work object class definitions with specific attributes. For example, when you use a Timestamp feed, you feed values into a user-defined attribute of a work object, which you must define in a class definition.

For more information on creating work object class definitions, see [Creating a New Class of Work Object](#).

Determining the Output Path Based on Its Type

Whenever you specify the path type of an output path of a block with multiple output paths, work objects pass onto the output path whose type most closely matches the object or a superior class of the object. This is true for all blocks with multiple output paths.

For example, if the input work object type is `order`, and the two output path types are `order` and `bpr-object`, the work object will flow on the output path whose type is `order`. Similarly, if the input work object type is `order`, which is a subclass of `form`, and the two output path types are `form` and `invoice`, the work object will flow on the output path whose type is `form`.

If no output path is capable of carrying the input work object, ReThink deletes the work object.

If two or more output paths are capable of carrying the same type of input work object, the output path of the work object is unpredictable.

Configuring the Animation of Paths

You might want to configure the color of paths to represent different categories of objects, for example: blue for physical materials and yellow for information, or red for phone calls, yellow for faxes, and blue for mail.

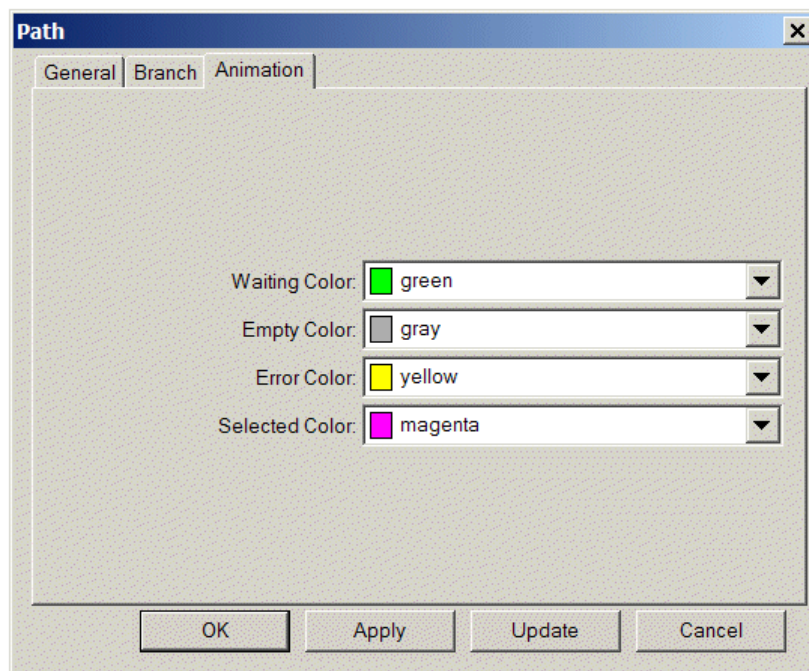
You can configure these colors of a path when it animates:

Animation Parameter	Description
Waiting Color	The color the path uses when there are work backups.
Empty Color	The color the path uses when no work backups exist.
Error Color	The color the path uses when it is in an error state.
Selected Color	The color the path uses when you select it for certain operations, such as when you use the Choose Original Output Path of a Copy block.

For information on...	See...
Work backups	Showing Work Backups on an Input Path.
Configuring the animation of all paths	<i>Customizing ReThink User's Guide.</i>

To configure the colors of a path when it animates:

- 1 Display the properties dialog for a path and click the Animation tab:



- 2 Choose a color from the dropdown list for the path color.

Configuring Path Types of Specific Blocks

When using blocks with multiple output paths, you must configure the output path types correctly. For example, when you use a Remove block to remove an invoice from a file container object, you must configure the output path types to carry an invoice object and a file object.

The following table lists each block that requires that you specify an output path type:

This block...	Requires that you configure the output path types of...
Branch	The various objects that the branch block processes when Branch Mode is Type.
Associate	The objects to be associated.
Reconcile	The objects to be reconciled.
Retrieve	The objects to be retrieved and the objects that do not meet the criteria.
Remove	The container, the empty container, and the objects in the container.
Copy Attributes	The object whose attributes are being copied and the target object.

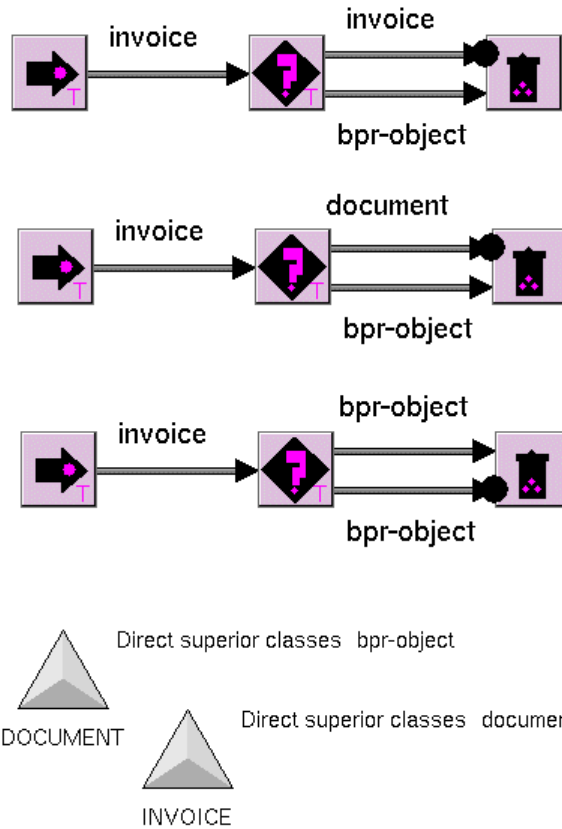
In general, you should always use the most specific class available as the output path type to ensure that only the desired types of work objects travel on a particular path. Remember, if a block has several output paths, a work object travels on the path that most closely matches the object's type in the object class hierarchy.

For example, suppose an object of type `invoice` passes to a Branch block that has two output paths, one whose type is `invoice` and the other whose type is `bpr-object`. Further suppose that `invoice` inherits not from `bpr-object` but from `document`, which in turn inherits from `bpr-object`. The `invoice` will pass onto the output path whose type most closely matches its class, which is `invoice`.

Now suppose the output path types were `document` and `bpr-object`. In this case, the `invoice` would pass onto the output path whose type is `document`, because `invoice` inherits its definition from `document`.

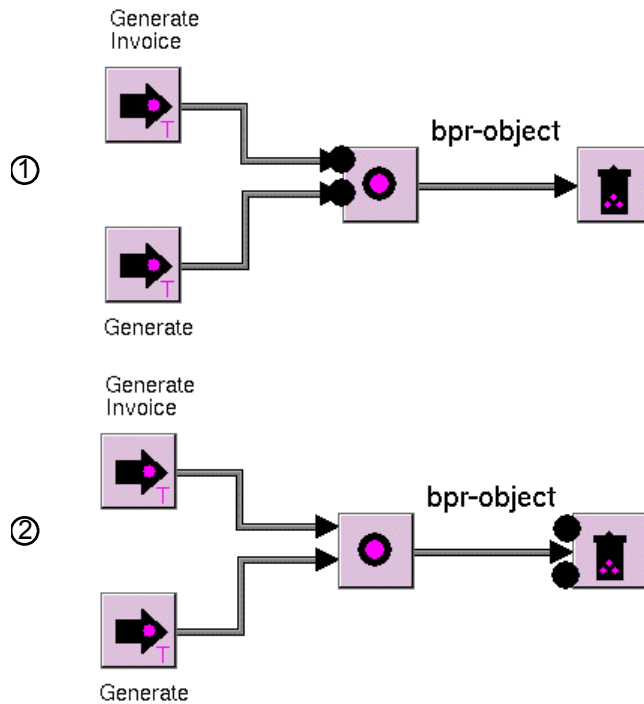
If no output path type named `invoice` or `document` exists, the `invoice` would pass onto the output path whose type is `bpr-object`, because `bpr-object` is at the top of the object class hierarchy of the `invoice` object.

The following figure illustrates this concept, using three different models:



To provide another example, suppose a Task block has two input paths, one of type invoice and the other of type order, and only a single output path of type bpr-object. Because bpr-object is at the top of the object class hierarchy, both invoices and orders travel on the output path.

The following figure shows two steps in such a model:



Configuring Blocks

Once you have created a block and placed it on your workspace, connected it to other blocks in the model, and configured the type of work object the block will process, you typically configure various aspects of the block.

To configure a block, you specify one or more of the following aspects of the block, either through the properties dialog or by using a specific menu choice:

- Common block parameters, such as the block label.
- Specific block parameters, such as the block mode.
- Path identity of specific blocks, such as the original output path of a Copy block.
- Block duration, such as the mean time between activities, using a random normal distribution.
- Fixed and variable block costs.
- Block animation and colors, such as active and inactive colors.

While all blocks behave similarly in terms of how you configure them, each block has unique requirements. The following sections describe how to configure blocks in general, as well as how to configure and use specific blocks.

Configuring General Block Parameters

Every block defines these parameters on the General tab of the properties dialog:

- Block Label, which is a mixed-case text string that identifies the block.
- Comments, which is a mixed-case text string that provides information about the purpose of the block in the model.
- Maximum Activities, which is an integer that represents the maximum number of activities the block can process concurrently.
- URL, which is a reference to an HTML file, either on the World Wide Web or on the file system, or to an RTF file, which describes how the model uses the block. When this attribute is configured, choosing Show URL or clicking the block displays the file in a browser window.

The Block Label appears next to the block as an attribute display.

For information on hiding the label associated with the block, see [Using Attribute Displays](#).

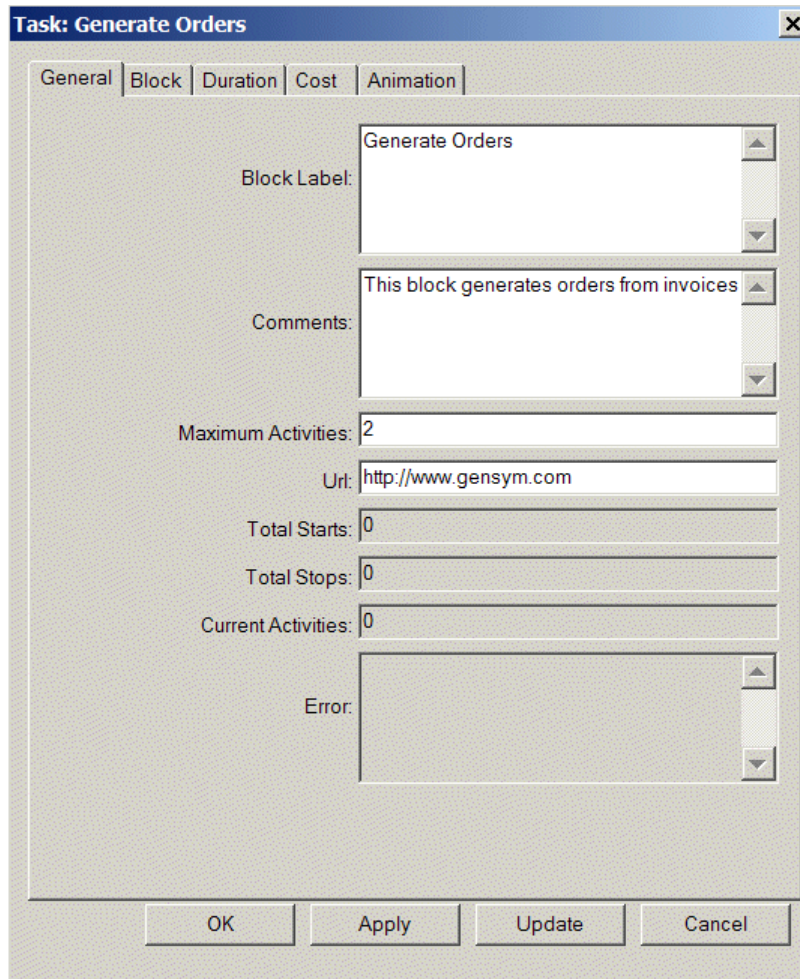
For information on configuring Maximum Activities, see [Limiting the Number of Concurrent Activities](#).

For information on configuring the URL, see [Using the Go Menu](#).

To configure the general block parameters:

- 1 Display the properties dialog for the block and click the General tab.
- 2 Configure the Block Label, Comments, Maximum Activities, and URL of the block.

For example, this dialog shows the general parameters of a Task block:



Configuring Specific Block Attributes and Features

Many blocks provide additional parameters that pertain only to the particular block. For example, a number of blocks provide a parameter for specifying the mode in which the block operates.

When you configure the mode of a certain block, you are often required to configure mode-specific parameters of the block. Depending on the mode, this typically involves configuring additional parameters in the dialog. However, certain modes require that you use a menu choice to configure a mode-specific feature of the block, for example, when storing work to a resource pool, you use a menu choice to identify the pool. In addition, certain blocks provide menu choices for configuring other specific features of the block, such as creating detail for a Task block.

Blocks provide default values for most of their specific parameters; thus, in general, you do not need to configure the specific block parameters unless you want the block to use non-default behavior.

Certain parameters do not provide a default value, which means you must configure them for the block to operate correctly. If you run the simulation without configuring a required parameter of the block, ReThink displays an indicator arrow next to the block when it attempts to execute.

You configure specific block parameters on the Block tab of the properties dialog, and you configure specific block features by using menu choices on the block.

The following table lists all the blocks, their specific parameters, and their specific menu choices, which you use to configure specific features of the block. All parameters and menu choices are optional unless identified as required. Unless otherwise stated, the specific parameters appear on the Block tab and in Modeler mode. For specific menu choices that pertain to configuring path identity, see [Configuring Path Identity of Specific Blocks](#).

This block...	Provides these specific parameters...	And these specific menu choices...
Source	Source Mode as type, object file, database, or custom Maximum Starts Start Time and End Time Output Count In object file mode, Object File Name and Repeat Object File In database mode, Database Interface Name and SQL Query (Database tab) In custom mode, Source Procedure Name (Developer mode only)	Single Shot Start
Task	Output Count Copy Attributes Copy All Attributes Detail Color (Animation tab)	Create Detail Show Detail Enable Detail Disable Detail
Copy	Output Count Add to Associations as true or false	None

This block...	Provides these specific parameters...	And these specific menu choices...
Branch	<p>Branch Mode as proportion, dynamic proportion, type, prompt, attribute value, or custom</p> <p>In prompt mode, Branch Prompt Message and Branch Timeout Period</p> <p>In attribute value mode, Rules Wait Interval, Branch Attribute, and Operation</p> <p>In custom mode, Branch Procedure Name (Developer mode only)</p>	<p>In attribute value mode, Create Rules lets you branch work based on multiple attribute values, and Show Rules show the rules workspace.</p>
Batch	<p>Batch Mode as number, sum, trigger, interval, or custom</p> <p>Container List Attribute</p> <p>Threshold</p> <p>In sum mode, Attribute Name and Minimum Threshold</p> <p>In interval mode, Start Time, End Time, Period, and Days</p> <p>In custom mode, Batch Procedure Name (Developer mode only)</p>	None
Associate	<p>Association Name (required)</p> <p>Mode as new, add, or remove</p> <p>Associate All as true or false</p>	None
Reconcile	<p>Association Name (required)</p> <p>Reconcile All as true or false</p>	None

This block...	Provides these specific parameters...	And these specific menu choices...
Store	<p>Store Mode as pool, file, database, or custom</p> <p>In file mode, Object File Name and Duration File Name</p> <p>In database mode, Database Interface Name, Database Table, Database Key, and SQL Query (Database tab)</p> <p>In custom mode, Store Procedure Name (Developer mode only)</p>	<p>In pool mode, Choose Pool identifies the storage pool (required), and Show Pool shows the pool.</p>
Retrieve	<p>Retrieve Mode as random, association, database, attribute value, or custom</p> <p>Retrieve All as true or false</p> <p>Retrieve Copy as true or false</p> <p>Add to Associations as true or false</p> <p>In association mode, Association Name.</p> <p>In attribute value mode, Retrieve Attribute, Attribute Value, Range Upper, Range Lower, and Operation.</p> <p>In database mode, Database Interface Name and SQL Query (Database tab)</p> <p>In custom mode, Retrieve Procedure Name (Developer mode only)</p>	<p>In random mode, Choose Pool identifies the pool (required), and Show Pool shows the pool.</p>
Insert	<p>Container List Attribute</p> <p>Mode as false, last, or all</p>	

This block...	Provides these specific parameters...	And these specific menu choices...
Remove	Container List Attribute Mode as false, last, or all	
Yield	Yield Mode Attribute to Split In random mode, Minimum Random Value and Maximum Random Value In random triangular mode, Minimum Random Value, Mode Random Value, and Maximum Random Value In work object mode, Work Object Yield Attribute Name In custom mode, Yield Procedure Name (Developer mode only)	

For information on these specific block parameters and menu choices, see the headings entitled “Specific Attributes” and “Specific Menu Choices” for individual blocks in [Blocks Reference](#).

For information on using database mode, see [Accessing External Databases](#).

To configure specific block parameters:

- 1 Display the properties dialog for the block and click the Block tab.
- 2 Configure the parameters on this tab, as needed.

Note When configuring the Source, Store, and Retrieve blocks in database mode, you configure database-related parameters on the Database tab.

Configuring Path Identity of Specific Blocks

A number of blocks require that you identify particular input and/or output paths of the block, by using menu choices on the block.

In most cases, configuring path identity for a block is a required feature of the block. Thus, if you run the simulation without configuring the path identity, ReThink displays an indicator arrow next to the block when it attempts to execute.

The following table lists all the blocks that define menu choices for configuring path identity. In addition to the menu choices listed in the table, each block defines corresponding menu choices for showing the identified path. All menu choices are required unless identified as optional.

This block...	Provides these specific menu choices for configuring path identity...
Batch	In trigger mode, Choose Trigger Input Path In trigger mode, Choose Trigger Output Path (optional)
Copy	Choose Original Output Path
Copy Attributes	Choose Original Input Path
Insert	Choose Container Input Path
Remove	Choose Empty Container Output Path Choose Nonempty Container Output Path
Retrieve	Choose Not Found Output Path (optional)
Yield	Choose Reject Path Show Reject Path

For information on these specific menu choices, see the heading entitled “Specific Menu Choices” for individual blocks in [Blocks Reference](#).

Configuring the Duration of Blocks

You determine the timing of events in the model by specifying the duration of blocks. The duration of a block determines the length of time that each activity in the model processes. An activity is the amount of work time applied to an object as a block processes it in the model. The duration of each activity in turn determines the:

- Duration of each block, work object, and resource.
- Total cost of each block, work object, and resource.
- Utilization of each resource.

If you do not specify the duration of a block, the values of the above metrics are all zero.

Typically, you specify a duration for these two types of blocks, at a minimum:

- Source blocks, to determine the frequency with which work objects flow into the model.
- Task blocks, to determine the duration of activities of all the value-added tasks in the model.

Depending on your model, it might or might not be necessary to configure the duration of other types of blocks, such as Branch blocks, Copy blocks, and Batch blocks.

You can specify a fixed or variable duration, depending on the requirements of the model.

For details on configuring the duration of a block, see [Working with the Duration of Blocks](#).

For information on activities, see [Understanding the Activities of Blocks](#).

To configure the duration of a block:

- 1 Display the properties dialog for the block and click the Duration tab.
- 2 Configure the duration parameters of the block, as needed.

For example, this dialog shows the duration parameters of a Task block that is configured to process work, using a random triangular distribution:

The screenshot shows a dialog box titled "Task" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Duration" tab is selected. Inside the "Duration" section, there is a "Distribution Mode" dropdown menu set to "Random Triangular". Below this are three rows of spinners for "Min:", "Max:", and "Mode:". The "Min:" row has three spinners with values "000", "000", and "01:00:00". The "Max:" row has three spinners with values "000", "000", and "03:00:00". The "Mode:" row has three spinners with values "000", "000", and "02:00:00". Below the "Duration" section is a "Miscellaneous" section with five text input fields: "Time Per Unit Attribute:" (empty), "Total Work Time:" (000.000.00:00:00), "Total Elapsed Time:" (000.000.00:00:00), "Creation Time:" (000.000.00:00:00), and "Average In Process:" (000.000.00:00:00). At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Configuring the Cost of Blocks

You might want to associate a fixed or variable cost with a block. You can also associate a fixed and variable cost with the resources associated with a block. ReThink adds the block costs and the resource costs, per activity, to compute the total cost of the block.

For details on configuring fixed and variable block costs, see [Working with Block Costs](#).

For information on specifying fixed and variable resource costs, see [Assigning Costs to Resources in a Model](#).

To specify the cost of a block:

- 1 Display the properties dialog for the block and click the Cost tab.
- 2 Configure the cost parameters of the block, as needed.

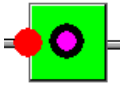
For example, this dialog shows the cost parameters of a Task block that is configured to have a \$10 fixed cost and a \$5 per hour variable cost:

The image shows a software dialog box titled "Task" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Cost" tab is currently selected. Inside the dialog, there are four rows of input fields:

- "Cost Per Use:" followed by a text box containing the number "10".
- "Cost Per Time Unit:" followed by a text box containing the number "5".
- "Time Unit:" followed by three separate input boxes. The first two contain "000" and the third contains "01:00:00".
- "Total Cost:" followed by a text box containing the number "0".

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Configuring the Animation of Blocks



You might want to use color to visually identify related blocks, such as blocks within the same organizational department.

You can configure these colors of a block when it animates:

Animation Parameter	Description
Active Color	The color the block uses when it is processing.
Inactive Color	The color the block uses when it is idle.
Error Color	The color the block uses when it is in an error state.
Detail Color	The color a Task block uses when it has detail.

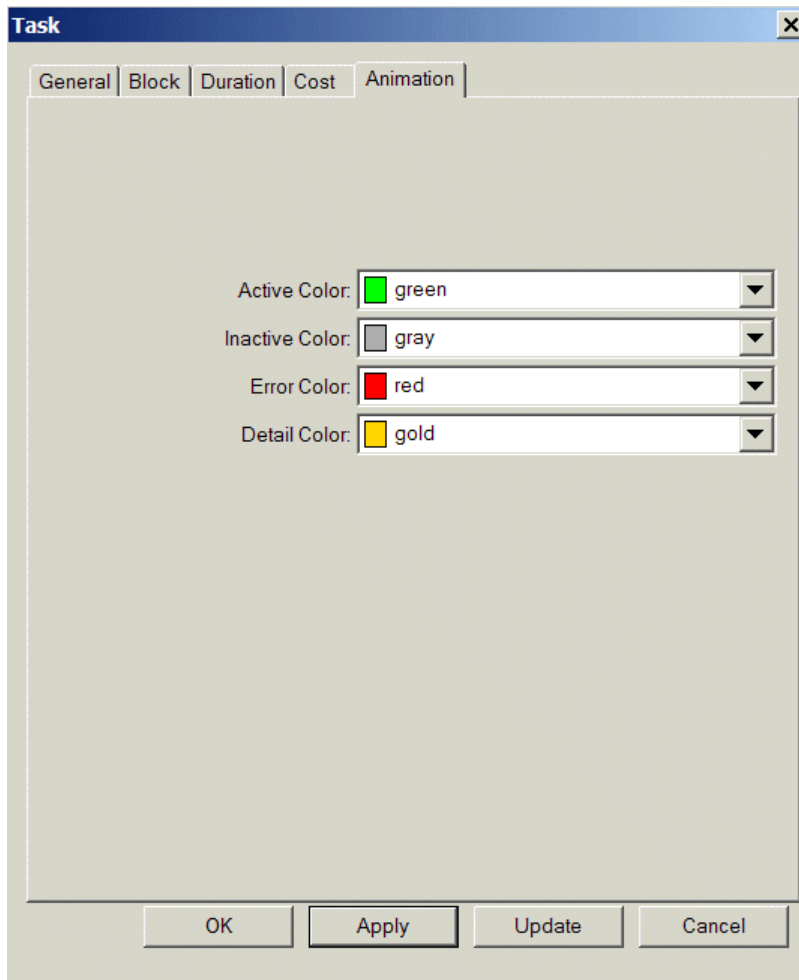
To configure the colors of a block when it animates:

- 1 Display the properties dialog for the block and click the Animation tab.

Note Detail Color is available for the Task block only.

- 2 Choose a color from the dropdown lists for each block color, as needed.

For example, this dialog shows the animation parameters of a Task block that is configured use non-default colors:



Configuring Specific Blocks

This section provides suggestions on how to use several of the most common blocks. For details on how to configure specific blocks, see [Blocks Reference](#).

Source Block

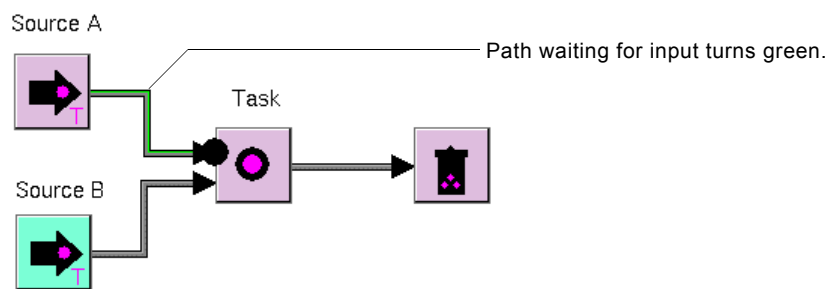
All models begin by introducing some kind of work object. In most cases, you use a Source block to generate work in a process, which you typically place on the left side of the workspace as the first block.

To determine the type of work object the Source block generates, you configure the Type parameter of the block's output path. If you do not specify a path type, the Source block generates a `bpr-object`, which is the default work object type.

The Source block can generate work objects by using the path type combined with a duration, by using an object file, or by using a database query.

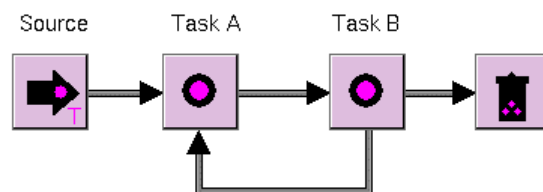
Task Block

You use a Task block for **synchronized processing**, which means the block processes its incoming work objects only when every input path has a work object. For example, if the Task block has two input paths and an object arrives at the block on only one of its input paths, the block waits to process the object until an object arrives on the other input path. While the block is waiting to synchronize its inputs, the path with the work object turns green, as the following figure shows:



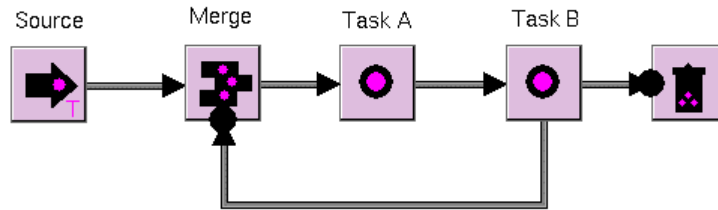
If you do not want the block to synchronize its inputs when it has multiple input paths, use a Merge block or a junction block instead, as shown in [Merge Block](#).

In particular, a common modeling mistake occurs when you attempt to create a loop in a diagram by feeding an output path of a downstream block into a Task block, as the following figure shows:

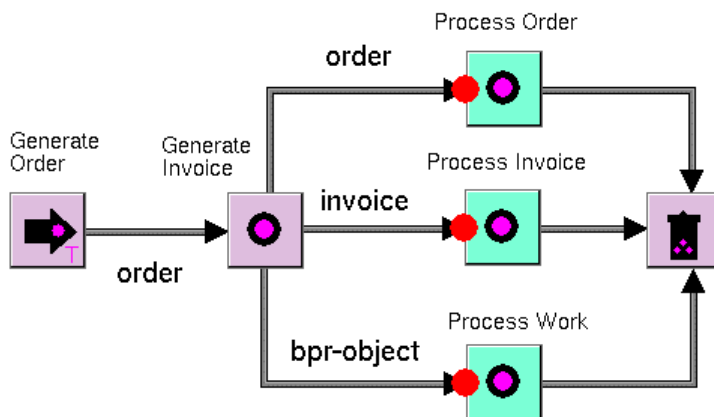


In this example, Task A must wait until an object arrives on the path that loops around the model from Task B before it can process the initial work object, which means Task A never processes.

If the desired behavior is to process work objects as they arrive at the block, feed the loop into a Merge block upstream of the Task block, as the following figure shows:



Another feature of a Task block is that it generates as many kinds of output work objects as it has output paths, regardless of the number of input paths, as this figure illustrates:



In this example, an order arrives at a Task block that has three output paths. The Task block processes the order and passes it to the output path of the corresponding type. At the same time, the Task block generates both an invoice and a bpr-object, and passes them onto the output paths with the corresponding type. If you do not specify an output path whose type is order, the Task block would send the order onto the bpr-object path because order inherits its definition from bpr-object. You can configure the Task block to copy attributes from the input to the output work objects.

When one task consists of several steps, it is good modeling practice to model the subtasks as detail. Creating Task blocks with detail makes the model cleaner to look at and easier to explain. However, it is still important to label the Task block with detail to provide someone viewing the model a high-level description of the subtasks without requiring them to look at the detail.

For more information on creating detail for Task blocks, see [Creating Hierarchical Views](#).

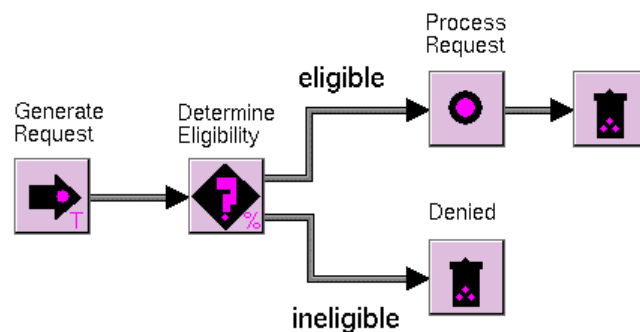
Branch Block

A Branch block can sort work objects by proportion, dynamic proportion, type, prompt, attribute value, or rule.

You can configure duration, cost, and resources for a Branch block, just as you can for a Task block. Therefore, when you need to branch work in a process, it is more efficient to use just a Branch block rather than a Task block followed by a Branch block.

It is also helpful to use free text to label the output paths of a Branch block so that someone viewing the model can understand the implication of each branch.

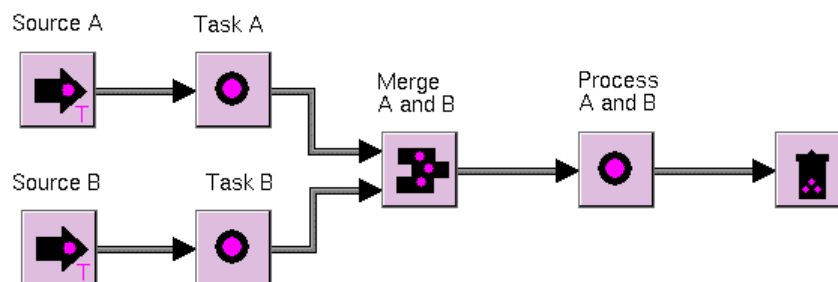
For example:



Merge Block

You use a Merge block for **sequential processing** to bring together work objects coming from different blocks when the process actually performs a value-added task of gathering together separate pieces of data. Unlike a Task block, a Merge block does not synchronize its inputs; instead, it processes work objects as they arrive at the block.

The following model merges work from Task A and Task B:



Insert and Remove Blocks

A Remove block removes items from a **bpr-container**, which you insert by using an Insert block. By default, a Remove block has the following three output paths:

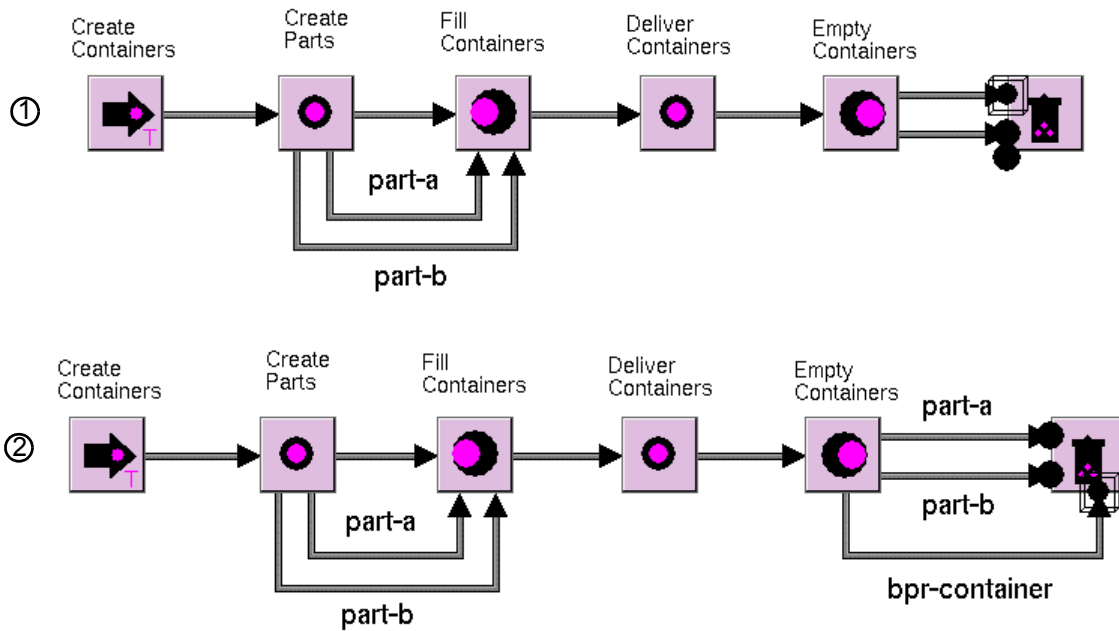
- Nonempty container path, which is the path that carries the container when it still has items in it. The block must identify this path when you configure it to remove objects one at a time, using a loop of some kind; if you configure the block to remove objects all at once, you can delete this path.
- Empty container path, which is the path that carries the container when all the items have been removed. The block must identify this path.
- The path that carries the items that the block removes from the container.

In order for the block to function properly, you must set the path types of these paths, and you must identify these paths by using the appropriate menu choices on the block.

If the container has more than one type of object, you have two options for setting the path that carries the items:

- Use the default path type of **bpr-object** so all types of objects pass onto a single output path.
- Create as many output paths as there are types of objects in the container and set each output path type accordingly.

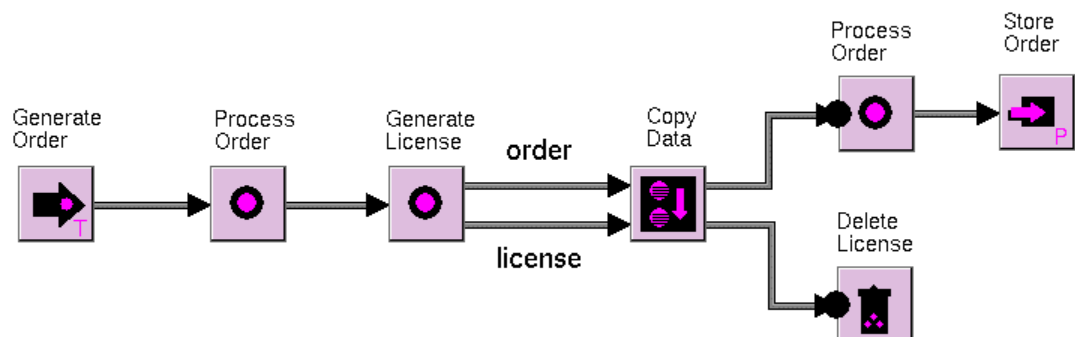
The following figure shows two versions of a model that fills and then empties containers. The first model uses a single output path for the objects removed from the container, whose type is **bpr-object**. The second model uses multiple output paths for the objects removed from the container, one for each type of object in the container. In the first model, the objects removed from the container have been moved on the path to show both objects.



Note that in the first model, if the container had a **part-c**, the Remove block would pass the part onto the **bpr-object** output path along with the other parts, whereas in the second model, the Remove block would delete the **part-c** object because no path of type **part-c** is specified.

Sink Block

You use a Sink block to eliminate work objects in a model when they are no longer needed. In the following example, an order and a license arrive at a Copy Attributes block named Copy Data. The order continues to a Task block and ultimately to a Store block, whereas the license is explicitly deleted, using a Sink block.



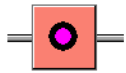
Custom Blocks

You can create custom blocks from standard blocks by customizing the procedures and methods that implement the blocks' behavior. For example, you might create a custom Retrieve block that retrieves two items from a pool each time it executes, rather than a single item.

Because the parent of a custom block is typically an existing block, by default, instances of custom blocks look identical to their parent. To avoid this, you should always edit the icon of a custom block to distinguish it from the standard block. The new icon should indicate the function that the block is performing. For example, you might add the number "2" to the custom Retrieve block's icon to distinguish it from the standard Retrieve block icon.

You store custom blocks in different locations, depending on how the model uses them. For more information, see the *Customizing ReThink User's Guide*.

Creating Hierarchical Views



When you create a model of a process, it is best to start at a very high level, understand the basic inputs and outputs for each process, then add the detail of each process. ReThink allows you to create **hierarchical views** in a model by creating progressively greater level of detail for a block.

For example, a Billing task might consist of several subtasks, processing the order and filing the order. You model these subtasks by creating blocks on the detail of the Billing task and configuring their parameters.

Note Once you create detail for a task, the duration and cost for the top-level task have no effect.

You create details for Task blocks only.

ReThink computes summary metrics for a task with detail. For example, each subtask computes its Total Work Time, which is the total amount of time the task spends processing work objects. The Total Work Time of the superior task is the sum of the Total Work Time values of each subtask. Similarly, Total Cost is the sum of the Total Cost values of all subtasks on the detail.

Note You cannot probe a Task block with detail. Probe the individual blocks on the detail instead.

For more information on how blocks compute total work time and total cost, see [Working with the Duration of Blocks](#) and [Working with Block Costs](#).

When you create a task with detail, ReThink computes metrics for both the blocks on the detail, as well as for the Task block. If you have many blocks with detail in

your model, performance can be affected. You can disable the detail so ReThink computes metrics for only the Task block.

For information about how to improve performance in a model that has Task blocks with detail, see [Configuring the Computation Behavior](#).

When you create block summary reports, you can control whether the report includes metrics on all blocks in the model or just the blocks on the detail.

For information about controlling the contents of a report for blocks with detail, see [Configuring the Scope of the Report](#).

Modeling the Detail of a Task

You model the details of a task by creating a detail subworkspace for the task. The task on the detail has the same specification as the top-level task, including its parameters, input and output path types, and label.

The detail has input and output **connectors** for each input and output path on the block. A connector on a detail causes work objects to flow from the input path of the superior task, to the blocks on the detail, and back up to the superior task.

When a Task block has detail, the color of the superior Task block changes to indicate that it has detail.

Note The path type of the output path leading into the right-most connector determines the path type of the object the superior task passes to the downstream block; the output path type of the superior task has no effect.

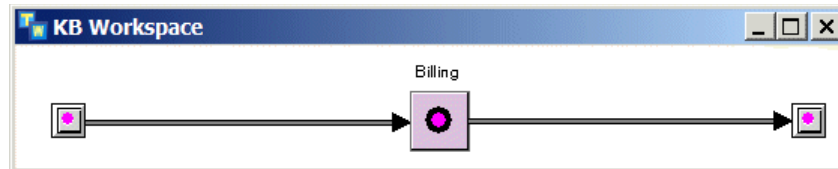
It is often useful to specify time delays in the superior task before you specify the detail so that the subtask on the detail inherits the specification of the superior task. That way, if you delete the task's detail, the superior task still has a meaningful specification. However, note that the duration and cost specifications in the superior task have no effect when the task has detail.

To model the detail of a task:

- 1 Choose Create Detail on the Task block.

ReThink displays the detail for the task. The detail contains input and output connectors, which are connected to a Task block with the same specification as the superior task.

Here is the detail for a Billing task with one input and one output path:

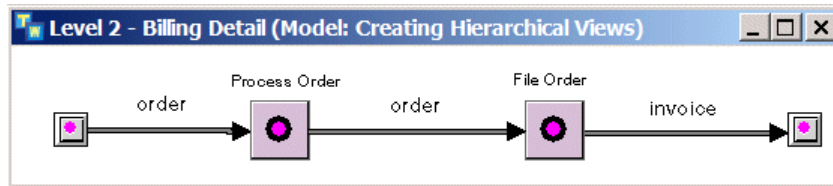


- 2 Create, connect, and configure other blocks on the detail.

For example, you might create a File Order subtask between the Process Order subtask and the connector at the right side of the detail.

To do this, you must delete the existing connection, as described in [Inserting a Block Between Two Connected Blocks](#).

The detail might look like this, where the labels indicate the path types that you configured for each path:



When you run the simulation, the work objects flow to the blocks on the detail of the Task block and up again.

Interacting with the Detail

You can hide and show the detail, and go to the connected path of the connectors on the detail. You can also enable and disable the detail, which allows you to run the simulation as if no detail existed for a task.

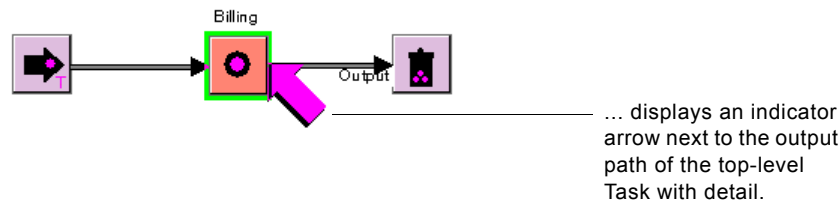
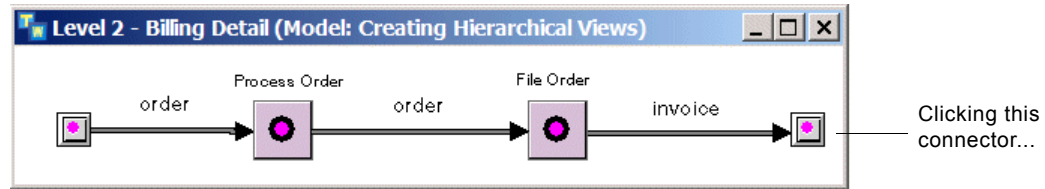
To display the detail of a task:

- ➔ Choose Show Detail on the block or click the equivalent toolbar button.

To show the connected paths of the detail:

- ➔ Click any connector on the detail.

ReThink displays an indicator arrow on the superior workspace next to the path to which the connector is connected and a label that indicates whether the connector is an input or output path, for example:



To disable detail:

→ Choose Disable Detail on a Task with detail.

The top-level Task block with detail behaves as if it had no detail, and its icon changes to indicate the detail is disabled:



To enable detail:

→ Choose Enable Detail on a Task with detail.

Work objects flow to blocks on the detail.

To delete the detail:

→ Select the detail workspace and choose Edit > Delete.

Understanding the Activities of Blocks



When a block processes its input work object or objects, it creates a single **activity** object. An activity represents the amount of work associated with processing all the inputs of a single block. You can view the activities associated with a task while the model is running by using a menu choice on the block.

The duration of an activity is determined by the duration mode and associated parameter values of the block. The number of **concurrent activities** depends on

the duration, the frequency with which work objects flow into the block, and the constraints associated with the block. If too many work objects exist for the given constraints on the block, work backs up on the input path to the block.

When a block processes multiple work objects, it creates multiple related activity objects, one for each set of inputs. Each activity computes the amount of work required to process a single block event, which includes all of the block's inputs, and the cost of processing that event.

The block sums the duration and cost of each activity to provide summary metrics for all the activities that the block processes.

Determining the Current Activities

ReThink keeps track of the activities of a block, using these metrics:

Attribute	Description
Total Starts	The total number of activities that the block has started processing since the start of the simulation.
Total Stops	The total number of activities that the block has finished processing since the start of the simulation.
Current Activities	The number of work objects that the block is currently processing. The Total Stops plus the Current Activities equals the Total Starts.

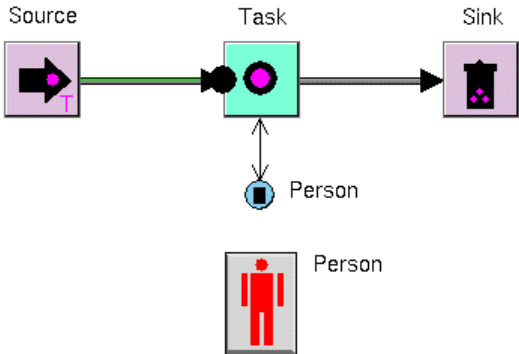
Each time the block starts processing a work object and creates a new activity, it increments Total Starts and Current Activities by one. Each time the block stops processing a work object and deletes the associated activity, it increments Total Stops by one and decrements Current Activities by one.

The Current Activities metric represents the number of activities associated with the block, which you can visualize while the model is running.

To see the metrics of a block related to activities:

➔ Display the properties dialog for the block and click the General tab.

The General tab of the properties dialog might look like this for a Task block with two current activities, given a pool of worker resources:



The screenshot shows a dialog box titled 'Task: Task' with a close button (X) in the top right corner. It has five tabs: 'General', 'Block', 'Duration', 'Cost', and 'Animation'. The 'General' tab is selected. The dialog contains the following fields and values:

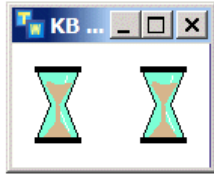
- Block Label: Task
- Comments: (empty text area)
- Maximum Activities: (empty text field)
- Url: (empty text field)
- Total Starts: 7
- Total Stops: 5
- Current Activities: 2
- Error: (empty text area)

At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Update', and 'Cancel'.

To visualize the current activities of a block:

→ Choose Snapshot Activities on a block.

ReThink displays a workspace that contains the current activities for the block:



This workspace can contain zero or more activities. As the block finishes processing an activity, ReThink dynamically deletes the activity from the workspace. However, ReThink does not dynamically add new activities to the workspace; you must take another snapshot to see the current activities of the block after the block processes another event.

Understanding the Attributes of Activities

Each activity has associated metrics, which compute the duration and cost associated with the individual activity. This table summarizes the metrics of an activity:

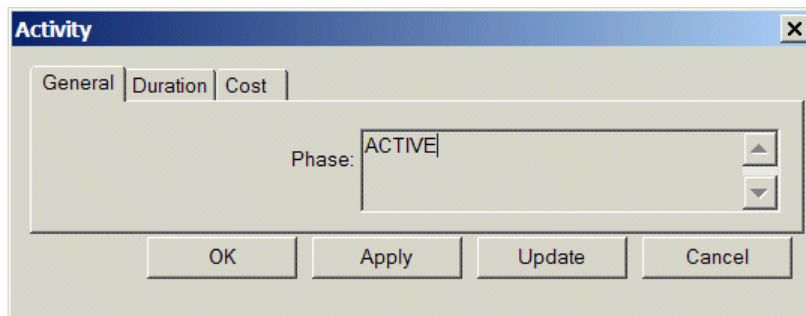
Attribute	Description
Phase	The status of the activity, which is active when the activity exists and inactive when the activity is deleted.
Work Time	The amount of simulation time that a particular activity is active processing work.
Elapsed Time	The amount of simulation time that represents the entire duration of the activity.
Cost	The cost associated with processing the activity, which includes costs assigned to the block and costs assigned to any associated resources.

By default, the block is busy processing work for the entire duration of each activity. Thus, the Work Time and the Elapsed Time of an activity are the same. However, you can customize the time delay of an activity.

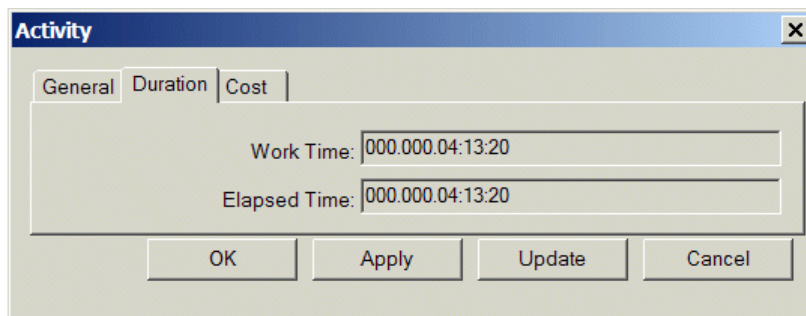
To display the metrics of an activity:

- 1 Run the simulation in step mode.
- 2 While a block is processing a work object, choose Snapshot Activities on the block.
- 3 Display the properties of an activity.

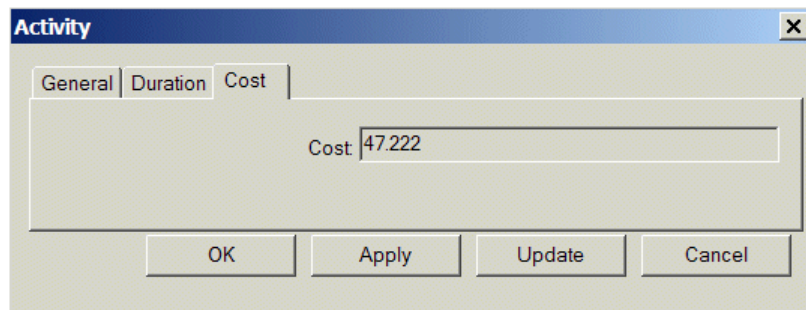
ReThink displays the General tab of the properties dialog for an activity that is active:



- 4 To see the metrics of the activity related to duration, click the Duration tab:



- 5 To see the metrics of the activity related to cost, click the Cost tab:



Customizing the Time Delay of Activities

You can customize the way ReThink calculates the time delay of activities. You do this to create variation in the amount of time that work is actually being processed (the Work Time) versus the amount of time that work is in the system (the Elapsed Time). For example, you might customize the time calculation to take into account the fact that work is only processed during normal business hours.

To illustrate, suppose an order takes three hours to process and it enters the system at 4:00 PM on a Monday. The total work time for the order would be three hours (assuming no deviation), however, the total elapsed time would include the hours from 5:00 PM on Monday to 9:00 AM on Tuesday, when the order was in the system but not being processed.

For information on customizing the duration of activities, see [Constraining the Availability of Resources](#) and see the *Customizing ReThink User's Guide*.

Using Resources to Constrain Concurrent Activities

If the block has no resource constraints, the block can process any number of activities concurrently. For example, if the block receives objects once an hour on average, and the block takes two hours to process on average, the block can process two concurrent activities.

If the block has associated resources, it can only process one activity for each available resource, assuming default resource utilization. Thus, the number of current activities is constrained by the number of available resources.

If a block has resource constraints, and if more work objects flow into the block than there are available resources, work backs up on the input path to the block.

For more information on...	See...
Assigning resources to a task	Using Resources to Constrain the Model.
Analyzing the work backups due to resource constraints	<ul style="list-style-type: none">• Showing Work Backups on an Input Path• Analyzing the Wait Time Due to Work Backups

Limiting the Number of Concurrent Activities

You can limit the number of concurrent activities a block can process, regardless of the available resources.

To limit the number of concurrent activities:

- 1 Display the properties dialog for a block and click the General tab.
- 2 Configure the Maximum Activities to be the number of concurrent activities.

If a block has specified a maximum number of activities, and if more work objects flow into the block than the maximum, work backs up on the input path to the block. In the following properties dialog, notice that the Current Activities plus the Total Stops equals the Total Starts, regardless of the specification for Maximum Activities; the block does not actually start processing the work objects that are waiting on the input path.

The screenshot shows the 'Task' properties dialog box with the 'General' tab selected. The fields are as follows:

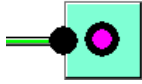
- Block Label: [Empty text box]
- Comments: [Empty text box]
- Maximum Activities: 2
- Url: [Empty text box]
- Total Starts: 4
- Total Stops: 2
- Current Activities: 2
- Error: [Empty text box]

A callout box on the right side of the dialog points to the Total Starts, Total Stops, and Current Activities fields, containing the following text:

$$\text{Total Stops} + \text{Current Activities} = \text{Total Starts}$$

You can analyze the work backups that result from limiting the maximum activities, as described in [Showing Work Backups on an Input Path](#) and [Analyzing the Wait Time Due to Work Backups](#).

Showing Work Backups on an Input Path



When you run a model, work objects can back up on the input path of a block creating a **work backup**. Work backups occur for one of two reasons:

- The block does not have enough resources to process all the work objects that are waiting on the input path.

See [Using Resources to Constrain Concurrent Activities](#).

- The block has specified a maximum number of activities that it can process at one time.

See [Limiting the Number of Concurrent Activities](#).

When work backups exist, ReThink colors the input path green and inserts the work objects into the **path queue**. You can visualize work backups by taking a snapshot of the path queue.

You can customize the color the path uses to indicate work backups, as described in [Configuring the Animation of Paths](#).

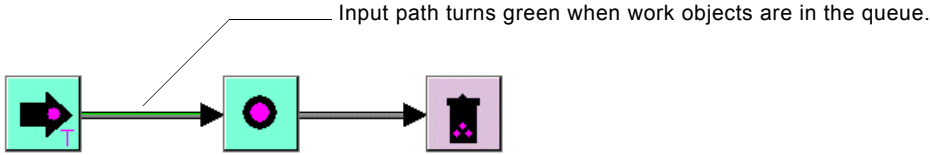
Note Input paths can turn green for one of two reasons: work backups due to constraints on the model or path synchronization. For example, a Task block with two input paths synchronizes its inputs by waiting to process until all the inputs have arrived at the block. This input path turns green for a different reason than work backing up on the input path.

To show work backups in the path queue:

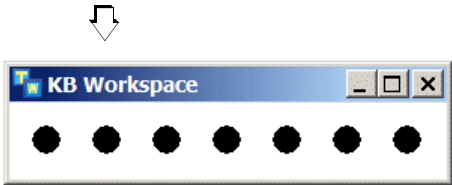
- ➔ Choose Snapshot Queue on a green input path that has work backups.

ReThink displays a temporary workspace that shows the work objects waiting in the path queue. ReThink transfers the work objects from the path to the workspace.

For example, here is a model where the task limits the number of current activities to two. When more than two work objects arrive at the block, the input path to the task turns green and the work objects appear in the queue.



Snapshot Queue



The number of work objects that the task can process is limited to 2. When more than 2 work objects arrive at the block, they back up on the path queue.

For an example of work backups due to resource constraints, see [Showing Work Backups Due to Resource Constraints](#).

Analyzing the Wait Time Due to Work Backups

ReThink computes summary metrics that report the total amount of time that work objects have spent waiting in the path queue. This table describes the metrics that relate to the path queue:

Attribute	Description
Total Insertions	The total number of work objects that have ever been in the path queue.
Current Waiting	The current number of work objects in the queue.

Attribute	Description
Total Wait Time	The total amount of time that all work objects have spent in the path queue since the start of the simulation.
Mean Wait Time	The total amount of time that all work objects have spent in the path queue (Total Wait Time) divided by the total number of work objects that have ever been in the path queue (Total Insertions).

You can create a summary report that includes metrics for all paths in the model. For more information, see [Summary of Input and Output Reports](#).

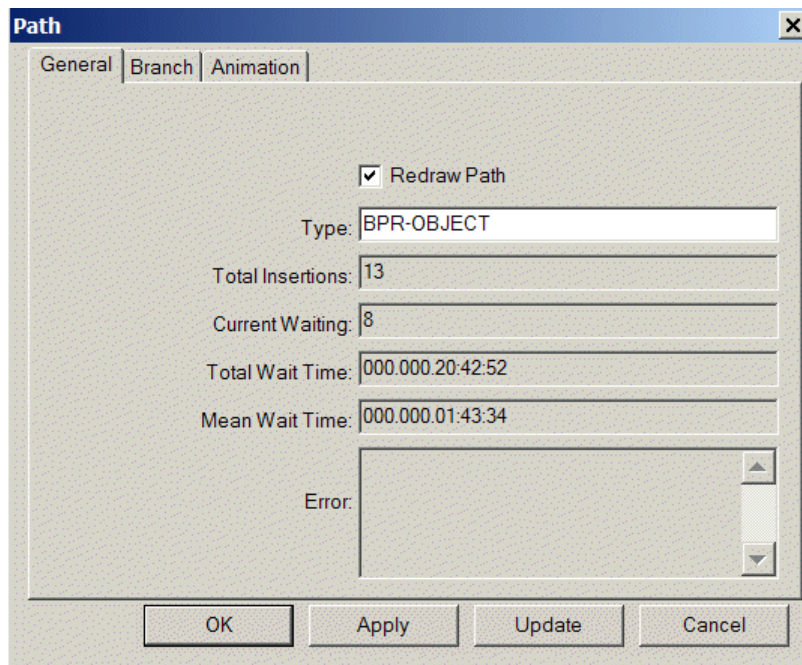
Another common technique for obtaining path metrics is to create a readout table. For details, see [Using Readout Tables](#).

To analyze the wait time due to work backups:

- ➔ Display the properties dialog for a green input path that has work backups and click the General tab.

The properties dialog for a path contains various metrics that compute the wait time of work objects due to work backups. The dialog also shows the total number of work objects in the queue.

Here is the properties dialog for a path with eight work objects in the path queue:



Showing Work Backups Interactively

To visualize where work backups occur most often in the model, you can interactively show the busiest path, the most used path, and the path with the longest wait time. To do this, you run, then pause the simulation.

To show work backups in the model interactively:

→ To show the path with the largest value for Current Waiting, choose Indicate Busiest Path from the popup menu for the workspace.

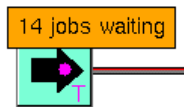
or

→ To show the path with the largest value for Total Insertions, choose Indicate Most Used Path from the popup menu for the workspace.

or

→ To show the path with the largest value for Total Wait Time, choose Indicate Longest Wait Time Path.

ReThink indicates the path and provides information. For example, here is the result of choosing Indicate Busiest Path on a path with 14 work objects waiting:



Working with the Duration of Blocks

Each block has an associated **duration**, which is the amount of time that the block spends processing its input work objects. Each block also computes summary duration metrics, which allow you to analyze the block's performance.

With the exception of the Source block, all ReThink blocks use a random normal distribution to compute duration, by default. A random normal distribution is a bell-shaped curve with a mean and a standard deviation. The Source block uses a random exponential distribution, by default, which uses just the mean. In addition, you can choose from among a number of additional distributions to compute duration, such as uniform and triangular, or you can specify a fixed duration.

By default, ReThink computes duration for a single work object. If your work object represents more than one object, you can configure an attribute of the work object that represents the number of units, which the block multiplies by the sampled duration to produce the overall duration.

You configure the duration of a block on the Duration tab of the properties dialog. You specify the duration by using an **interval**, which is an expression such as 10 seconds, 30 minutes, 3 hours, 5 days, or 4 weeks.

You view metrics related to block duration in the properties dialog for a block or in the Metrics toolbar. You can also create a Block Summary Report, which displays duration metrics for all blocks in your model.

Note The duration of an individual block cannot exceed 17 years; however, the Total Elapsed Time of the simulation can be any length of time.

Specifying a Fixed Duration

You might want to specify a fixed duration for an activity as a way of verifying the model. With no variation, you can more easily cross-check metrics that the model computes.

To specify a fixed duration:

- 1 Display the properties dialog for the block and click the Duration tab.
Use the default Distribution Mode, which is Fixed Distribution.
- 2 Configure the Mean to be a time interval.

Here is the Duration tab of the properties dialog with a fixed duration:

The screenshot shows a dialog box titled "Task: Task" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Duration" tab is selected. Inside the "Duration" section, there is a "Distribution Mode" dropdown menu set to "Fixed Distribution". Below it, the "Mean" is displayed as "000 000 02:00:00". The "Miscellaneous" section contains five input fields: "Time Per Unit Attribute", "Total Work Time", "Total Elapsed Time", "Creation Time", and "Average In Process", all of which are currently empty or set to "000.000.00:00:00". At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Specifying a Random Duration

Most models simulate real-world durations, which are random, based on some standard distribution. Depending on the activity, you might specify a random normal, random exponential, random triangular, or some other random distribution.

To specify a random duration:

- 1 Display the properties dialog for the block and click the Duration tab.
- 2 Choose a Distribution Mode from the dropdown list.
- 3 Configure the parameters associated with the Distribution Mode you specify.

For details on configuring each of the distribution modes, see the descriptions of each distribution below.

Here is the Duration tab for a Source block, which specifies a random exponential distribution, by default:

Source: Source

General | Block | Database | Duration | Cost | Animation

Duration

Distribution Mode: Random Exponential

Mean: 000 000 01:00:00

Miscellaneous

Time Per Unit Attribute: 000.000.00:00:00

Total Work Time: 000.000.00:00:00

Total Elapsed Time: 000.000.00:00:00

Creation Time: 000.000.00:00:00

Average In Process: 000.000.00:00:00

OK Apply Update Cancel

Here is the Duration tab for a Task block that specifies a random triangular distribution:

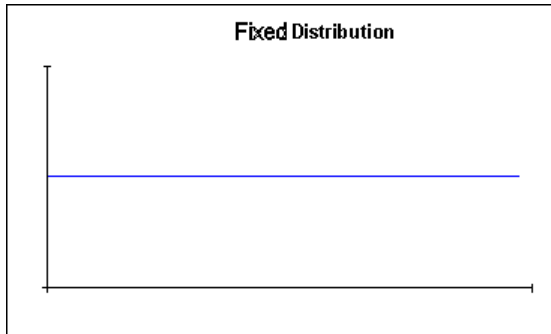
The screenshot shows the 'Task: Task' dialog box with the 'Duration' tab selected. The 'Distribution Mode' is set to 'Random Triangular'. The 'Min' value is 000, the 'Max' value is 000, and the 'Mode' value is 000. The 'Miscellaneous' section contains fields for 'Time Per Unit Attribute', 'Total Work Time', 'Total Elapsed Time', 'Creation Time', and 'Average In Process', all set to 000.000.00:00:00. Buttons for 'OK', 'Apply', 'Update', and 'Cancel' are at the bottom.

You can choose from the following random distributions for computing the duration of a block:

- [Fixed Distribution](#)
- [Random Exponential](#)
- [Random Normal](#)
- [Random Uniform](#)
- [Random Triangular](#)
- [Random Erlang](#)
- [Random Weibull](#)
- [Random Lognormal](#)

- [Random Gamma](#)
- [Random Beta](#)

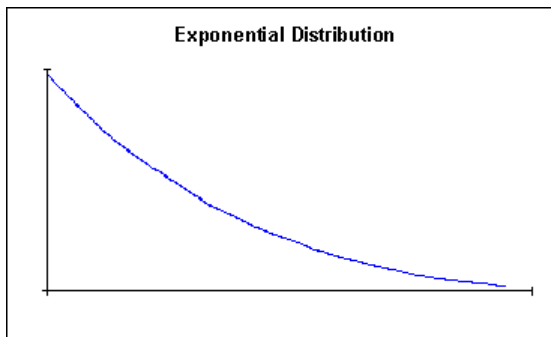
Fixed Distribution



Parameters: Mean

Specifies a fixed distribution given a mean, which results in the same value being used for each sample.

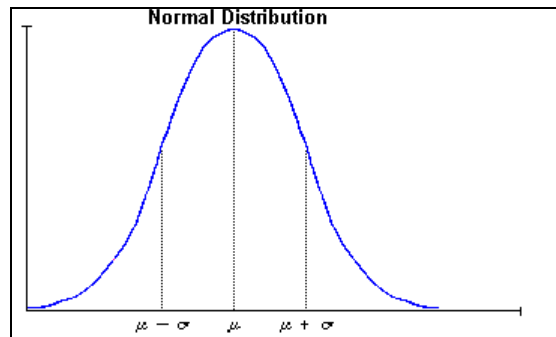
Random Exponential



Parameter: Mean

Specifies a random sample with a built-in deviation, where the likelihood is greatest that the value will be less than the Mean or somewhat greater than the Mean; however, some small percentage of the time, the value is significantly greater than the Mean. A random exponential function most closely models the frequency with which a process receives inputs in the real world.

Random Normal

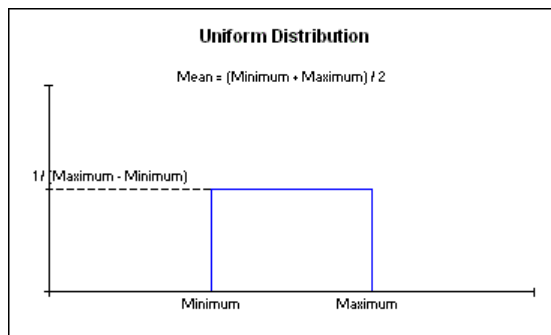


Parameters: Mean, Standard Deviation

Specifies that the value varies around the Mean, based on the Standard Deviation. For example, if the Mean is two hours and the Standard Deviation is one hour, then, on average, the value is two hours. The value varies from some amount less than two hours to some amount greater than two hours, based on a normal distribution, where 95% of the sample points fall within two standard deviations of the Mean.

You can configure the Standard Deviation to be zero, in which case the model uses the same value each time.

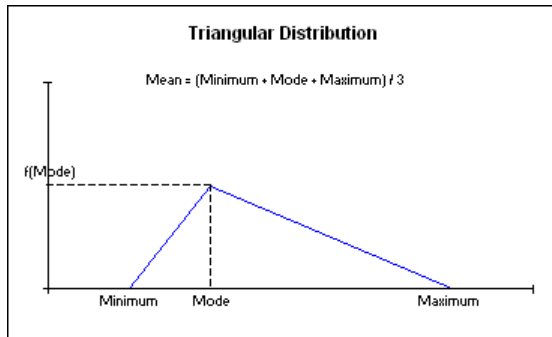
Random Uniform



Parameters: Min, Max

Specifies that every value between the Min and the Max is equally likely. The use of this distribution often implies a complete lack of knowledge about the shape of the data, other than the minimum and maximum values.

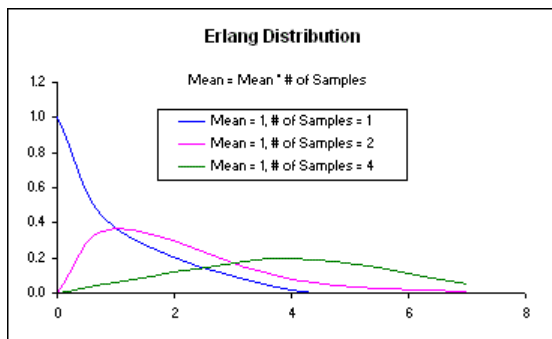
Random Triangular



Parameters: Min, Mode, Max

Appropriate when a most likely value, called the Mode, is known, and a linear distribution between the Min and the Mode and between the Mode and the Max can be assumed. Random-Triangular is the default Mode Type.

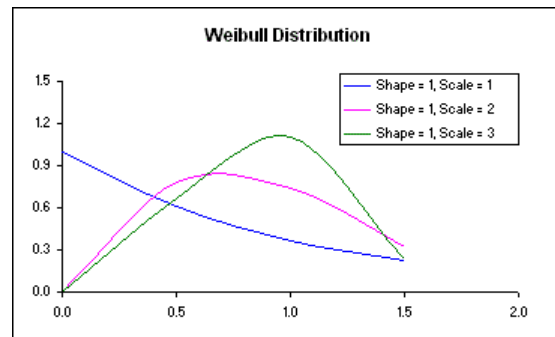
Random Erlang



Parameter: Mean

The sum of independent and identically distributed exponential samples with the specified Mean. This distribution is a special case of the Random-Gamma distribution where the Beta parameter is an integer that represents the number of samples.

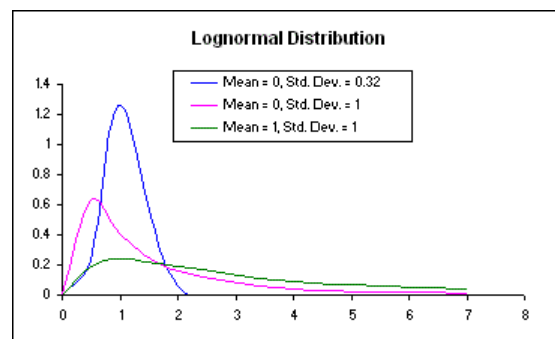
Random Weibull



Parameters: Shape, Scale

Often used for modeling the time to failure, called the reliability, of independent, identical components. When the Shape equals 1, the Random-Weibull distribution reduces to the Exponential distribution.

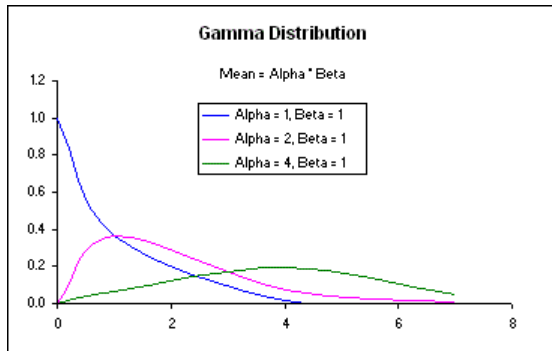
Random Lognormal



Parameters: Mean, Standard Deviation

The distribution of data whose natural logarithm follows the normal distribution, given a Mean and Standard Deviation. This distribution is appropriate for situations where the value of a data point is a random proportion of the previous data point, for example, the distribution of personal incomes.

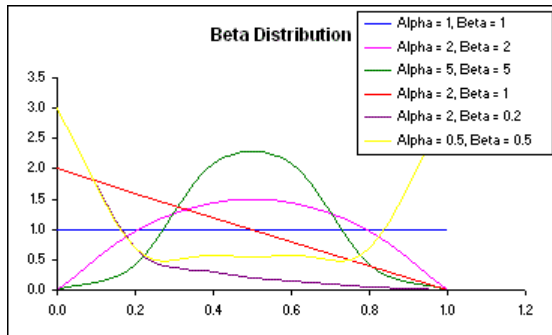
Random Gamma



Parameters: Alpha, Beta

A generalization of the Erlang distribution where conceptually the number of exponential samples need not be an integer value. With different parameter settings, the Random-Gamma distribution can take on many different shapes and can, therefore, represent a wide variety of different physical processes.

Random Beta



Parameters: Alpha, Beta, Min, Max

Takes on a wide variety of different shapes for different values of the Alpha and Beta parameters, including bell-shaped, U-shaped, symmetric, or asymmetric. The Random-Beta distribution is defined over a finite range (0, 1) that is then scaled, using the Min and Max parameter values.

Specifying Duration from a File

You might know exactly when an activity occurs in your model, based on durations in a file. The file should be a list of numbers, separated by a carriage return, representing the difference in arrival times between each subsequent object that the block processes, interpreted as seconds. For example:

```
720
650
640
620
840
```

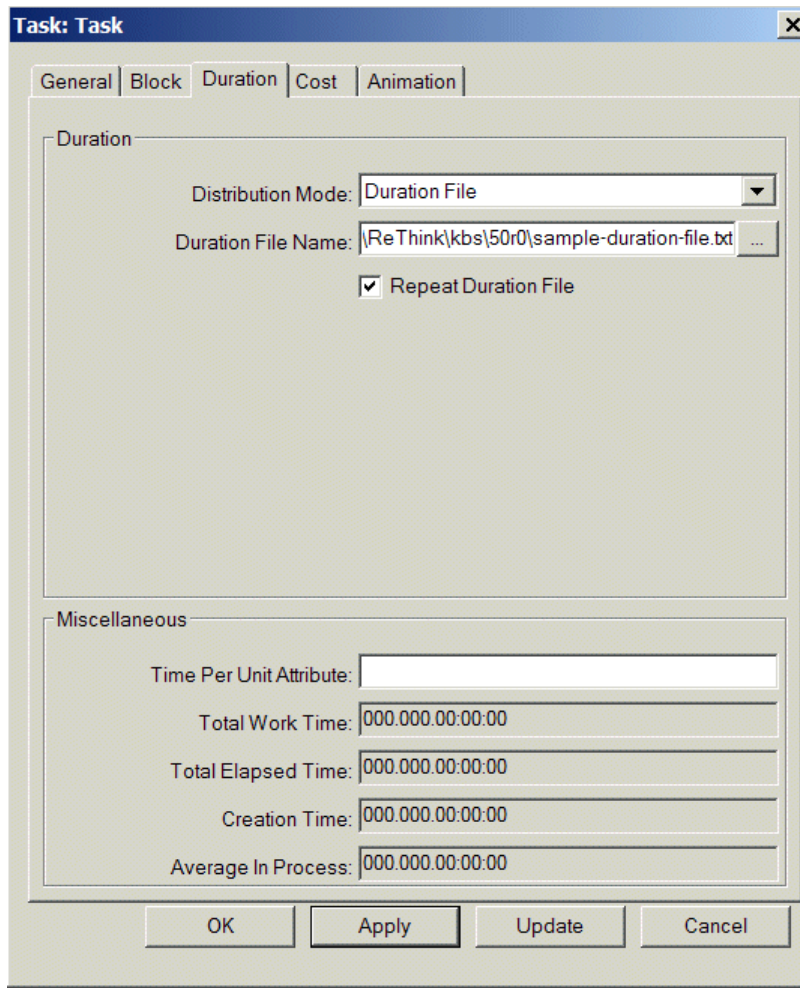
By default, the block continues processing work objects when it reaches the end of the file by looping back to the beginning of the file.

To specify duration from a file:

- 1 Display the properties dialog for the block and click the Duration tab.
- 2 Choose Duration File as the value of the Distribution Mode.
- 3 Configure the Duration File Name to specify a text file that contains the durations.
- 4 To cause the block to stop processing work objects when it reaches the end of the file, disable the Repeat Duration File option.

Note If you disable the Repeat Duration File option, be sure that the upstream block does not continue to send work objects to the block that obtains its durations from a file; otherwise, work objects will accumulate on the input path of the block without being processed.

For example, here is the Duration tab for a Task block that determines its duration from a file:



Specifying Duration Based on an Indexed Report Lookup

You might know exactly when an activity occurs in your model, based on durations in a report. You can create an Indexed Lookup Report with a column that is a list of numbers representing the difference in arrival times between each subsequent object that the block processes, interpreted in the time unit of the

report. For example, the following values would be interpreted as hours, the default time unit:

2
3
4.5
3.5
2
5

To do this, you create a work object definition that defines an attribute that specifies the duration. You then use an Indexed Lookup Report to specify the durations. By default, the block stops processing work objects when it reaches the end of the report.

To specify duration, based on an indexed report lookup:

- 1 Create a work object with an attribute whose value you want to look up in a report, for example, *order-processing-time*.
- 2 Configure the model to create work objects of this type.
- 3 Display the Reports palette of the ReThink toolbox:



- 4 Create an Indexed Lookup Report with a column that is the name of the attribute of the work object that defines the durations you want to look up. In this example, the column would be *order-processing-time*.

For details, see:

- [Creating Reports.](#)
- [Configuring the Attributes to Appear in a Report.](#)

- 5 Create as many rows as needed in the report and enter values for each duration you want to look up.

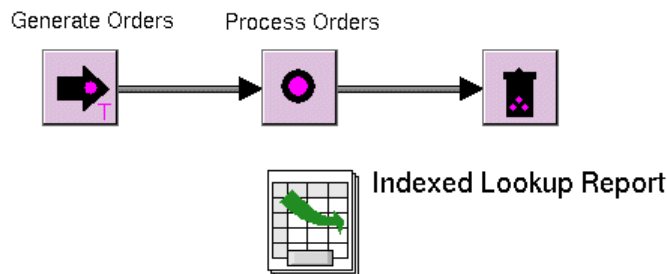
For details, see [Applying Input Report Data to the Model.](#)

- 6 Display the properties dialog for the block whose durations you want to look up in the report and click the Duration tab.
- 7 Choose Report Indexed Lookup as the value of the Distribution Mode.
- 8 Configure the Report Title to be the name of the Indexed Lookup Report you created above.
- 9 Configure the Lookup Attribute Name to be the name of the column to use as an index when looking up values in the report.

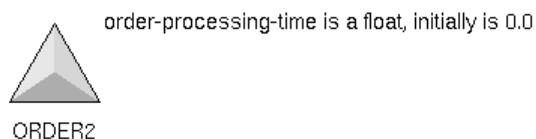
In the example above, the Lookup Attribute Name would be order-processing-time.

- 10 To cause the block to continue processing work objects when it reaches the end of the report, enable the Repeat Indexed Lookup option.

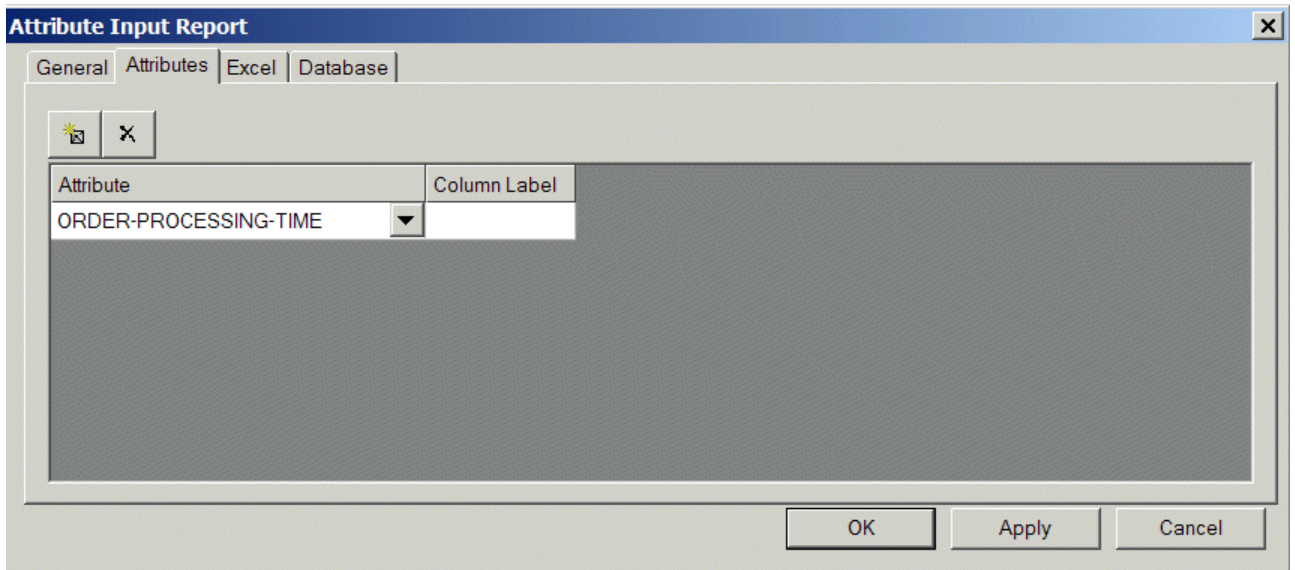
Here is a model that looks up durations in the report named Indexed Lookup Report:



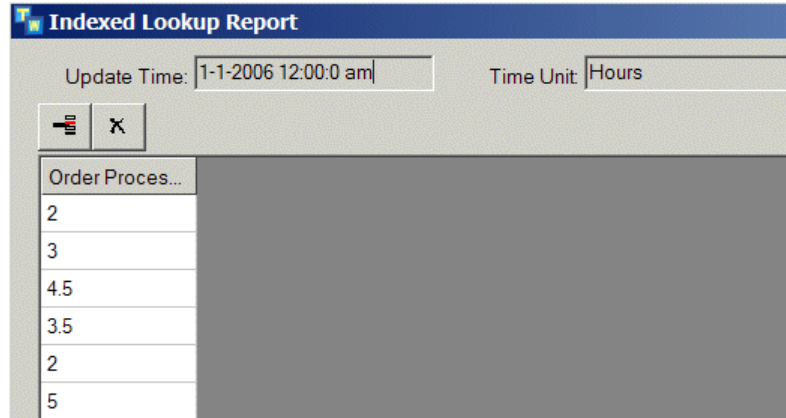
Here is the class definition for the work object that defines the order-processing-time attribute, whose values the model will look up:



Here is the Attributes tab of the Indexed Lookup Report, which specifies **order-processing-time** as the attribute of the work object whose values to look up:



Here is the contents of the Indexed Lookup Report with values for **order-processing-time**, which are given in hours, the default time unit of the report:



Here is the Duration tab for a Task block that determines its duration by looking up values for the order-processing-time attribute in the report named Indexed Lookup Report:

Task [X]

General | Block | **Duration** | Cost | Animation

Duration

Distribution Mode: Report Indexed Lookup

Report Title: Indexed Lookup Report

Lookup Attribute Name: ORDER-PROCESSING-TIME

Repeat Indexed Lookup

Miscellaneous

Time Per Unit Attribute: []

Total Work Time: 000.000.00:00:00

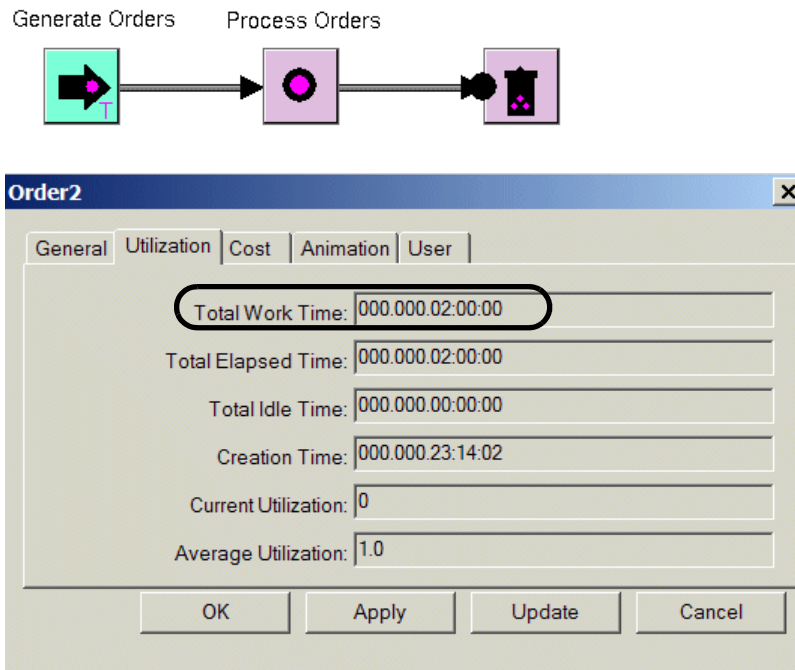
Total Elapsed Time: 000.000.00:00:00

Creation Time: 000.000.00:00:00

Average In Process: 000.000.00:00:00

OK Apply Update Cancel

When you run the simulation, the first work object uses a duration of 2 hours:



Specifying Duration Based on an Attribute of a Work Object

You might define a class of work objects that specify their own durations. For example, you might define the Packing Time for different types of boxes, or you might feed the packing time into the model, using a feed. When different types of objects arrive at a block that packs boxes, it determines the duration of the activity, based on the value of the Packing Time.

To specify durations, based on an attribute of a work object, you must first define the class or classes of objects that define the duration.

To specify duration, based on an attribute of a work object:

- 1 Create one or more class definitions for objects that the model will use to determine duration.

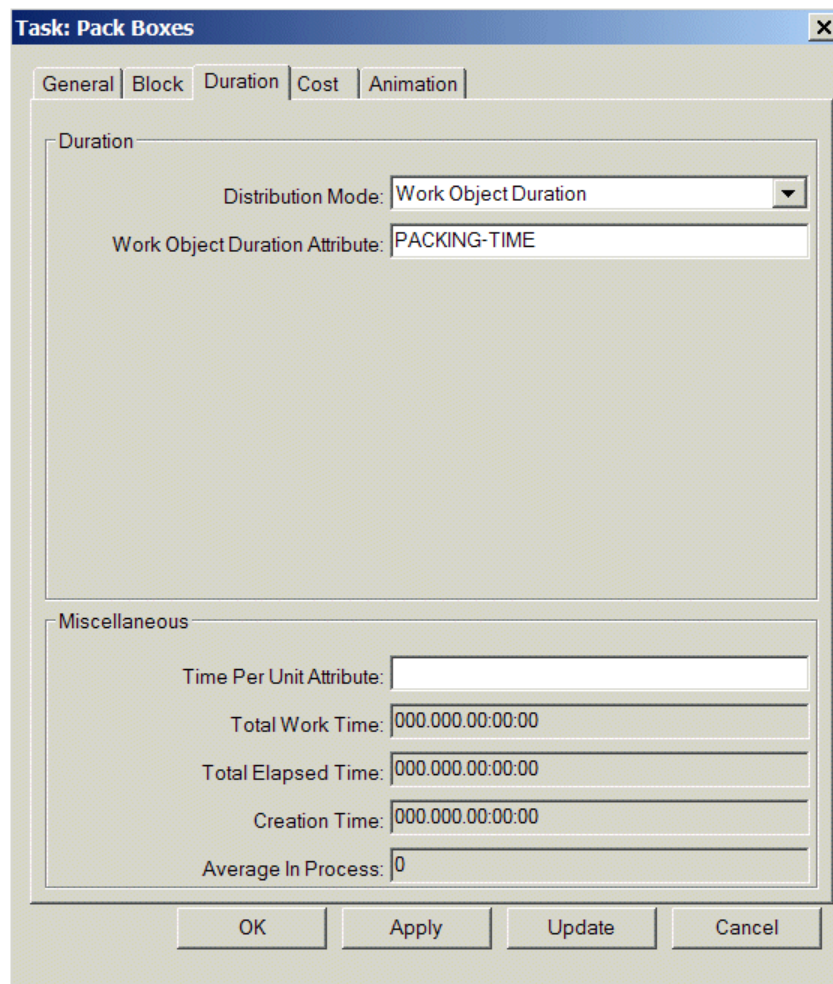
For example, you might define two classes named **small-box** and **big-box**, each with a **packing-time** attribute, where the initial values depend on the type of box.

- 2 Configure the model to generate the user-defined objects such that they arrive at the input path of the block whose duration should be based on an attribute of a work object.

For example, you might create one Source block that generates small boxes and another Source block that generates large boxes.

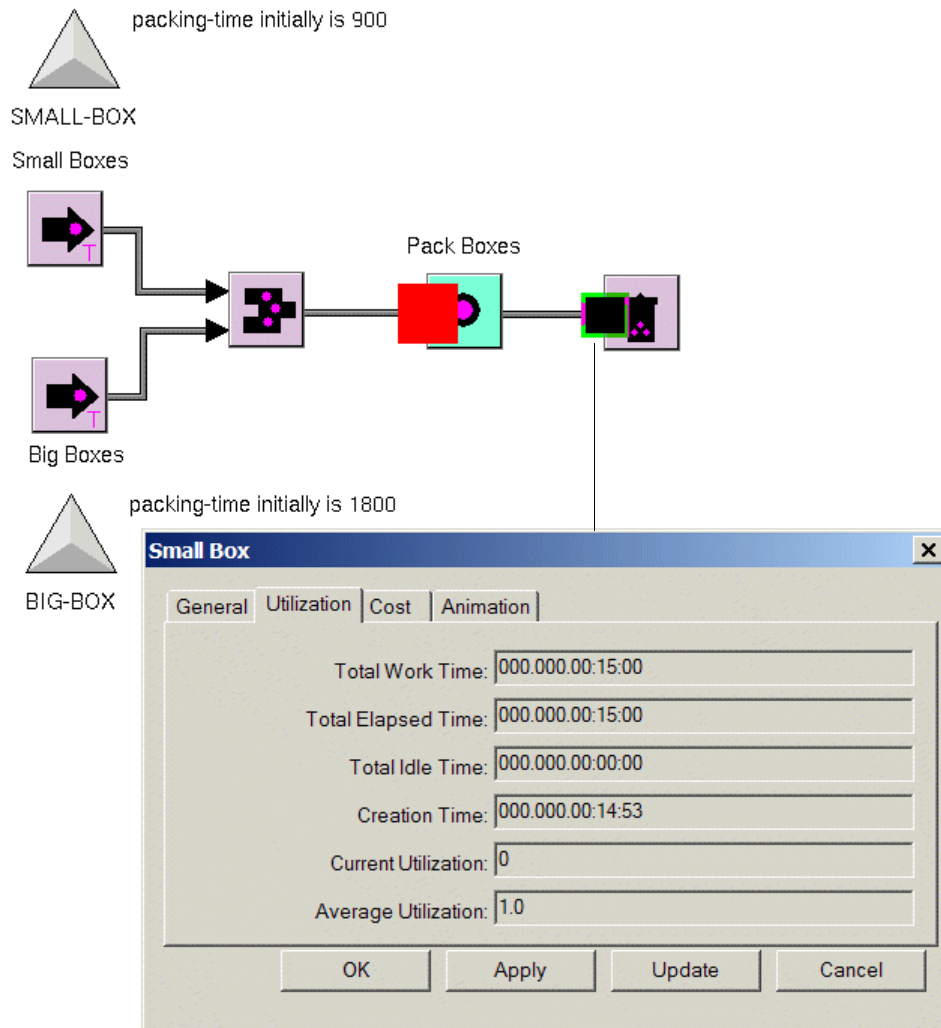
- 3 Display the properties dialog for the block and click the Duration tab.
- 4 Choose Work Object Duration as the value of the Distribution Mode.
- 5 Configure the Work Object Duration Attribute to be the attribute value of the user-defined work object that determines the duration of the block.

Here is the Duration tab for the Pack Boxes task in the following model, which determines duration by using the packing-time attribute of the work objects it processes:



This diagram shows the result of running a model where the Pack Boxes task uses the packing time of the small-box or large-box to determine the duration of the activity. The class definitions show the class-specific attributes, which specify the packing-time attribute with different initial values, as well as the associated icons. In the running model, the Pack Boxes task is currently processing a big-box and

just finished processing a small-box. The properties dialog for the small-box on the output path of the Pack Boxes task show a Total Work Time of 15 minutes or 900 seconds, which is the initial value for the **packing-time** attribute of the **small-box** class. When the Pack Boxes task finishes processing the big-box, the Total Work Time for the big-box will be 30 minutes or 1800 seconds, the initial value for the **packing-time** of the **big-box** class.



Specifying Duration Based on an Attribute Report Lookup

You might have a block that processes several different types of objects, where an attribute of the work object is an index that the block uses for looking up the duration in a report. For example, you might define an `order` object with a `carrier` attribute, which you feed into the model. The value of the `carrier` is the index that the block uses to look up its duration in a report, where `mail` would take 3 days, `ups` would take 2 days, and `fedex` would take 1 day.

You can use multiple attributes as indexes for the report lookup, for example, the lookup might be based on a combination of the `carrier` attribute and the `priority` attribute. You can also perform arithmetic tests that compare the attribute value of the work object with values in the report. For example, you might define a `distance` attribute on the `order`, which you feed into the model as a random value. To determine which value for `distance` to use as an index in the report, the block would compare the actual distance of the work object with a distance value in the report, where if the distance is less than 1000 miles, it uses one value and if the distance is 1000 miles or greater, it uses a different value.

To specify durations, based on an attribute report lookup, you must first define the class or classes of objects that define the attribute to use as the index for the report lookup. You must then define an Attribute Lookup Report with the appropriate columns to use for the lookup. You can use the same report for multiple blocks. Finally, you must configure the duration of the block specify the report name, the column to use as the duration, the column to use as the label, and the search criteria.

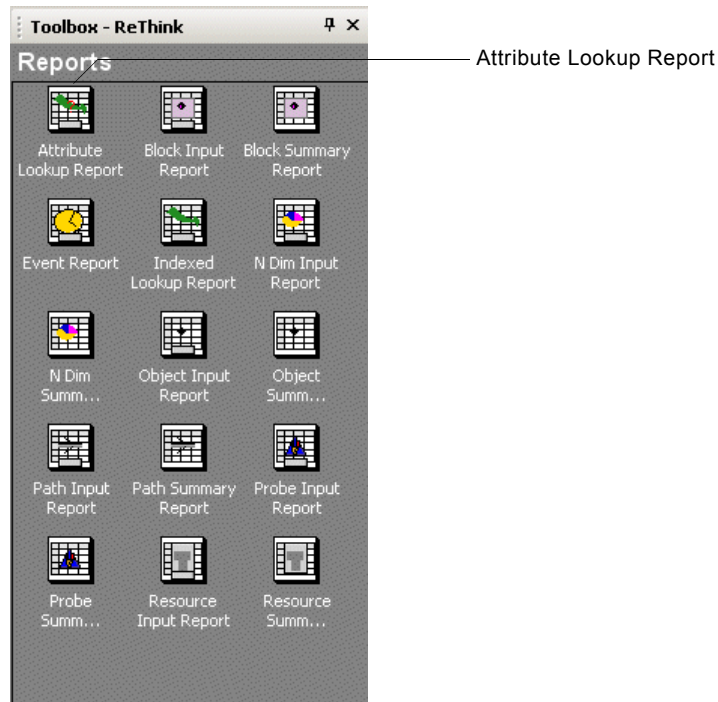
To specify duration, based on a report lookup:

- 1 Create one or more class definitions for objects that the model will use to determine the attribute or attributes to use as an index for a report lookup.

For example, you might define a class named `order` with a `carrier` attribute and a `distance` attribute, whose values you will feed into the model.
- 2 Configure the model to generate the user-defined objects such that they arrive at the input path of the block whose duration should be based on an attribute report lookup.

For example, you might create a Source block that generates orders, which feeds the value of the `carrier` attribute and the `duration` into each order, then passes the orders to a downstream Task block.

- 3 Display the Reports palette of the ReThink toolbox:



- 4 Create an Attribute Lookup Report and configure the report to have the following columns:

Column	Description	Example
Lookup Attribute Name	The report column that the block uses for the duration.	transportation-time
Lookup Label Attribute Name	The report column that uniquely identifies the block.	block-label
Search Criteria Report Column(s)	The report column or columns that the block uses as the index for comparing with the Work Object Attribute value.	carrier-value distance-min-value distance-max-value

Note When configuring the report columns in the report, you must use symbols; do not use spaces.

- 5 Create as many rows as needed and provide values for each column of the report.

For example, you might enter these values for different distances for three types of carriers:

Block Label	Carrier Value	Distance Min Value	Distance Max Value	Transportation Time
Ship Orders	mail	1000	10000	120
Ship Orders	mail	0	1000	72
Ship Orders	ups	1000	10000	72
Ship Orders	ups	0	1000	48
Ship Orders	fedex	1000	10000	48
Ship Orders	fedex	0	1000	24

Note You must enter the value for duration in the time unit of the report, whose default is 1 hour.

- 6 Display the properties dialog for the block and click the Duration tab.
- 7 Choose Report Lookup as the value of the Distribution Mode.
- 8 Choose the Report Title from the dropdown list to be the report you created above.

Note You must create the report before you configure the Report Title; otherwise, it will not appear in the list.

- 9 Configure the Lookup Attribute Name to be the report column that the block uses as its duration.

In the example above, the Lookup Attribute Name would be transportation-time.

- 10 Configure the Lookup Label Attribute Name to be the report column that uniquely identifies the block.

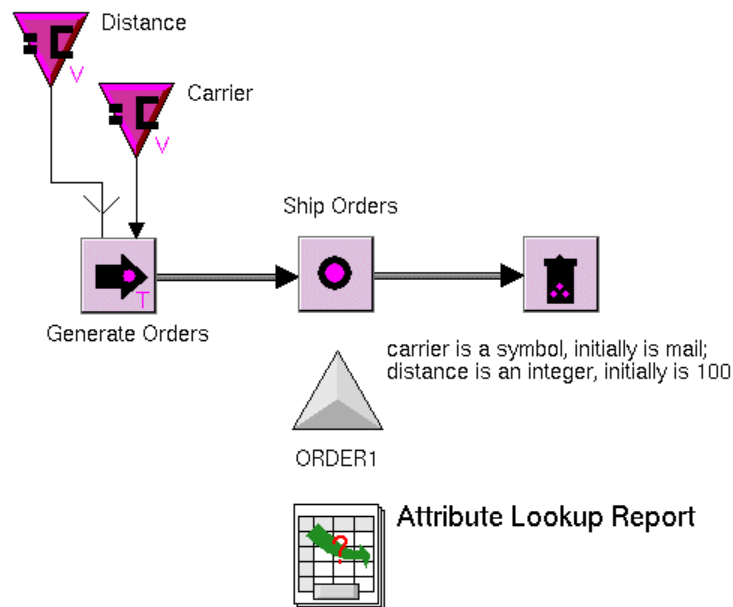
In the example, the Lookup Label Attribute Name would be block-label.

- 11 Configure the Search Criteria by editing the Work Object Attribute, Operator, and Report Column to specify the index to use for the report lookup, adding search criteria, as needed.

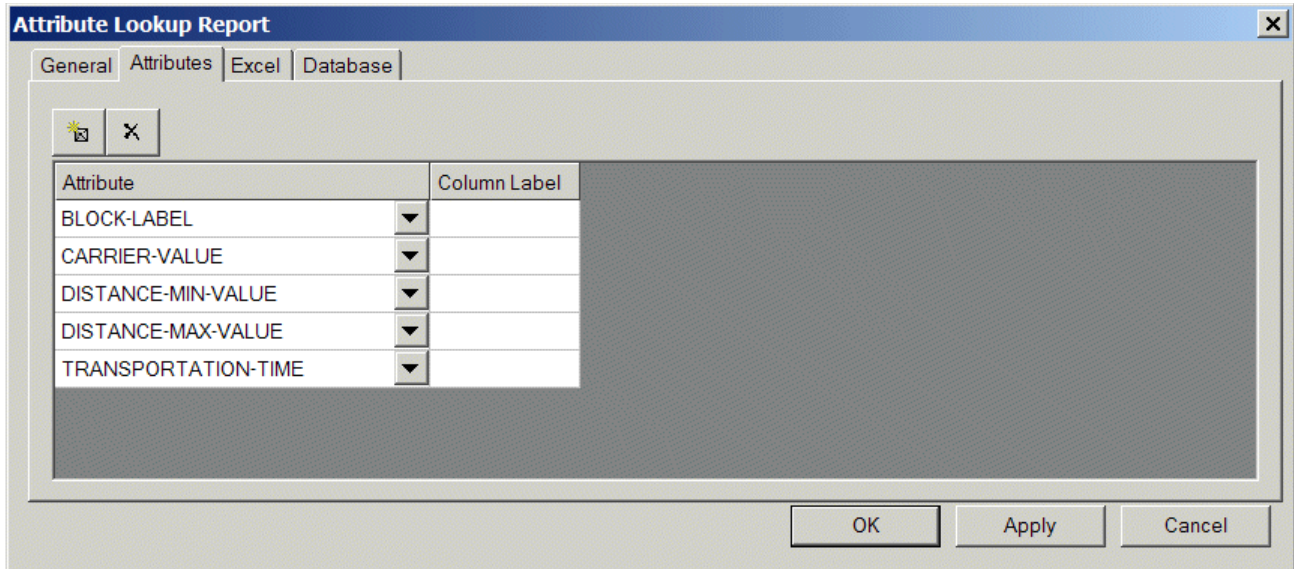
In the example, you would create these three criteria for determine the index to use for the report lookup:

Work Object Attribute	Operator	Report Column
CARRIER	=	CARRIER-VALUE
DISTANCE	>=	DISTANCE-MIN-VALUE
DISTANCE	<	DISTANCE-MAX-VALUE

This model generates orders and feeds values for the carrier and distance into each order, using Change feeds. The Ship Order task looks up values in the Attribute Lookup Report, based on the specified criteria. The `order` class defines the `carrier` and `distance` attributes.



Here is the Attributes tab of the Attribute Lookup Report:



Here is the Attribute Lookup Report for the model, which specifies values for each attribute in the report:

Block Label	Carrier Value	Distance Min ...	Distance Max ...	Transportatio...
Ship Orders	mail	1000	10000	120
Ship Orders	mail	0	1000	72
Ship Orders	ups	1000	10000	72
Ship Orders	ups	0	1000	48
Ship Orders	fedex	1000	10000	48
Ship Orders	fedex	0	1000	24

Here is the Duration tab for the Ship Order task in the following model, which determines duration by looking up the transportation-time in the Attribute Lookup Report, based on the carrier and distance attributes of an order:

Task [X]

General | Block | **Duration** | Cost | Animation

Duration



Distribution Mode: Report Lookup

Report Title: Attribute Lookup Report

Lookup Attribute Name: TRANSPORTATION-TIME

Lookup Label Attribute Name: BLOCK-LABEL

Search Criteria:

Work Object	Oper...	Report Column	
carrier	=	carrier-value	
distance	<=	distance-max-val	

Miscellaneous

Time Per Unit Attribute:

Total Work Time: 000.000.00:00:00

Total Elapsed Time: 000.000.00:00:00

Creation Time: 000.000.00:00:00

Average In Process: 000.000.00:00:00

OK Apply Update Cancel

Here is the User tab for a work object on the output path of the Ship Order block:

The screenshot shows a dialog box titled "Order1" with a close button (X) in the top right corner. It has five tabs: "General", "Utilization", "Cost", "Animation", and "User". The "User" tab is selected. Inside the dialog, there are two input fields: "Carrier:" with the value "UPS" and "Distance:" with the value "1950". At the bottom, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Here is the Utilization tab for the work object, which shows that the Total Work Time is 3 days (72 hours), which the Ship Order task looked up in the report, based on the Carrier (UPS) and Distance (1000 < 1950 < 10000):

The screenshot shows the same "Order1" dialog box, but with the "Utilization" tab selected. It displays several time and utilization metrics in input fields: "Total Work Time: 000.003.00:00:00", "Total Elapsed Time: 000.003.00:00:00", "Total Idle Time: 000.000.00:00:00", "Creation Time: 000.000.00:00:00", "Current Utilization: 0", and "Average Utilization: 1.0". The same four buttons ("OK", "Apply", "Update", "Cancel") are at the bottom.

Using a Graph to Specify Duration



You might have a general picture of the frequency with which work objects arrive into your process. For example, you might know that in the morning there are a moderate number of calls per hour, a peak before lunch, a lull at midday, a large number of calls in the afternoon, and a moderate number of calls approaching the end of the day. You can represent this visually in ReThink by using an Arrival Rate Input Graph.

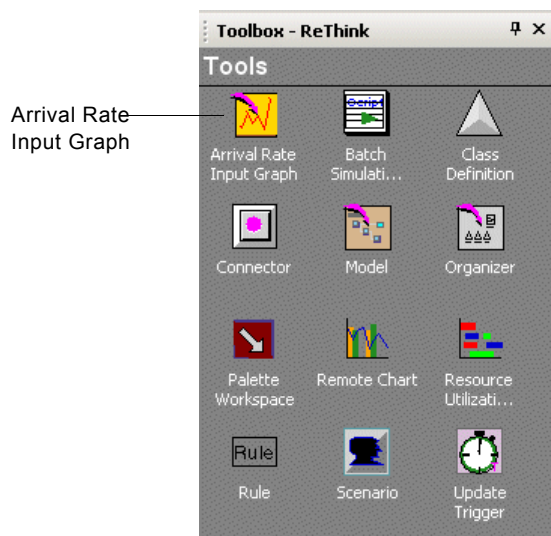
To use an Arrival Rate Input Graph, you must:

- [Create the graph.](#)
- [Configure the graph.](#)
- [Edit the shape of the graph.](#)
- [Configure the Source block to use the graph.](#)

Creating an Arrival Rate Input Graph

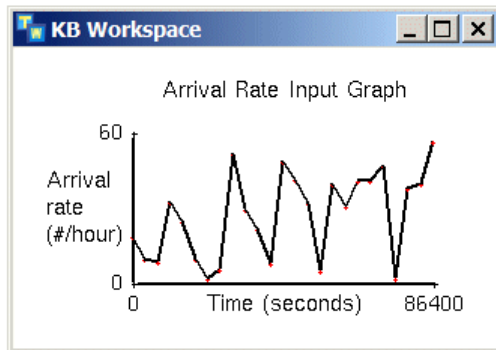
To create an Arrival Rate Input Graph:

- 1 Display the Tools palette of the ReThink toolbox:



- 2 Select an Arrival Rate Input Graph and place it on the model detail.
- 3 Choose Create Input Graph on the tool.

ReThink creates a special-purpose graph for specifying the input arrival rate of work objects:



Note This graph represents the input arrival rate of work objects to a block; it does *not* represent the time between arrival of the objects. ReThink interprets this graph to compute arrival times.

Configuring the Arrival Rate Input Graph

The horizontal axis of an Arrival Rate Input Graph represents the time period over which the Source block generates objects. The default is 1 day, represented in seconds.

The vertical axis represents the number of objects that arrive per hour. The default maximum is 60 objects.

By default, a point at the maximum vertical position on the graph means that 60 objects will arrive per hour. ReThink interprets this point as an arrival time of 1 minute (60 objects per 60 minutes = 1 object per minute).

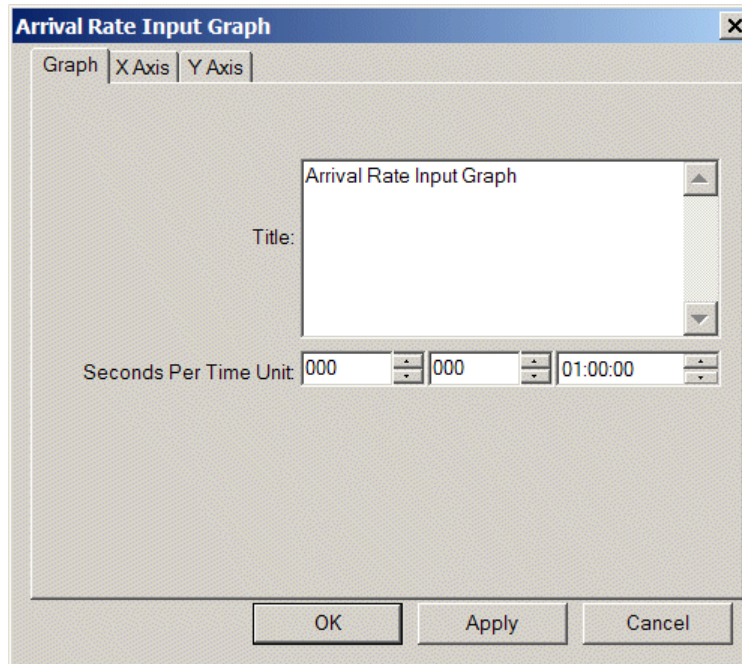
A point in the exact middle vertical position on the graph means that 30 objects will arrive per hour, which ReThink interprets as an arrival time of 2 minutes (30 objects per 60 minutes = 1 object every 2 minutes).

A point at the minimum vertical position on the graph means that 1 object will arrive per hour, which ReThink interprets as an arrival time of 60 minutes.

To configure the graph:

- 1 Choose properties on the Arrival Rate Input Graph.

Here is the General tab:



By default, ReThink interprets the value of the y axis as the number of objects that arrive per hour. You can change this attribute to interpret the arrival rate by using some other time interval, such as per day.

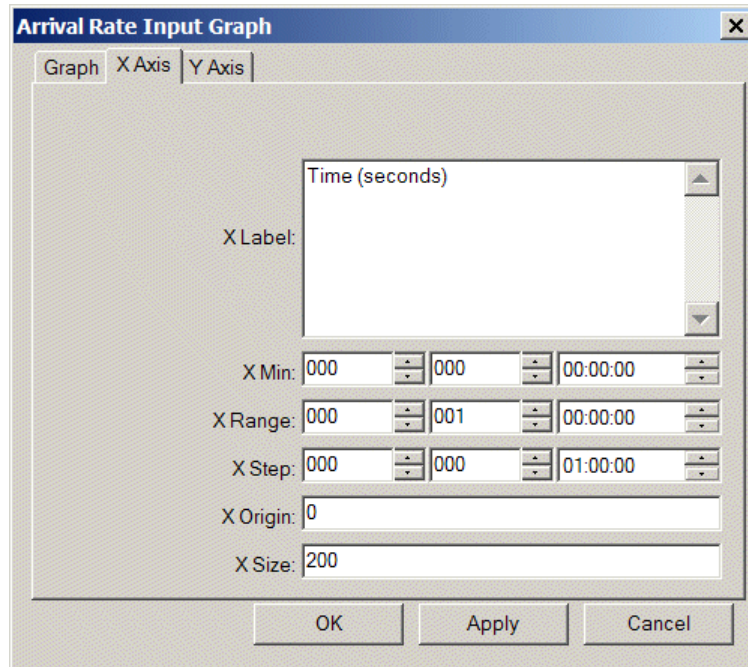
- 2 Configure the Time Unit attribute to be the time unit that ReThink uses to interpret the arrival rate.

Tip If you edit the Time Unit, edit the Label attribute of the X and Y axes to reflect the unit you specify.

By default, the graph allows you to model the arrival rate of objects over a 24 hour period.

- 3 To specify the time range over which the block creates objects, click the X Axis tab and configure the X Range attribute.

Here is the default X Axis tab:



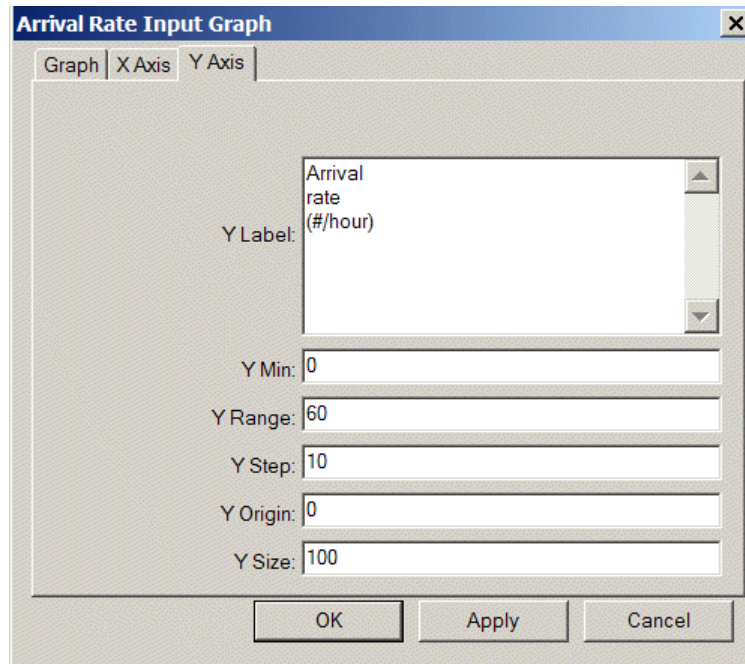
For example, if you are modeling a 10-hour period during the work day, as opposed to a 24-hour day, specify the Range of the x axis to be 10 hours.

- 4 To specify a starting time other than 0, configure the X Min attribute.
- 5 To specify the default number of points in the graph, configure the X Step attribute.

For the x axis, the X Step attribute in combination with the Range attribute determine the default number of points that ReThink uses to generate the graph. The number of points is the X Range minus the X Min divided by the X Step, plus one. For example, if the X Step is 2 hours and the Range is 10 hours, the graph will have 6 points ($36000/7200 + 1$).

- 6 To specify the maximum number of objects per time period, click the Y Axis tab and configure the Y Range attribute.

Here is the default Y Axis tab:



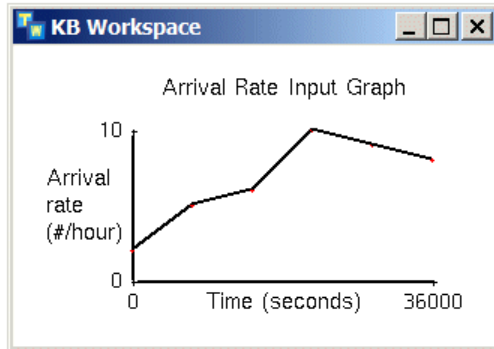
For example, if you want the block to emit a maximum of 10 objects per hour instead of 60 objects per hour, specify the Y Range to be 10.

- 7 To specify a minimum number other than 0, configure the Y Min attribute.
The default size of a graph is 200 pixels along the x axis by 100 pixels along the y axis.
- 8 To specify the size of the graph in pixels, click the X Axis tab and configure the X Size attribute and click the Y Axis tab and configure the Y Size attribute.

To update the graph after configuring it:

- Choose Update Input Graph on the Arrival Rate Input Graph.

For example, here is a graph configured to generate a maximum of 10 objects per hour, over a 10-hour period, with only six points by default:



ReThink adds and deletes points based on the values of the Range and Step attributes of the x axis.

Note If you have already edited the shape of the graph by moving or editing points, you might want to choose the Create Input Graph menu choice instead of Update Input Graph to regenerate the graph according to the new specification.

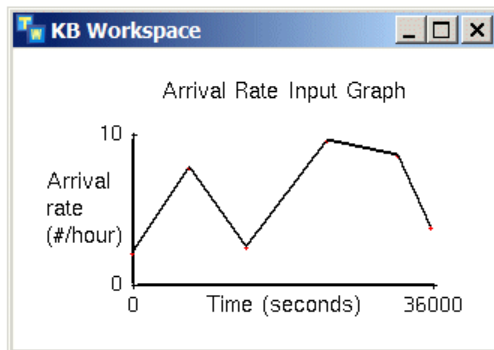
Editing the Shape of the Arrival Rate Input Graph

Once you have configured the graph, you edit its shape by manually dragging points to new locations. You can also edit the exact location of each point by specifying the x, y coordinates as attributes. You add and delete points, as necessary.

To adjust the position of points on the graph manually:

➔ Click a point in the graph and drag it to a new location.

For example, the following graph represents the arrival rate of calls described at the beginning of this section:



To specify the x, y coordinates of the point directly:

- 1 Click a point in the graph and choose Set Point.
- 2 Edit the X and Y attributes to specify the exact coordinates of the point.

To add points to the graph:

- 1 Click a point in the graph near where you want a new point to appear to display its table.
- 2 Choose New Point to add a point on top of the old point.
- 3 Drag the new point off the existing point to adjust the shape of the graph.

To delete points from the graph:

- ➔ Drag an existing point on top of another point.

or

- ➔ Click a point and choose Delete.

Configuring the Block to Use the Graph

Now that you have configured the graph, you must configure the block to use the Arrival Rate Input Graph for its durations.

To configure the block to use a graph:

- 1 Display the properties dialog of the block whose duration you want to configure and click the Duration tab.
- 2 Configure the Distribution Mode to be Arrival Rate Input Graph.
- 3 Choose the Choose Input Graph menu choice, then choose Select on the Arrival Rate Input Graph that is to supply duration times for the block to select the graph.

ReThink displays an indicator arrow next to the selected graph, which is now available as input for the block.

- 4 Choose Show Source on the Arrival Rate Input Graph to show the block that uses the graph.

Specifying a Custom Duration

You can specify a custom duration by creating your own procedure that specifies exactly how to compute the duration. To do this, you work in Developer mode and you must be familiar with creating G2 procedures.

For more information, see the *Customizing ReThink User's Guide*.

To specify a custom duration:

- 1 Create a custom procedure that computes duration.
- 2 Choose Tools > User Mode > Developer.
- 3 Display the properties dialog for the block and click the Duration tab.
- 4 Choose Custom as the value of the Distribution Mode.
- 5 Configure the Procedure Name to be the name of the procedure you created above.

Understanding Total Work Time and Total Elapsed Time

The Total Work Time and Total Elapsed Time of a block are related to the Work Time and Elapsed Time metrics of the block's activities, as follows:

Attribute	Description
Total Work Time	The sum of all the Work Time values for each activity currently scheduled by the block, as of the current simulation time.
Total Elapsed Time	The amount of time that has elapsed since the simulation began or since you created the block in a running simulation.

The block also computes the Average in Process, which is based on the Total Work Time and Total Elapsed Time.

For an explanation of Average in Process, see [How the Block Uses Total Work Time and Total Elapsed Time](#).

You can view the Total Work Time and Total Elapsed Time of a block in the properties dialog for the block.

To display the total work time and total elapsed time of a block:

- ➔ Run the simulation, then display the properties dialog for the block and click the Duration tab.

Here is the Duration tab of the properties dialog for a block that is currently processing:

Task [X]

General | Block | **Duration** | Cost | Animation

Duration

Distribution Mode: Random Normal

Mean: 000 000 01:00:00

Standard Deviation: 000 000 01:00:00

Miscellaneous

Time Per Unit Attribute: []

Total Work Time: 000.000.21:53:38

Total Elapsed Time: 000.000.13:25:22

Creation Time: 000.000.00:00:00

Average In Process: 1.631

OK Apply Update Cancel

Total Work Time

The Total Work Time of a block depends on the number of concurrent activities the block is processing. If the block is processing more than one activity, it updates the Total Work Time of the block each time an activity is created and each time an activity ends. Thus, depending on the frequency with which work objects flow into a block, as well as the duration of the block, the Total Work Time can include partial work times for some of its activities.

For an example of how the block computes Total Work Time, see [Relating Work Time and Elapsed Time of Activities and Blocks](#).

Total Elapsed Time

The Total Elapsed Time of a block is a measure of the total processing time of a block from the beginning of the simulation. The Total Elapsed Time is the same for each block in the simulation, assuming they were all created at the same time. Note that the block does not update the duration metrics that are based on the Total Elapsed Time unless the block is currently processing work objects.

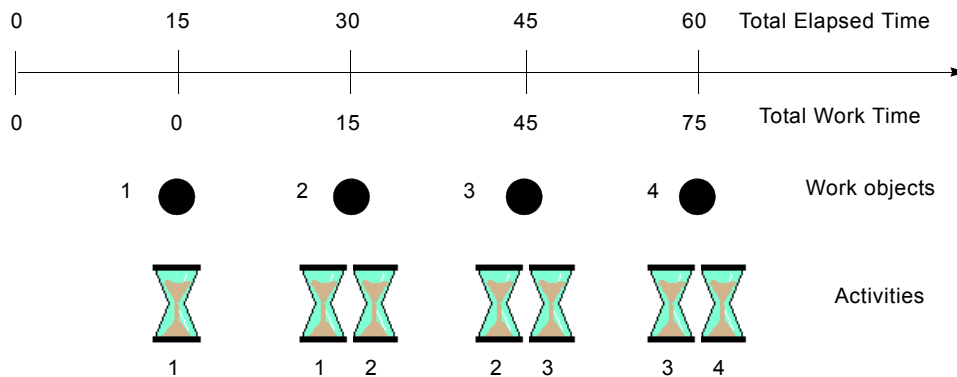
During the course of the simulation, you can update the Total Elapsed Time of the block and the metrics that depend on it, as described in [Updating Duration Metrics for Blocks](#).

If you create a block and add it to the model while the model is running, the Total Elapsed Time for that block begins when you connect it to the model, and the Creation Time is the simulation time at which the block was created.

Relating Work Time and Elapsed Time of Activities and Blocks

Suppose a Source block generates an object exactly once every 15 seconds, and a Task block takes exactly 30 seconds to complete. The task updates every 15 seconds, each time it receives a new work object. The Total Work Time is a cumulative measure of the amount of work applied by each activity at each update interval.

The following time line shows the Total Work Time and Total Elapsed Time for the Task block at the point when each new work object arrives at the block. It also shows a snapshot of the associated activities at the same point in time. The Work Time of each activity is always 30 seconds.



At 15 seconds of Total Elapsed Time, one activity is associated with a single work object. At the point at which the work object arrives at the block, the activity has not yet contributed any work time to the task, thus the Total Work Time is zero.

At 30 seconds, two activities are associated with work objects 1 and 2. The Total Work Time represents the amount of work time that each activity has contributed

at 30 seconds. The first activity has only contributed half of its total work time, which is 15 seconds, and the second activity has contributed nothing.

At 45 seconds, the block has finished processing the first work object, and now two activities are associated with work objects 2 and 3. The Total Work Time represents the total work that the first activity has contributed, which is 30 seconds, plus the partial work that the second activity has contributed, which is 15 seconds, for a total of 45 seconds.

Finally, at 60 seconds, the block is processing work objects 3 and 4. The Total Work Time represents the total work of the first and second activities, which is 30 seconds each, plus the partial work of the third activity, which is 15 seconds, for a total of 75 seconds.

How the Block Uses Total Work Time and Total Elapsed Time

ReThink uses the Total Work Time and Total Elapsed Time metrics to compute the Average in Process, which is the Total Work Time divided by the Total Elapsed Time. The Average in Process gives an indication of the average number of concurrent activities that a block has performed since the start of the simulation.

To see the Average in Process of a block:

- ➔ Run the simulation, then display the properties dialog for the block and click the Duration tab.

The following table shows the Average in Process at each time interval:

When Total Elapsed Time is...	And Total Work Time is...	Average in Process is...
0	0	0
15 seconds	0	0
30 seconds	15 seconds	.5
45 seconds	45 seconds	1
60 seconds	75 seconds	1.25

For example, if the Average in Process of a block is less than one, you know that the block has been processing less than one activity on average over the course of the simulation. On the other hand, if the Average in Process is a number greater than one, you know that on average the block has been processing more than one activity concurrently over the course of the simulation.

Updating Duration Metrics for Blocks

ReThink does not update the Total Elapsed Time for all blocks in a model, by default; it only updates the Total Elapsed Time of the block currently processing work objects. Thus, the Total Elapsed Time and the Average in Process, which is computed based on the Total Elapsed Time, do not always reflect the current values for all blocks in the model. For example, the Total Elapsed Time of a block is zero until it starts processing.

To update the duration metrics for a block:

- Choose Update on the block or click the Update button in the properties dialog.

Computing Duration for Multiple Units of Work

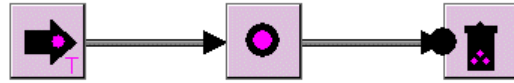
If the work object your model is processing represents multiple units of work, ReThink can compute the duration by multiplying the sampled duration by the number of units to produce the overall duration. The time unit is hours.

To compute the duration for multiple units of work:

- 1 Create an attribute of the work object that represents the number of units of work.

For example, if a work object represents a set of documents, you might call the attribute `number-of-documents`.
- 2 Display the properties dialog for a block and click the Duration tab.
- 3 Configure the Time per Unit Attribute to be the name of the attribute of the work object that specifies the number of units.

When the block computes its duration, it multiplies the duration computed, based on the Distribution Mode by the number of units to produce the overall block duration, as this example shows:



XYZ

number-of-documents is an integer, initially is 2

Task [X]

General | **Block** | Duration | Cost | Animation

Duration

Distribution Mode: Fixed Distribution

Mean: 000 000 01:00:00

Miscellaneous

Time Per Unit Attribute: NUMBER-OF-DOCUMENTS

Total Work Time: 000.001.03:00:00

Total Elapsed Time: 000.000.15:00:00

Creation Time: 000.000.00:00:00

Average In Process: 1.8

OK Apply Update Cancel

Working with Block Costs

One way of analyzing cost is to assign fixed and variable costs directly to a block. You use this technique as a quick way of representing cost, when you are not modeling the resources associated with a task.

For information on modeling costs associated with resources, see [Working with Resource Costs](#).

Note You typically assign costs to either the resources associated with a particular block or to the block itself; you do not typically specify costs for both a block and the block's resources.

You configure fixed and variable costs by using the Cost tab of the properties dialog. Each block also computes the total cost of the block, which you can view in the properties dialog for a block or in the Metrics toolbar. You can also create a Block Summary Report, which displays cost metrics for all blocks in your model.

Configuring the Cost of a Block

You can specify a fixed cost or a variable cost for a block. The variable cost is based on the duration of the block and a time unit, which is 1 hour, by default.

You can specify both a fixed and variable cost for a block, in which case ReThink sums the two costs to compute the cost of the activity.

Specifying a Fixed Cost

To specify a fixed cost for a block:

- 1 Display the properties dialog for the block and click the Cost tab.
- 2 Configure the Cost Per Use to be the fixed cost for the block.

Each time ReThink creates an activity for the block, it assigns the specified fixed cost to the activity.

Here is the Cost tab of the properties dialog with a fixed cost specified:

The screenshot shows a dialog box titled "Task" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Cost" tab is selected. Inside the dialog, there are four rows of input fields:

- "Cost Per Use:" followed by a text box containing the number "10".
- "Cost Per Time Unit:" followed by a text box containing the number "0".
- "Time Unit:" followed by three spinners. The first two contain "000" and the third contains "01:00:00".
- "Total Cost:" followed by a text box containing the number "0".

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Specifying a Variable Cost

ReThink computes variable cost by multiplying the specified cost per time unit by the duration of the activity, given a time unit. For example, if the Cost Per Use is \$20 per hour and the task takes one hour, the cost of the activity is 20. However, if the task only takes 30 minutes, the cost of the activity is only 10.

By default, ReThink computes variable costs based on an hourly cost. You can specify a different time unit for computing variable cost. For example, you might specify the cost per day of operating a particular piece of equipment.

To specify a variable cost for a block:

- 1 Display the properties dialog for the block and click the Cost tab.
- 2 Configure the Cost Per Time Unit to be the variable cost for the block.
- 3 Configure the Time Unit to be the time unit for computing variable cost.

ReThink computes the variable cost based on the time unit you specify.

Computing the Total Cost of a Block

The **total cost** of a block includes the cost of all of its individual activities, which includes the fixed and variable costs assigned to the block and the fixed and variable costs assigned to any resources associated with the block.

As mentioned earlier, you typically assign costs to either the block or the resources, but not to both. However, keep in mind that total cost takes both of these costs into account.

For information on assigning costs to resources, see [Working with Resource Costs](#).

To display the total cost of a block:

- ➔ Run the simulation, then display the properties dialog for the block and click the Cost tab.

Here is the Cost tab of the properties dialog for a block with a fixed cost of \$10 and a variable cost of \$5 that has processed three work objects:

The screenshot shows a dialog box titled "Task" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Cost" tab is selected. Inside the dialog, there are four rows of input fields:

- Cost Per Use: 10
- Cost Per Time Unit: 5
- Time Unit: 000, 000, 01:00:00
- Total Cost: 45.0

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

ReThink updates the Total Cost of a block each time the block processes a work object. Thus, the total cost of a block at a particular moment depends upon the frequency with which the block receives work objects and the block's duration.

In the above example, suppose the Source block generates work objects once every hour exactly and a Task block takes exactly one hour to process each work object. If the Task block has a \$10 fixed cost and a \$5 hourly cost, the cost per

activity of the Task block is \$15. The block updates the Total Cost every hour, as follows:

When Total Elapsed Time is...	Total Cost is...
0	0
1 hour	15
2 hours	30
3 hours	45

The current total cost of a block depends on the number of activities the block is processing at a given time, which in turn depends on the duration of the current block and the arrival rate of the work objects.

For information on displaying the cost of an individual activity, see [Understanding the Attributes of Activities](#).

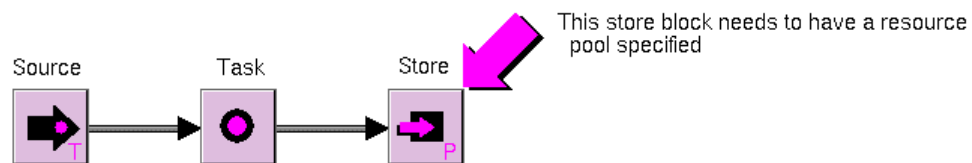
Debugging Blocks

When you build a model, the blocks sometimes do not behave as intended. You might encounter an error in the configuration of a block, or the block metrics might produce unexpected results. You can use various techniques to reset errors and verify that the model is behaving as expected.

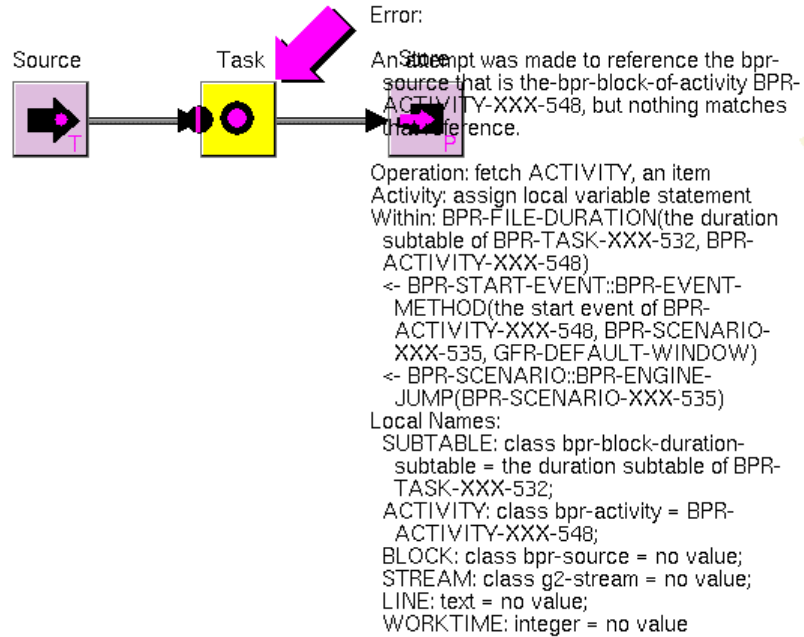
Viewing and Resetting Errors

If you have not configured a block correctly, one of two things can happen:

- An indicator arrow appears next to the block with a message telling you what you need to configure. For example, if you do not choose a pool for the Store block when the Store Mode is Pool, you would receive this message:



- The block turns yellow and an indicator arrow appears next to the block with the word Error. For example, if you have configured a block's Mode Type to be Duration File and you have not specified a Duration File Name, you would receive this type of error:



You can view the error message for a block in the properties dialog. Depending on the type of error, sometimes errors also appear in the Logbook or on the Message Board. For information about viewing these types of errors, see [Viewing Messages](#).

To view an error message in the properties dialog:

- 1 Display the properties dialog for the block and click the General tab.
- 2 Scroll to the top of the Error scroll area to view the message.

For example:

The screenshot shows a dialog box titled "Task: Task" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Block" tab is selected. The "Block Label" field contains the text "Task". The "Comments" field is empty. The "Maximum Activities" field is empty. The "Url" field is empty. The "Total Starts" field contains the value "0". The "Total Stops" field contains the value "0". The "Current Activities" field contains the value "0". The "Error" field contains the text: "Error: An attempt was made to reference the bpr -source that is the-bpr-block-of-activity BPR-ACTIVITY-XXX-552, but nothing". At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

To reset an error condition:

- ➔ Click the indicator arrow to remove it, reset the simulation, then configure the block correctly to fix the error condition.

Verifying Model Metrics

Much of the computation that ReThink performs is based on the duration of the activities associated with a block. Because of deviation in the duration of an activity, it is not always easy to tell how ReThink computes a particular metric.

Before you put a model into operation, you will probably want to verify the metrics that the model computes. You have several strategies for doing this.

You can verify metrics that the model computes by:

- Choosing Single Shot on the Source block to track the performance of a single work object to verify the results.
- Setting the maximum number of work objects a Source block creates by specifying a value for Maximum Starts to control the number of work objects that flow into the model.
- Running the model in Step mode, analyze summary metrics for blocks, work objects, and resources at various stages of the process, and compare against expected results.
- Running the model in jump mode and set a break point in the model where you want to analyze metrics by choosing Set Break on a block.
- Run the simulation by using a fixed duration by configuring the Distribution Mode to be Fixed Distribution. You can also use a Random Normal distribution and set the Standard Deviation to 0.

For more information on configuring these types of durations, see [Specifying a Fixed Duration](#) and [Random Normal](#).

Testing Every Possible Outcome

When you branch work in a model, one of the output paths might be used very rarely, due to probabilities. To test every possible outcome in a model, you might want to identify explicitly the path onto which you want a work object to flow. For example, you might want to do this when you are branching work objects based on probabilities to test the lowest probability outcome.

To branch work onto an explicit output path:

- ➔ On the Block tab of the properties dialog of a Branch block, configure the Branch Mode to be Prompt.

When you run the simulation, you identify the output path interactively by clicking on the desired output path. For more information, see [Interactively Selecting the Output Path](#).

Customizing Blocks

All ReThink block definitions are available to you for viewing and customizing. You can also create directly a subclass of an existing ReThink block by using a menu choice on the block.

You can also customize the default behavior of the path queue.

For detailed information about how to customize blocks and path, see the *Customizing ReThink User's Guide*.

Using Instruments

Describes how to use ReThink instruments to probe and chart the performance of your model, and how to feed input parameters into your model.

Introduction	205
Creating Instruments	207
Connecting Instruments	212
Probing the Performance of Your Model	215
Charting Performance Metrics	228
Exporting Probed Data to a CSV File	230
Feeding Values into the Model	236
Creating User Interface Objects for Feeding Values	243
Creating a Chart Directly from a Probe	246
Customizing Instruments	259

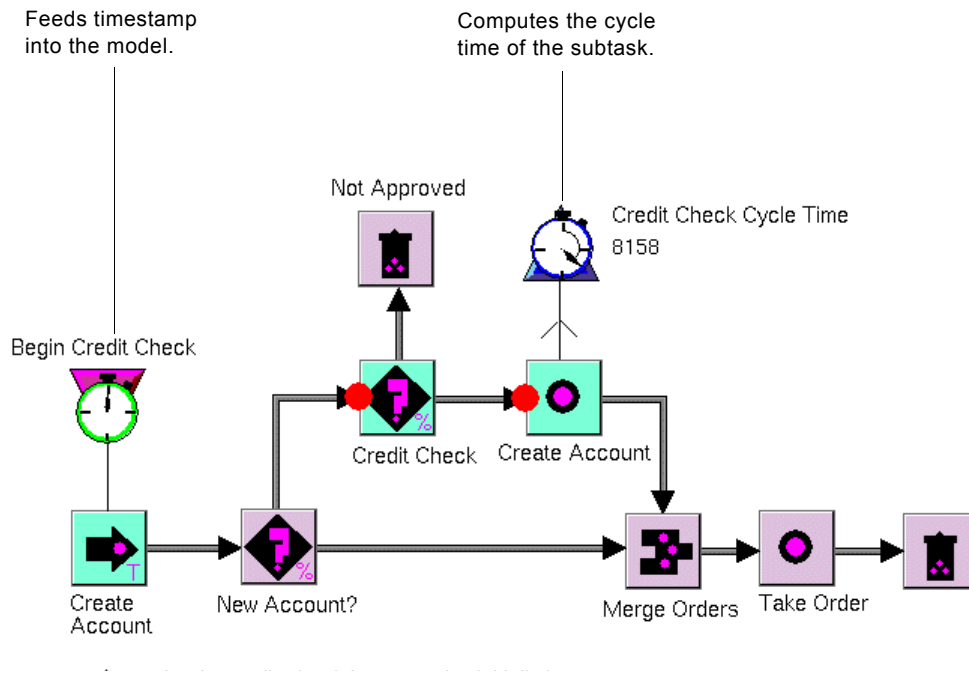


Introduction

ReThink **instruments** allow you to obtain performance metrics from your model and to control key parameters in your model, both while the model is running. Using instruments, you can create a complete user interface for performing “what-if” analysis on your model.

Here is the detail of a Take Order task, which details the credit check approval process. The detail uses instruments to feed a timestamp into the model at the

beginning of the subprocess and to probe the model at the end of the subprocess to compute a partial cycle time for the subtask.



ReThink provides these two categories of instruments:



- **Feeds**, which supply values to your model, such as the mean time of a block, the hourly wage of a resource, or the timestamp at a particular point in the model's processing. When you use feeds to supply duration and cost, you typically create **type-in boxes** and **sliders** from your feeds to provide a user interface for supplying values to your model.



- **Probes**, which obtain information from your model, such as the total work time of a work object, the total cost of a block, the average utilization of a resource, the cycle time from one point in time to another, or the moving average of any value. Because probes maintain a history of their values, you can easily chart these values over time to provide a visual representation of key performance metrics in your model.

To use ReThink instruments, first you connect the instrument to a block, resource, or another instrument, depending on the type of instrument. You then configure the instrument to specify the object to which the instrument applies and the attribute or attributes that the instrument is either feeding a value into or probing.

You can create **charts** to plot the current values of probes over time. To create the chart, you create a **remote** from a probe, which keeps a history of probed values. You configure the chart to plot the history of one or more remotes, using a variety of chart types.

You can also create a simple chart directly from a probe. Note that this technique has been superseded by remote charts.

Creating Instruments

You create instruments from the Instruments palette in the ReThink toolbox.

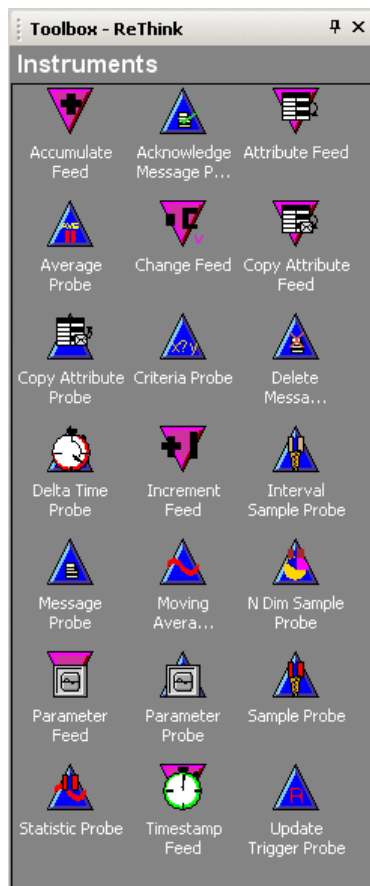
You can also copy existing instruments. For details, see [Cutting, Copying, Pasting, and Deleting Objects](#).

You connect instruments to blocks, resources, other instruments, or variables, depending on the requirements of the model. For more information, see [Connecting Instruments](#).

Creating Instruments

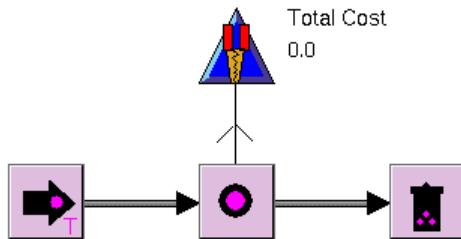
To create instruments:

- 1 Display the Instruments palette of the ReThink toolbox:



- 2 Select an instrument in the toolbox, then click just above the object on which it will operate to place it on the workspace.
- 3 Connect the instrument to the object on which it will operate.
For details, see [Connecting Instruments](#).
- 4 Display the properties dialog for the instrument and configure the Label as a text, then drag the label to the desired location next to the instrument.

Here is a Sample probe connected to a Task block that probes the total cost of the work object that it processes:



Following is a summary of each ReThink instrument.

See also [Instruments Reference](#).

Timestamp Feed



The Timestamp feed supplies a timestamp into an attribute of a work object at a particular point in the process. You use the Timestamp feed with a Delta Time probe to compute a partial cycle time for a model. For example, you might want to compute the cycle time of the distribution subtask of an order fulfillment model.

Accumulate Feed



The Accumulate feed increments an attribute of a work object by the value specified in another attribute of the object, thereby accumulating its value. For example, suppose you are modeling a sales process that processes local and regional sales calls, where each type of sales call has an associated mileage attribute. You use an Accumulate feed to increment a total mileage attribute of each sales call object by each specific mileage amount.

Increment Feed



The Increment feed increments a counter of the feed by a value specified in the feed. For example, you use this feed to report on the number of times a work object has gone around a loop in the model.

Change Feed



The Change feed modifies an attribute of the model by using a new value. For example, you use a Change feed to modify the frequency with which work objects flow into a model from a Source block or the hourly wage of a resource allocated by a task. You can configure the Change feed to generate random values, unique IDs, or random values, based on a function. You can also create sliders and type-in boxes from a Change feed, thereby creating a user interface for supplying input parameters to the model.

Parameter Feed



The Parameter feed gets the value of a parameter and sets its current value as the value of an attribute of the model. You can create the parameter from the feed.

Attribute Feed



The Attribute feed copies the value of an attribute of one object to another attribute, either in the same object or in a different object. For example, you might want to copy the work time of an activity of a block into an attribute of the output work object.

Copy Attributes Feed



The Copy Attributes feed copies attributes *from* a source object *to* the object to which the feed applies. You can use the Copy Attributes feed to copy metrics computed in a higher-level Task block to work objects on the detail.

Delta Time Probe



The Delta Time Probe compares a timestamp of an object with the current simulation time. You use this probe to compute cycle times in your model. For example, you can probe the creation time of an object and compare it to the current time at the end of the process to obtain the cycle time of the overall process.

You often use the Delta Time probe with a Timestamp feed to compute a partial cycle time. The probe compares the timestamp you feed into the model to the current simulation time.

Sample Probe



A Sample probe obtains any attribute value that the model computes. For example, you use the Sample probe to obtain the total work time of a block, the total cost of a work object, or the average utilization of a resource. You can also use a Sample probe to probe the current value of a quantitative parameter.

Average Probe



The Average probe computes an average of all sampled values. The probe keeps track of the minimum and maximum values. You use the Average probe to compute the average of an attribute of a ReThink object, whose value does not depend on how long it has persisted, such as the average duration of a block. You can also use an Average probe to probe the current value of a quantitative parameter.

Moving Average Probe



The Moving Average probe computes a time-weighted average of any attribute in the model whose value depends on how long it has persisted, such as the number of activities of a block. You can specify the time period over which the probe computes the average.

You typically use the Moving Average probe to probe another probe. For example, you can use a Delta Time probe to compute a cycle time and then probe the Delta Time probe to obtain a moving average of the cycle time.

Interval Sample Probe



The Interval Sample probe averages or sums the value of an attribute of the model at regular time intervals, based on simulation time. You use an Update Trigger to determine when to sample the model. For example, you would use an Interval Sample probe to chart on a weekly basis the average total cost of work objects at a particular point in the model.

Parameter Probe



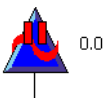
The Parameter probe sets the value of a parameter to the value of an attribute of the model. You can create the quantitative parameter from the probe.

Copy Attributes Probe



The Copy Attributes probe copies attributes *from* the object to which the probe applies *to* a destination object. You can use the Copy Attributes probe to “roll up” metrics computed on a detail to the higher-level Task block. For example, you might want to “roll up” the Total Cost of the task on the detail to the superior task.

Statistics Probe



The Statistics probe obtains any attribute value that the model computes and computes various metrics on the value, given a time window. You can compute

the average, moving average, time-weighted average, standard deviation, minimum, and maximum.

Criteria Probe



The Criteria probe compares a sample value in the model against criteria you configure in the probe to determine the percentage of time the sampled value meets the criteria. You configure the value to compare and the operator to use in the comparison. For example, you would use a Criteria probe to determine the percentage of time that the total cost of a work object goes above a certain value.

Update Trigger Probe



The Update Trigger probe triggers updates in the model, based on model events. You connect the Update Trigger probe to a block, instrument, or resource in the model. When the attached object evaluates, the Update Trigger probe triggers updates for all its associated objects. Compare an Update Trigger probe with an Update Trigger tool, which triggers updates, based on simulation time.

N-Dimensional Sample Probe



You use an N-Dimensional Sample probe to obtain multiple sample values from the model, using a single probe, and optionally keep a history of those values over time. To view the sampled data, you export the n-dimension samples to Excel by using an Excel Export tool.

For example, you might use an N-Dimensional Sample probe to collect a history of the Total Cost of a task, the Total Cost of a work object, and the Average Utilization of a resource. You could then export this data to Excel to create a report and a graphical representation of the data over time.

Message Probe



You use a Message probe to pause the model at a particular location and display a message. ReThink displays an indicator arrow with the message text next to the probe when the Message probe triggers. For example, you might want to pause the model and display a message when the Retrieve block retrieves an object from a pool. You can also use this probe to generate messages in the Message Browser.

Acknowledge Message Probe



You use an Acknowledge Message probe to acknowledge messages generated by the Message probe in the Message Browser.

Delete Message Probe



You use a Delete Message probe to delete messages generated by the Message probe in the Message Browser.

Connecting Instruments

You connect instruments to objects in your model in the same way you connect blocks, using the **wire** attached to the instrument. Wires on instruments are similar to input and output stubs on blocks.



Connecting Instruments to Objects

You can connect instruments to these types of objects in your model:

- Blocks
- Instruments
- Resources
- Parameters and variables

Parameters and variables are objects that maintain a history of values over time, which you create from a Parameter feed or Parameter probe. You can only attach a Sample probe or an Average probe to a quantitative parameter.

For more information on connecting instruments to parameters, see:

- [Average Probe](#).
- [Sample Probe](#).
- [Parameter Probe](#).
- [Parameter Feed](#).

By default, instruments probe attributes and feed values *after* the attached block, instrument, or resource evaluates and applies its duration metrics to the simulation. You can also configure any instrument to probe and feed values *before* the attached object evaluates and applies its duration. Thus, when connecting instruments to blocks, consider carefully where you connect the instrument and when you want it to evaluate depending on the object you want to feed or probe.

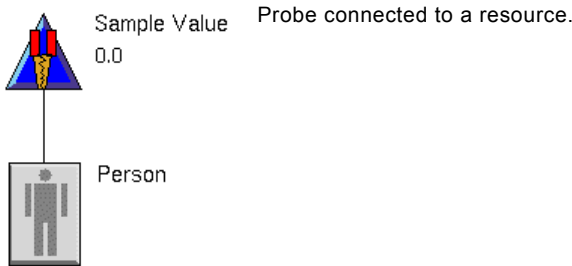
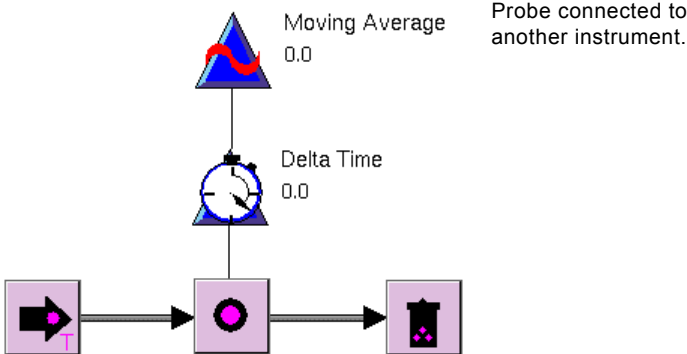
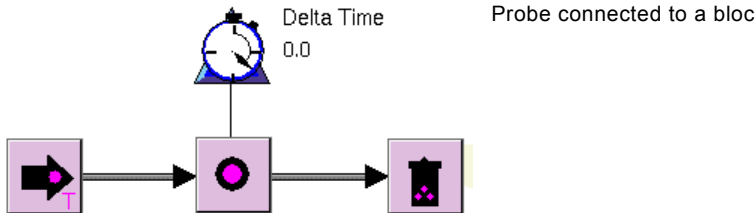
For information on specifying when an instrument evaluates, see:

- [Configuring the Probe.](#)
- [Configuring the Feed.](#)

To connect an instrument to an object:

➔ Click the wire attached to the instrument, move it to the middle of the object, and click to connect.

These examples show a Delta Time probe connected to a block, a Moving Average probe connected to a Delta Time probe, and a Sample probe connected to a resource:

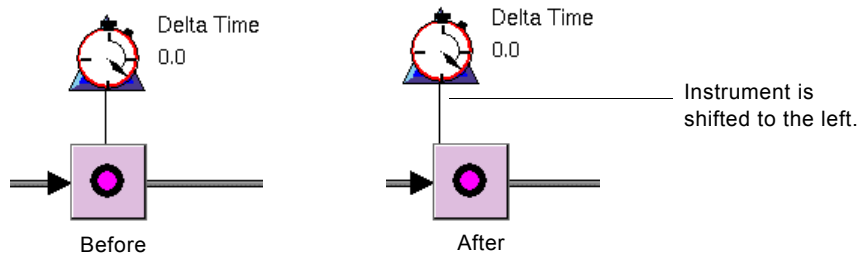


Note If you attach a probe to a Sink block to obtain metrics about work objects, you must probe the Sink block before the attached block applies its duration to the simulation by configuring the Phase to be **Start**.

To redraw the connection between an instrument and an object:

➔ Drag the instrument to a new location on the connected object.

ReThink reconfigures the connection:



If you delete the connection stub on an instrument, you can create it again, using a menu choice. The menu choice is only available if you have deleted the connection stub.

To create a new connection stub on an instrument:

➔ Choose Create Connection on the instrument.

Replacing Instruments

You can drop a new instrument on top of an existing connected instrument to replace the instrument. ReThink maintains all existing connections. The new instrument copies the common configuration information from the existing instrument. For example, if the Apply to Class Name of the original instrument is order, the Apply to Class Name of the new instrument will also be order, even if they are different types of instruments.

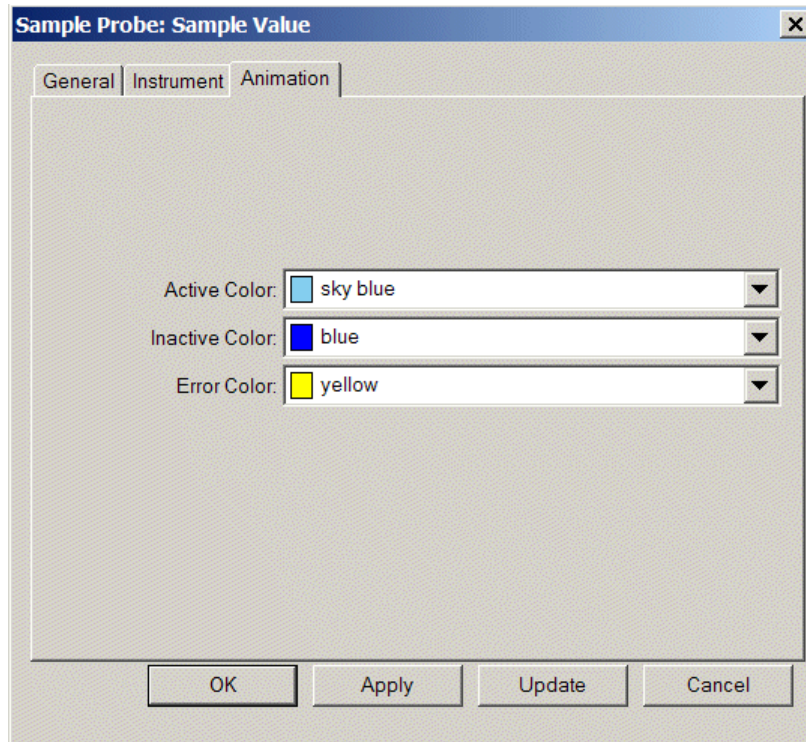
Configuring the Animation of Instruments

You can configure these colors of an instrument when it animates:

Animation Parameter	Description
Active Color	The color the instrument uses when it is processing.
Inactive Color	The color the instrument uses when it is idle.
Error Color	The color the instrument uses when it is in an error state.

To configure the colors of an instrument when it animates:

- 1 Display the properties dialog for the instrument and click the Animation tab to display this dialog:



- 2 Choose a color from the dropdown list for each instrument color.

Probing the Performance of Your Model



ReThink lets you obtain performance metrics of various aspects of the model, which you present to the end user in the form of charts.

You can obtain performance metrics about these categories of objects:

- Blocks
- Activities
- Input and output paths
- Work objects
- Resources
- Instruments
- Parameters

For example, you might want to know the total number of concurrent activities performed by the current task, the total cost of all activities applied to the current work object at the end of a process, or the average utilization of the current resource allocated by a task.

To obtain performance metrics about the model, you use probes.

You can connect a probe to these types of objects to obtain these metrics:

- A block to obtain metrics of:
 - The block itself.
 - The current activity of the block.
 - The input or output path of the block.
 - The current work object the block is processing.
 - The resource that the block is currently using.
- Another probe to obtain metrics about the probe, for example, to compute a moving average of a probed value.
- A resource to obtain metrics about the resource directly.

Note When you probe the performance of work objects, keep in mind that if the model creates and deletes work objects as part of processing, the performance metrics reflect only the duration of the simulation during which the work object exists.

Configuring the Probe

Once you have connected a probe to an object, you must configure:

- The class of objects to which the probe applies.
- The name of the attribute of the class to probe.
- The **phase**, which determines when the probe evaluates relative to the attached block.

By default, probes evaluate *after* the blocks computes its duration.

Depending on the Phase you choose, the probe evaluates at different times and can probe different types of objects, as this table describes:

This Phase...	Causes the probe to evaluate...	And means you can probe these types of objects...
Start	Before the connected block applies its duration to the simulation	Input work objects, resources, blocks, activities or input paths.
Stop	After the connected block applies its duration to the simulation	Output work objects, resources, blocks, activities, or output paths.

Note If you are probing the output object and the Phase is set to Stop, and if the output object is different from the input object, the probe obtains its values from the output object just after the object is created.

To configure the probe:

- 1 Connect the probe to a block, probe, or resource, depending on the needs of your model.
- 2 Display the properties dialog for the probe and click the General tab.
- 3 Configure the Apply to Class Name to refer to one of the following classes or any user-defined subclass of these classes:

To probe a...	Configure Apply to Class Name as this class or any subclass...
Block	bpr-block
Work object	bpr-object
Resource	bpr-resource
Instrument	bpr-instrument
Activity	bpr-activity
Path	bpr-path

Tip Be as specific as possible when you specify the class name. For example, if you are probing a work object, be sure to specify the subclass; do not use bpr-object.

Note If you are probing the `bpr-object` of a block and the block you are probing has an attached Resource Manager, the probe will update each time a work object *or* a resource becomes active, because a resource is a subclass of `bpr-object`.

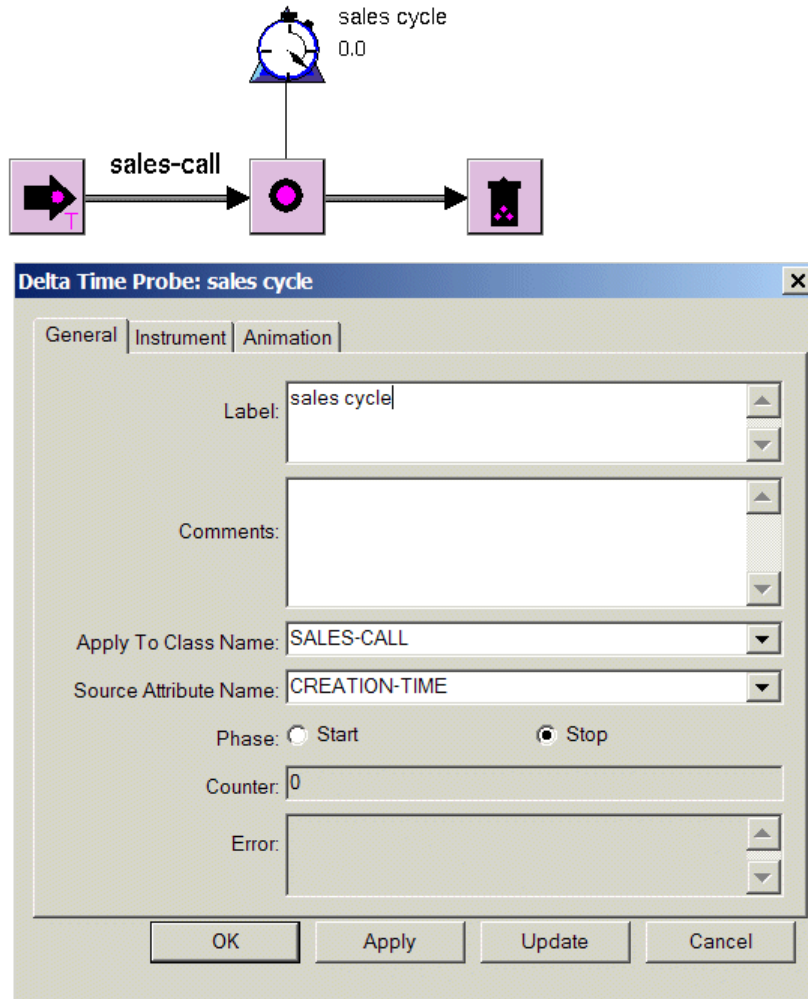
- 4 Configure the Source Attribute Name to be the attribute of the specified class that you want to probe, as a symbol.

For example, if you are using a Delta Time probe to compute the cycle time from the creation time of a work object to the end of the process, the Source Attribute Name is `creation-time`.

To determine the attribute name to probe, use the attribute name as it appears in the properties dialog and insert hyphens in place of spaces in the attribute name.

- 5 Configure the Phase to determine when the probe evaluates, and, when probing work objects and paths, whether the probe applies to the input or output work object or path.

This simple example probes the creation-time of the sales-call object that the Task block processes to compute the cycle time. The Delta Time probe specifies sales-call as the class and creation-time as the attribute to probe, where sales-call is a subclass of bpr-object and creation-time is an attribute of the sales call.



Showing the Current Value of the Probe

Each probe has a metric that shows the current probed value, which appears as an attribute display of the probe. The name of this metric depends on the type of probe, as follows:

This type of probe...	Defines this metric(s) that shows the current value...
Delta Time probe	Delta Time
Sample probe	Sample Value
Interval Sample probe	Sample Value
Average probe	Average Value Minimum Value Maximum Value
Moving Average probe	Moving Average Moving Standard Deviation
Statistics probe	See Statistics Probe .
Criteria probe	Criteria True Count Criteria True Percent

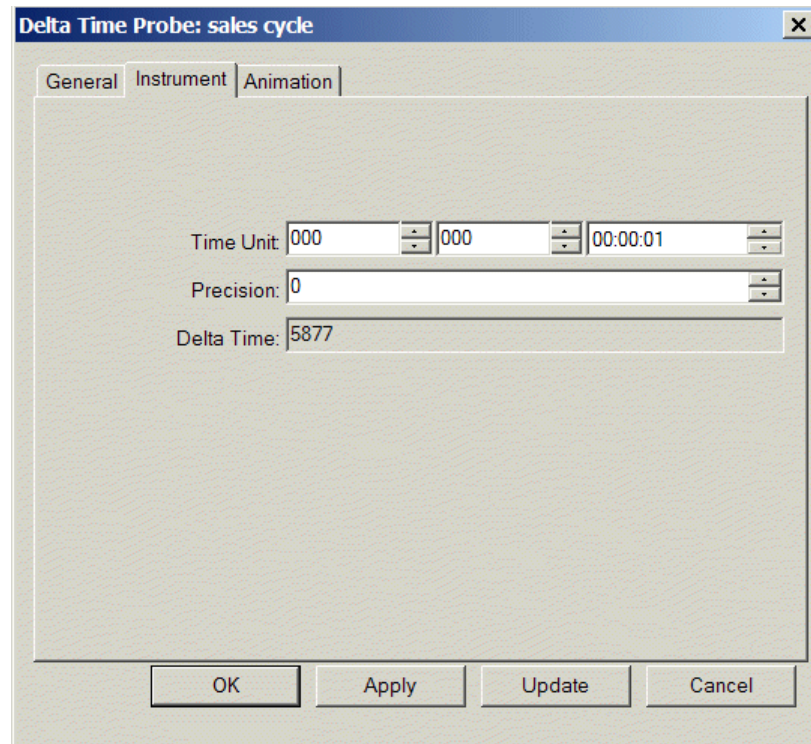
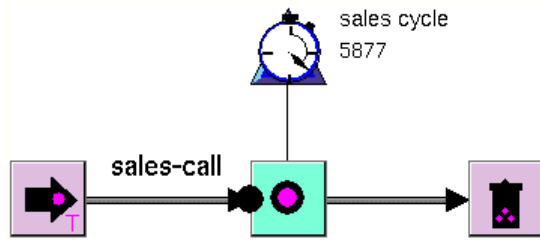
The current value of this metric is the current probed value. You can display the current value in a dialog to observe the value as it changes. The dialog also displays the number of work objects that the probe has processed.

You can also create a Probe summary report to create a report on all the probed values in the model.

To display the current value of the probe:

- ➔ Run the simulation, then display the properties dialog for the probe and click the Instrument tab.

This example shows the current value of the Delta Time probe:



Probing the Performance of Blocks

You can obtain performance metrics about the following attributes of blocks, using the following types of probes:

To obtain performance metrics on the...	Use a...	To probe the...
Total amount of work performed by a block	Sample probe	Total-work-time of the block.
Total cost of all activities performed by a block	Sample probe	Total-cost of the block.
Average number of concurrent activities performed by the block	Sample probe	Average-in-process of the block.
Total number of concurrent activities performed by a block	Sample probe	Current-activities of the block.
Total number of activities that a block processes	Sample probe	Total-starts of the block.
Average, minimum value, and maximum value of any metric listed above that is not time-persistent, such as Total-work-time	Average probe	Sample-value of the Sample probe, or any non-time-persistent metric listed above directly.
Time-weighted average and standard deviation of any time-persistent metric listed above, such as Current-activities or Total-starts	Moving Average probe	Sample-value of the Sample probe or any time-persistent metrics listed above directly.

For a complete description of each of these attributes, see [Using Blocks](#).

Note You cannot probe a Task block with detail. Probe the individual blocks on the detail instead.

Probing the Performance of Work Objects

You can obtain performance metrics about the following attributes of work objects, using the following types of probes:

To obtain performance metrics on the...	Use a...	To probe the...
Cycle time from the creation of a work object to a particular point in the process	Delta Time probe	Creation-time of the work object.
Cycle time of a subprocess within the overall model	Delta Time probe	Timestamp that you feed into the model, using a Timestamp feed.
Total amount of work time applied to a work object	Sample probe	Total-work-time of the work object.
Total amount of time that the work object has been waiting for resources or other work objects	Sample probe	Total-idle-time of the work object.
Total amount of time that a work object has been worked on compared with the total amount of time that it has been waiting for resources, over the entire simulation	Sample probe	Average-utilization of the work object.
Total cost of all activities applied to a work object	Sample probe	Total-cost of the work object.
Total number of activities that have been applied to a work object	Sample probe	Total-starts of the work object.

To obtain performance metrics on the...	Use a...	To probe the...
Average, minimum value, and maximum value of any metric listed above that is not time-persistent, such as Total-work-time	Average probe	Sample-value of the Sample probe, or any non-time-persistent metric listed above directly.
Time-weighted average and standard deviation of any time-persistent metric listed above, such as Current-activities or Total-starts	Moving Average probe	Sample-value of the Sample probe or any time-persistent metrics listed above directly.

For a complete description of these attributes, see [Using Work Objects](#).

Be sure to take the following precautions when you probe the performance of work objects:

- When you probe the performance of work objects, keep in mind that if the model creates and deletes work objects as part of processing, the performance metrics reflect only the duration of the simulation during which the work object exists.
- Do not attach a probe to a Task block with detail to probe the performance of a work object; the probe will not obtain any values. Instead, attach the probe to the last block on the detail.
- When probing work objects, be as specific as possible when you specify the class name, especially if the block to which the probe is attached also has an attached Resource Manager. If you probe the **bpr-object** of a block, rather than a subclass of **bpr-object**, and the block you are probing has an attached Resource Manager, the probe will update each time a work object *or* a resource becomes active, because a resource is a subclass of **bpr-object**.

Probing the Performance of Resources

You can obtain performance metrics about the following attributes of resources, using the following types of probes:

To obtain performance metrics on the...	Use a...	To probe the...
Total cost of all activities performed by a resource	Sample probe	Total-cost of the resource.
The amount of the resource currently allocated by a task	Sample probe	Current-utilization of the resource.
Average amount of the resource allocated by a task over the entire simulation	Sample probe	Average-utilization of the resource.
Amount of time that the resource has been allocated	Sample probe	Total-work-time of the resource.
Amount of time that the resource has been idle	Sample probe	Total-idle-time of the resource.
Total number of activities that the resource has performed	Sample probe	Total-starts of the resource.
Average, minimum value, and maximum value of any metric listed above that is not time-persistent, such as Total-work-time	Average probe	Sample-value of the Sample probe, or any non-time-persistent metric listed above directly.
Time-weighted average and standard deviation of any time-persistent metric listed above, such as Current-activities or Total-starts	Moving Average probe	Sample-value of the Sample probe or any time-persistent metrics listed above directly.

For a complete description of each of these attributes, see [Using Resources](#).

Three Techniques for Probing Resources

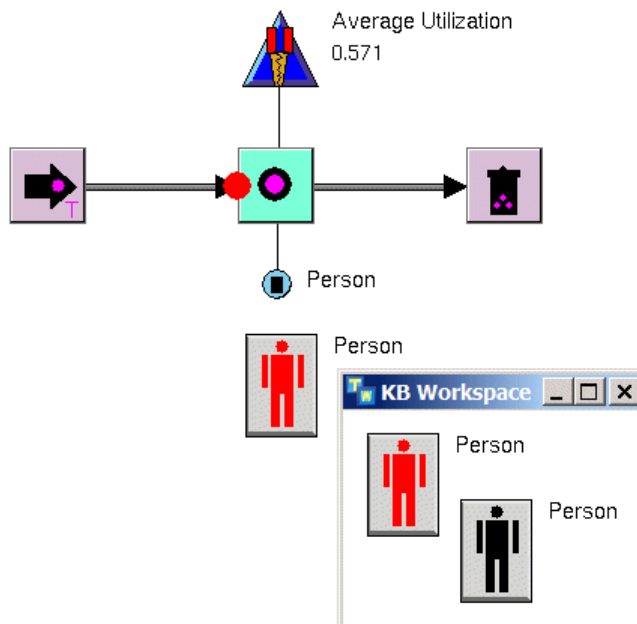
When you probe the performance of resources, you have three options:

To obtain performance metrics about...	Probe the...
The resource that is currently allocated by an activity	Block that requires the resource.
The sum of all resources in a resource pool	Top-level resource that is the pool.
A particular resource in a resource pool	Individual resource in the pool.

Probing the Average Utilization of the Current Resource

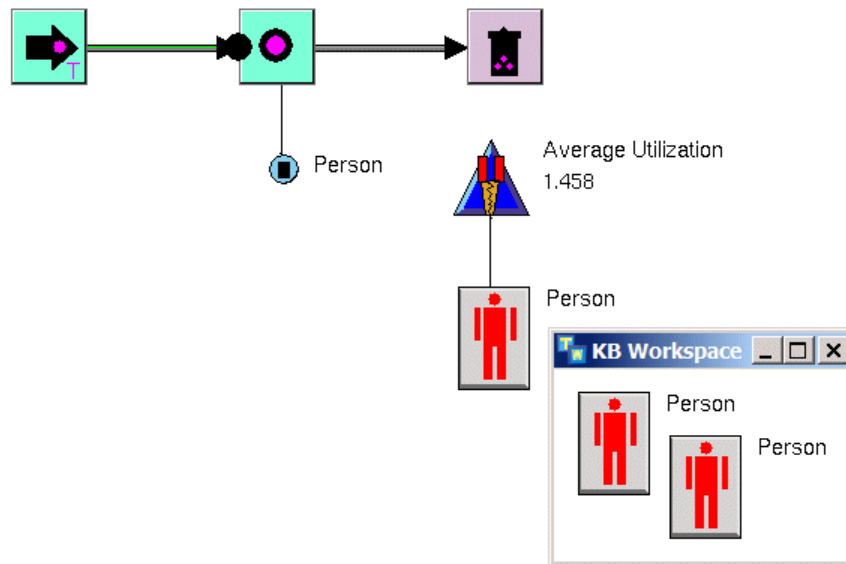
By probing a block that requires a resource, you obtain performance metrics about the currently allocated resource. Because the instrument probes the block at the end of processing its work objects, the probe updates its value at same time that the model deallocates the resource.

This figure shows a running model that probes the average utilization of the resource previously allocated by the task. The next time the probe updates, the sample value will be for the person resource currently allocated.



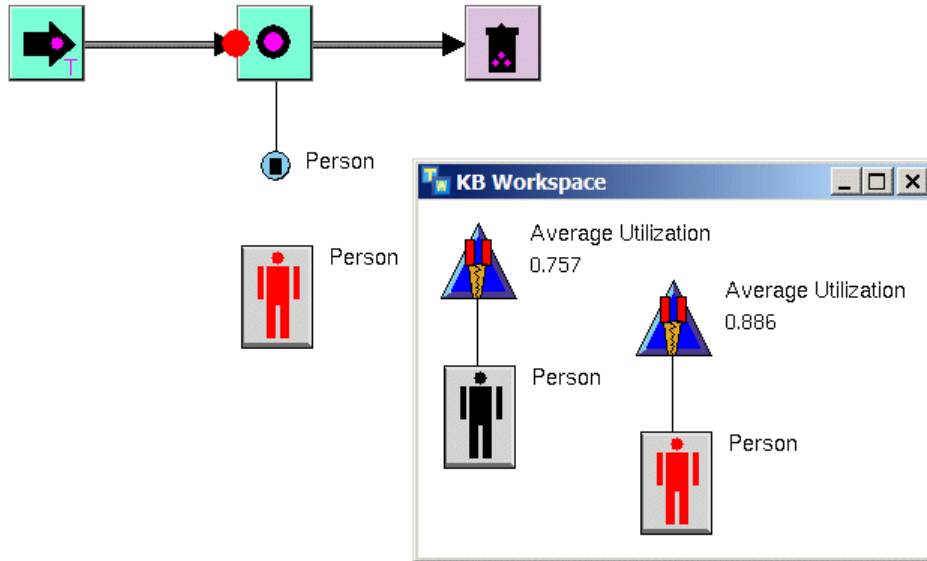
Probing the Average Utilization of the Top-Level Resource

By probing the top-level resource in a pool, you obtain performance metrics about the sum of all resources in the pool. This figure shows a running model that probes the average utilization of the top-level resource in a resource pool. Notice that the average utilization is greater than 1, which means that on average, the task uses more than one resource.



Probing the Average Utilization of a Resource in a Pool

By probing individual resources in a pool, you obtain performance metrics about each resource in the pool. This figure shows a running model that probes the average utilization of the each resource in the pool:



Charting Performance Metrics

One common output of a ReThink model is a visual representation, in the form of a chart, of the performance metrics that the probes in the model obtain. Charts provide visual feedback about the performance of the model, which you can use to perform “what-if” analysis.

Note If you have many charts in your model or charts with many data points, you should not update them too frequently; otherwise, the performance of your simulation can be adversely affected.

Note We recommend that you use remote charts for charting performance metrics, which allows you to create a wide variety of chart types that are visually appealing. You can also create simple line charts directly from probes, as described in [Creating a Chart Directly from a Probe](#). Currently, remote charts do not support all the functionality of the simple line charts; therefore, line charts continue to be supported. However, in a future release when remote charts are fully supported, classic charts may no longer be supported.

You place the remote charts associated with a model:

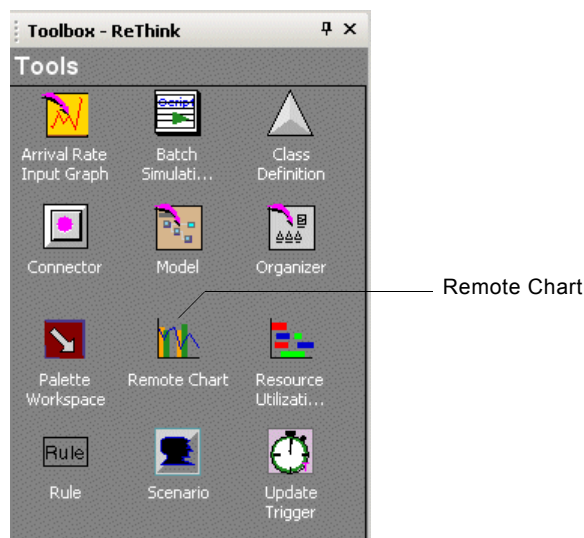
- On the same workspace as the model.
- On the detail of an Organizer associated with the model.

Creating a Remote Chart

You can create a remote chart to plot the history of one or more probes. Each time the probe receives new data, the chart updates. To chart probed data, first you must create a remote from each probe whose history you want to plot. You then configure the remote chart to specify which remotes to plot. You can also configure various properties of the chart, including labels and colors.

To create a remote chart:

- 1 Create one or more probes whose history you want to plot.
- 2 Choose Create Remote on each probe.
ReThink creates a remote, which keeps a history of the probed data.
- 3 Create a Remote Chart from the Tools palette of the ReThink toolbox:



- 4 Display the properties dialog for the Remote Chart, and choose the remotes you want to chart.
A list of all remotes associated with the scenario appears in the Remotes list.
- 5 Choose Show Chart to show the chart view.
- 6 Run the simulation to view the remote data in the chart.

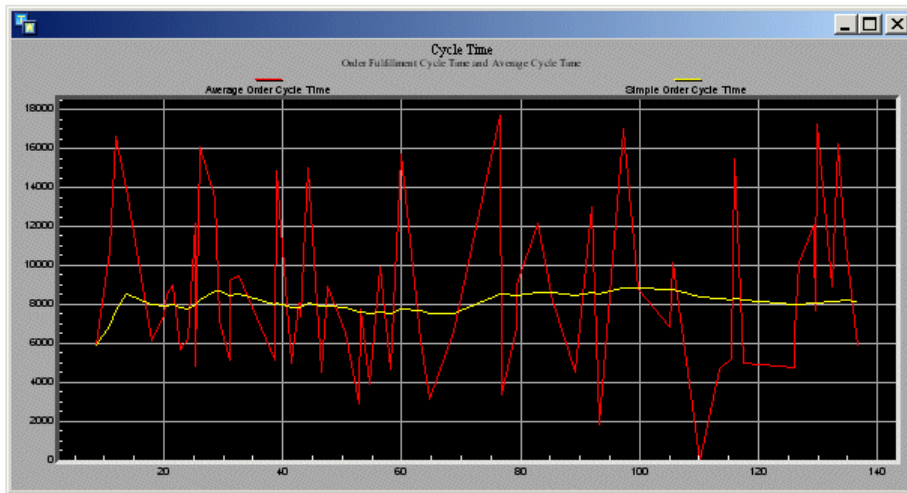
To configure chart views:

➔ To configure the chart, mouse right on the chart and choose Properties.

You can configure the title, subtitle, type, colors, grid, margins, axes, subsets, line annotations, and graph annotations.

For more information about configuring chart views, see the *G2 Reporting Engine User's Guide*.

Here is a remote chart that plots the history of two remotes:



Exporting Probed Data to a CSV File

You can export probed data to a `.csv` file. To do this, you connect an Excel Export tool to the probe whose data you want to export. If the probe to which the export tool is connected has an associated remote, the export tool exports the history of probed values contained in the remote. You can export data from one or multiple objects to the same sheet in Excel.

To determine when to export the probed data, you must associate the Excel Export tool with an Update Trigger tool or probe. You use an Update Trigger tool to export data at regular time intervals, based on simulation time. You use an Update Trigger probe to export data, based on model events. This means that you can collect the data, using the probe, and export the probed data, using the export tool, at different time intervals.

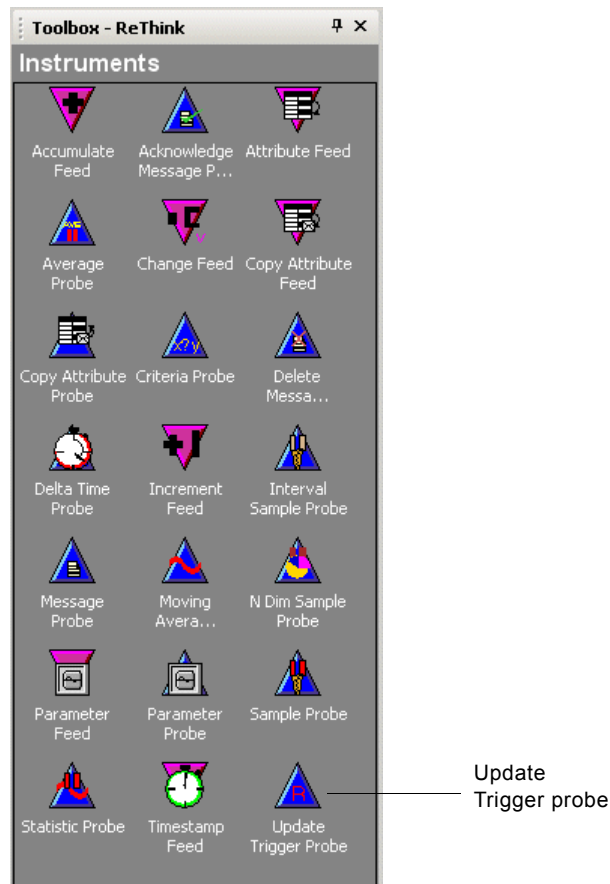
The first time the Excel Export tool updates, it creates the specified sheet in the currently connected Excel spreadsheet, then exports the data. Thereafter, it exports the data each time the export tool is triggered.

Exporting Probed Data Based on Model Events

To export probed data, based on model events, you use an Update Trigger probe.

To export probed data based on model events:

- 1 Create an Update Trigger probe from the Instruments palette of the ReThink toolbox:



- 2 Connect the Update Trigger probe to an object in the model that should trigger updates.

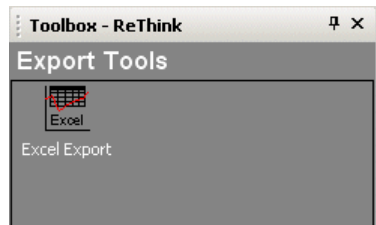
For example, to trigger updates when the block updates or when a work object arrives at the block, connect the probe to a block. To trigger updates when a resource is allocated, connect the probe to a resource.

- 3 Display the properties dialog for the Update Trigger probe and configure the Apply to Class Name to be the class that triggers updates.

The default value is `bpr-object`, which means the probe is configured to trigger updates when a work object arrives at the block to which the probe is attached.

If the probe is attached to a block, configure the class to be `bpr-block` to trigger updates when the block updates. If the probe is connected to a resource, configure the class to be `bpr-resource` to trigger updates when the resource is allocated.

- 4 Create and configure a probe whose data you want to export to Excel, and connect it to an object in the model whose attributes you want to probe and export.
- 5 Display the Export Tools palette of the ReThink toolbox:



- 6 Create an Export Excel tool and connect it to the probe whose data you want to export.
- 7 Choose the Choose Update Trigger menu choice on the Excel Export tool, then choose Select on the Update Trigger probe.

The Excel Export tool is now configured to export data each time the Update Trigger probe updates.

- 8 Display the properties dialog for the Excel Export tool and configure the Sheet Name to be the name of a worksheet within the current Excel spreadsheet in which to create the data.

By default, the export tool begins inserting data into the upper-left cell. If you are exporting data from multiple probes to the same spreadsheet, you must configure the starting column and row to use for inserting the data.

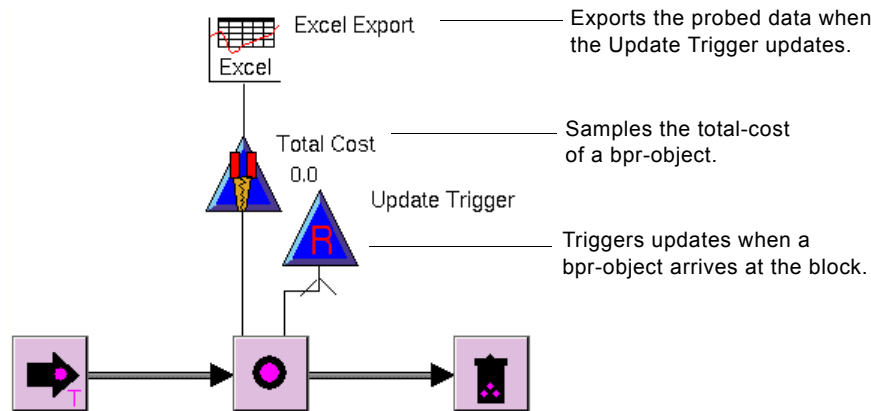
- 9 Configure the Column and Row to be the starting column and row in which to insert the data, as needed.

For example, if you are exporting data from two export tools, you might specify the Column and Row of one to be 0 and 0, and the other to be 1 and 0, respectively, which would result in two columns of data.

- 10 Configure the Excel File Name to be a `.csv` file to which to export the data.
- 11 Enable the Excel CSV File Reporting Enabled option to export data when the Update Trigger updates.
- 12 Enable the Date and Time as Durations option to export the update time as a time interval as opposed to seconds, as needed.

When you run the simulation, ReThink writes the data to this sheet each time the associated Export Excel tool updates, based on the Update Trigger probe.

This figure shows a model that exports the Total Cost of a work object when a work object arrives at the Task block. The Update Trigger probe is configured to trigger updates each time the work object arrives at the block. The Sample probe is configured to probe the total-cost of a bpr-object.

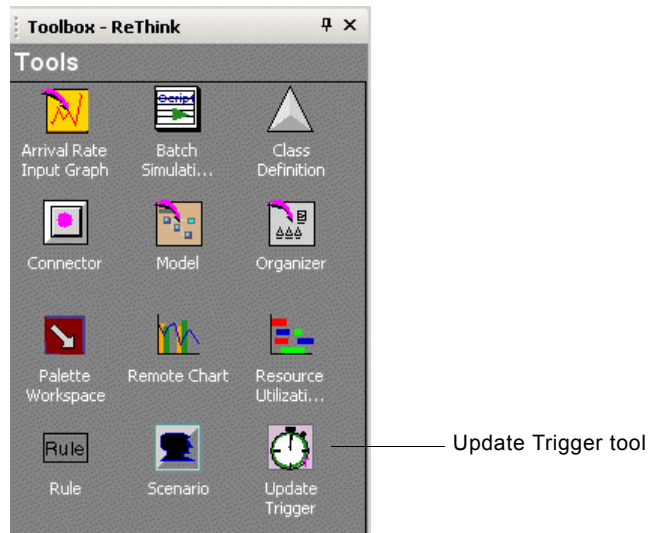


Exporting Probed Data at Regular Time Intervals

To export probed data at regular time intervals, you use an Update Trigger tool.

To export probed data to Excel at regular time intervals:

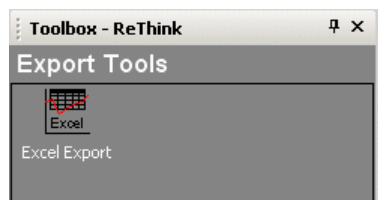
- 1 Create an Update Trigger tool from the Tools palette of the ReThink toolbox:



- 2 Display the properties dialog for the Update Trigger tool, click the Block tab, and configure the Start Time and End Time, as needed.

You configure the Start Time to probe the model and export the data after a delay.

- 3 Click the Duration tab and configure the Period to be the frequency with which the Update Trigger should trigger updates, for example, 1 day or 1 week.
- 4 Create, connect, and configure a probe whose data you want to export to Excel.
- 5 Display the Export Tools palette of the ReThink toolbox:



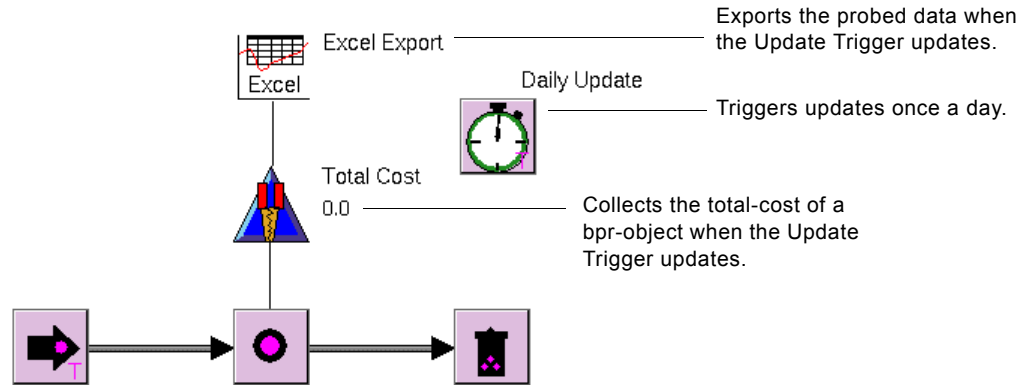
- 6 Create an Excel Export tool and connect it to the probe whose data you want to export.
- 7 Choose the Choose Update Trigger menu choice on the Excel Export tool, then choose Select on the Update Trigger tool.

The Excel Export tool is now configured to export data each time the Update Trigger tool is scheduled to update.
- 8 Display the properties dialog for the Excel Export tool and configure the Sheet Name to be the name of a worksheet within the current Excel spreadsheet in which to create the data.
- 9 Configure the Column and Row to be the starting column and row in which to insert the data, as needed.
- 10 Configure the Excel File Name to be a `.csv` file to which to export the data.
- 11 Enable the Excel CSV File Reporting Enabled option to export data when the Update Trigger updates.
- 12 Enable the Date and Time as Durations option to export the update time as a time interval as opposed to seconds, as needed.

For details on how to open the default spreadsheet and how Excel connects to the server, see [Creating Reports in Excel](#).

When you run the simulation, ReThink creates the specified worksheet in the currently connected Excel spreadsheet if it does not exist and writes the data to this sheet each time the associated Excel Export tool updates, based on the Update Trigger tool.

This figure shows a model that exports the Total Cost of a work object once a day. The Update Trigger tool is configured to trigger updates once a day. The Sample probe is configured to probe the total-cost of a bpr-object.



Exporting Historical Data

Instead of exporting a single probed value each time the Excel Export tool updates, you can export a history of probed values. To do this, you create a remote from the probe whose data you want to export, and the Excel Export tool exports the history.

You can export historical data, based on model events or at regular time intervals, depending on whether you use an Update Trigger probe or tool, respectively.

For details, see:

- [Exporting Probed Data Based on Model Events.](#)
- [Exporting Probed Data at Regular Time Intervals](#)

To export historical data:

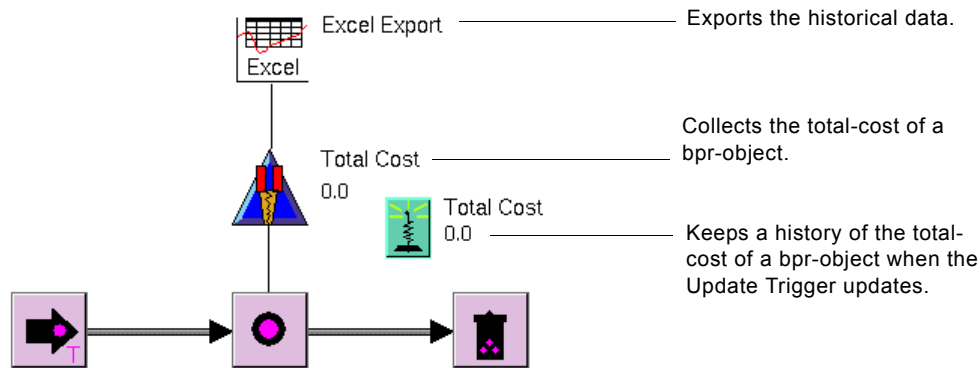
- 1 Create a model that uses an Excel Export tool to export probed data. You do not need an Update Trigger probe or tool to trigger updates.

For details, see:

- [Exporting Probed Data Based on Model Events.](#)
- [Exporting Probed Data at Regular Time Intervals.](#)

- 2 Choose Create Remote on the probe.

This figure shows a model that exports a history of the Total Cost of a work object once a day. The Update Trigger tool is configured to trigger updates once a day. The Sample probe is configured to probe the total-cost of a bpr-object. The remote was created from the Sample probe.



Feeding Values into the Model



One powerful use of ReThink is to perform “what-if” analysis on a model. For example, you might want to test:

- Different arrival rates of work objects into a process to see the effect on cycle time and performance.
- Different hourly wages for workers to see the effect on total cost of work objects.

To supply values to the model, you use feeds. Typically, you create sliders or type-in boxes from feeds to facilitate entering values into the model.

For a description of creating user interface objects for feeds, see [Creating User Interface Objects for Feeding Values](#).

You also use feeds to timestamp a work object as it passes through a particular task, to count the number of times that an object executes, or to accumulate values in an attribute of an object.

You can connect a feed to these types of objects to feed these types of values:

- Blocks to feed values into attributes of:
 - The block, for example, the Mean time of a source.
 - The current work object the block is processing, for example, a timestamp.

- The current resource the block is using, for example, the variable cost of the resource.
- The input or output path, in particular, its type.
- Resources to feed values directly into a particular resource.

Configuring the Feed

Once you have connected a feed to a block, you must configure:

- The class to which the feed is supplying a value.
- The target attribute whose value the feed is supplying.
- The **phase**, which determines when the feed evaluates relative to the attached block.

By default, feeds evaluate after the duration of the block is computed.

Depending on the Phase you choose, the feed evaluates at different times and can feed values into different types of objects, as this table describes:

This Phase...	Causes the feed to evaluate...	And means you can feed values into these types of objects...
Start	Before the connected block applies its duration to the simulation	Input work objects, resources, blocks, or input paths.
Stop	After the connected block applies its duration to the simulation	Output work objects, resources, blocks, or output paths.

Note If you are feeding values into the output object and the Phase is set to Stop, and if the output object is different from the input object, the instrument feeds its values into the output object just after the object is created.

Depending on the type of feed, you might also be required to supply additional attribute values, such as the source of the information for the feed.

You use a feed to update these types of attributes:

- A user-defined attribute of a work object or resource, for example, the arrival time of a work object at a particular task.
- A system-defined attribute of the model, for example, the mean time of a Source block.

Updating User-Defined Attributes of a Work Object

To update user-defined attributes of a work object, you must first create a class definition for the object and declare class-specific attributes for the attribute to update. You can view the value of the user-defined attribute on the User tab of the properties dialog for the work object.

For information about how to create a work object, see [Creating a New Class of Work Object](#).

To update a user-defined attribute of a work object:

- 1 Create a class definition that is a subclass of `bpr-object` with a class-specific attribute whose value the feed will update.

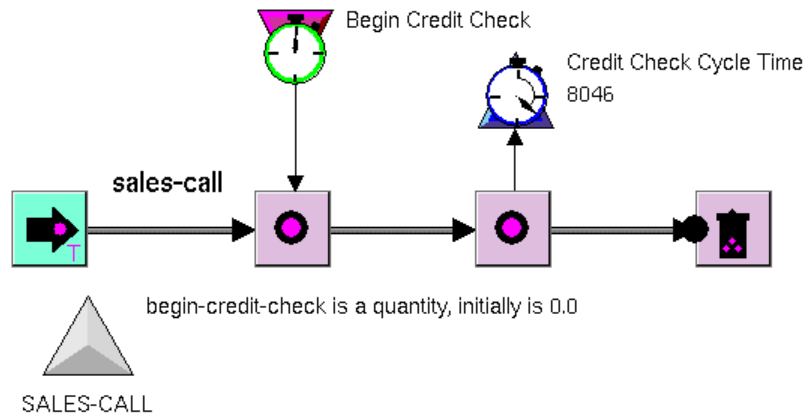
For example, if you are feeding a timestamp into a `sales-call` object at the beginning of a credit check subtask, the name of the class-specific attribute might be `begin-credit-check`.

- 2 Connect the feed to a block.
- 3 Display the properties dialog for the feed and click the General tab.
- 4 Configure the Apply to Class Name of the feed to refer to the name of the user-defined subclass of `bpr-object`.

Tip Be as specific as possible when you name the class.

- 5 Configure the Destination Attribute Name to specify the attribute of the class whose value the feed will update.
For example, if you are feeding a timestamp into the `begin-credit-check` attribute of a sales call, the Destination Attribute Name is `begin-credit-check`.
- 6 Configure the Phase to determine when the feed evaluates, and, when feeding values into work objects and paths, whether the feed applies to the input or output work object or path.

This simple model uses a Timestamp feed to supply a timestamp into the `begin-credit-check` attribute of a `sales-call`, which is a subclass of `bpr-object`. The model then computes the credit check cycle time by probing the `begin-credit-check` attribute of the `sales-call` with a Delta Time probe.



Timestamp Feed: Begin Credit Check

General | Animation

Label:

Comments:

Apply To Class Name:

Destination Attribute Name:

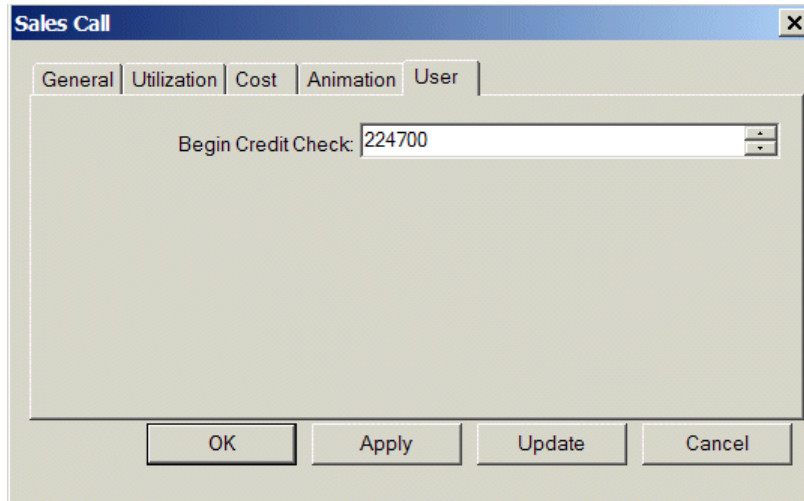
Phase: Start Stop

Counter:

Error:

OK Apply Update Cancel

Here is the User tab of the properties dialog of the sales call before the end of the simulation, which shows the value of the user-defined attribute, Begin Credit Check:



Updating System-Defined Attributes of the Model

You can update system-defined attributes of a block, resource, or path. To do this, you use a Change feed, which allows you to feed values, either directly or through a slider or type-in box, into the attribute. When feeding values directly, you can feed individual values, random values, unique IDs, or random values, based on a distribution.

For details, see [Change Feed](#).

To update system-defined attributes of the model:

- 1 Connect a Change feed to the object whose attribute you want to update.
- 2 Display the properties dialog for the feed and click the General tab.
- 3 Configure the Apply to Class Name of the Change feed to refer to one of the following classes or any subclass of these classes:

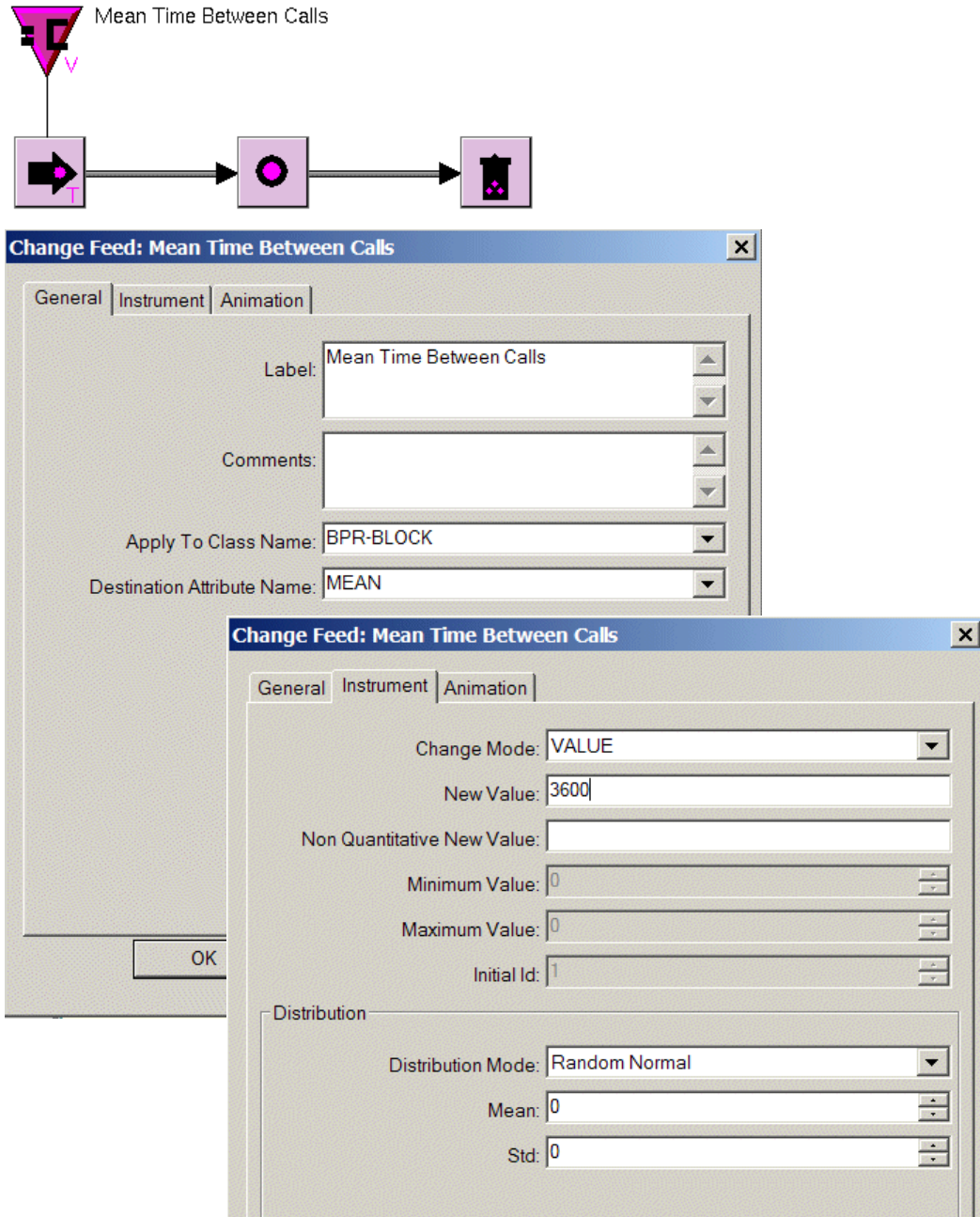
To feed values into a...	Configure Apply to Class Name as...
Block	bpr-block
Resource	bpr-resource
Path	bpr-path

Tip Be as specific as possible when specifying the class name.

- 4** Configure the Destination Attribute Name to specify the attribute of the class whose value the feed will update.

For example, if you are using a feed to update the Mean of a Source block, the Destination Attribute Name is `mean`.
- 5** Click the Instrument tab and configure the New Value to specify the value to feed into the system-defined attribute.

This simple model feeds the mean time into the Source block. The Apply to Class Name is bpr-block, which feeds values into the attached block, and the Destination Attribute Name is mean, which is the attribute of the block to update. The New Value is interpreted as seconds.



Creating User Interface Objects for Feeding Values

When you use a Change feed to update attributes of the model, you typically create a user interface object to feed the value.

For example, if you are feeding the time mean into a Source block to control the rate at which work objects flow into the model, you might create a slider or a type-in box for entering the value.

You place sliders and type-in boxes:

- On the same workspace as the model.
- On the detail of an organizer associated with the model.

You create a slider or type-in box directly from the feed, using a menu choice.

Note ReThink also creates a remote that is associated with the feed and places it on the feed's detail. You do not need to use the remote associated with a feed.

Creating a Slider

You create a slider for feeding numeric values into a Change feed. When you specify a number on the slider, the number is in units of seconds.

To configure the slider, you specify the minimum and maximum values for the slider and the initial value.

If you specify an initial value for the slider, the slider uses this default value when you initially run the simulation. If you do not specify an initial value, the slider uses the current slider value when you run the simulation.

To create a slider from a Change feed:

- 1 Display the properties dialog for the Change feed and click the Instrument tab.
- 2 Choose **Value** as the value of Change Mode.
- 3 Choose **Create Slider** on the Change feed.
- 4 Display the properties dialog for the slider.
- 5 Configure the **Minimum Value** and **Maximum Value** to set the minimum and maximum values for the slider.

The values must be in seconds.

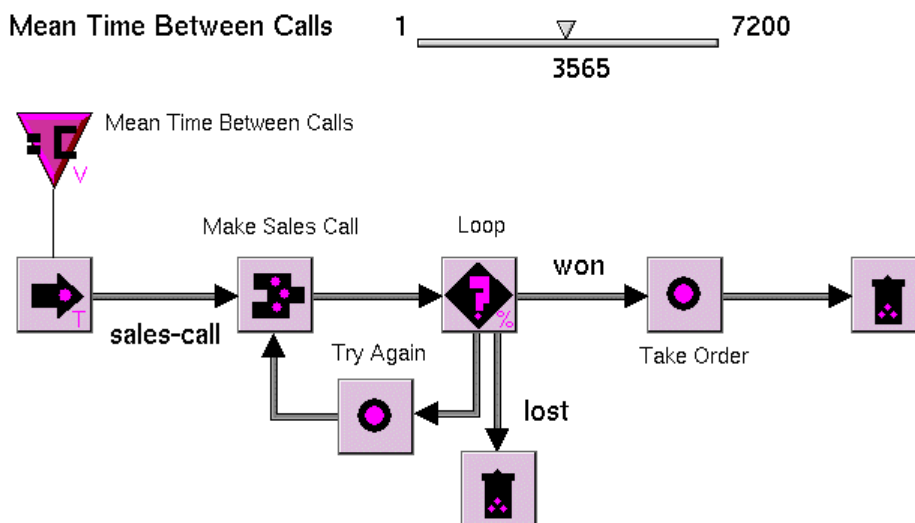
- 6 Configure the initial value in the Value on Activation.

If you specify a number, the value must be in seconds. ReThink uses this value as the default when you reset the model. If you specify a value of **none**, ReThink uses the current slider value as the default when you reset the model.

- 7 Using the right mouse button, move the slider to the desired location.
- 8 Set the value of the slider by moving the arrow.

When you run the simulation, the Change feed uses the value of the slider for the value of the attribute you specify.

This example provides a slider for entering the time mean of the Source block:



Creating a Type-in Box

You create a type-in box for feeding numeric, symbolic, or textual values. When you specify a number, the number is in units of seconds. The default value of the Change feed's New Value determines the data type of the type-in box.

To configure the type-in box, you specify the initial value. If you specify an initial value for the type-in box, the type-in box uses this default value when you initially run the simulation. If you do not specify an initial value, the type-in box uses the current type-in box value when you run the simulation.

To create a type-in box from a Change feed:

- 1 Display the properties dialog for the Change feed and click the Instrument tab.
- 2 Choose **Value** as the value of Change Mode.
- 3 Configure the New Value to be the default value for the feed.

- 4 Choose Create Type In on the feed.

The type-in box accepts values whose type corresponds with the default value you specified.

- 5 Display the properties of the type-in box.

- 6 Configure the Value on Activation to be the initial value.

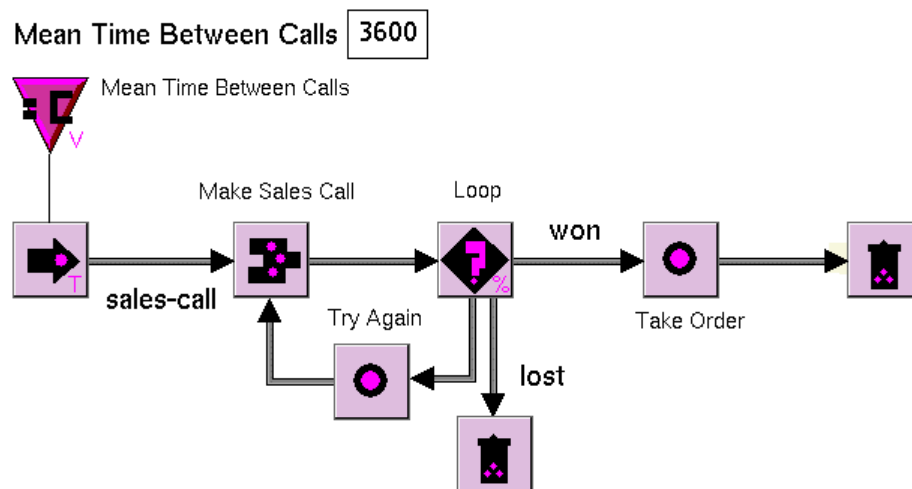
If you specify a number, the value must be in seconds. ReThink uses this value as the default when you reset the model. If you specify a value of *none*, ReThink uses the current type-in box value as the default when you reset the model.

- 7 Using the right mouse button, move the type-in box to the desired location.

- 8 Enter a value in the type-in box.

When you run the simulation, the Change feed uses the value of the type-in box for the value of the attribute you specify.

Here is the same example with a type-in box for entering the time mean of the Source block:



Creating a Chart Directly from a Probe

You can create a chart directly from any type of probe except the Parameter probe. To view the plot, you must update the chart, using one of several techniques.

Note This technique for creating charts has been superseded by remote charts.

When you create a chart from a probe, ReThink creates a remote, which keeps a history of the values of the probe. You use remotes to:

- Display multiple plots on a single chart.
- Configure various aspects of the chart, such as the line color of plot, the background color of the chart, or the x and y axes.
- Perform computation on the probed values.

The remote shows the current value for the probe, which it stores in its Latest Value metric. The default label for the remote also includes the label for the probe and the type of probe.

The chart does not begin plotting automatically. For information on updating the chart, see [Updating Charts](#).

Note If you have many charts in your model or charts with many data points, you should not update them too frequently; otherwise, the performance of your simulation can be adversely affected.

Creating a Chart

To create a chart from a probe:

- 1 Choose Create Chart on a probe.

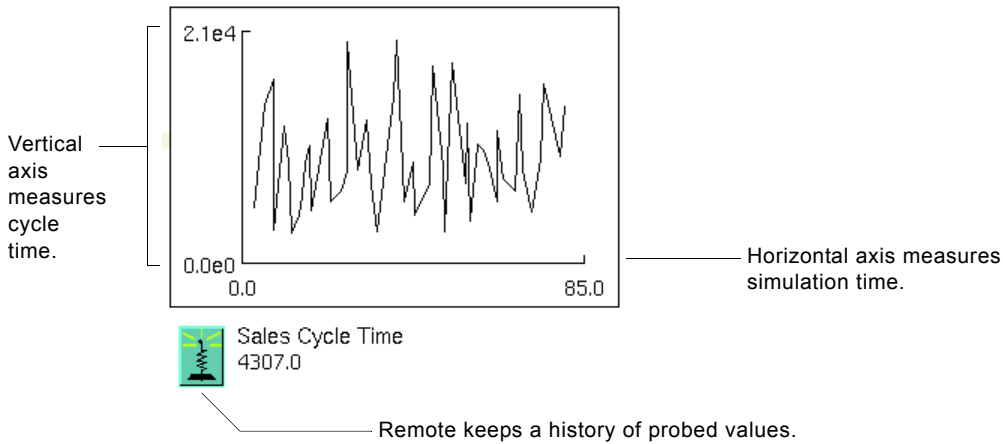
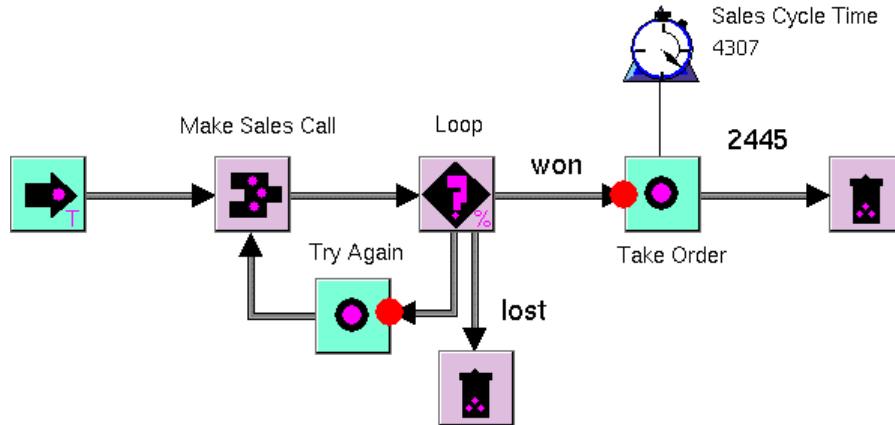
ReThink creates a chart and a remote and places them on top of the probe. The x axis is the amount of time that has passed in the simulation, and the y axis is the probed value. Depending on what the instrument is probing, the vertical axis is measured in seconds or in the time unit of the probed value, for example, total cost or average utilization.

- 2 Move the chart to a new location on the workspace to expose the remote.

The remote appears just above the probe.

- 3 Move the probe near the chart, if desired.

Here is a model of a sales process with a probe that computes the sales cycle time from the creation of the Sales Call work object to the Take Order task. In this example, the y axis is the number of seconds that correspond to the sales cycle time, or the delta time.



Updating Charts

If your model contains many charts or charts with many data points, the performance of your simulation can be significantly degraded by redrawing the chart each time a new value is added. Therefore, by default, the scenario does not update charts automatically.

You can update charts manually, using a menu choice or an action button, or you can create rule to update charts on a regular basis.

For information on configuring the update behavior of charts, see [Configuring the Computation Behavior](#).

Updating Charts Manually

To update charts manually:

→ Choose Update Chart on the chart.

or

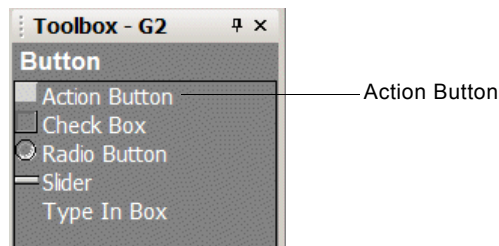
→ Choose Update Chart on the remote associated with the chart.

Using an Action Button to Update Charts

You can create an action button that updates charts in the model when you click the button.

To create an action button that updates charts:

1 Choose View > Toolbox - G2 and click the Buttons tab:



2 Select an Action Button and place it on the workspace that contains the chart you want to update.

3 Display the properties dialog for the action button.

4 Configure the Label to be the name of the button, surrounded by quotation marks, for example, "Update Charts".

5 Configure the Action to specify the following action statement:

update every chart

Tip If you want to create an action button that updates every chart on a particular workspace, rather than every chart in the model, use this action statement:
update every chart upon this workspace.

You can now click the action button to update every chart in the model.

Here is an action button with its table, which updates every chart in the model:

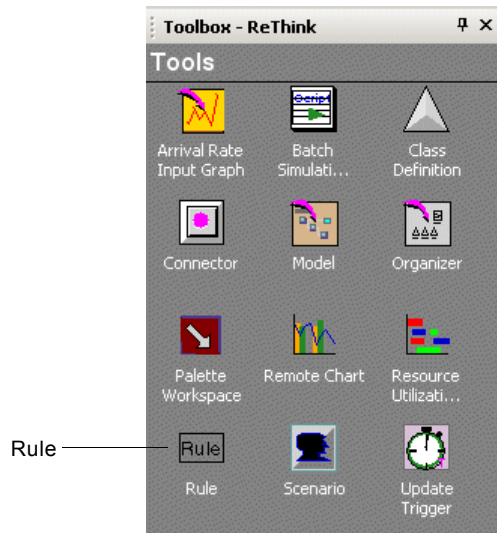
Update Charts	
UUID	"d1cdf30884d11dab4ef00095b4ac3d7"
Notes	ACTION-BUTTON-XXX-227: OK
Item configuration	none
Names	none
Label	"Update Charts"
Action	update every chart
Action priority	2

Using a Rule to Update Charts

You can create a rule that updates charts in the model at a periodic interval.

To create a rule that updates charts:

- 1 Display the Tools palette of the ReThink toolbox:



- 2 Select a Rule and place it on a workspace.
Typically, you place a rule on the detail of an Organizer.
- 3 Double-click the rule to display the text editor.
- 4 Enter the following rule:
unconditionally update every chart

Tip If you want to create a rule that updates every chart on a particular workspace, rather than every chart in the model, use this rule: **unconditionally update every chart**.

- 5 Display the properties dialog for the rule.
- 6 Configure the Scan Interval to be the time interval at which you want to update every chart, for example, 10 seconds or 1 minute.

Every chart now updates according to the scan interval you specify.

Here is a rule with its properties table, which updates every chart in the model every ten seconds:

unconditionally update every chart	
UUID	"87d19ea3884e11dab4ef00095b4ac3d7"
Options	invocable via backward chaining, invocable via forward chaining, may cause data seeking, may cause forward chaining
Notes	RULE-XXX-228: OK
Authors	nrs (18 Jan 2006 1:19 p.m.)
Change log	0 entries
Item configuration	none
Names	none
Tracing and breakpoints	default
unconditionally update every chart	
Scan interval	2 seconds
Focal classes	none
Focal objects	none
Categories	none
Rule priority	6
Depth first backward chaining precedence	1
Timeout for rule completion	use default

Configuring the Colors and Data Points of the Chart



You can configure the line color of the plot in the chart and the maximum number of data points to display at one time. To do this, you configure the associated remote.

To configure the colors of the chart:

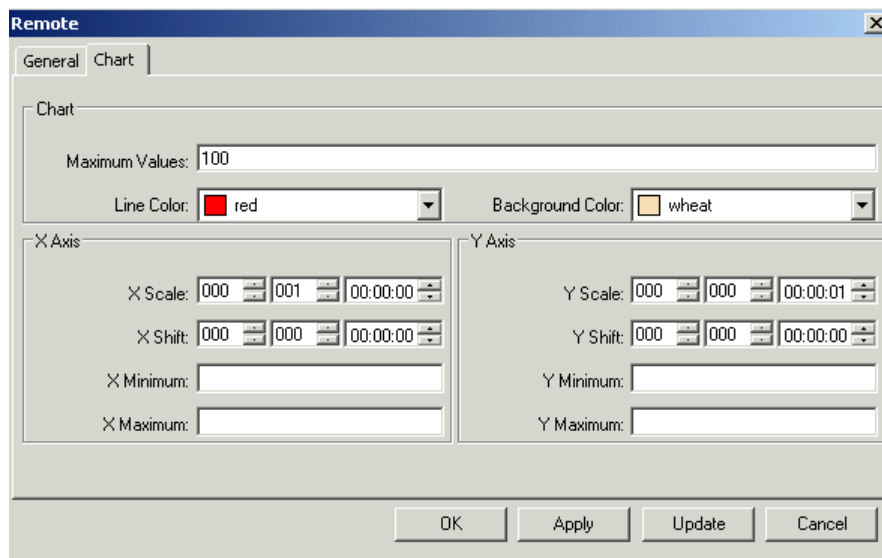
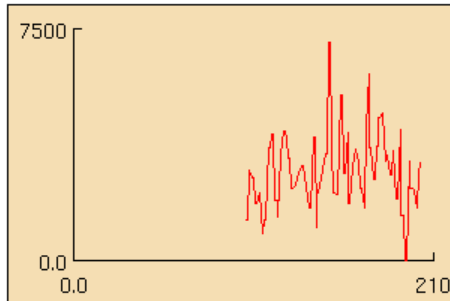
- 1 Display the properties dialog for the remote associated with the probe whose values you are plotting.
- 2 Click the Chart tab and edit the Line Color to specify the color of the plot.
- 3 Configure the Background Color to specify the color of the background of the chart.

The icon for the remote shows the line color of the plot.

To configure the maximum number of points to plot:

- 1 Display the properties dialog for the remote associated with the probe whose maximum values you want to configure.
- 2 Click the Chart tab and edit the Maximum Values to be the maximum number of data points to show on the chart at any one time.

Here is a chart and associated properties dialog for a Delta Time probe, whose Line Color is red, whose background color is wheat, and which shows only the last 100 data points:



Configuring the Axes of the Chart



You can configure the scale and offset that the chart uses to compute data, as well as the minimum and maximum values along the x and y axes. To do this, you configure the associated remote.

By default, the chart plots data in hours along the x axis. Depending on the durations that your model uses, you might want to use a different time scale as the default.

To configure the time scale of the x axis:

- 1 Display the properties dialog for the remote associated with the probe whose x axis you want to configure.
- 2 Click the Chart tab.
- 3 Configure the X Scale to be the time unit of the x axis.

The horizontal axis of the chart now shows the number of minutes, hours, days, or weeks that have passed since the start of the simulation.

When you are plotting time data, the chart plots the y value in seconds, by default. Similarly, these values can be difficult to interpret, depending on what you are plotting.

- 4 Configure the Y Scale as a number by which to divide the current plotted value.

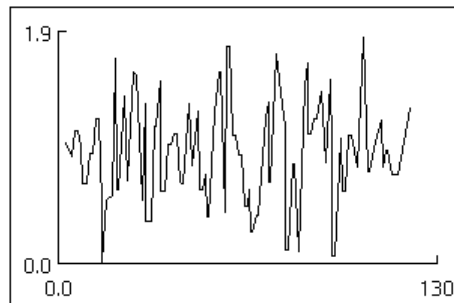
The chart divides the current probed value along the y axis by the Y Scale you specify.

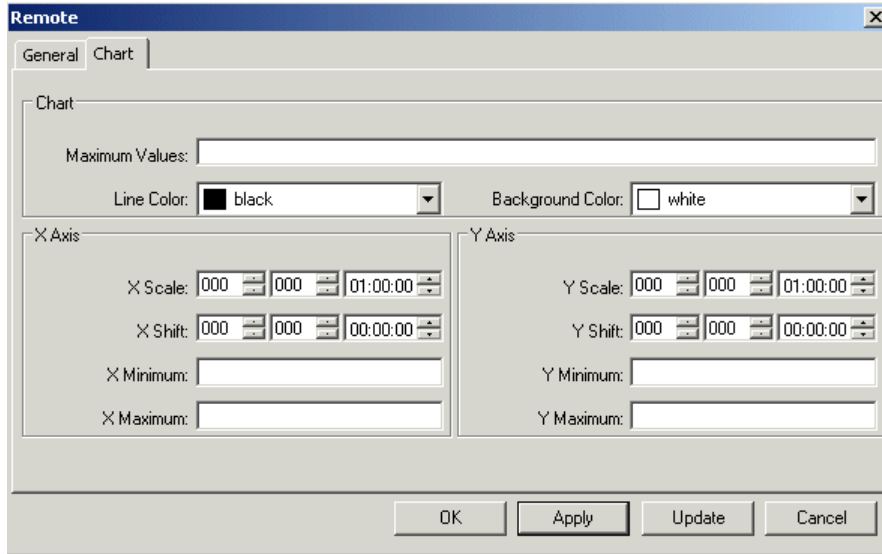
For example, if you are probing time values and you want to plot hourly values, specify the Y Scale as 1 hour. If you want to plot values in minutes, specify Y Scale as 1 minute.

Note The associated remote computes its value in seconds.

For information on how to scale the value of the remote, see [Scaling the Current Value of a Remote](#).

For example, here is a plot for a Delta Time probe, whose X Scale is 1 hour and whose Y Scale is 1 hour. The chart has been plotting for approximately 130 hours along the x axis, and the maximum delta time plotted so far along the y axis is 1.9 hours.





If you are plotting values other than time values, for example, the number of activities or the cost of an object, you might or might not want to scale the y axis. For example, suppose you are probing the Average Utilization of the top-level resource of a pool with two resources. This probed value yields the sum of the Average Utilization values of the two resources in the pool. By setting the Y Scale to 2, you can plot the average of this sum on the chart.

To configure the data offsets for the plotted values:

- 1 Display the properties dialog for the remote.
- 2 Click the Chart tab and configure the X Shift and Y Shift to be a number to which the current x or y value, respectively, is added.

The plotted values are offset by the specified number.

To configure the minimum and maximum values along an axis:

- 1 Display the properties dialog for the remote.
- 2 Click the Chart tab and configure the X Minimum or Y Maximum to be the minimum or maximum value along the x or y axis, respectively.

Any values that fall outside the specified range do not appear on the chart.

Plotting Multiple Values on the Same Chart



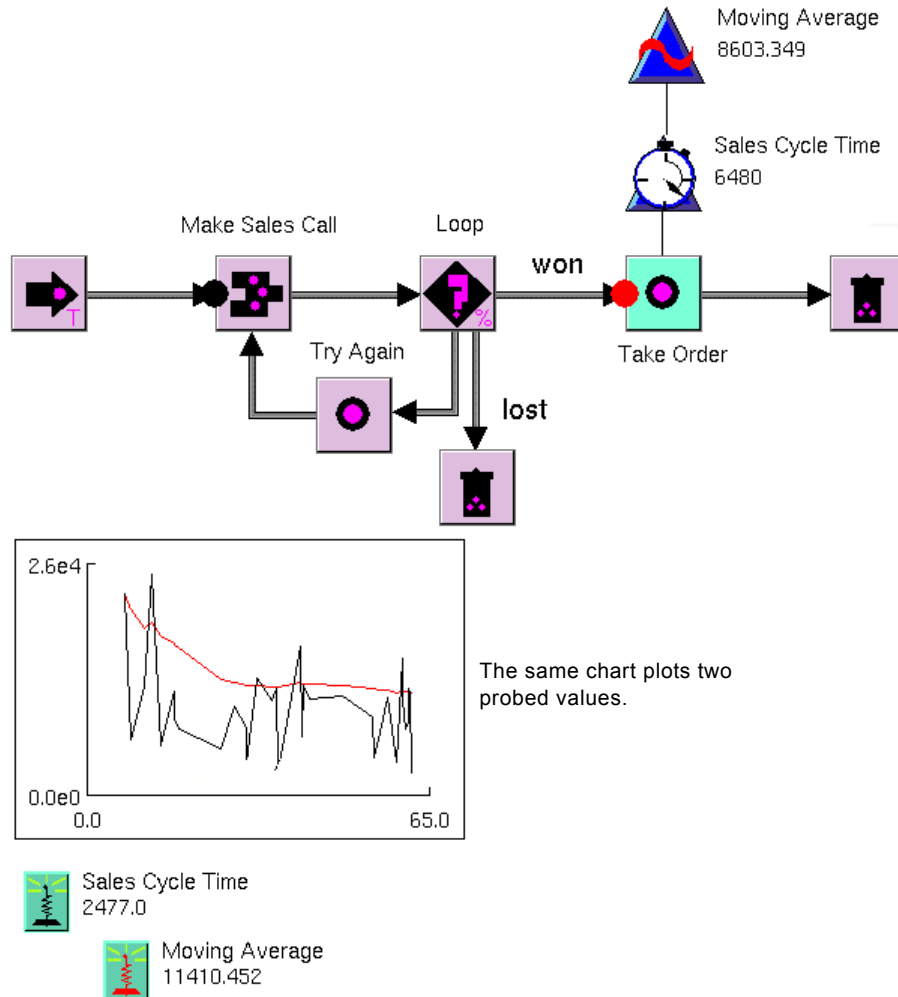
Often you need to plot multiple values on the same chart. For example, when you probe the sales cycle time of a process, you might also want to probe and plot a moving average of the sales cycle time on the same chart. To do this, you create a remote from a probe and associate the remote with an existing chart.

To plot multiple values on the same chart:

- 1** Create, connect, and configure two probes whose values you want to plot on the same chart.
- 2** Choose Create Chart on one of the probes to create a chart that plots a single probed value.
- 3** Choose Create Remote on the second probe to create a remote only.
ReThink displays the remote above the probe, without creating the associated chart.
- 4** Choose Add Remote on the chart, then choose Select on the remote associated with the second probe to add it to the chart.
An indicator arrow appears next to the remote you select. The additional values begin plotting immediately.
- 5** To distinguish the plots from one another using color, configure the Line Color on the Chart tab of the properties dialog for the remote.

ReThink plots the values of both the original and the new remote on the same chart.

This example shows a running sales process model with a Moving Average probe and a Delta Time probe, and the associated remotes and chart. The chart plots both the sales cycle time and average sales cycle time.



Scaling the Current Value of a Remote



You might want to scale the current value of a remote, for example, to compute the value in hours instead of seconds or to average the probed value.

To scale the current value of a remote:

- ➔ Display the properties dialog for the remote and configure the Scaling Divisor to be a number by which the current probed value is divided.

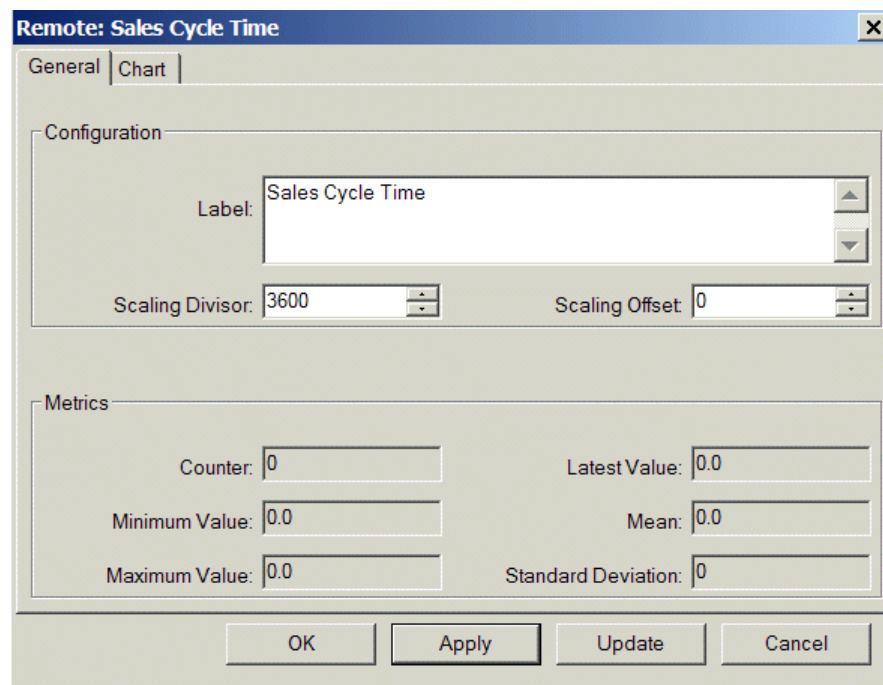
The probed values of the remote are now scaled by the Scaling Divisor.

Note The Scaling Divisor of a remote has no effect on the probed or plotted value; it only affects the value of the remote.

When you edit the Scaling Divisor of the remote, you typically also edit the Scale of the y axis to match the scaled values. For details, see [Configuring the Axes of the Chart](#).

Note If you set the Scaling Divisor to be a number that is too large, the ReThink model might produce somewhat inaccurate results when it computes very large numbers. This inaccuracy is due to round-off errors when computing large numbers, which is unavoidable.

For example, here is the properties dialog for the remote associated with a Delta Time probe that divides the current probed value by 3600, which causes the remote to compute the delta time in hours instead of seconds:



For another example, suppose you are probing the average utilization of a resource by connecting a Sample probe to a top-level resource with three resources in the pool. By default, the remote obtains values for the *sum* of all the resources in the pool. Suppose you want the remote to compute the *average* of all the average utilizations of the resources in the pool. To do this, you would edit the Scaling Divisor of the remote to be 3 to compute an average.

Offsetting the Current Value of a Remote

You might want to offset the current value computed by a remote.

To offset the current value of a remote:

- ➔ Display the properties dialog for the remote and configure the Scaling Offset to be a number to which the current probed value is added.

The probed values of the remote are now offset by the Scaling Offset.

Note The Scaling Offset of a remote has no effect on the probed or plotted value; it only affects the value of the remote.

When you edit the Scaling Offset of the remote, you typically also edit the Shift of the y axis of the remote to match the scaled values. For details, see [Configuring the Axes of the Chart](#).

Showing Metrics for a Remote

The remote computes a number of time-weighted metrics, for example, a time-weighted mean and a time-weighted standard deviation. The default time unit on which these metrics are based is one hour.

To view metrics for a remote:

- ➔ Display the properties dialog for the remote and click the General tab.

Here is the properties dialog for the remote associated with a Delta Time probe:

Remote: Sales Cycle Time

General | Chart

Configuration

Label: Sales Cycle Time

Scaling Divisor: 1 Scaling Offset: 0

Metrics

Counter: 0 Latest Value: 8907.0

Minimum Value: 0.0 Mean: 8907.0

Maximum Value: 19082.0 Standard Deviation: 0

OK Apply Update Cancel

Customizing Instruments

All ReThink instrument definitions are available to you for viewing and customizing.

For detailed information about how to customize instruments, see the *Customizing ReThink User's Guide*.

Using Resources

Describes how to use resources and temporal constraints to constrain your model and compute cost and utilization statistics.

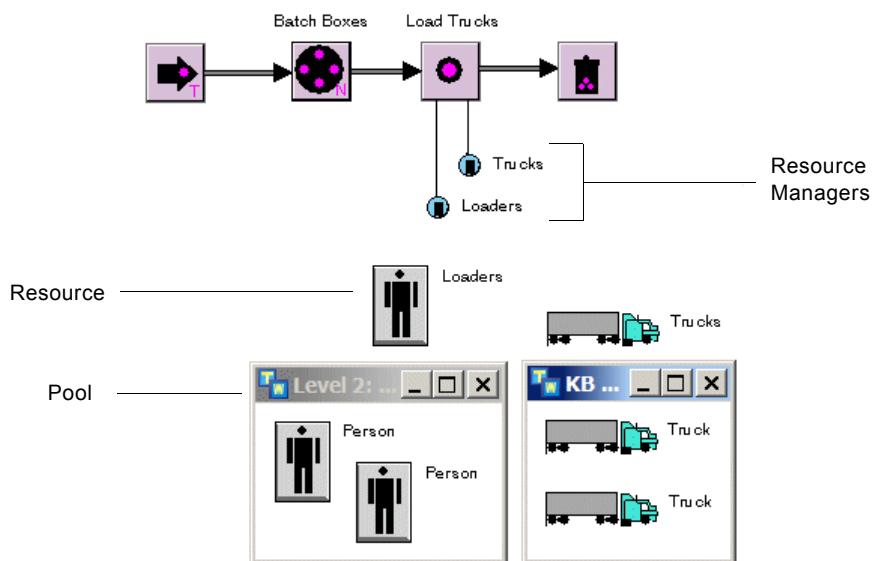
Introduction	262
Using Resources to Constrain the Model	264
Creating a Pool of Resources	271
Computing Utilization and Duration Metrics	275
Working with Resource Costs	281
Allocating Multiple Resources to a Task	285
Allocating the Same Pool to Multiple Tasks	286
Sharing the Same Resource in Multiple Pools	287
Allocating Partial and Multiple Resources	289
Allocating the Same Resource for Multiple Sequential Steps	300
Choosing Particular Resources from a Pool	303
Allocating Resources Associated to Work Objects	312
Allocating the Same Resource to Different Blocks Based on Priority	317
Creating Resources with Different Efficiency Factors	319
Showing the Metrics of Resources	322
Constraining the Availability of Resources	325
Configuring the Animation of Resources	340
Probing the Performance of Resources	341
Populating Resource Pools Dynamically	342
Customizing Resources	343

Introduction

One important element of process modeling involves modeling the resources that individual activities require. By modeling resource requirements, you can analyze the overall performance of your process.

For example, in a sales process model, the activity of processing an order might require a clerk; in a distribution process model, a delivery task might require a truck; and in a financial model, a task might require capital.

In this model, the Load Trucks task requires two different sets of resources, people and trucks.



You use these types of objects to model resource requirements:

- **Resources**, which constrain the amount of work that a model can process based on available personnel, equipment, or cash.
- **Resource pools**, which represent a set of available resources on the detail of a resource.
- **Resource managers**, which associate a particular resource with a particular task in the model.
- **Temporal constraints**, which constrain the availability of resources.

By default, a Resource Manager **allocates** the first available resource from the pool when the associated block begins processing, which makes the resource unavailable for other activities. When the block has finished processing, the Resource Manager **deallocates** the resource, which makes it available for other block activities.

You can use resources and Resource Managers to constrain a ReThink model in various other ways:

- A task can require one or more resources. For example, in a distribution process model, a delivery task might require a truck and a driver.
- A task can require one or more different resources. For example, a delivery task might require a pool of available trucks and drivers. The more resources that are available, the more concurrent activities the task can perform; the fewer resources that are available, the more the resources constrain the model.
- A task can require a partial resource, such as half of a computer.
- A task can require multiple identical resources, for example, \$50.
- Two tasks can share the same resource, for example, a person might work half-time for two different departments.
- The same resource might be allocated for several sequential steps in a process, for example, a loading task and a delivery task.
- A Resource Manager can allocate resources based on cost, utilization, or priority.

You can assign fixed and variable costs to resources, which you use to compute costs in a model. Resources also compute various metrics that report on their performance, such the average amount of time that the resource is allocated over the course of the simulation and its overall cost. You obtain key performance metrics about your model by probing the resources that the model requires.

You can constrain the amount of time that a resource is available by using temporal constraints. You can constrain the availability of resources on a monthly, weekly, and hourly basis.

In addition to using resources to constrain a model, you can use resources to model database operations or inventory fluctuations by storing objects in a resource pool and retrieving objects from the pool.

Using Resources to Constrain the Model



Person To constrain the model by using resources, you create a resource from the Resources palette of the ReThink toolbox, then associate the resource with the task, using a Resource Manager. When you run the simulation and too few resources are available to process the work that flows into the block, work backs up on the input path to the task requiring the resource.

When you create a resource, you place the resource:

- On the same workspace as the blocks.
- On the detail of an organizer that contains all the resources for a model.
- On the detail of a resource pool.

For large models, you typically place all resources on the detail of an organizer.

Note When you place resources on the detail of an organizer, the organizer must be on the same workspace as the scenario or on a detail; the organizer should not be at the same level as the model.

For more information about using organizers, see [Creating an Organizer](#).

Creating a Resource

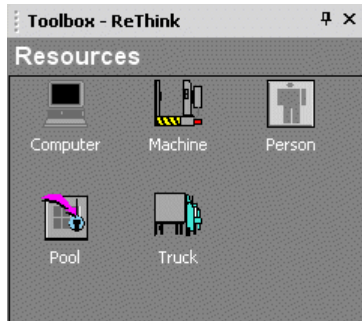
The Resources palette of the ReThink toolbox provides several default types of resources, depending on the type of model. The first step is to create a resource from the toolbox. You can also copy existing resources on a workspace, in which case ReThink copies the selected resource, including all of its configured attributes.

The toolbox also contains a Pool resource, which you can use when the resource does not fall into any other category. The only difference between a Pool resource and the other resources on this tab is the fact that a Pool resource defines a detail by default, whereas the other resources do not. You can add a detail to any resource on this tab.

For information about creating resource pools, see [Creating a Pool of Resources](#).

To create a resource:

- 1 Display the Resources palette of the ReThink toolbox:



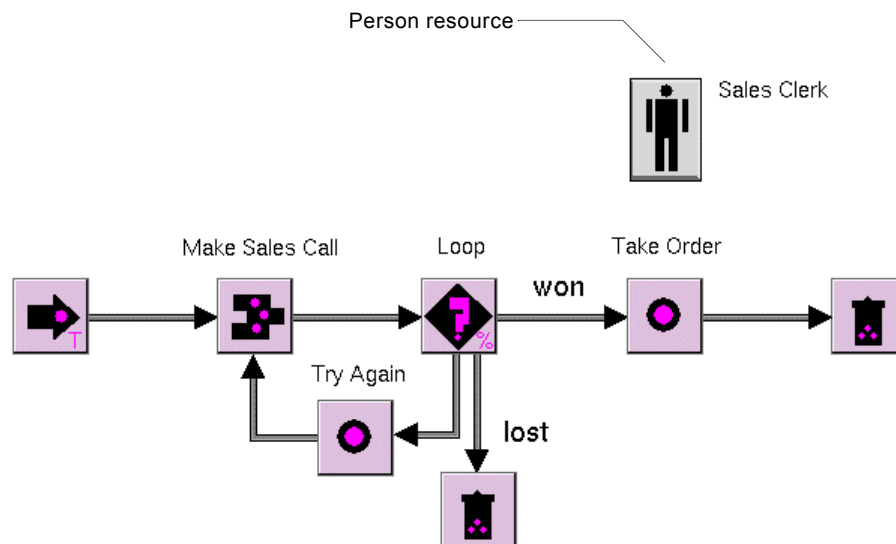
- 2 Select a resource and click to place it on the workspace.

If you place a resource on the same workspace as the model, you typically position it near the block that requires it.

- 3 Display the properties for the resource and configure the Label, as needed.

For information on how to display other attributes of the resource with the resource, see [Using Attribute Displays](#).

In the following sales process model, a sales clerk resource is positioned above the Take Order task:



So far, the model does not yet use the resource. To do this, you create a Resource Manager and attach it to the task that requires the resource to allocate the resource to a task.



Person

Allocating a Resource to a Task

You use Resource Managers to choose, allocate, and deallocate resources for a task. You create Resource Managers directly from a resource, using a menu choice.

By default, the Resource Manager allocates a single resource at the beginning of each activity of the block and deallocates the resource at the end of each activity.

You can change the default behavior of a Resource Manager to allocate the resource at the beginning of one task and deallocate the resource at the end of a different task. You can also specify the amount of the resource that the manager allocates to a task. For example, you might want the manager to allocate only a half of the resource or two resources.

For more information, see [Allocating the Same Resource for Multiple Sequential Steps](#) and [Allocating Partial and Multiple Resources](#).

To allocate a resource to a task, using a Resource Manager:

- 1 Choose Create Manager on a resource.

ReThink creates a Resource Manager and places it just above the resource on the workspace.

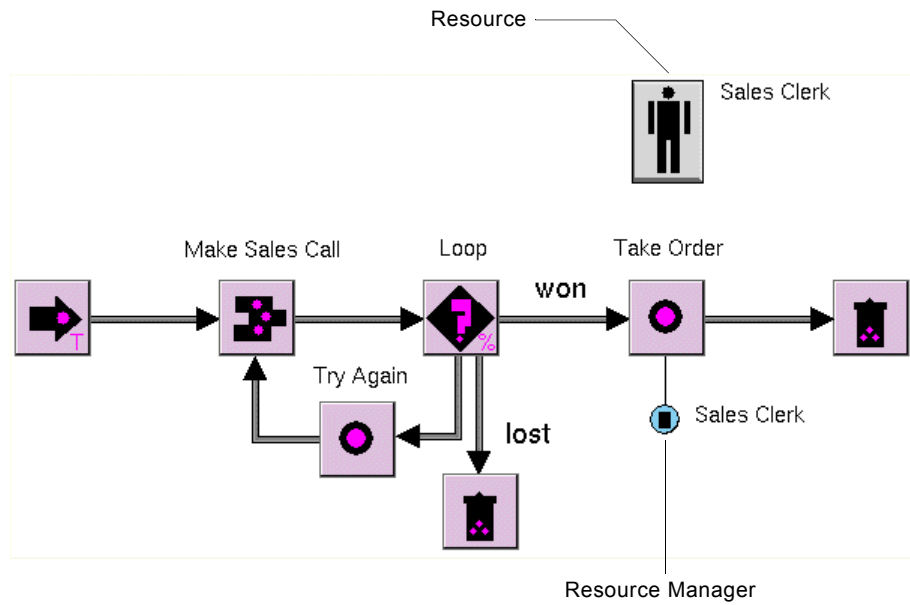
The label for the Resource Manager corresponds to the label for the resource. If you edit the resource label, the label associated with the Resource Manager changes, assuming the associated Scenario is active.

The wire leading out of the Resource Manager serves as a connection path.

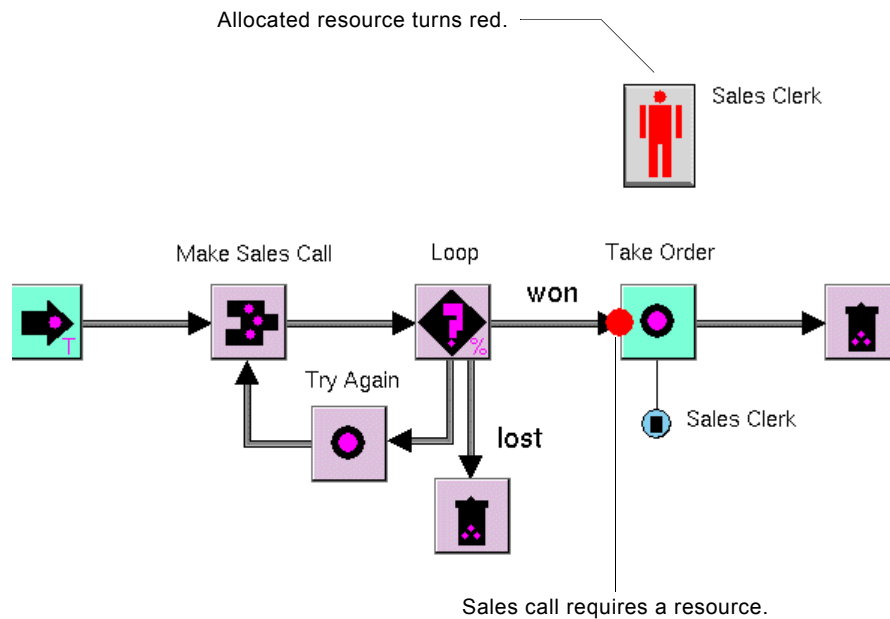
- 2 Position the Resource Manager just below the block that requires the associated resource and connect the manager to the block.

Tip You can reconfigure the connection between the Resource Manager and the block by dragging the Resource Manager to a new location along the edge of the block.

In this example, the Take Order task of the sales process model now requires the sales clerk resource to process orders:



When the model is running and a work object arrives at the Take Order task, the resource turns red, indicating that it is currently allocated by the task:



Identifying the Associated Resource

Once you create a Resource Manager from a resource, the Resource Manager adds the Show Resource menu choice to its menu, which identifies the resource from which the manager allocates and deallocates resources.

To show the resource associated with a manager:

→ Choose Show Resource on the Resource Manager.

ReThink places an indicator arrow next to the associated resource.

Associating the Manager with a Different Resource

When you first create a Resource Manager, ReThink automatically associates the manager with the resource from which it was created. You might want the Resource Manager to allocate resources from a different pool after you have already connected the manager to the task.

To associate the manager with a different resource:

- 1 Click the Resource Manager and choose the Choose Resource menu choice.
- 2 Click a new resource to display its menu and choose Select.

ReThink places an indicator arrow next to the new resource.

The manager allocates resources from the new resource instead of the original resource.

Tip You should make the resource and manager labels match so you can easily identify which resource corresponds to which manager.

Replacing Resources

You can drop a new resource on top of an existing resource to replace the resource. Any associated Resource Managers point to the new resource. The new resource copies the configuration information from the existing resource, including its label and its detail. For example, if the Cost per Use of the original resource is 10, the Cost per Use of the new resource will also be 10.

Showing Work Backups Due to Resource Constraints

When a task does not have any resource constraints, it can process any number of work objects concurrently. The number of concurrent activities depends on frequency with which work objects flow into the task. When a task requires a resource, however, it can only process as many work objects as there are available

resources, up to the maximum activities of the block. If only one resource exists, for example, it can only process one work object at a time. This assumes the default Utilization of the Resource Manager, the default Maximum Utilization of the resource, and the default Maximum Activities of the block.

If work flows into a block more frequently than the block has resources to handle, work objects back up on the input path of the block, and the input path turns green.

The input path to the block keeps track of the work objects that are waiting for resources. By default, ReThink allocates the resource to the first work object in the queue.

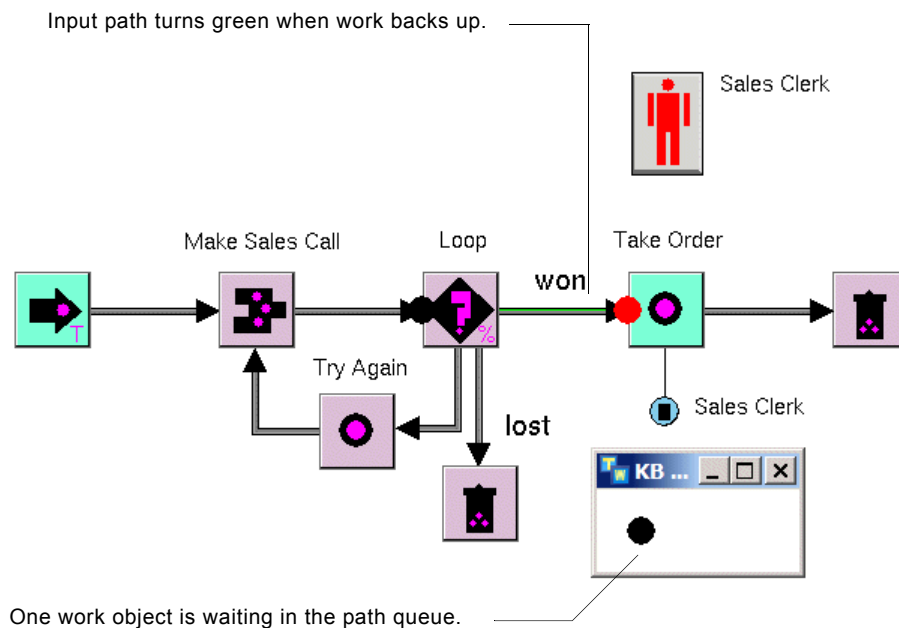
You can visualize work backups on the input path to a block to see how many objects are waiting for resources.

To visualize work backups due to resource constraints:

- ➔ Choose Snapshot Queue on the input path to a block that has turned green due to resource constraints.

ReThink displays a temporary workspace that shows the work objects waiting for resources as of the moment you choose Snapshot Queue. ReThink dynamically removes work objects from this workspace when a resource becomes available; however, it does not dynamically add work objects to the workspace when new work objects arrive at the block and are waiting for resources.

This figure shows the input path queue for the Take Order task, which has one order waiting to be processed:



For information about the other reason for work backups in a model, see [Showing Work Backups on an Input Path](#).

Showing Currently Allocated Resources

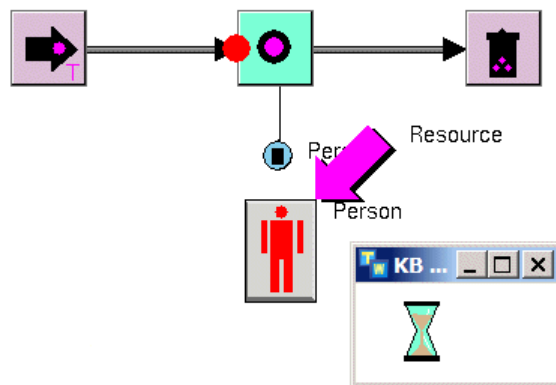
You can show the resources that are currently allocated to an activity.

You can also show the resources that are currently allocated to a work object. However, because work objects are associated with resources only when the resource has been allocated by an upstream block but not yet deallocated, you can only show allocated resources for a work object when you allocate and deallocate the resource as independent steps.

To show allocated resources for an activity:

- ➔ Choose Snapshot Activities on an active block, then choose Show Resources on the activity.

ReThink displays an indicator arrow next to the resources that are currently allocated. For example, this figure shows the result of choosing Show Resources on an activity:



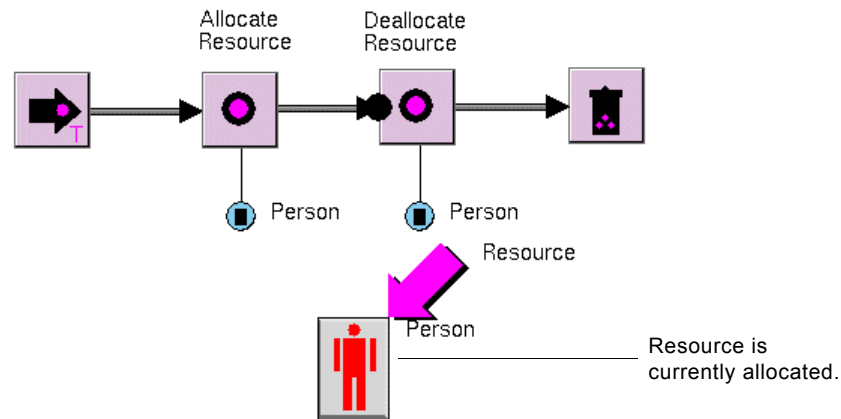
To show allocated resources for a work object:

- 1 Create two Resource Managers, one of which allocates resources upstream in a process, and the other of which deallocates the resources downstream in the process.

For information on how to do this, see [Allocating the Same Resource for Multiple Sequential Steps](#).

- 2 Choose Show Resources on a work object when the resource has been allocated but not yet deallocated.

The following example shows the result of choosing Show Resources on a work object when the resource has been allocated but not yet deallocated:



Disabling a Resource

By default, a resource is always available for allocation. You might want to disable it temporarily, for example, to model a broken down truck in a fleet of trucks.

To disable a resource:

- 1 Show the properties dialog for a resource.
- 2 Click the Utilization tab.
By default, the State is active.
- 3 Configure the State to be inactive or failed.

Creating a Pool of Resources



Often, more than one resource is available for a task. For example, you might have a fleet of delivery trucks, a department of sales clerks, or a supply of computers. One way to model this is to create a resource pool, which is a resource with detail that contains other resources. You often use resource pools to model organizational departments, where a resource within a pool is itself a pool that contains other resources.

Another common way of modeling multiple resources is by configuring the Maximum Utilization of an individual resource and the Utilization of the associated Resource Manager. You often use resource pools in conjunction with resource utilization to model hierarchical views of multiple resources.

When a task requires a pool of resources, as opposed to an individual resource, the task can process as many work objects as there are resources in the pool. For example, if you have a pool of five trucks available for a delivery task, the task can deliver a maximum of five truckloads concurrently before work backs up on the input path to the task. This assumes the default Utilization of the Resource Manager and the default Maximum Utilization of the resource.

When a task requires a pool of resources, by default, the Resource Manager chooses resources at random from the pool. If resources in the pool are currently allocated, the manager chooses an idle resource at random. If there are no resources available, ReThink places the work object that is waiting for a resource in the path queue of the block requiring the resource.

You can customize the way in which a Resource Manager chooses resources from a pool. For example, a manager can choose the lowest cost, lowest utilization, or highest priority resource that is available.

For more information on...	See...
Configuring Maximum Utilization of resources and Utilization of managers	Allocating Partial and Multiple Resources.
Determining whether to use a resource pool or specify individual resource utilization	Determining Whether to Use a Pool or an Individual Resource.
Customizing the way ReThink allocates resources	Choosing Particular Resources from a Pool.

Creating a Pool for Any Resource

To create a pool of resources, you create a detail subworkspace for the resource and add resources to the detail. You can use any resource on the Resources palette of the ReThink toolbox to create a pool.

To create a pool of resources from any resource:

- 1 Display the Resources palette of the ReThink toolbox:



- 2 Select a resource and place it on the model detail.

You can create a person, truck, computer, or machine resource and use it as a pool.

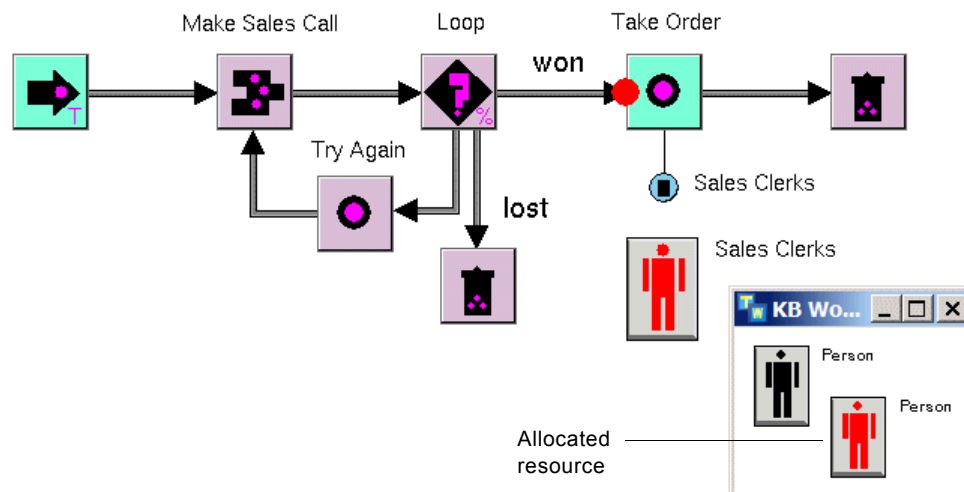
- 3 Click the resource and choose Create Detail.

ReThink creates detail for the resource.

- 4 From the Resources tab, select as many resources as needed and place them in the pool.

- 5 Once you have added resources to the pool, shrink wrap the pool detail.

This running model shows a pool of two sales clerks available for the Take Order task, one of which is currently allocated. The Resource Manager allocates the resources from the pool at random. The allocated resource and pool both turn red.



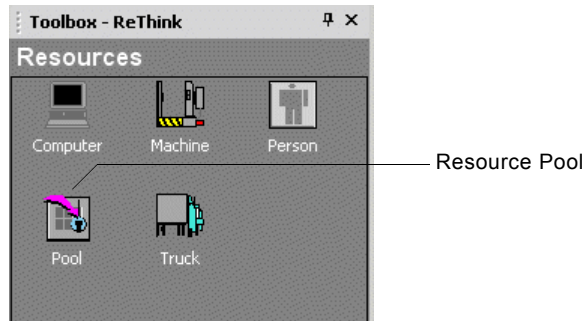
Creating a Generic Pool



If you do not want to use one of the standard resource icons to represent a resource, you can use the Pool resource. You use this icon to represent any type of generic resource pool. A pool has detail by default.

To create a generic pool:

- 1 Display the Resources palette of the ReThink toolbox:



- 2 Select a pool and place it on the model detail.
- 3 Choose Show Detail on the resource pool.
- 4 Select as many resources as needed from the Resources tab and place them in the pool.
- 5 Once you have added resources to the pool, shrink wrap the pool detail.

Showing Pool Details

To display the detail of a resource pool:

- Choose Show Detail on a pool with detail.

Deleting Pool Details

You might decide you only want to allocate a single resource to a task when a pool already exists for the resource. To do this, simply delete the pool detail.

To delete pool details:

- Show the pool detail and choose Edit > Delete.

Computing Utilization and Duration Metrics

ReThink computes utilization and duration metrics for:

- Individual resources in the pool.
- The resource pool itself.

The metrics for individual resources depend on the amount of time the resource is allocated over the life of the simulation. The metrics for the resource pool depend on the number of resources in the pool.

- [Compute utilization history.](#)
- [Charting resource utilization.](#)

Computing Utilization Metrics

ReThink computes the following **utilization** metrics for individual resources and resource pools:

This metric...	Computes this value...	
	For an individual resource...	For the resource pool...
Maximum Utilization	A default value of 1, which means the entire resource is being allocated.	The sum of the Maximum Utilization values of all the resources in the pool. You can also edit this value manually.
Current Utilization	The amount of the resource currently allocated, which is 0, by default, when the resource is idle, and 1, by default, when the resource is allocated.	The sum of the Current Utilization values of all the resources in the pool.
Average Utilization	The amount of time the resource is allocated compared to the amount of time it is available for allocation, which is the Total Work Time divided by the Total Elapsed Time of the resource.	The sum of the Average Utilization values of all the resources in the pool.

The value of the Current Utilization and Maximum Utilization metrics in the table above assumes the default Utilization of the Resource Manager and the default Maximum Utilization of the resource.

If you specify non-default values for the Utilization of a manager or for the Maximum Utilization of a resource, ReThink computes the Current Utilization

and Average Utilization metrics based on the specified utilizations. This means, for example, that when you allocate multiple resources to a task, the Average Utilization of an individual resource can be a number greater than 1.

Note The value of the Average Utilization metric is an average over the entire duration of the simulation. This implies that the metric might take some time to increase, even when the resource is allocated most of the time.

For more information on how to specify non-default utilizations for resources and Resource Managers, see [Allocating Partial and Multiple Resources](#).

Computing Duration Metrics

ReThink computes the following duration metrics for individual resources and resource pools:

This metric...	Computes this value...	
	For an individual resource...	For a resource pool...
Total Work Time	The total amount of time that the resource has been allocated.	The sum of the Total Work Time values of all resources in the pool.
Total Elapsed Time	The total amount of simulation time that a resource has existed.	The total amount of simulation time that the resource pool has existed.
Total Idle Time	The total amount of simulation time that the resource has not been allocated.	The sum of the Total Idle Time values of all resources in the pool.
Not Available Time	The amount of time that the resource is not available during the course of the simulation, due to temporal constraints on the resource.	The total amount of time that all resources in the pool are not available during the simulation, due to temporal constraints on any of the resources.
Creation Time	The simulation time at which the resource was created.	The simulation time at which the resource pool was created.

Assuming Maximum Utilization is 1, Total Work Time + Total Idle Time + Not Available Time = Total Elapsed Time.

Computing Metrics for Individual Resources

ReThink computes utilization metrics for each individual resource in a pool.

For information about the Efficiency Factor, see [Creating Resources with Different Efficiency Factors](#).

To display utilization metrics for an individual resource:

- ➔ Display the properties dialog for a resource in a pool and click the Utilization tab.

The Utilization tab contains utilization metrics, as well as duration metrics. In the following example, the Maximum Utilization is 1, which means the resource can process one activity, assuming the default configuration of the Resource Manager. Here is the Utilization tab for a resource in the pool that is currently allocated by a task:

Property	Value
State	ACTIVE
Maximum Utilization	1.0
Efficiency Factor	1
Total Work Time	000.000.04:51:56
Total Elapsed Time	000.000.12:41:42
Total Idle Time	000.000.07:49:46
Not Available Time	000.000.00:00:00
Creation Time	000.000.00:00:00
Current Utilization	1
Average Utilization	0.383

Current Utilization is 1, which means the resource is currently allocated.

Computing Metrics for the Resource Pool

ReThink computes utilization metrics for the resource pool, which shows the sum of all the resources in the pool.

For example, if two resources are in the pool, the Current Utilization is 0 when no resources are allocated, 1 when one resource is allocated, or 2 when both

resources are allocated. These metrics assume the default utilization of the Resource Manager.

The Average Utilization of the resource pool reflects the sum of the average utilizations of each resource in the pool. Thus, if two resources are in the pool, the Average Utilization will never be greater than 2.

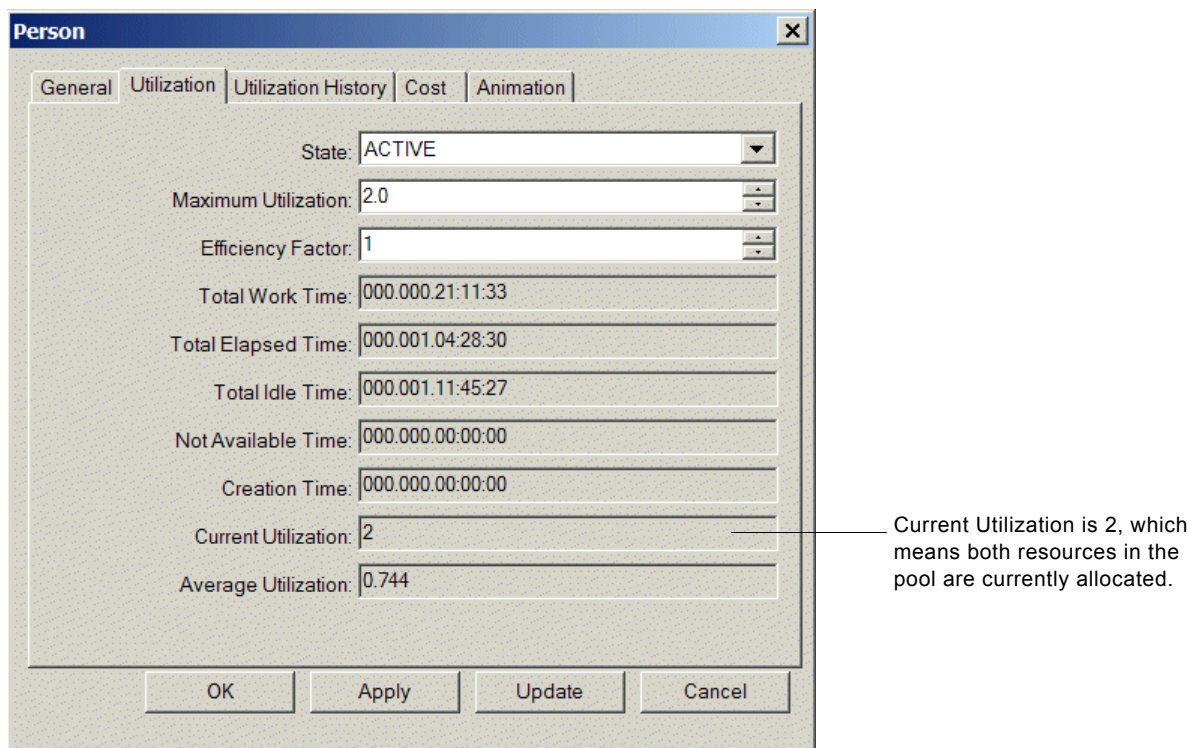
The Maximum Utilization of the resource pool is the sum of the maximum utilizations of each resource in the pool, which is 2, in this example. The Maximum Utilization of the resource pool is read-only in the dialog.

For information about the Efficiency Factor, see [Creating Resources with Different Efficiency Factors](#).

To display utilization metrics for a resource pool:

- ➔ Display the properties dialog for a resource in a pool and click the Utilization tab.

In the following example, the Maximum Utilization is 2 because two resources are in the pool. Here is the Utilization tab for a resource pool when both resources are currently allocated:



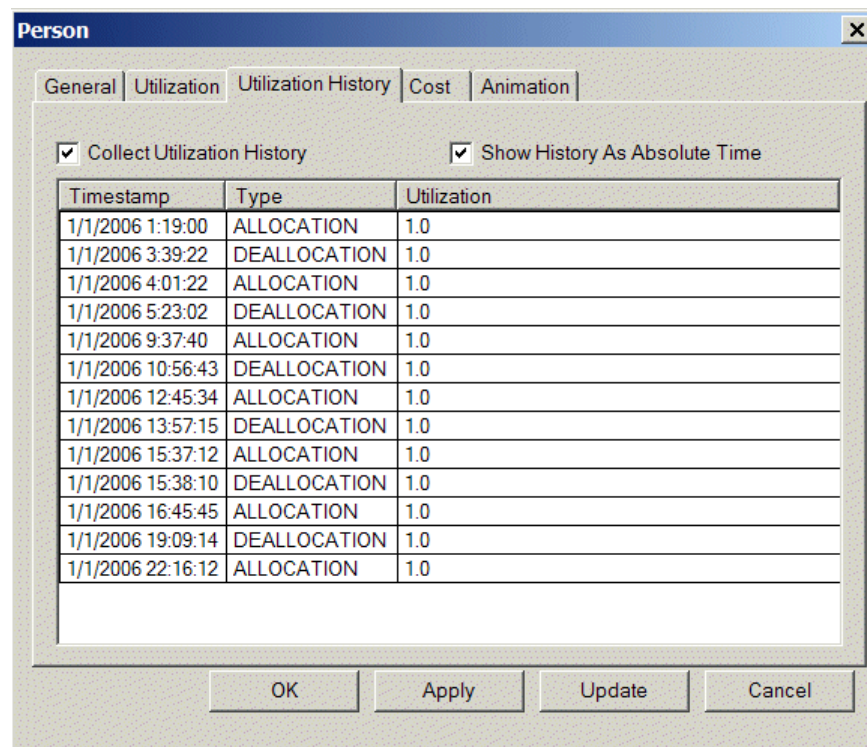
Keeping a History of Resource Utilization

You can configure resources to keep a history of current utilization over time.

To keep a history of resource utilization:

- 1 Display the properties dialog for a resource and click the Utilization History tab.
- 2 Enable the Collect Utilization History option.
By default, ReThink shows timestamps, based on simulation time. You can also show timestamps in absolute time.
- 3 Enable the Show History as Absolute Time, as needed.
- 4 Accept the dialog and run the simulation.
- 5 Display the Utilization History tab again.

The utilization history updates each time the resource is allocated. For example:

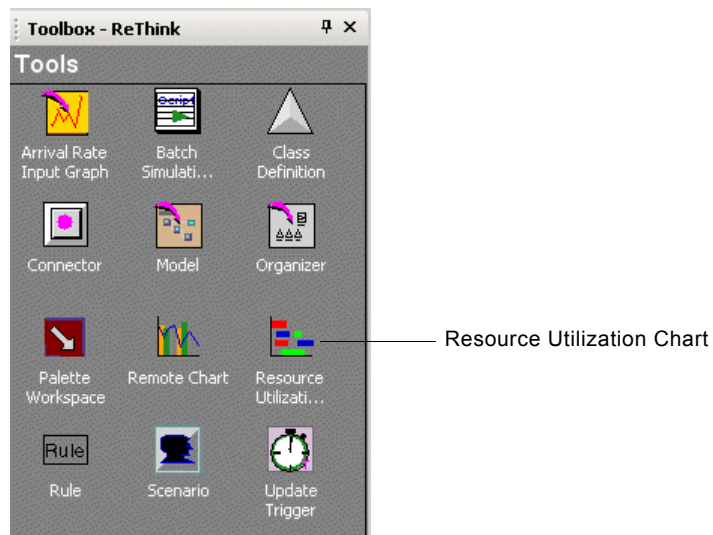


Charting Resource Utilization

You can chart the utilization (Current Utilization / Maximum Utilization) of one or more resources when a simulation runs. The chart represents utilization using colors, where 100% utilization is red, 50% utilization is yellow, 1% utilization is green, and 0% utilization is black. The color is interpolated, for example, 75% utilization is halfway between yellow and red.

To chart resource utilization:

- 1 Display the Tools palette of the ReThink toolbox and create a Resource Utilization chart:



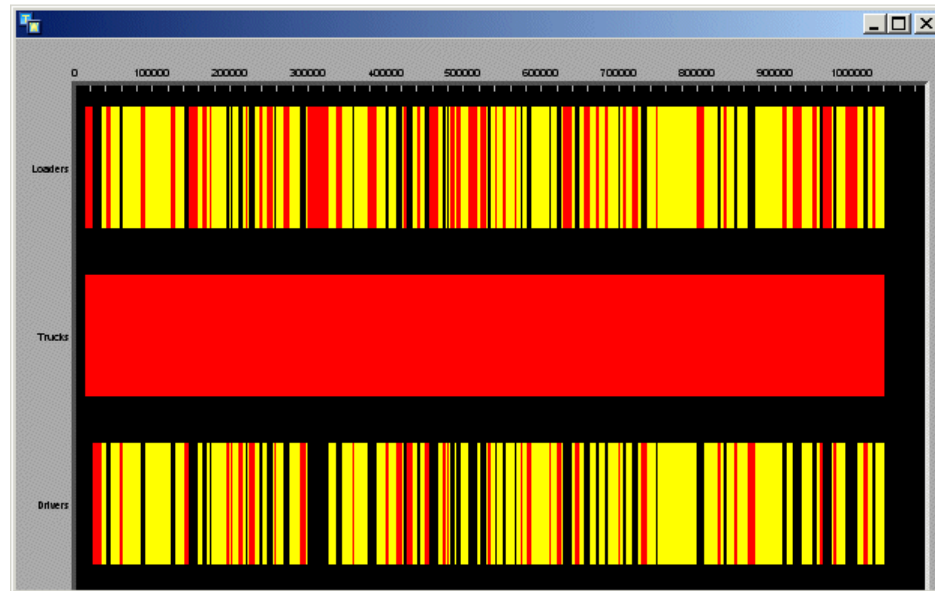
- 2 Display the properties dialog for the Resource Utilization Chart, and choose the resources you want to chart.
- 3 Choose Show Chart to show the chart view.
- 4 Run the simulation to view the resource utilization in the chart.

To configure chart views:

- ➔ To configure the chart, mouse right on the chart and choose Properties.

For more information about configuring chart properties, see the *G2 Reporting Engine User's Guide*.

For example, here is a resource utilization chart that shows % utilization for Trucks, Drivers, and Loaders, where utilization is very high:



Working with Resource Costs

One way you can assign cost in a model is to assign costs to the resources that tasks allocate. You can assign fixed and variable costs to individual resources in the model; you cannot assign costs to resource pools with detail.

Each activity that requires a resource keeps track of the cost of the activity, based on its duration. If you have assigned costs to the resources, the activity tracks the cost of using the resource, based on the duration of the activity. The cost of each activity, in turn, contributes to the total cost of work objects, blocks, and resources themselves.

For information on how to assign fixed and variable costs to blocks, see [Configuring the Cost of a Block](#).

Assigning Costs to Resources in a Model

Suppose you assign a fixed cost of \$10 each time you allocate a truck for a delivery task and a variable cost of \$20 per hour. Each time a resource is allocated, ReThink assigns the fixed cost to the activity and computes the variable cost, based on the duration of the activity.

If the delivery task takes exactly one hour, the cost of the activity is \$30. This \$30 contributes to the total cost of the delivery task, the truck load object, and the truck resource.

To assign fixed and variable costs to a resource:

- 1 Display the properties dialog for the resource whose costs you want to assign and click the Cost tab.

If you are assigning costs to resources in a pool, click a resource in a pool, not the resource pool.

- 2 Configure the Cost Per Use to be a fixed cost.
- 3 Configure the Cost Per Time Unit to be a variable cost.

ReThink computes the variable cost on an hourly basis, by default.

- 4 To compute variable cost based on some other time unit, configure the Time Unit.

For example, to compute costs on a per minute basis, enter 1 minute.

Here is the Cost tab for a resource with fixed and variable costs assigned:

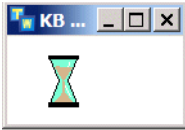
The screenshot shows a dialog box titled "Person" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Utilization", "Utilization History", "Cost", and "Animation". The "Cost" tab is selected. The dialog contains the following fields and controls:

- Cost Per Use:** A text box containing the value "5".
- Cost Per Time Unit:** A text box containing the value "10".
- Time Unit:** Three spinners. The first two are labeled "000" and the third is labeled "01:00:00".
- Total Cost:** A text box containing the value "0".

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Computing the Cost of Individual Activities

Using the sales process model shown under [Creating a Pool of Resources](#), assume the Take Order task takes exactly 30 minutes. This figure shows the activity for the Take Order task and its dialog. The Work Time of the activity is 30 minutes, which is the duration of the activity. The Cost of the activity is \$10, which is the \$5 fixed cost plus half of the \$10 variable cost, because the activity took half an hour.



Result of choosing snapshot activities on the Take Order task block.

$5 + 10(1800/3600)$

Computing Total Costs Based on Resource Costs

ReThink computes the individual cost of an activity based on the duration of the activity and the fixed and variable costs of the resource. The cost of an activity in turn contributes to the total cost of:

- Work objects, which is the sum of the costs of each activity that contributes to the work object.
- Blocks, which is the sum of the costs of each activity that the block performs.
- Individual resources in a pool, which is the sum of the cost of each activity for which a resource was allocated.
- Resource pools, which is the sum of the cost of each resource in the pool.

Assume you have specified a \$5 fixed cost, a \$10 per hour variable cost for each of the two resources in the pool, and a 30 minute duration for the Take Order task

that requires the resources. This table shows the total cost for each type of object when the Take Order task has processed two sales calls:

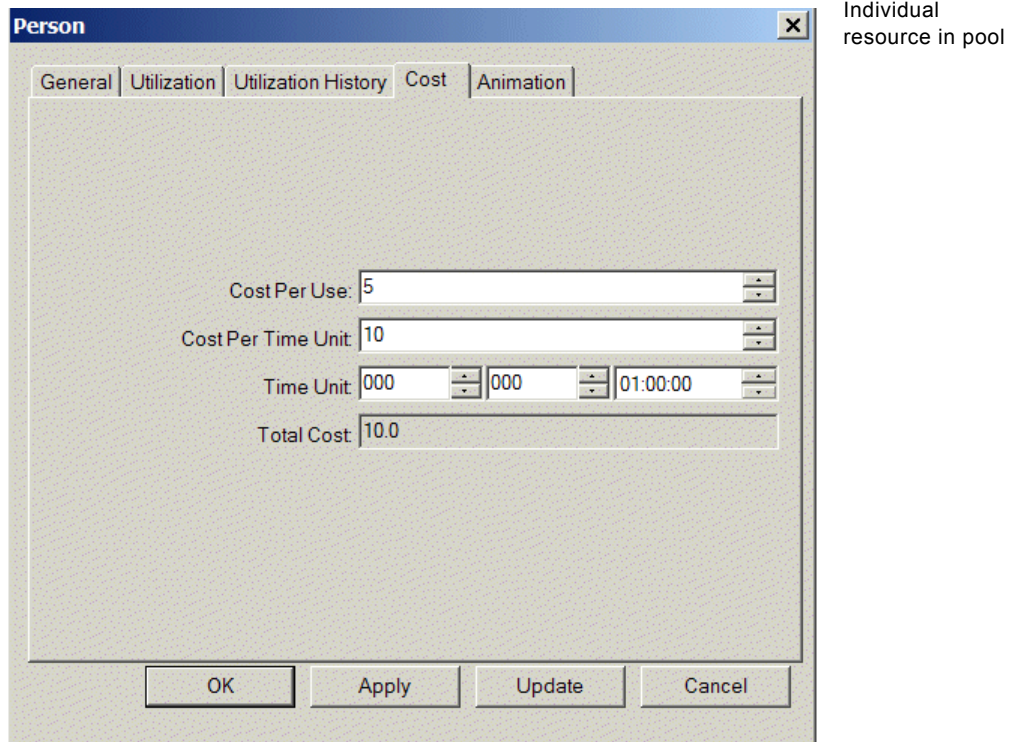
For this object...	The Total Cost is...
Sales call work object	10
Take Order Task block	20
Individual sales clerk resource in pool	10
Sales clerk resource pool	20

The total cost of the individual sales clerks in the resource pool assumes that each resource was allocated once.

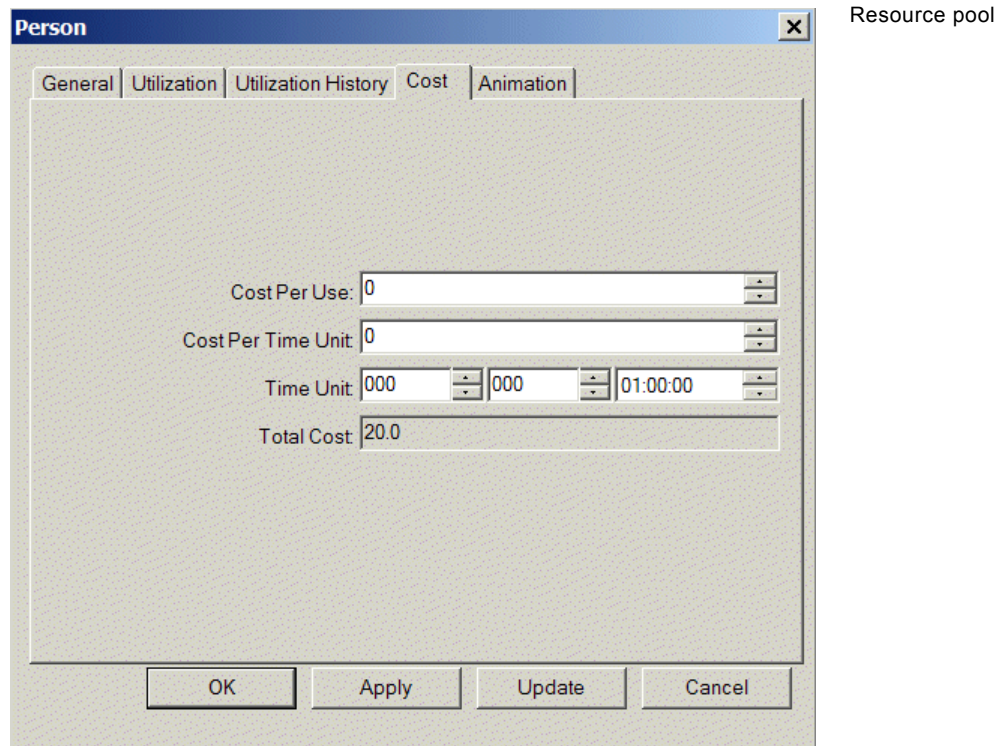
To display the total cost of a resource:

➔ Display the properties dialog for the resource and click the Cost tab.

For example, here is the Cost tab of the properties dialog for an individual resource in the pool after it has been used to process one sales calls:



Here is the Cost tab of the properties dialog for the resource pool after the Take Order task has processed two sales calls:



Note ReThink computes the Total Cost each time a new activity is created or deleted. Thus, Total Cost sometimes reflects the cost of partial activities.

For an explanation of when ReThink updates the metrics of blocks, see [Relating Work Time and Elapsed Time of Activities and Blocks](#).

Allocating Multiple Resources to a Task

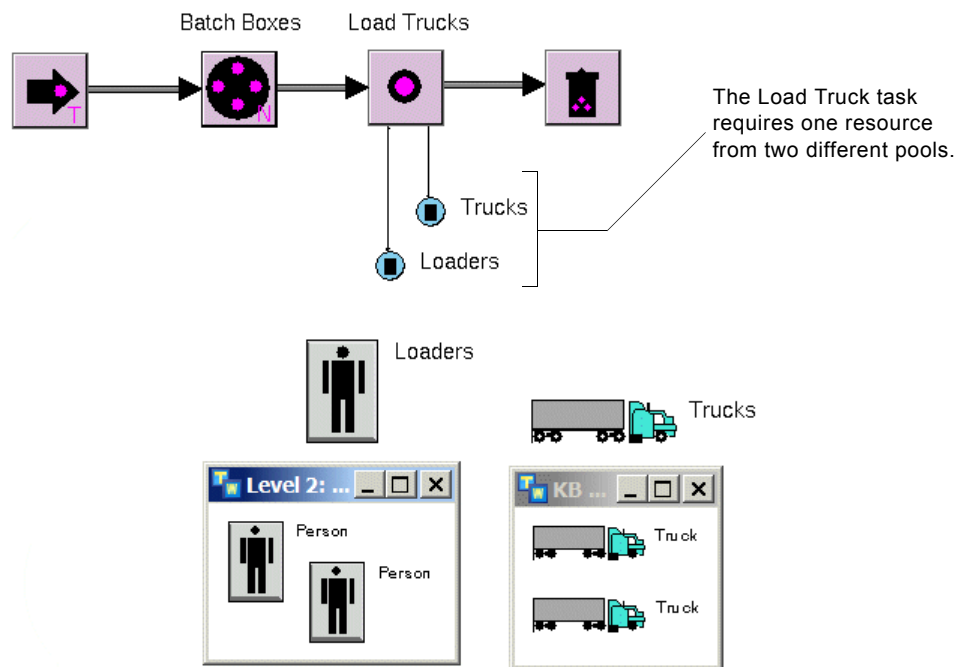
A single task can require multiple resources from different pools. For example, a load truck task might require a truck and a driver.

To allocate multiple resources to a task:

- 1 Create two resources from the Resources palette of the ReThink toolbox and place them on the model detail.
- 2 Create a Resource Manager for each resource by choosing Create Manager on each resource.
- 3 Attach each Resource Manager to the same task.

The task now requires both resources to perform the task.

This model of a loading process batches boxes before loading them onto trucks. The Load Truck task requires truck resources and driver resources:



When you run the simulation, the Load Truck task is constrained by both resources. Notice that the resource pools have the same number of resources.

Tip To optimize the resources in the model, place a similar number of resources in each pool. If more trucks exist than loaders, some trucks will be underused, because not enough loaders exist, and vice versa.

Allocating the Same Pool to Multiple Tasks

Often, two different tasks require the same resource. For example, in a model of a delivery process, the task that loads trucks might require the same truck resources as a task that delivers the trucks.

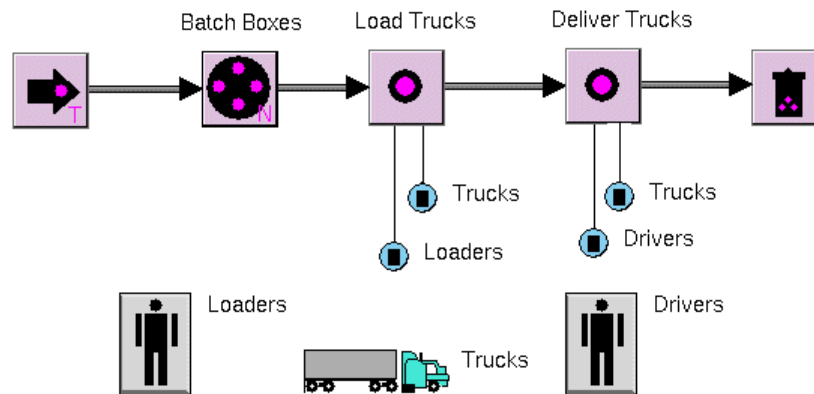
To allocate the same pool of resources to more than one task:

- 1 Create a resource from the Resources palette of the ReThink toolbox and place it on your model workspace.
- 2 Create a Resource Manager for each resource by choosing Create Manager.
- 3 Attach the Resource Manager to a task that requires the resource.

- 4 Create another Resource Manager for the same resource by choosing Create Manager again.
- 5 Attach the second manager to another task.

Two different tasks now require the same resource.

This model shows a delivery process that batches boxes, loads trucks, and delivers boxes. The Load Truck task and the Deliver Truck task require the same truck resources and different people resources.

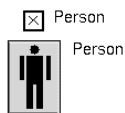


The resource keeps track of which blocks are waiting for resources. If multiple blocks are waiting for the same resource, by default, ReThink allocates the resource at random to the blocks that are waiting, so that the resources are evenly distributed.

You can change the default behavior so that ReThink allocates resources to blocks that are waiting based on priority. For more information, see [Allocating the Same Resource to Different Blocks Based on Priority](#).

When multiple blocks are waiting for the same resource, the Blocks Waiting on the General tab of the properties dialog indicates the number of blocks that are waiting. For more information, see [Showing the Metrics of Resources](#).

Sharing the Same Resource in Multiple Pools



Sometimes, an individual resource is available for more than one task. For example, in a model of a delivery process, a truck loader might also act as a truck driver.

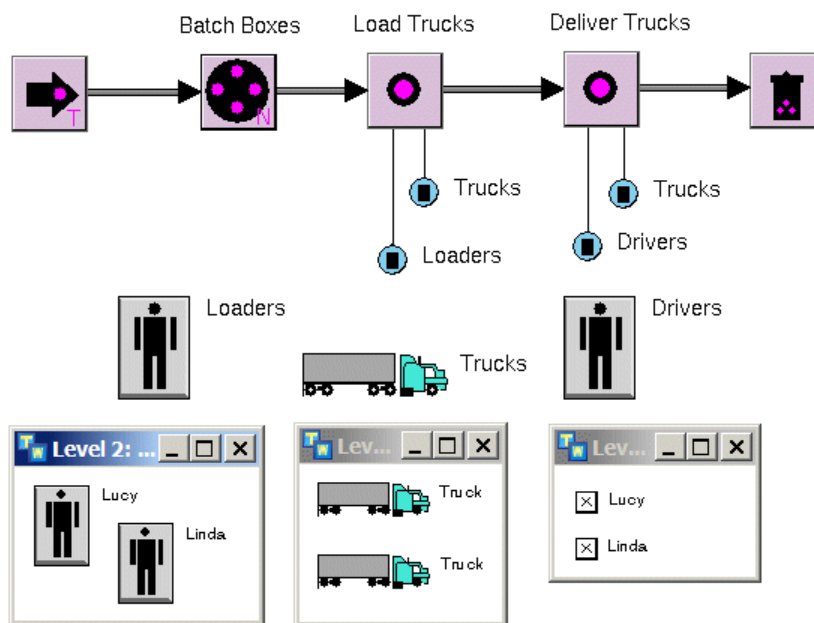
You can model this in ReThink by creating **surrogates**, which are different manifestations of the same resource that appear in multiple pools.

To share the same resource in more than one pool:

- 1 Create two resources from the Resources palette of the ReThink toolbox and place them on a workspace.
- 2 Create detail for one of the resources by choosing Create Detail.
- 3 Create individual resources from the toolbox and place them on the detail of the pool.
- 4 Create detail for the other resource.
- 5 Choose Create Surrogate on each individual resource that will be shared.
ReThink creates a surrogate object and attaches it to the mouse.
- 6 Click to place the surrogate in the empty pool.

The surrogate's label corresponds to the resource label.

This example shows a delivery process in which loader resources for the Load Truck task also act as driver resources for the Deliver Truck task:



When the Deliver Truck task processes, ReThink allocates resources from the Loaders pool, highlighting both the surrogate and the associated resource.

The Drivers resource pool computes summary utilization and cost metrics based on the resource allocation of the surrogates by the Deliver Truck task. Similarly, the Loaders resource pool computes summary utilization and cost metrics based on the allocation of the loader resources by the Load Truck task.

The loader resources themselves compute summary metrics based on the allocation by both the Load Truck task and the Deliver Truck task. Thus, when

you use surrogates, because the resources are shared, the current and average utilization of the resource pool that contains the original resources will be less than the sum of the current and average utilization of the resources in the pool.

To show the resource associated with a surrogate:

→ Choose Show Original on the surrogate.

Allocating Partial and Multiple Resources

Often, a task requires only part of a resource to process a work object. For example, a person might work 50% of the time for two different departments, or a computer resource might be shared between two individuals.

Alternatively, certain tasks might require multiple identical resources. For example, a truck loading task might require two loaders, as opposed to just one. Another example is a financial model in which your resources represent cash, where a task might require \$50 to process a work object.

To allocate partial or multiple resources to a task, you specify the Utilization of the Resource Manager.

You determine how many resources are available for each activity by specifying the Maximum Utilization of individual resources. The Maximum Utilization of a resource represents the amount of the resource that is available for a task.

ReThink computes summary metrics for individual resources and the resource pool based on the utilization you specify.

Specifying the Utilization of the Resource Manager

The default Utilization of a Resource Manager is 1, which means the manager allocates a single resource to each activity, by default.

To allocate a partial resource to an activity:

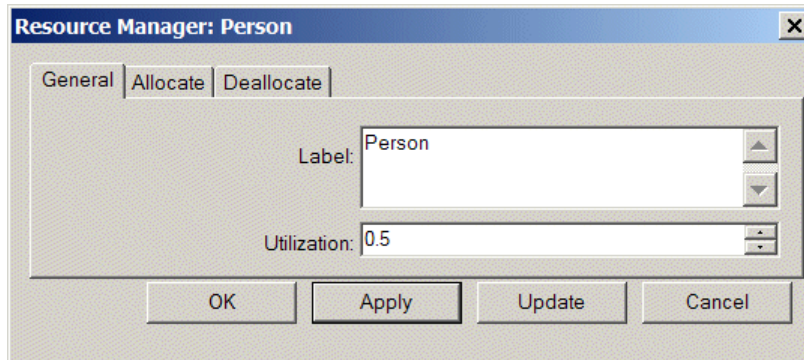
- 1 Display the properties dialog for a Resource Manager and click the General tab.
- 2 Configure the Utilization of the Resource Manager to be a number less than 1.

To allocate multiple resources to an activity:

- 1 Display the properties dialog for a Resource Manager and click the General tab.
- 2 Configure the Utilization of the Resource Manager to be a number greater than 1.
- 3 Ensure that the resource pool has at least the specified number of resources.

For example, when you want to allocate half of a computer resource, set the Utilization of the Resource Manager to 0.5, and when you want to allocate two loaders, set the Utilization to 2.

Here is the properties dialog with the Utilization set to 0.5, which allocates half a resource to an activity:



Specifying the Number of Available Resources

You determine how many resources are available for an activity by specifying the Maximum Utilization of an individual resource. Depending on the requirements of the model, you specify a pool of resources or a single resource. You can specify any number for the Maximum Utilization of the resource, depending on how many resources are available.

To specify the availability of a resource:

- ➔ Display the properties dialog for an individual resource, click the Utilization tab, and specify a number for the Maximum Utilization of the resource.

Typically, you specify the Maximum Utilization of a resource to be a number that is greater than or equal to the Utilization of the associated manager and evenly divisible by the Utilization. Thus, if the manager allocates half of a resource to a task, the Maximum Utilization of each resource in the pool could be 0.5, 1, 1.5, 2, and so on.

However, you can also specify the Maximum Utilization to be a number that is less than the Utilization of the manager and/or not evenly divisible by the Utilization of the manager. In this way, ReThink can obtain partial utilizations from more than one resource to obtain the total utilization that the manager requires. For example, if the Utilization of the Resource Manager is 0.5, and the Maximum Utilization of the two resources in the pool are 0.2 and 0.3, ReThink can obtain the required utilization by combining these two resources.

Similarly, suppose you have two resources with a Maximum Utilization of 1 each, and they are each being partially allocated by another manager for another

activity such that they each have a Maximum Utilization of 0.4 available. ReThink will allocate 0.4 from one and 0.1 from another to obtain a Utilization of 0.5.

Determining the Maximum Number of Activities

ReThink determines the maximum number of concurrent activities for which the model can allocate each resource as follows:

The Maximum Utilization of available resource divided by the Utilization of Resource Manager, rounded down to the nearest integer

The following table illustrates the results of different combinations of Resource Manager Utilization and resource Maximum Utilization when you have two resources in the pool:

If the Utilization of the Resource Manager is...	And the Maximum Utilization of the resource is...	Then the maximum number of activities the block can process concurrently is...
0.5	0.5	1
0.5	1	2
0.5	2	4
1	1	1
1	2	2
2	2	1
2	4	2
2	5	2

Determining Whether to Use a Pool or an Individual Resource

You specify a pool of resources in which individual resources each specify a Maximum Utilization, under these circumstances:

- When you are modeling a relatively small number of resources.
- When you want to visualize the allocation of individual resources by a task.
- When each resource has a different cost associated with it.
- When each resource has a different availability.

You specify an individual resource with a Maximum Utilization, under these circumstances:

- When you are modeling large numbers of resources, such as financial resources, when it would be cumbersome to create individual resources in a pool; for example, a financial model where a single resource represents \$50 cash.
- When you do not care about visualizing the allocation of individual resources.
- When no cost is associated with each allocated resource or when the cost of each resource is the same.
- When each allocated resource has the same availability.

Example of Allocating Partial Resources from a Pool

To allocate partial resources, you configure the Utilization of the Resource Manager to be a number less than one, and you configure the Maximum Utilization of either a single resource or several individual resources in a pool.

To allocate partial resources to a task:

- 1 Display the properties dialog for a Resource Manager and click the General tab.
- 2 Configure the Utilization to be the amount of each resource that the manager will allocate for each task.

The number should be less than one.
- 3 Create a single resource or create a resource pool, depending on how you want to specify the available resources.
- 4 Display the properties dialog for each resource, click the Utilization tab, and configure the Maximum Utilization to be the amount of the resource that is available for the task.

For information on how to specify the availability of resources, see [Specifying the Number of Available Resources](#).

Here is the properties dialog for a Resource Manager that allocates half a resource for each activity and the Utilization tab of the properties dialog for a resource where the availability of the resource is also one half:



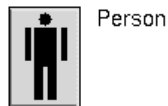
Resource Manager: Person

General | Allocate | Deallocate

Label: Person

Utilization: 0.5

OK Apply Update Cancel



Person

General | Utilization | Utilization History | Cost | Animation

State: ACTIVE

Maximum Utilization: 0.5

Efficiency Factor: 1

Total Work Time: 000.000.00:00:00

Total Elapsed Time: 000.000.00:00:00

Total Idle Time: 000.000.00:00:00

Not Available Time: 000.000.00:00:00

Creation Time: 000.000.00:00:00

Current Utilization: 0

Average Utilization: 0

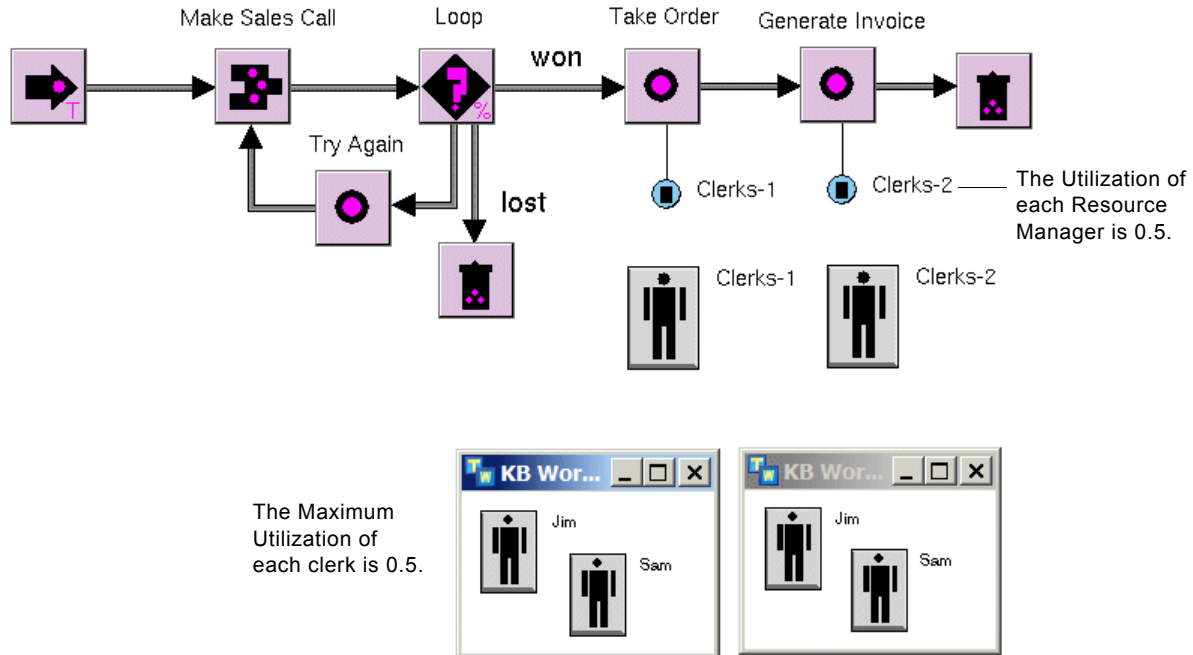
OK Apply Update Cancel

When the Maximum Utilization of a resource is less than one, you can model the use of the balance of the partial resource by allocating the balance of the resource to another task. For example, you might have a pool of clerks that split their time equally between taking orders and generating invoices.

To model this, create two resource pools, where each resource in each pool has a Maximum Utilization of 0.5 and specify the utilization of the associated manager as 0.5. Although the resources in each pool represent two halves of a complete resource, they are in fact different resources, each with a Maximum Utilization of 0.5.

This model of a sales process has two tasks that require resources: the Take Order task and the Generate Invoice task, each of which uses half of a clerk resource.

The Utilization of the manager is 0.5, and the Maximum Utilization of each clerk resource is 0.5. Although the labels of the resources in each pool are the same, the resources are in fact unique resources that represent the same person.



ReThink computes utilization metrics for all resources in the model, based on the Utilization of the Resource Manager and the Maximum Utilization of the individual resources.

For general information on displaying and computing utilization metrics for resources and managers, see [Computing Utilization and Duration Metrics](#).

Computing Metrics for Individual Resources in a Pool

If the Utilization of the manager is 0.5, and the Maximum Utilization of a resource is 0.5, the Current Utilization of an individual resource is 0 when it is not allocated, and 0.5 when it is. The Average Utilization in turn will always be less than 0.5.

Here is the Utilization tab of the properties dialog for a resource that is currently allocated:

The screenshot shows the 'Person' properties dialog with the 'Utilization' tab selected. The dialog contains the following fields and values:

Field	Value
State	ACTIVE
Maximum Utilization	0.5
Efficiency Factor	1
Total Work Time	000.000.01:39:03
Total Elapsed Time	000.000.11:29:00
Total Idle Time	000.000.04:05:26
Not Available Time	000.000.00:00:00
Creation Time	000.000.00:00:00
Current Utilization	0.5
Average Utilization	0.144

A callout box points to the 'Current Utilization' field, stating: "Current Utilization is .5, which means that half a resource is currently allocated."

Computing Metrics for the Resource Pool

If two resources are in the pool, the Current Utilization of the resource pool is 0 when the model does not allocate any resources, 0.5 when the model allocates one resource, and 1 when the model allocates both resources.

The Average Utilization will be greater than 0.5 if the model allocates both resources concurrently on average, but it will always be less than 1.

The Maximum Utilization of the resource pool is 1, which is the sum of each resource's Maximum Utilization.

Here is the Utilization tab of the properties dialog for the resource pool in the previous model, when the model is currently allocating both resources, each of which has a Maximum Utilization of 0.5:

The screenshot shows the 'Person' properties dialog with the 'Utilization' tab selected. The fields are as follows:

Field	Value
State	ACTIVE
Maximum Utilization	1
Efficiency Factor	1
Total Work Time	000.000.05:46:22
Total Elapsed Time	000.000.13:33:15
Total Idle Time	000.000.07:46:52
Not Available Time	000.000.00:00:00
Creation Time	000.000.00:00:00
Current Utilization	1.0
Average Utilization	0.426

Current Utilization is 1, which means that both resources in the pool are allocated, where each individual resource in the pool represents half a resource.

Example of Allocating Multiple Resources from a Pool

To allocate multiple resources to a task, you configure the Utilization of the Resource Manager to be a number greater than one, and you configure the Maximum Utilization of either a single resource or several individual resources in a pool.

To allocate multiple identical resources from a pool to a task:

- 1 Display the properties dialog for a Resource Manager and click the General tab.
- 2 Configure the Utilization of the Resource Manager to be the number of resources that the task requires.
This number should be greater than 1.
- 3 Create a single resource or create a resource pool, depending on how you want to specify the available resources.

- 4 Display the properties dialog for each resource, click the Utilization tab, and configure the Maximum Utilization to be the amount of the resource that is available for the task.

For information on how to specify the availability of resources, see [Specifying the Number of Available Resources](#).

Here is the properties dialog for a Resource Manager that allocates two resources at a time and the Utilization tab of the properties dialog for a resource where the availability of the resource is also 2:



Loaders

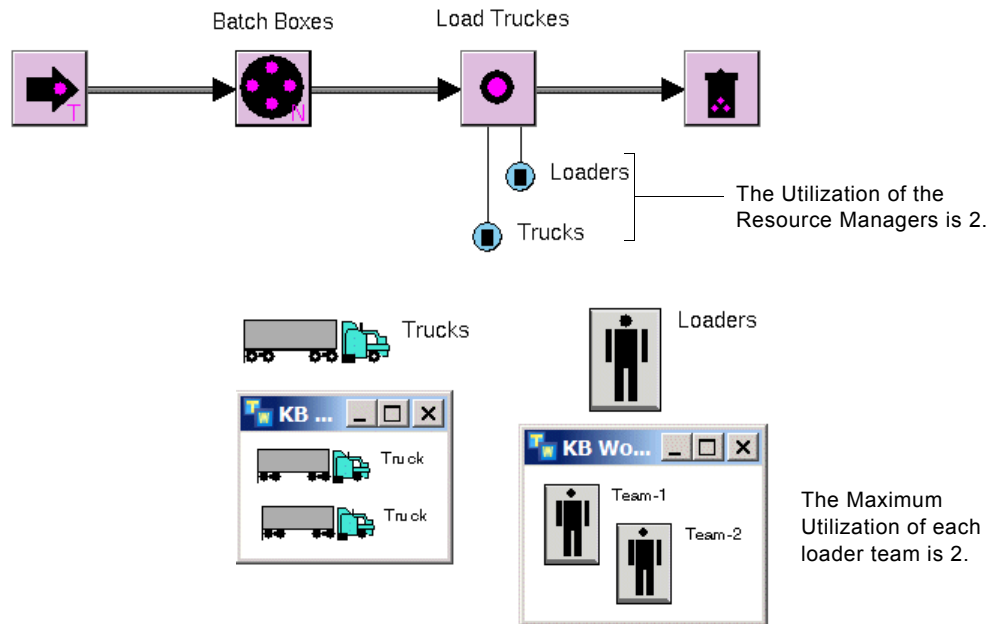
The dialog box titled "Resource Manager: Loaders" has three tabs: "General", "Allocate", and "Deallocate". The "General" tab is selected. It contains a "Label:" field with the text "Loaders" and a "Utilization:" field with the value "2". At the bottom, there are four buttons: "OK", "Apply", "Update", and "Cancel".



Team-1

The dialog box titled "Person" has five tabs: "General", "Utilization", "Utilization History", "Cost", and "Animation". The "Utilization" tab is selected. It contains several fields: "State:" with a dropdown menu set to "ACTIVE"; "Maximum Utilization:" with a value of "2"; "Efficiency Factor:" with a value of "1"; "Total Work Time:", "Total Elapsed Time:", "Total Idle Time:", "Not Available Time:", and "Creation Time:" all with values of "000.000.00:00:00"; "Current Utilization:" with a value of "0"; and "Average Utilization:" with a value of "0". At the bottom, there are four buttons: "OK", "Apply", "Update", and "Cancel".

This model of a delivery process requires two resources: a single truck and a team of loaders, where each team of loaders represents two identical resources:



ReThink computes utilization metrics for all resources in the model based on the utilization of the Resource Manager and the Maximum Utilization of the individual resources.

For general information on displaying and computing utilization metrics for resources and Resource Managers, see [Computing Utilization and Duration Metrics](#).

[Computing Utilization and Duration Metrics](#).

Computing Metrics for Individual Resources in a Pool

If the Utilization of the manager is 2 and the Maximum Utilization of each resource in a pool is 2, the Current Utilization of each resource is 0 when the model does not allocate any resources, or 2 when it does.

The Average Utilization is computed based on the Current Utilization. Thus, depending on the duration of the task that requires the resource, the Average Utilization of the resource will be between 0 and 2.

Here is the Utilization tab of the properties dialog for the Team-1 resource, which represents two identical resources that are currently allocated:

The screenshot shows the 'Person' properties dialog with the 'Utilization' tab selected. The fields and their values are as follows:

Field	Value
State	ACTIVE
Maximum Utilization	2
Efficiency Factor	1
Total Work Time	000.000.21:00:38
Total Elapsed Time	000.000.16:08:36
Total Idle Time	000.000.11:16:34
Not Available Time	000.000.00:00:00
Creation Time	000.000.00:00:00
Current Utilization	2
Average Utilization	1.302

Buttons at the bottom: OK, Apply, Update, Cancel.

Current Utilization is 2, which means that the equivalent of two resources are allocated.

Computing Metrics for the Resource Pool

ReThink computes utilization metrics for the resource pool, which reflects the sum of the utilizations of each resource in the pool.

If the Utilization of the manager is 2, the Current Utilization of the resource pool is 0 when the model does not allocate any resources, 2 when the model allocates one physical resource, or 4 when the model allocates two physical resources. Again, the current utilization depends on the utilization of the manager.

The Average Utilization reflects the sum of the average utilizations of the resources in the pool. Thus, depending on the duration of the task, the Average Utilization of the resource pool will be between 0 and 4.

The resource pool computes its Maximum Utilization based on the sum of the Maximum Utilizations of each resource in the pool, which is 4.

Here is the Utilization tab of the properties dialog for the Loaders resource pool when both resources are allocated:

The screenshot shows a dialog box titled "Person" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Utilization", "Utilization History", "Cost", and "Animation". The "Utilization" tab is selected. The dialog contains the following fields:

- State: ACTIVE (dropdown menu)
- Maximum Utilization: 4 (spin box)
- Efficiency Factor: 1 (spin box)
- Total Work Time: 000.001.18:01:16 (text box)
- Total Elapsed Time: 000.000.16:08:36 (text box)
- Total Idle Time: 000.000.22:33:08 (text box)
- Not Available Time: 000.000.00:00:00 (text box)
- Creation Time: 000.000.00:00:00 (text box)
- Current Utilization: 4 (spin box)
- Average Utilization: 2.603 (text box)

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Current Utilization is 4, which means that both resources are allocated, where each individual resource in the pool represents two resources.

Allocating the Same Resource for Multiple Sequential Steps

Often you need to allocate a resource for several sequential steps in a process. For example, the same clerk might process the order and file the order. When you allocate the same resource for several sequential steps in a process:

- The resource remains allocated for the duration of both tasks, which means the resource is unavailable to any other block in the model during the time it is allocated.
- You guarantee that the same resource is allocated to each sequential step in a process, as opposed to allocating potentially different resources.

One way to model this is to allocate the resource at the beginning of one task and deallocate the resource at the end of another task. You model this by configuring attributes of the Resource Manager.

Be aware that if a work object arrives at a block that allocates a resource, and the work object flows to a downstream block that is required to wait for an input

from some other block while the resource is still allocated, the Total Work Time of the resource includes the time the block spent waiting for the other input.

Note You must explicitly deallocate every resource in the model; deleting work objects at the end of processing does not automatically deallocate resources.

To allocate the same resource for multiple sequential steps in a process:

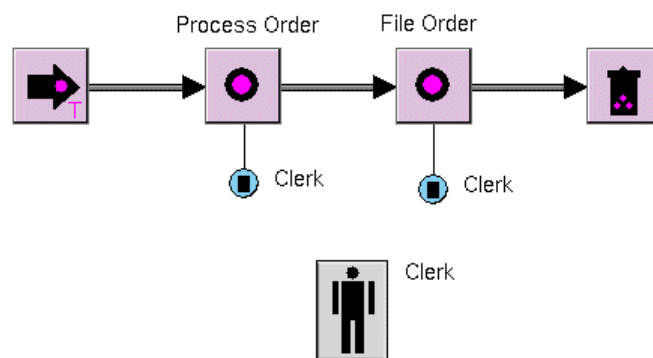
- 1 Create a resource that a task will allocate for multiple sequential steps in a process.
- 2 Create two Resource Managers from the same resource by choosing Create Manager twice on the resource.
- 3 Connect one Resource Manager to the task that will allocate the resource.
- 4 Connect the other Resource Manager to the task that will deallocate the resource.
- 5 Display the properties dialog for the Resource Manager that will allocate the resource.
- 6 Click the Deallocate tab and click the Deallocate Resource option off.

The first Resource Manager allocates the resource for the task, but it does not deallocate it at the end of processing.

- 7 Display the properties dialog for the Resource Manager that will deallocate the resource.
- 8 Click the Allocate tab and click the Allocate Resource option off.

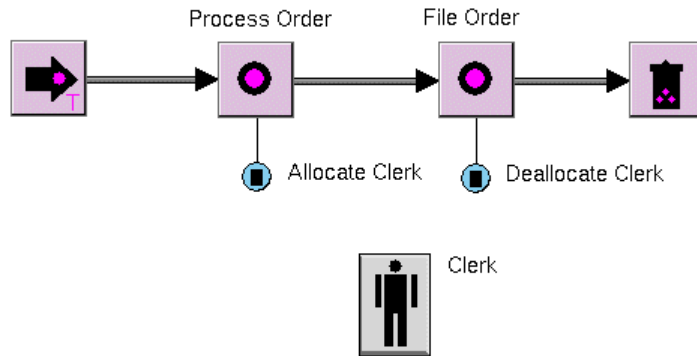
The second Resource Manager deallocates the resource at the end of processing; the resource is already allocated when it arrives at the block.

For example, suppose your model allocates and deallocates the same resource for each of two tasks, using the default behavior:



In this model, the Resource Managers allocate the resource twice, once for each task. The first manager allocates the clerk resource when the Process Order task processes an order, and it deallocates the clerk resource when the order passes to the input path of the File Order task. The second manager allocates the clerk resource again when the File Order task processes the order, and it deallocates the clerk resource when the order passes to the input path of the Sink block.

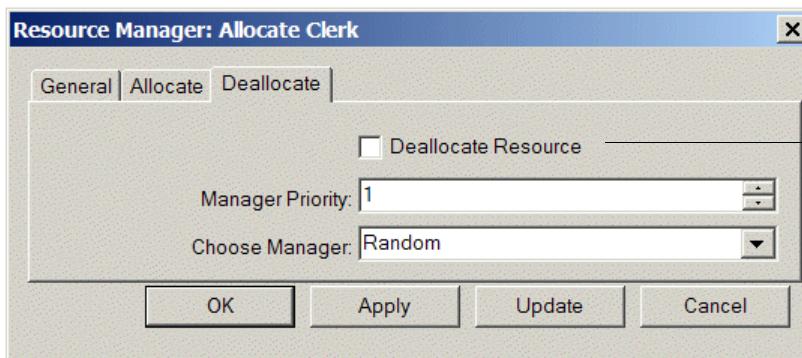
Now, compare the previous model to this model, which specifies two Resource Managers that allocate the same clerk resource for two sequential tasks:



In this model, the Allocate Clerk manager allocates the resource only once, at the beginning of the Process Order task. The clerk resource remains allocated when the order passes to the input path of the File Order task. The Deallocate Clerk manager deallocates the clerk resource when the order passes to the input path of the Sink block.

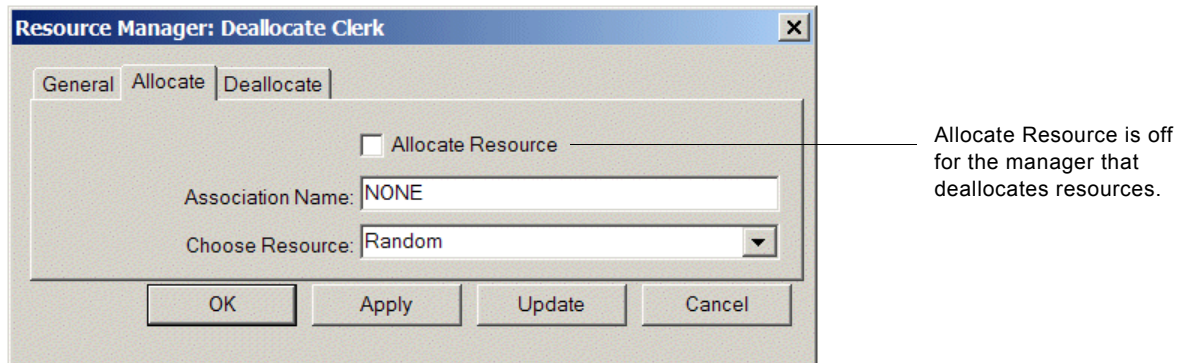
Your model might create work object in between allocating and deallocating the resource. In the model above, you might have a Task block with two output paths or a Copy block between the Process Order and File Order tasks. In this case, be sure to specify the output path types such that the same work object that caused the resource to be allocated also causes the resource to be deallocated.

Here is the Deallocate tab of the properties dialog for the Allocate Clerk Resource Manager:



Deallocate Resource is off for the manager that allocates resources.

Here is the Allocate tab of the properties dialog for the Deallocate Clerk Resource Manager:



Choosing Particular Resources from a Pool



Person

By default, ReThink allocates available resources at random from a pool. If a pool contains multiple resources, all of which are allocated, ReThink chooses the first resource to become available.

You might want to allocate a particular resource for an activity, based on some other criteria. For example, you might want to allocate the available resource with the lowest cost. You might also want to allocate the available resource with the lowest utilization. Alternatively, you might have some other criteria for determining which available resource to choose first. In all cases, ReThink chooses the first *available* resource that meets the criteria.

You can configure the Resource Manager to choose resources from a pool, based on one of these criteria:

- [Lowest cost](#)
- [Lowest utilization](#)
- [Highest or lowest priority](#)

You can also choose resources from a pool, based on a custom procedure. For details, see the *Customizing ReThink User's Guide*.

Choosing the Lowest Cost Resource

Each resource computes its cost based on the fixed and variable cost of the resource and the duration of the activity for which it is allocated. You can choose to allocate the resource with the lowest cost per activity before you allocate resources with higher cost per activity by configuring the Resource Manager. The resource cost of the activity is:

$$\text{Cost Per Use} + \text{Cost Per Time Unit/Time Unit}$$

To choose the lowest cost resource:

- 1 Create a resource pool and associated Resource Manager and connect the manager to a task.

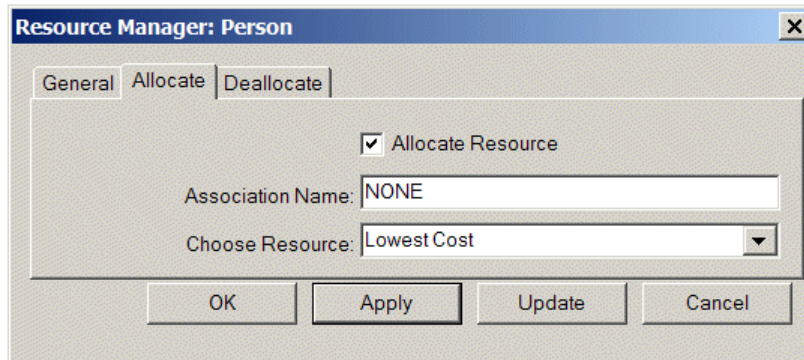
For information on how to do this, see [Creating a Pool of Resources](#) and [Using Resources to Constrain the Model](#).

- 2 Display the properties dialog for each resource in the pool, click the Cost tab, and configure the fixed and variable costs.

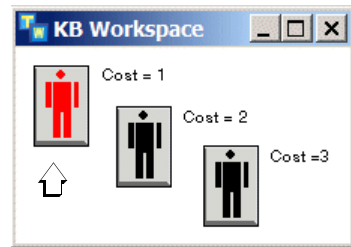
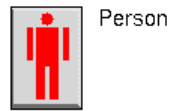
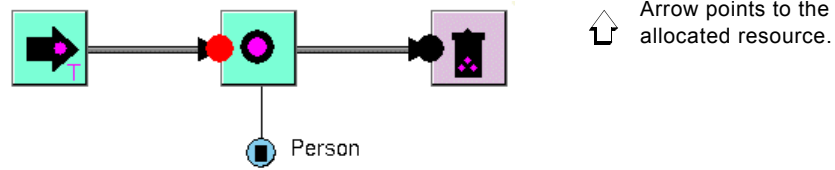
For information on how to do this, see [Working with Resource Costs](#).

- 3 Display the properties dialog for the Resource Manager, click the Allocate tab, and configure Choose Resource to be **Lowest Cost**.

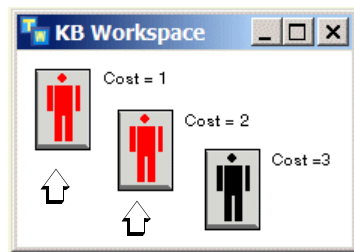
Here is the Allocate tab of the properties dialog for a Resource Manager configured so that it chooses the available resources with the lowest cost from the pool first:



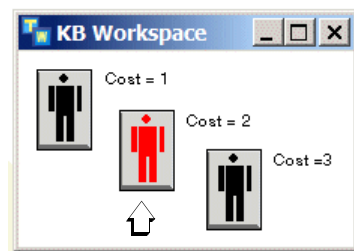
This example shows how ReThink chooses the lowest cost resource in the pool. The label of each resource in the pool indicates the value of the Cost Per Use of the resource. The example shows the same resource pools at different points in the simulation. The arrows show the allocated resources.



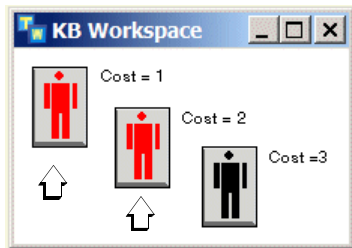
ReThink allocates the resource with the lowest cost first.



ReThink allocates the resource with the next lowest cost next, because the lowest cost resource is already allocated.



ReThink deallocates the lowest cost resource.



ReThink allocates the lowest cost resource again because it is available.

Choosing the Resource with the Lowest Utilization

Each resource computes its average utilization, which is the average amount of time that the resource has been allocated over the course of the simulation. Specifically, the Average Utilization of a resource is the amount of time the resource has been allocated (Total Work Time) divided by the duration of the simulation (Total Elapsed Time).

If you compare the average utilization of the resources in a pool, you can tell which resources have been allocated relatively less frequently over the course of the simulation. You can choose to allocate resources with a relatively lower average utilization before resources with a relatively higher utilization by configuring the Resource Manager.

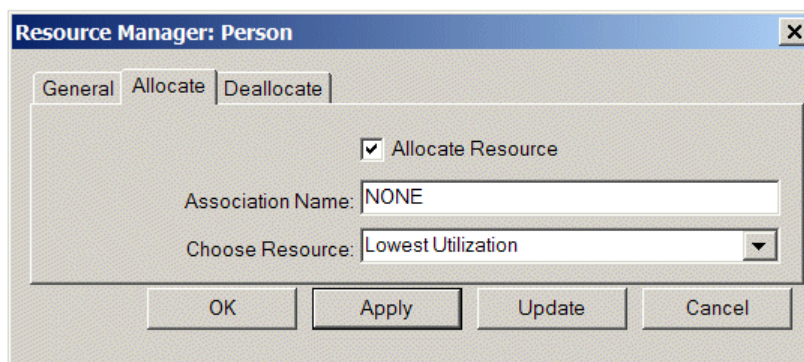
To choose the resource with the lowest average utilization:

- 1 Create a resource pool and associated Resource Manager, and connect the manager to a task.

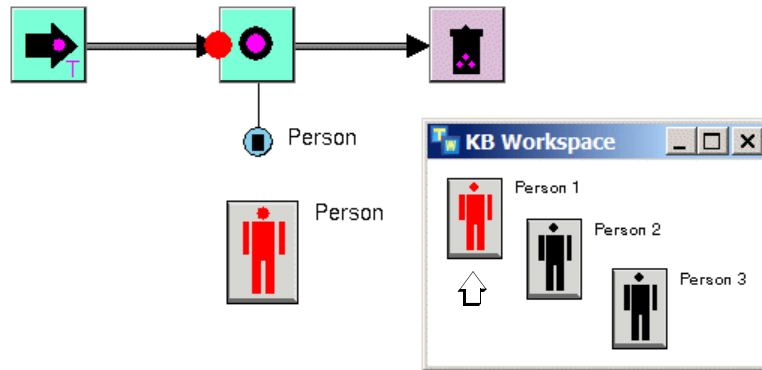
For information on how to do this, see [Creating a Pool of Resources](#) and [Using Resources to Constrain the Model](#).

- 2 Display the properties dialog for the Resource Manager, click the Allocate tab, and configure Choose Resource to be Lowest Utilization.

Here is the Allocate tab of the properties dialog configured so that it chooses the resource with the lowest utilization from the pool first:



This example shows how ReThink chooses the resource with the lowest average utilization from the pool before other resources. The example shows the Utilization tab of the properties dialog for each resource at a particular point in the simulation. Notice that ReThink allocates the resource with the lowest utilization first.



ReThink allocates the resource with the lowest utilization first.

Person-3

Person [X]

General Utilization Utilization History Cost Animation

State: ACTIVE

Maximum Utilization: 1.0

Efficiency Factor: 1

Total Work Time: 000.000.06:43:58

Total Elapsed Time: 000.000.14:19:56

Total Idle Time: 000.000.07:35:58

Not Available Time: 000.000.00:00:00

Creation Time: 000.000.00:00:00

Current Utilization: 0

Average Utilization: 0.47

OK Apply Update Cancel

Person-2

Person

General Utilization

Maxi

Et

Tc

Total

Not /

Cu

Average Utilization: 0.432

OK Apply Update Cancel

Person-1

Person

General Utilization

Average Utilization: 0.417

OK Apply Update Cancel

Lowest average utilization

Choosing Resources Based on Priority

You might have certain criteria for choosing resources from a pool other than cost or utilization. For example, you might always want to choose one worker over another worker based on skills, or you might want to choose one truck over another based on the mileage of the truck.

You can assign priorities to each resource in a pool and choose resources from the pool based on priority. You can allocate resources with the highest priority over resources with the lowest priority, or vice versa.

You assign the priority of a resource by configuring the resource, and you specify which resource to allocate by configuring the Resource Manager.

To choose a resource based on priority:

- 1 Create a resource pool and associated Resource Manager, and connect the manager to a task.

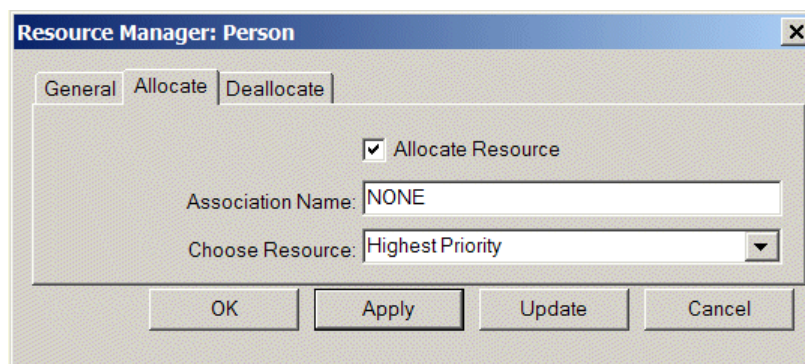
For information on how to do this, see [Creating a Pool of Resources](#) and [Using Resources to Constrain the Model](#).

- 2 Display the properties dialog for each resource in the pool, click the General tab, and configure Resource Priority.

You can specify any number as the priority, where 1 is the highest priority; the larger the number, the lower the priority.

- 3 Display the properties dialog for the Resource Manager, click the Allocate tab, and configure Choose Resource to be either **Lowest Priority** or **Highest Priority**, depending on your needs.

Here is the Allocate tab of the properties dialog configured so that it chooses the resource with the highest priority from the pool first:



Here is the properties dialog for the resource with the highest priority:

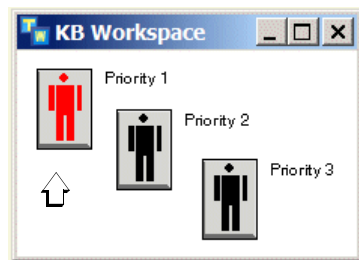
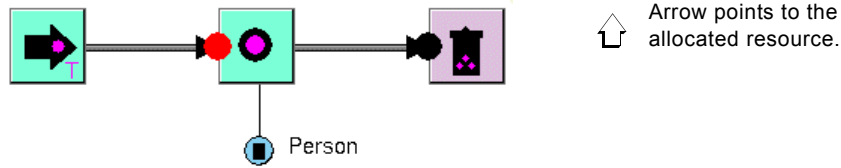
The screenshot shows a dialog box titled "Person" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Utilization", "Utilization History", "Cost", and "Animation". The "General" tab is active. The fields are as follows:

- Label: Priority 1
- Comments: (empty)
- Resource Priority: 1
- Total Starts: 0
- Total Stops: 0
- Current Activities: 0
- Blocks Waiting: (empty)
- Error: (empty)

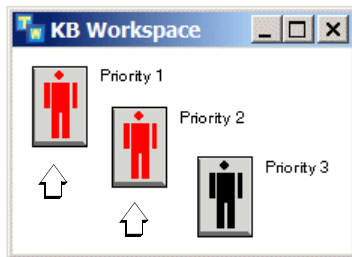
At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Resource Priority is 1 for the resource with the highest priority.

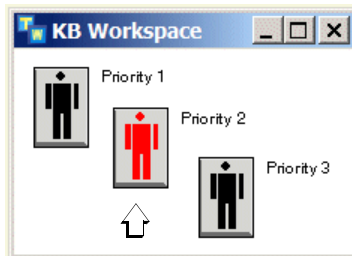
This example shows how ReThink chooses the highest priority resource in the pool. The label of each resource in the pool indicates the value of the Priority of the resource. The example shows the same resource pools at different points in the simulation. The arrows point to the allocated resources.



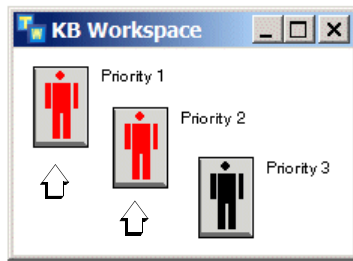
ReThink allocates the highest priority resource first.



ReThink allocates the resource with the next highest priority next, because the highest priority resource is already allocated.



ReThink deallocates the highest priority resource.



ReThink allocates the highest priority resource again (Priority 1) because it is now available.

You can also add the current value of the Priority as an attribute display of a resource. For information on how to do this, see [Using Attribute Displays](#).

Allocating Resources Associated to Work Objects

In some cases, you might have a process in which resources are associated with a particular type of work object. You can associate resources with work objects by using an Associate block, then allocate only the associated resources to a task. To do this, you generate resources as part of the process, associate them with the designated type of work objects, then store the resources in a pool. A downstream task then allocates resources from the pool based on the association name.

You can allocate associated resources from a pool, using any of the standard methods for choosing a resource: random, lowest cost, lowest utilization, or priority. You specify the name of the association in the Resource Manager.

The Utilization of the Resource Manager determines how many of the associated resources to allocate to the task, as follows:

- A positive Utilization allocates *the specified number* of associated resources to the task.
- A Utilization of 0 allocates *all* associated resources to the task.
- A negative Utilization allocates *all but* the specified number of associated resources to the task.

You can use this feature in conjunction with resource priorities to determine which specific associated resources the manager allocates first.

For example, suppose you have a POR (purchase order request) approval process that allocates to an approval task different approving managers, depending on the POR amount:

- For amounts \$100 or under, the approval task requires one approval signature.
- For amounts between \$101 and \$2500, the approval task requires two approval signature.
- For amounts \$2500 or over, the approval task requires signatures from all approving managers, plus the CFO.

The model might associate three levels of approving managers with a POR, then store these resources in a pool. The model would also store a CFO resources in the pool, although this resource would not be associated with the others.

The Resource Managers associated with each approval task would specify the association name and the number of required signatures from approving managers. The approval task for POR amounts over \$2500 would also allocate the CFO resource.

The following set of steps describe how to build such a model.

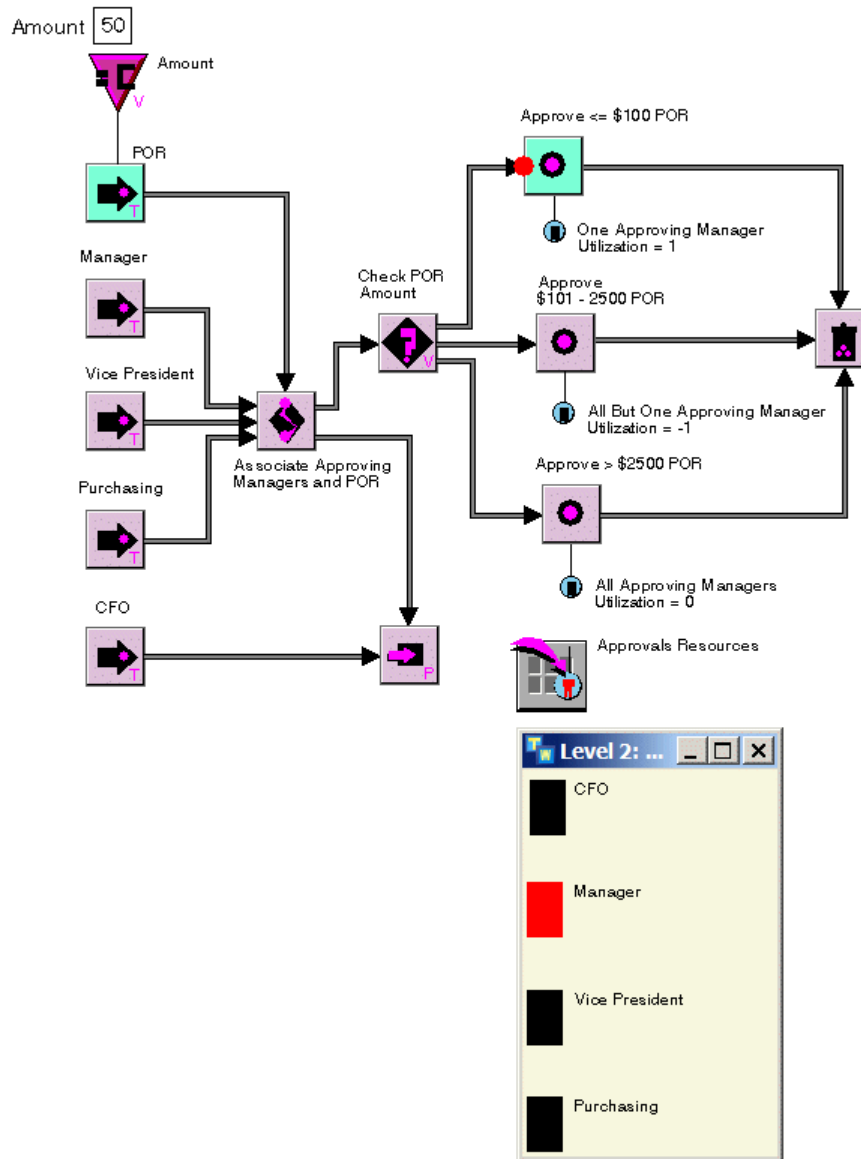
To allocate resources associated to work objects:

- 1 Create class definitions for one or more resources, which inherit from the `bpr-resource` class, or any subclass.
- 2 Build a model that associates the resources with a work object and store the resources in a pool.
To do this, use an [Associate block](#).
- 3 Create one or more Resources Managers from the pool and connect them to tasks that require those resources.
- 4 Display the properties dialog for the Resource Manager, click the Allocate tab, and configure the Association Name to be the name of the association created in step 2.
- 5 Configure Choose Resource on the Allocate tab to determine how the manager allocates resources from the pool.
- 6 Click the General tab and configure the Utilization to determine how many associated resources to allocate.

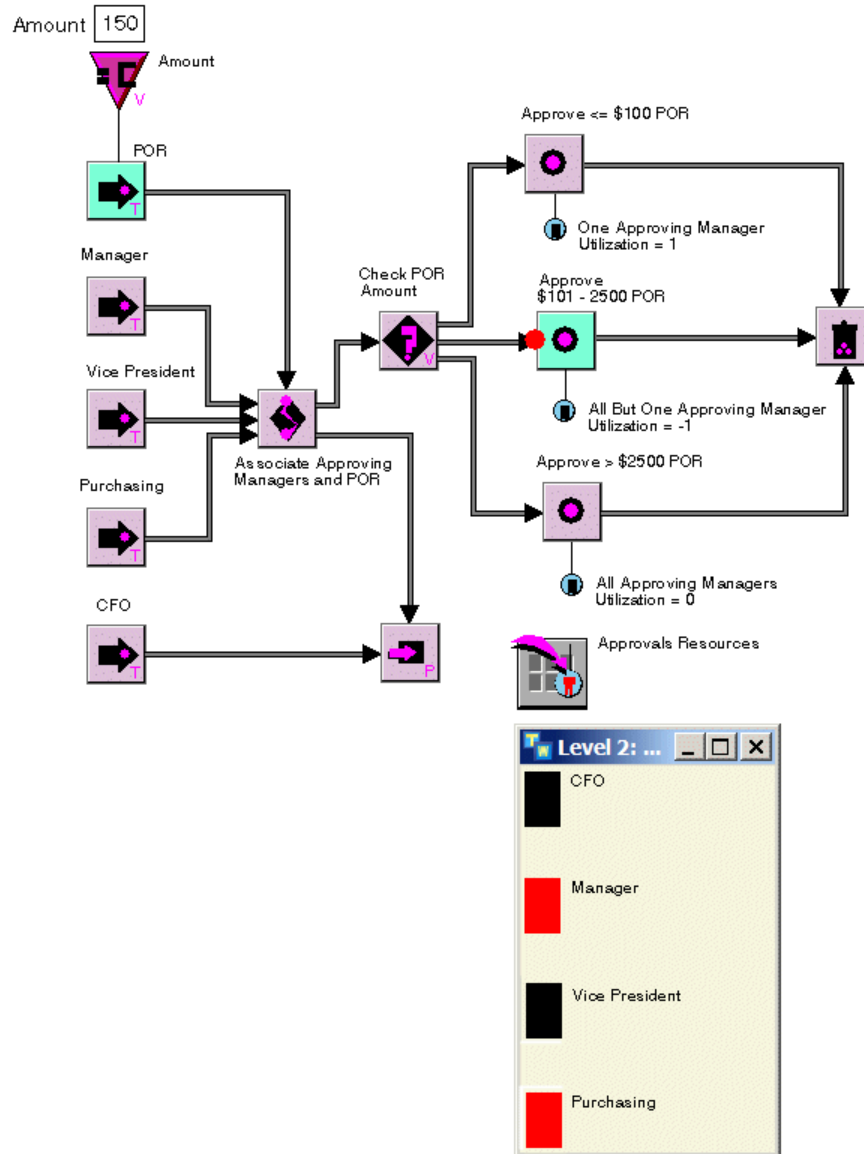
The following figures show three runs of a model for such a POR approval process. The model associates three categories of approving managers, a manager, vice president, and purchasing resource, with a POR, using the `approvals` association name. The model then stores the approving managers and a CFO resource in a pool. A Branch block tests the POR amount against a range of values to determine which POR approval task to execute.

The Approve POR tasks allocate associated resources based on priority, where each associate resource has a different priority.

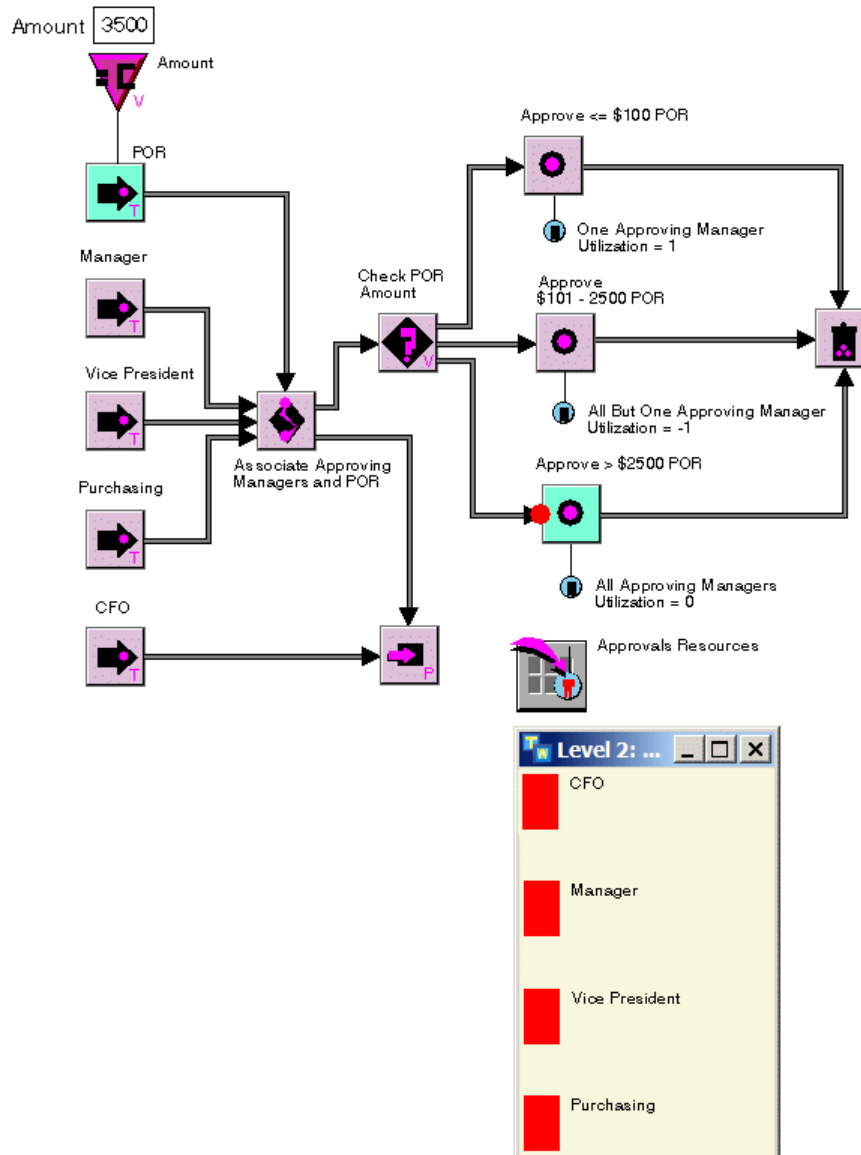
In the first simulation, the POR amount is \$50, which requires a single approving manager. The Manager resource has a priority of 1, thus the approval task allocates this associated resource to the task.



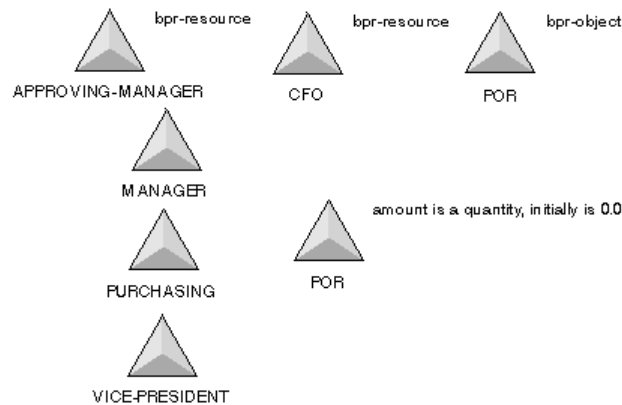
In the second simulation, the POR amount is \$150, which requires all but one approving manager. To specify this, the Utilization of the Resource Manager is -1. The Purchasing resource has a priority of 2, thus the approval task allocates the Manager and the Purchasing associated resources to the task.



In the third simulation, the POR amount is \$3500, which requires all approving managers, plus the CFO. To specify this, the Utilization of the Resource Manager that allocates associated resources is 0. Thus, the approval task allocates all associated resources plus the CFO to the task.



Here are the class definitions for the resources, the superior class of which inherits from the bpr-resource class, and the POR, which is a bpr-object:



Allocating the Same Resource to Different Blocks Based on Priority

You can allocate resources from the same pool to different blocks, using separate Resource Managers. For example, you might have two separate inventory tasks that require resources from the same clerks pool. By default, ReThink allocates resources at random to each block that requires the same resource.

You might have two tasks that require the same resources, one of which has a higher priority. You can specify the priority of each Resource Manager and allocate resources to the block with the highest priority first. If work objects are waiting on the input path of two tasks that allocate resources from the same pool, ReThink will always allocate resources to the task whose manager has the highest priority. This configuration ensures that the highest priority work is always performed first.

To allocate resources to different blocks based on priority:

- 1 Create a resource pool, then create and connect multiple Resource Managers, one for each task that requires resources from this pool.

For information on how to do this, see [Creating a Pool of Resources](#) and [Allocating the Same Pool to Multiple Tasks](#).

- 2 Display the properties dialog for each Resource Manager, click the Deallocate tab, and configure Choose Manager to be Priority.

The default value of Choose Manager is Random.

Note The value of Choose Manager for each manager should be the same.

3 Configure Manager Priority for each Resource Manager.

The smaller the number, the higher the priority. The manager with the highest priority allocates resources before the manager with a lower priority.

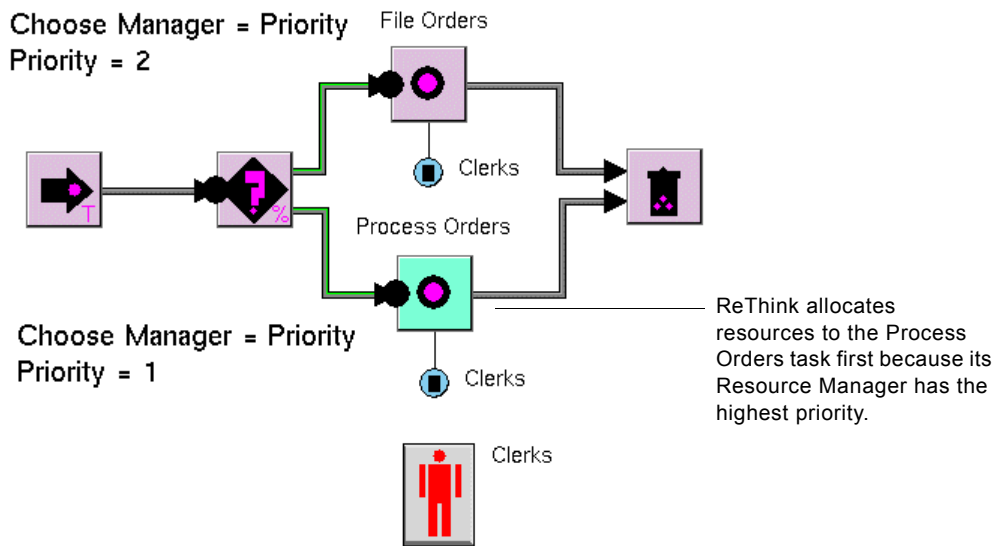
Note ReThink can only choose blocks based on the highest priority. If you want to allocate resources to blocks in the reverse order, redefine the priorities in the Manager Priority of each manager.

The following example illustrates how you can control the order in which various tasks execute. In this simple model, a Branch block passes orders to two separate tasks, each of which allocates resources from the same pool.

The Resource Manager connected to the Process Orders task specifies a Manager Priority of 1, and the Resource Manager connected to the File Orders task specifies a Manager Priority of 2. This means that the Process Orders task has a higher priority than the File Orders task; when resources become available, they will process orders before they file orders.

Both Resource Managers specify Choose Manager as Priority, which means the resources will go to the block with the highest priority, in this case, the Process Orders block.

This running model shows that when work is backed up on the input path to both the Process Orders task and the File Orders task, ReThink allocates resources to the Process Orders task first:



Creating Resources with Different Efficiency Factors

You might have a pool of resources in which certain resources are more efficient than others. You can set the efficiency factor of individual resources, which ReThink multiplies by the duration of the block to determine the duration of work objects and the utilization of resources. Combining the efficiency factor with priorities, you can cause ReThink to allocate the most efficient resources first in a model.

To specify the efficiency of individual resources in a pool:

- 1 Create a resource pool.
- 2 Display the properties dialog for each resource in the pool, click the Utilization tab, and specify the Efficiency Factor of each resource.

The Efficiency Factor is a multiplier that modifies the duration of the block to which the resource is allocated.

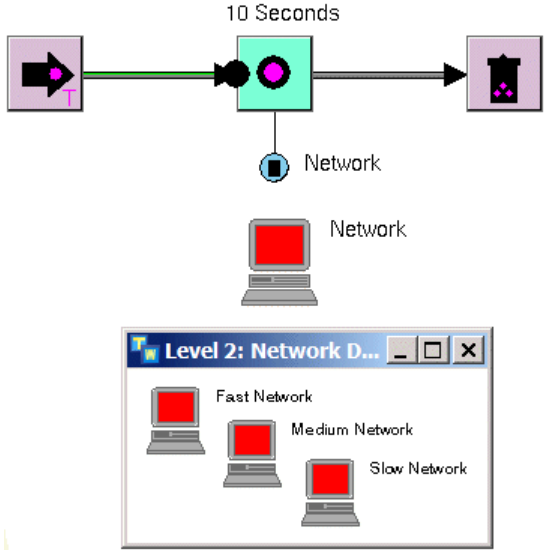
The following model shows how you can use efficiency factors combined with priorities to model the network speed of a pool of computer resources. The network consists of three networks:

- A Fast Network, whose Efficiency Factor is .75 and whose Priority is 1.
- A Medium Network, whose Efficiency Factor is 1.0 and whose Priority is 2.
- A Slow Network, whose Efficiency Factor is 1.25 and whose Priority is 3.

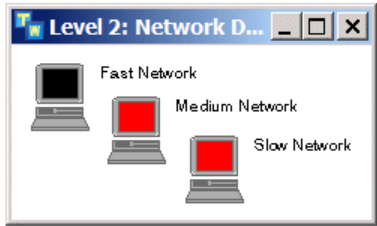
The Resource Manager is configured to choose the highest priority resource first, which means the fastest available network is always used first.

The duration of the task is set to exactly 10 seconds. The variation in the actual duration of the block is due to the allocated resource.

In the first step of the simulation, all resources are currently allocated. In the second step, the Fast Network is deallocated, and the work object that used this resource passes to the downstream block.

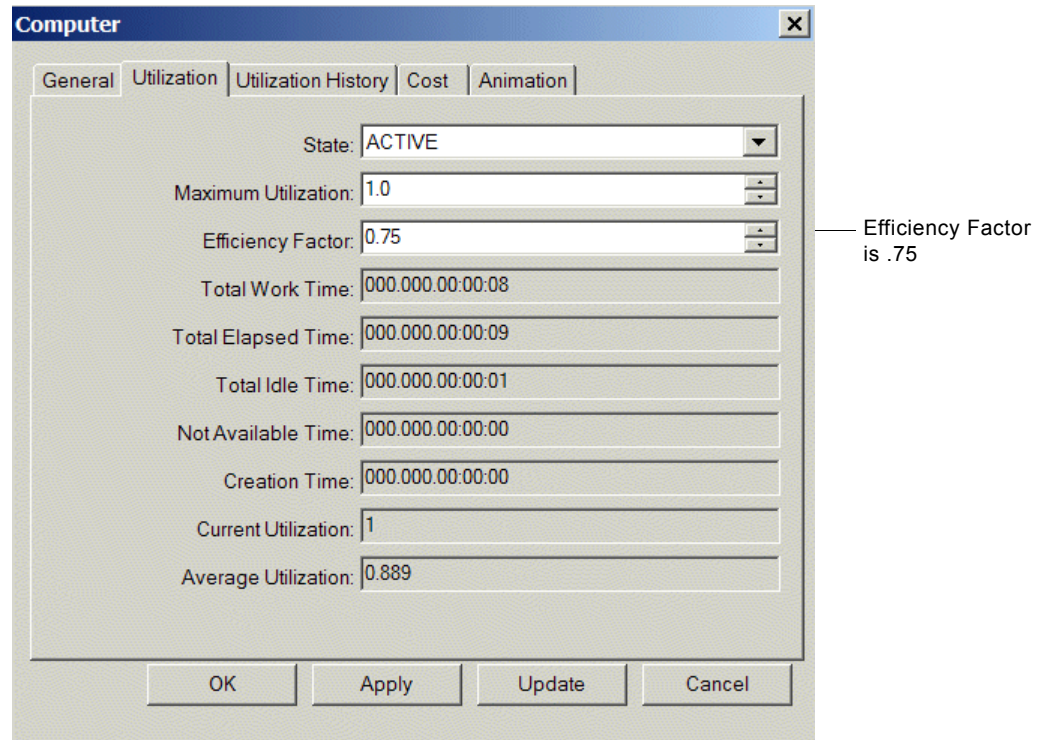


All resources are currently allocated.

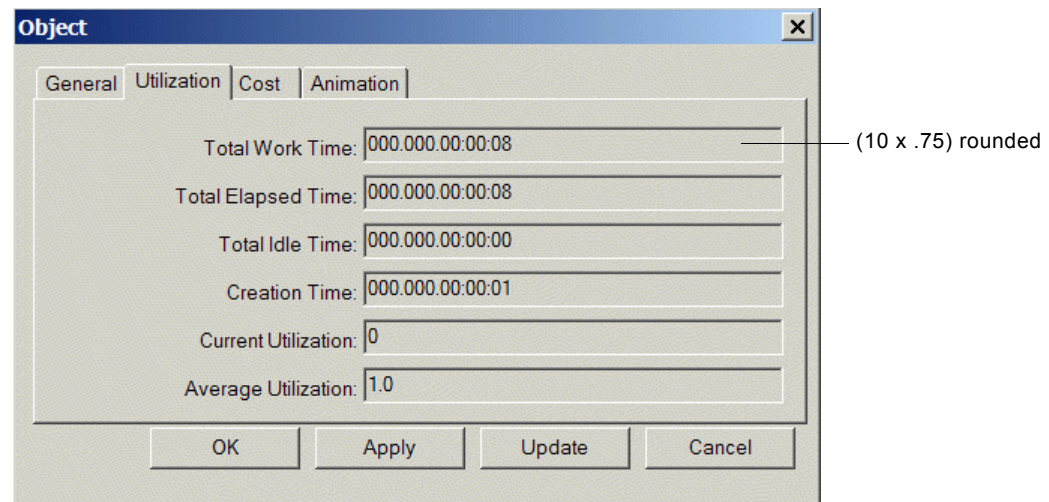


The Fast Network is deallocated.

Here is the Utilization tab of the properties dialog for the Fast Network:



Here is the Utilization tab of the properties dialog of the work object on the output path of the Task block. The Total Work Time reflects network speed of the chosen resource, which is the duration of the task times the Efficiency Factor, rounded to the nearest second.



Showing the Metrics of Resources

Just as you can show the metrics of a block, instrument, or work object, you can show the metrics of a resource in a dialog. The system-generated metrics of a resource are the same as those of a block or work object, with the addition of one metric.

Displaying Resource Metrics

The General tab of the properties dialog computes these metrics:

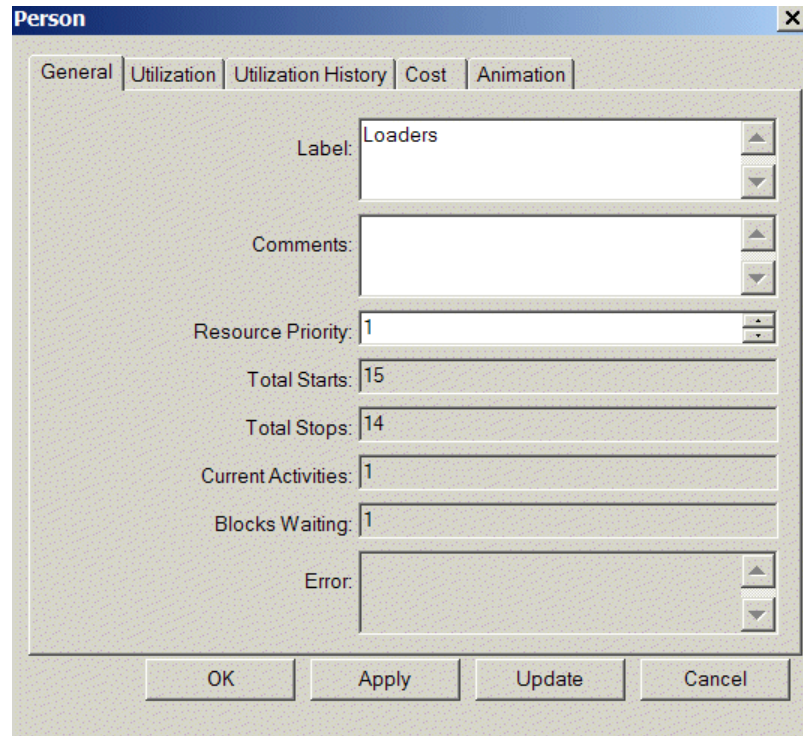
Computes this value...

This metric...	For an individual resource...	For the resource pool...
Total Starts	The total number of activities to which the resource has been allocated so far.	The sum of the total number of activities to which all the resources in the pool have been allocated so far.
Total Stops	The total number of activities to which the resource has been allocated and deallocated so far.	The sum of the total number of activities to which all the resources in the pool have been allocated and deallocated so far.
Current Activities	The current number of activities to which the resource is currently allocated.	The sum of the current number of activities to which all the resources in the pool are currently allocated.
Blocks Waiting	The total number of blocks that are waiting for that particular resource. In general, this metric will be 0 for a resource in a pool.	The total number of blocks that are waiting for any resource in the pool. The value of this metric is 0 when no blocks are waiting. The value is 1 when a single task is waiting for the resource. The value is 2 or greater when multiple tasks are allocating resources from the same pool, and more than one task is waiting for the resource.

To show the metrics of a resource:

- ➔ While the model is running, display the properties dialog for a resource and click the General tab.

ReThink displays a dialog similar to this:



For information on how to allocate the same resource pool to more than one task, see [Allocating the Same Pool to Multiple Tasks](#).

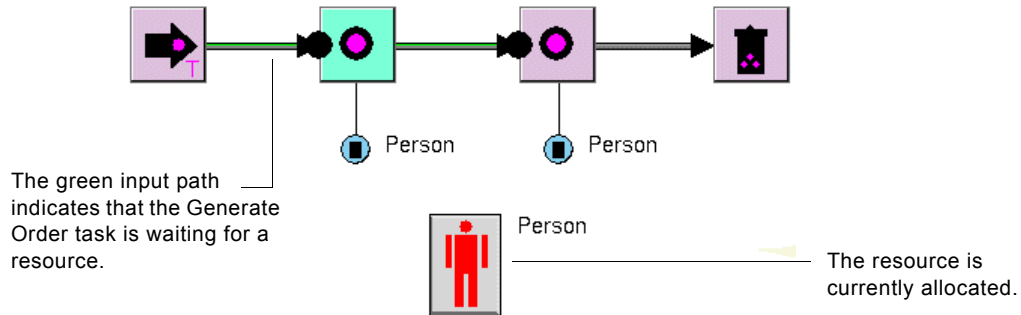
Note The Utilization of the Resource Manager does not affect any of these metrics for a resource.

Example of Allocating Resources

The following figure shows the General tab of the properties dialog for a running model in which two different tasks allocate the same individual resource. The resource has started processing 135 activities since the model started, and it has finished processing 134 activities. The difference between the Total Starts and the Total Stops is the Current Activities, which is always either 0 or 1 in this example. Because the resource is currently allocated, the Current Activities is 1.

Notice that the input path to the Generate Order task is green, indicating that it has work objects waiting in the queue for a resource. Thus, there is one block that

is waiting for the resource, which is indicated by Blocks Waiting. If a resource is allocated to a single task, Blocks Waiting will always be either 0 or 1, depending on whether the block is waiting for the resource.



Person

General Utilization Utilization History Cost Animation

Label: Loaders

Comments:

Resource Priority: 1

Total Starts: 15

Total Stops: 14

Current Activities: 1

Blocks Waiting: 1

Error:

OK Apply Update Cancel

Current Activities is 1, which means the resource is allocated.

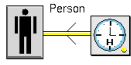
Blocks Waiting indicates that a single block is waiting for a resource.

Displaying Attributes with a Resource

You can add the current value of any attribute on the General tab of the properties dialog as an attribute display of a resource. For example, you might want to show the Priority of the resource as an attribute display.

For information on how to do this, see [Using Attribute Displays](#).

Constraining the Availability of Resources



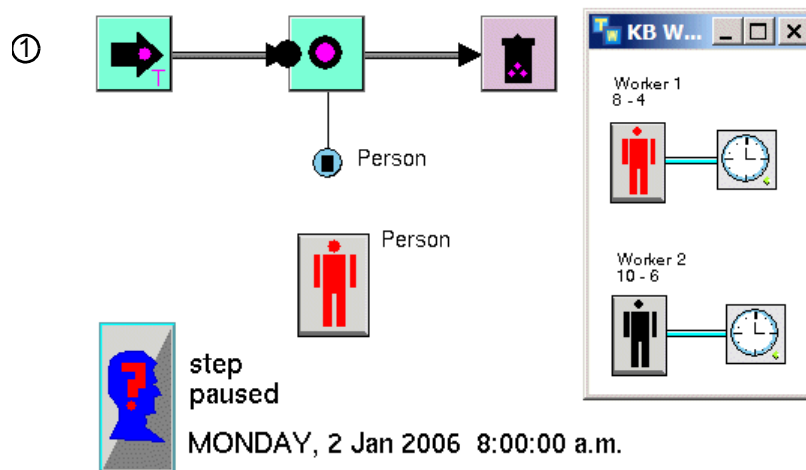
Typically, resources are only available for certain hours during the day and for certain days during the week. For example, most human resources work a nine to five work day and a five day work week. While hardware resources might be available twenty-four hours a day, they might be down for maintenance on a periodic basis.

You can constrain the availability of any resource in your model by using Temporal Constraints, or **constraints**, to specify the hours, days, months, and dates during which a resource is available. To do this, you attach a Temporal Scheduler to any individual resource and configure the constraints on the detail of the scheduler.

Note You can only constrain the availability of individual resources, not resource pools.

Allocating Resources With Constraints

ReThink takes into account temporal constraints when determining which resource to allocate. If a work object arrives at a block and no resources are available due to temporal constraints, ReThink allocates the resource that is available soonest. For example, suppose you have a pool of two resources, where Worker 1 is available from 8:00 to 4:00 and Worker 2 is available from 10:00 to 6:00, and suppose a work object arrives at the task requiring a resource at 7:00 AM. ReThink allocates Worker 1 to the task when the simulation time is greater than or equal to the first available time of the resource. The following figure shows this situation:



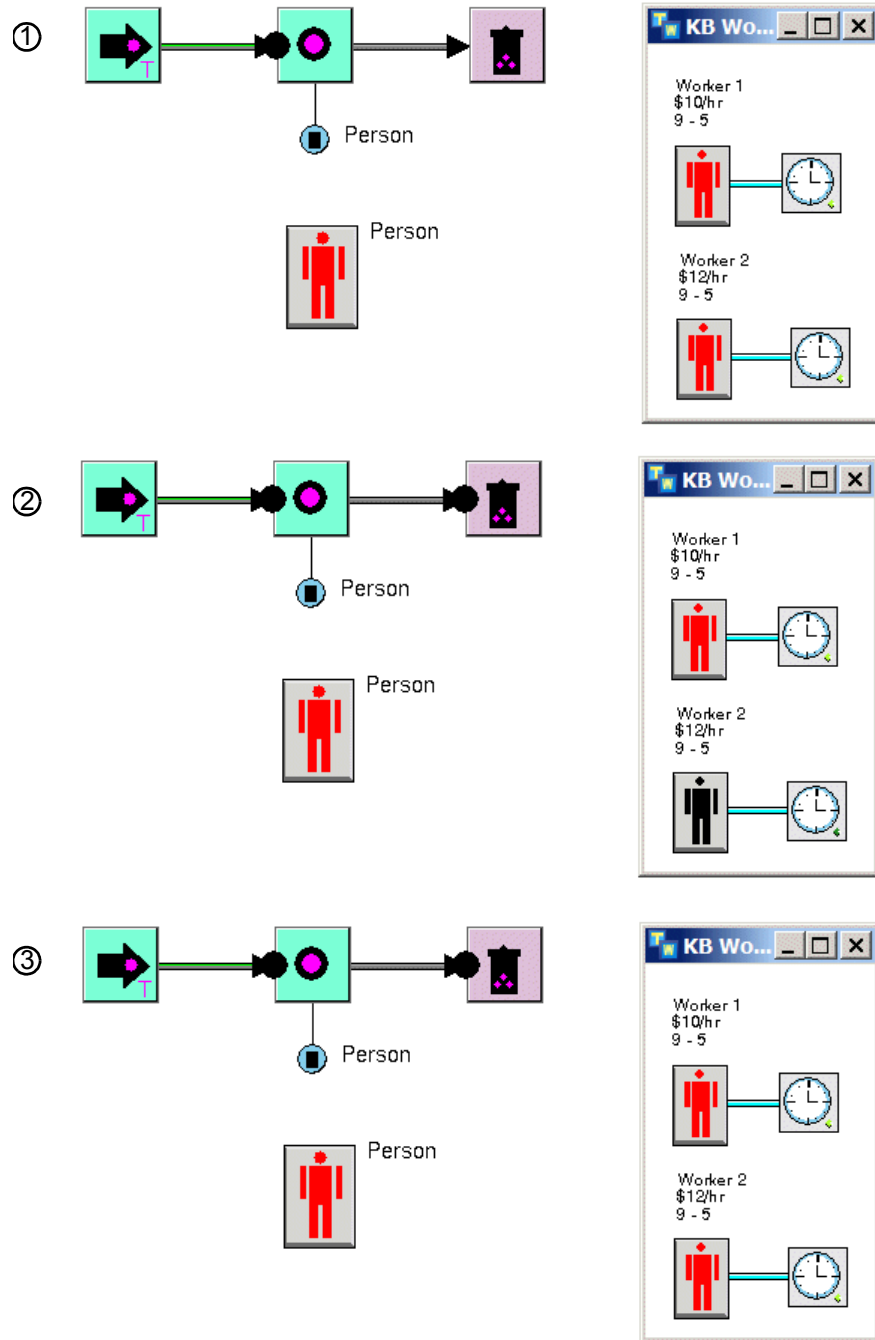
When work objects arrive at the block and no resource is available due to temporal constraints, ReThink assigns one work object to each resource that is not

currently allocated. ReThink places the rest of the work objects in the path queue until the simulation time matches the available time of the resource. For example, in the above model, between the hours of 6:00 PM and 8:00 AM, ReThink assigns two work objects to each resource in the pool, and it places the rest of the work objects that arrive during these hours in the path queue. The work objects that are assigned to the resources wait on the input path for the resources to become available.

If a work object arrives when both resources are available due to temporal constraints and when neither resource is currently allocated, ReThink allocates the resource according to the value of Choose Resource on the Allocate tab of the Resource Manager's properties dialog. For example, suppose the Resource Manager chooses resources based on the lowest cost, and suppose a work object arrives at 11:00 when both resources are available due to temporal constraints. If neither resource is currently allocated, the Resource Manager allocates the lowest cost resource, regardless of which resource is available soonest.

If both resources are available due to temporal constraints but are currently allocated, ReThink allocates the first resource in a pool to become available, regardless of how the Resource Manager chooses resources. For example, suppose you have a pool of resources, each of which is available from 9:00 to 5:00 and each with a different hourly cost, and suppose the Resource Manager chooses resources based on lowest cost. If all resources are currently allocated and a work object arrives at the block, ReThink allocates the first available resource to the waiting work object, regardless of the resource cost.

The following three steps in a model illustrate this situation, where each resource in the pool has a different cost. In the first model, both resources are currently allocated. In the second model, the resource that costs \$12 per hour becomes available. In the third model, the Resource Manager allocates the first available resource, even though it is the more expensive resource.



If a resource is currently allocated to a task and that resource becomes unavailable due to temporal constraints, ReThink does not switch the allocation of the current resource to use a different resource. The current resource remains allocated to the current task, even if a new resource is available due to temporal constraints and is not currently allocated or becomes deallocated.

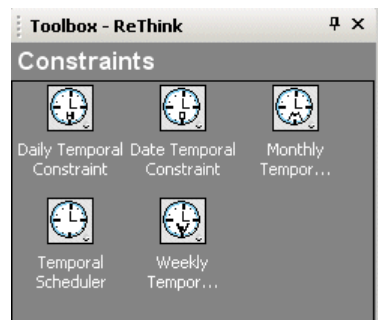
For example, if a resource is available from 9:00 to 5:00 and a work object arrives at a 2 hour task at 4:00, ReThink does not switch the allocation of the current resource to use a resource that is available from 5:00 to 12:00, even if that resource is not currently allocated or becomes deallocated after 5:00. Instead, the work object waits on the input path of the task until 9:00 the next day when the currently allocated resource is again available.

Displaying Constraints

The temporal constraints are located on the Constraints palette of the ReThink toolbox.

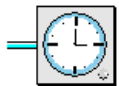
To display the Constraints tab of the ReThink toolbox:

➔ Display the Constraints palette of the ReThink toolbox:



The Temporal Scheduler contains a Monthly, Weekly, and Hourly Constraint on its detail, by default. You configure these constraints by showing the Temporal Scheduler detail. You can also create additional constraints and place them on the detail of a Temporal Scheduler to configure the date availability of a resource.

Constraining a Resource to Normal Business Hours



By default, the Temporal Scheduler object is configured to constrain the availability of the attached resource to normal business hours: eight o'clock to twelve o'clock and one o'clock to five o'clock, five days a week, fifty-two weeks a year. To constraint the model by using normal business hours, you simply attach a temporal scheduler object to a resource and run the simulation.

To constrain a resource by using normal business hours:

- 1 Create a resource and associated Resource Manager, and connect the manager to a block in the model.
- 2 Create a Temporal Scheduler from the Constraints palette of the ReThink toolbox and place it to the right of the resource.

Note You can also use the Make Temporal Connector menu choice on a resource to create a connection for a temporal scheduler.

- 3 Connect the temporal connector stub from the scheduler to the resource.
- 4 Run the simulation.

ReThink constrains the resource according to the default configuration of the scheduler, which has the following effect on the utilization metrics for the resource:

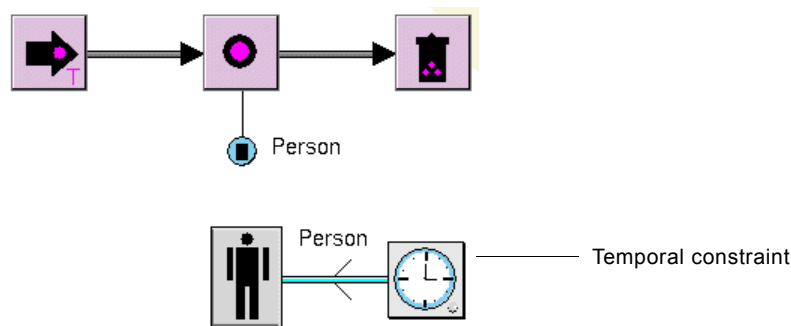
- The Total Work Time now only includes work time when the resource is allocated during normal business hours.
- The Total Elapsed Time includes the time when the resource is unavailable.
- The Not Available Time is the total amount of time that the resource was not available during the simulation.
- The Total Idle Time is the time that the resource was available but not allocated.

Thus, the utilization metrics of a resource relate to one another somewhat differently when the model uses constraints. With constraints, the relationship is:

$$\text{Total Work Time} + \text{Not Available Time} + \text{Total Idle Time} = \text{Total Elapsed Time}$$

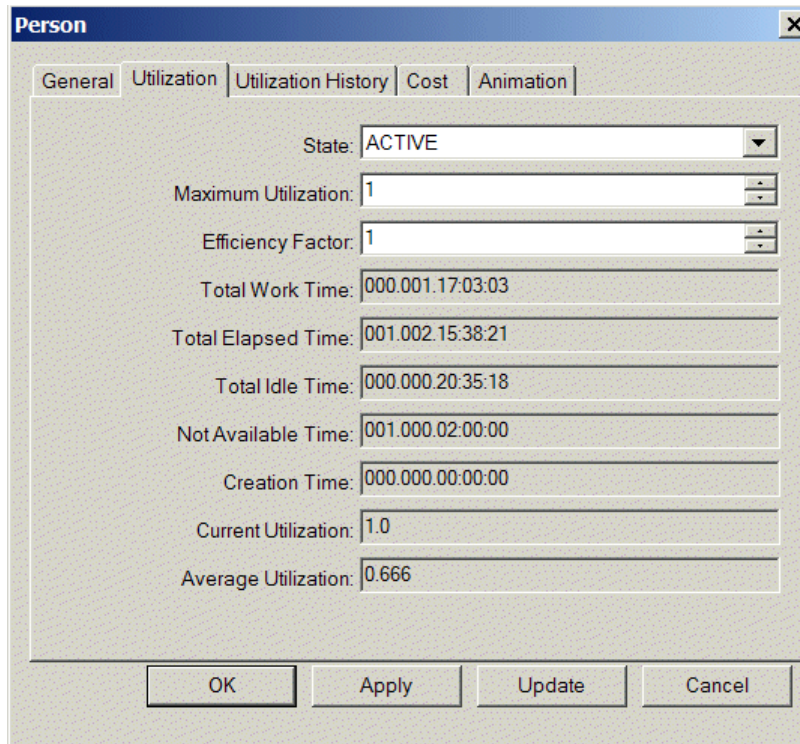
Keep in mind that the model can be constrained for reasons other than those due to temporal constraints: work backups and path synchronization can also cause Total Work Time to be less than Total Elapsed Time. Also, this relationship assumes the Maximum Utilization of the resource is 1.

This simple model shows a resource that is constrained by using normal business hours:



Here is the Utilization tab of the properties dialog for the resource after the model has been running for approximately one month. Notice that the Total Work Time is less than the Total Elapsed Time, because the resource was only available

during normal business hours. Also notice the Not Available Time, which reflects the amount of time the resource was not available to be allocated, and the Total Idle Time, which is the amount of time that the resource could have been allocated but wasn't.



The screenshot shows a dialog box titled "Person" with a close button (X) in the top right corner. The "Utilization" tab is selected, showing the following fields:

State:	ACTIVE
Maximum Utilization:	1
Efficiency Factor:	1
Total Work Time:	000.001.17:03:03
Total Elapsed Time:	001.002.15:38:21
Total Idle Time:	000.000.20:35:18
Not Available Time:	001.000.02:00:00
Creation Time:	000.000.00:00:00
Current Utilization:	1.0
Average Utilization:	0.666

At the bottom of the dialog are four buttons: OK, Apply, Update, and Cancel.

Configuring the Availability of the Resource

You configure the availability of the resource by configuring the constraints on the detail of the temporal scheduler. You can configure these constraints to determine when the resource is available:

- Hourly Constraint to determine the hours during the day when the resource is available.
- Weekly Constraint to determine the days of the week when the resource is available.
- Monthly Constraint to determine the months during the year when the resource is available.
- Date Constraint to determine the dates during the month when the resource is available.

Temporal Scheduler Detail

By default, the temporal scheduler detail contains three constraints: a Monthly Constraint, a Weekly Constraint, and a Hourly Constraint. You can attach a Date Constraint to the Monthly Constraint to configure the days during the month that the resource is available.

The constraints on the scheduler detail are connected in a particular order according to the colored temporal connector stubs. You can only connect constraints to other constraints with like-color stubs. Thus, you can only connect a Date Constraint to a Monthly Constraint, using the red stub.

Default Configuration of the Temporal Constraint Detail

By default, the constraints are configured so that the resource is available from 8:00am to 5:00pm, with one hour for lunch at 12:00pm, Monday through Friday, every month of the year.

Determining the Availability of Each Type of Constraint Visually

Each type of temporal constraint has a small dot in the lower-right corner of the icon. The color of the dot indicates the availability of resource for the particular constraint, according to this table:

This color...	Means the resource is...
Red	Completely unavailable.
Yellow	Partially available.
Green	Completely available.

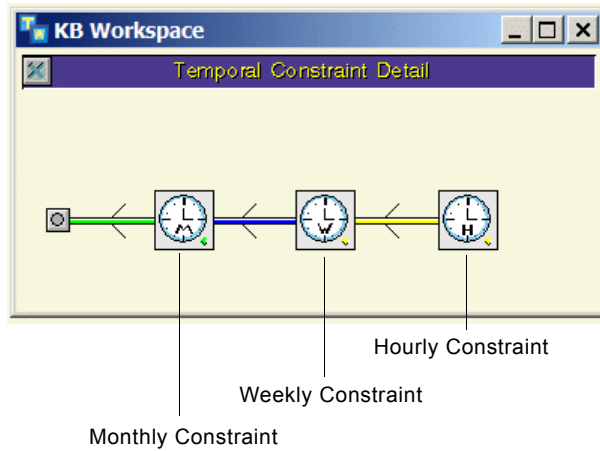
Displaying the Temporal Scheduler Detail

To configure the availability of resources, you must first display the detail of the temporal scheduler.

To display the detail of a temporal scheduler:

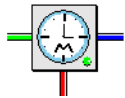
→ Choose Show Constraint on a temporal scheduler.

ReThink displays this detail:



The detail contains a Monthly Constraint, a Weekly Constraint, and a Hourly Constraint, all connected together using the colored temporal connector stubs. The constraint icons contain letters indicating the type of constraint.

Configuring the Monthly Availability

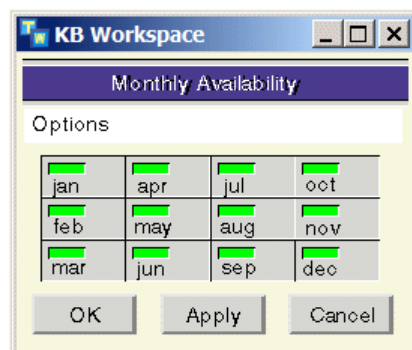


The Monthly Constraint allows you to configure the months during the year when the resource is available. The dot in the lower-right corner of the icon visually indicates the availability of the resource for the month, as described in [Determining the Availability of Each Type of Constraint Visually](#).

To configure the monthly availability of the resource:

- 1 Choose Show Constraint on the Monthly Constraint on the temporal scheduler detail, which is the first constraint from the left.

ReThink displays the detail of the Monthly Constraint:

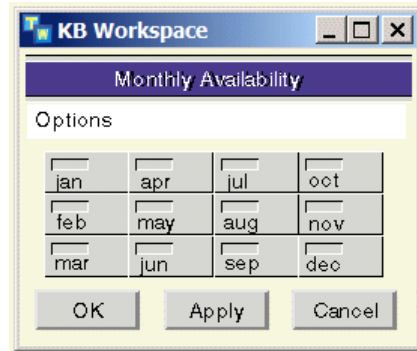


Each month has an associated colored region, which is green by default; the resource is available twelve months a year.

- 2 Do a combination of one or more of the following to specify the availability of the resource for the month:

➔ To make the resource unavailable for an entire month, click the right mouse button on the Options area at the top of the workspace and choose Set All Not Available.

The regions turn gray, indicating the resource is unavailable, as this figure shows:



➔ To make the resource available for an entire month, click the right mouse button on the Options area at the top of the workspace and choose Set All Available.

The regions turns green, indicating the resource is available.

➔ To make the resource available or unavailable for particular months, click the colored region above a month to toggle its availability.

- 3 Click OK to accept the changes.

Configuring the Weekly Availability



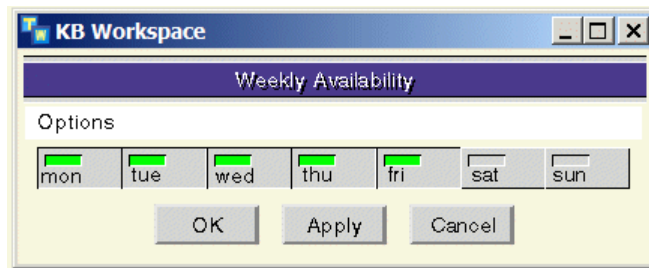
The Weekly Constraint allows you to configure the days during the week when the resource is available. The dot in the lower-right corner of the icon visually indicates the availability of the resource for the week, as described in [Determining the Availability of Each Type of Constraint Visually](#).

Note When using the Weekly and Hourly constraints together, you should set all the weekly constraints (Monday - Sunday) to **true**. Otherwise, the Hourly constraint might indicate that a resource is available on a certain day when the Weekly constraint indicates that it is not available. ReThink would actually jump ahead 1 year in the simulation.

To configure the weekly availability of the resource:

- 1 Click the Weekly Constraint on the detail of the temporal scheduler, which is the middle constraint, and choose Show Constraint.

ReThink displays the Weekly Constraint's detail:



Each weekday has an associated colored region. The regions associated with the five work days are green, which means the resource is available during the five work days of the week; the resource is unavailable on the weekend.

- 2 Do a combination of one or more of the following to specify the availability of the resource for the week:

→ To make the resource unavailable for an entire week, click the right mouse button on the Options area at the top of the workspace and choose Set All Not Available.

The regions turn gray, indicating the resource is unavailable.

→ To make the resource available for an entire week, click the right mouse button on the Options area at the top of the workspace and choose Set All Available.

The regions turn green, indicating the resource is available.

→ To make the resource available or unavailable for particular days, click the colored region above a day to toggle its availability.

- 3 Click OK to accept the changes.

Configuring the Hourly Availability



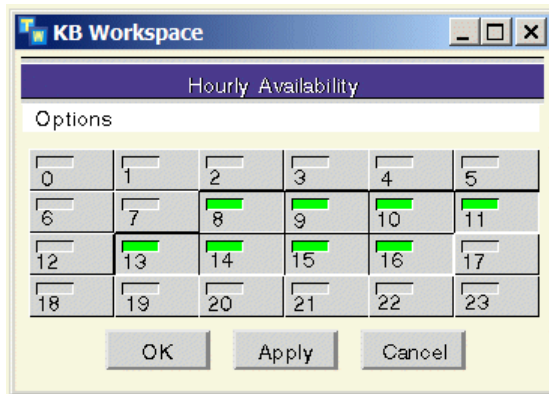
The Hourly Constraint allows you to configure the hours during the day when the resource is available. The dot in the lower-right corner of the icon visually indicates the availability of the resource for the day, as described in [Determining the Availability of Each Type of Constraint Visually](#).

Note When using the Weekly and Hourly constraints together, you should set all the weekly constraints (Monday - Sunday) to true. Otherwise, the Hourly constraint might indicate that a resource is available on a certain day when the Weekly constraint indicates that it is not available. ReThink would actually jump ahead 1 year in the simulation.

To configure the hourly availability of the resource:

- 1 Click the Hourly Constraint on the detail of the temporal scheduler, the last constraint on the right, and choose Show Constraint.

ReThink displays the Hourly Constraint's detail:



Each hour of the day has an associated colored region. The hours from 9:00 AM to 12:00 PM and from 1:00 PM to 5:00 PM are green. The detail uses a 24-hour clock.

- 2 Do a combination of one or more of the following to specify the availability of the resource for the day:
 - ➔ To make the resource unavailable for an entire day, click the right mouse button on the Options area at the top of the workspace and choose Set All Not Available.

The regions turn gray, indicating the resource is unavailable.
 - ➔ To make the resource available for an entire day, click the right mouse button on the Options area at the top of the workspace and choose Set All Available.

The regions turn green, indicating the resource is available.
 - ➔ To make the resource available or unavailable for particular hours, click the colored region above a hour to toggle its availability.
- 3 Click OK to accept the changes.

You might want a resource to become available on the half hour or some other fraction of an hour.

To indicate the minutes in the hour that the resource is available:

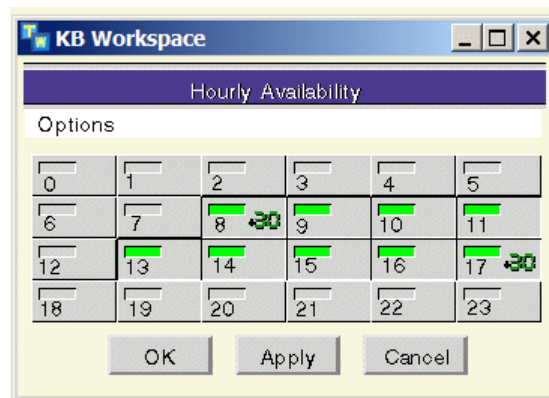
- 1 Click the right mouse button on the Options area at the top of the detail.
- 2 Choose the desired time interval in minutes that the resource is available.
For example, choose **+30** to specify that the resource is available on the half hour.

ReThink creates a time token and places it on the detail.

- 3 Drag the time interval to the desired hour to associate it with that hour.
For example, to make the resource available at 8:30 AM, drag the **+30** time token so that it is on the 8:00 hour.

- 4 Repeat this process for whatever hours you need to specify minutes.

For example, here is the Hourly Constraint detail such that the resource is available from 8:30 AM to 5:30 PM with an hour for lunch:



- 5 Click OK to accept the changes.

To delete a time token:

- Choose Remove on the time token.

Configuring the Date Availability



For any given Monthly Constraint, you can configure the individual days during the month when the resource is available by using a Date Constraint. For example, you might want to schedule vacation time during a particular month when a resource is unavailable.

You connect a Date Constraint to a Monthly Constraint, using the red temporal connector stub.

If you use a Date Constraint to constrain the availability of a resource in a single month, you must create a separate Monthly, Weekly, and Hourly Constraint for the month whose dates you want to constrain. Otherwise, if you want to constrain the date availability of all months identically, you can use a Date Constraint with a single Monthly Constraint.

The dot in the lower-right corner of the icon visually indicates the availability of the resource for the month, as described in [Determining the Availability of Each Type of Constraint Visually](#).

To configure the date availability of a resource:

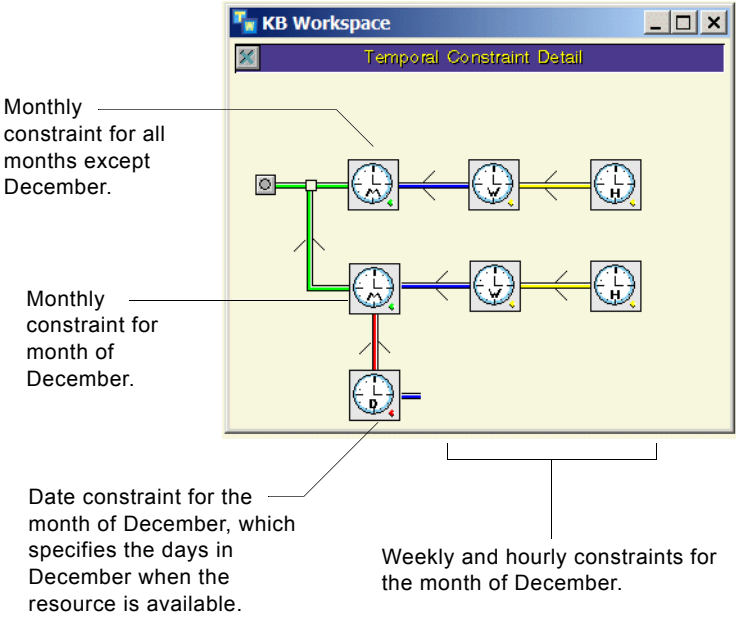
- 1** On the detail of a temporal scheduler, connect a Monthly Constraint to the path between the connection post and the existing Monthly Constraint, using a junction.
- 2** Configure the availability of the Monthly Constraint whose dates you want to constrain.

For example, if you are configuring a two week vacation in December, click the colored region above the month of December.

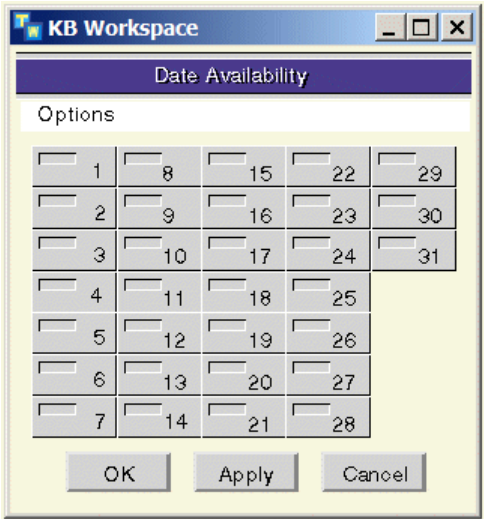
- 3** Connect a Date Constraint below the Monthly Constraint.
- 4** Connect a Weekly Constraint and an Hourly Constraint in sequence to the Date Constraint and configure the weekly and hourly availability of the resource during the dates when the resource *is* available.
- 5** Configure the availability of the other Monthly Constraint to make the resource available for the remaining months of the year.

In this example, you would click the highlighted regions above all months except December.

The temporal scheduler detail looks like this:



6 Click the Date Constraint and choose Show Constraint to display its detail:



Each date during the month has an associated colored region, which is gray by default; the resource is unavailable on every day of the month.

Note If the Weekly Constraint is configured so that the resource is unavailable on a particular day, and the Date Availability constraint is configured so that the resource is available on the same day, or vice versa, the Weekly Constraint takes precedence.

- 7 Do a combination of one or more of the following to specify the availability of the resource for the month:
 - ➔ To make the resource available for an entire month, click the right mouse button on the Options area at the top of the workspace and choose Set All Available.
The regions turn green, indicating the resource is available.
 - ➔ To make the resource unavailable for an entire month, click the right mouse button on the Options area at the top of the workspace and choose Set All Not Available.
The regions turn red, indicating the resource is unavailable.
 - ➔ To make the resource available or unavailable for particular days, click the colored region above a day to toggle its availability.
- 8 Click OK to accept the changes.

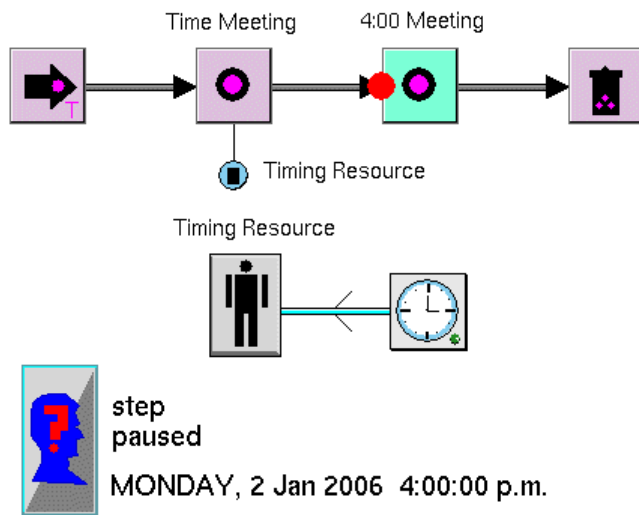
Using Constraints with Timing Resources

You can use resources in conjunction with Temporal Constraints strictly for timing purposes.

When a model requires this type of resource, you should not place it in either the Resource and Storage Pool organizers. Instead, we recommend that you place it near its associated Resource Manager with a brief text description of how the model uses it.

For example, suppose you wanted to model a meeting that takes place at exactly four o'clock. To do this, you would create a resource named Timing Resource with a Temporal Constraint. You would then attach the Resource Manager associated with the Timing Resource to a Task block, which limits the availability of the resource to four o'clock. You would then create a second Task block that would represent the meeting. When an object arrives at the first Task block, it waits until the Timing Resource is available, which is at four o'clock. It then passes the object to the 4:00 Meeting task.

The following example illustrates this process:



Note An alternative way of accomplishing the same task is to use a Batch block whose Batch Mode is Interval.

Configuring the Animation of Resources

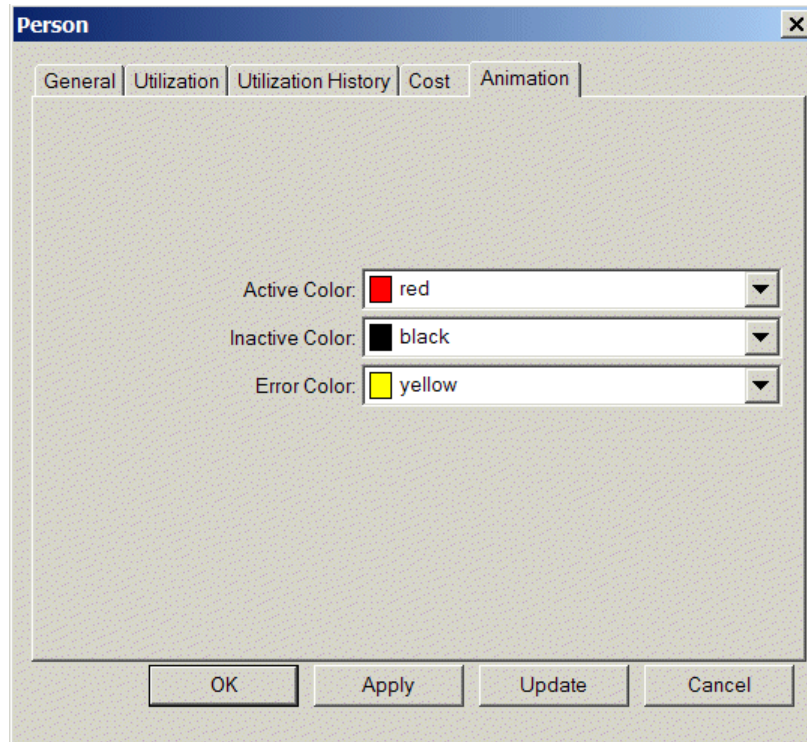
You can configure these colors of a resource or a surrogate when it animates:

Animation Color	Description
Active Color	The color the resource or surrogate uses when it is processing.
Inactive Color	The color the resource or surrogate uses when it is idle.
Error Color	The color the resource or surrogate uses when it is in an error state.

For information on using surrogates, see [Sharing the Same Resource in Multiple Pools](#).

To configure the colors of a resource or surrogate when it animates:

- 1 Display the properties dialog for the resource or surrogate and click the Animation tab to display this dialog:



- 2 Choose a color from the dropdown list for each resource color.

The Animation tab of the properties dialog for a surrogate has the same attributes.

Probing the Performance of Resources

When you are creating models with resources, you often want to obtain performance metrics regarding the resources in the model. For example, you might want to create a chart that plots a history of the:

- Average utilization of the current resource allocated by a task.
- Total cost of all the resources in a pool.
- Average total cost of all resources in a pool.

To probe the performance of resources, you have three options:

To obtain performance metrics about...	Probe the...
The resource that is currently allocated by a task	Block that requires the resource.
The sum of all resources in a pool	Resource pool.
A specific resource in a pool	Individual resource.

For information about and examples of these three techniques of probing resources, see [Three Techniques for Probing Resources](#).

To obtain performance metrics that represent the average of the sum of all resources in a pool, you perform computations on the probed values, using the remote. For example, if three resources are in a pool and you want to compute the average total cost of all the resources, you probe the resource pool and divide the resulting values by 3.

Populating Resource Pools Dynamically

Rather than manually adding resources to a pool, you might want to add resources to a pool dynamically as part of processing. For example, the number of computer resources that are available to a task might depend on financial factors that the model determines upstream in the process.

Once the model populates a pool with resources, the model allocates the resources to tasks, using Resource Managers.

Note When you reset the model, ReThink deletes dynamically created resources.

To populate a resource pool dynamically, you must create a new resource class definition with the desired appearance. You then store these resources dynamically to a pool.

To store resources to a pool dynamically:

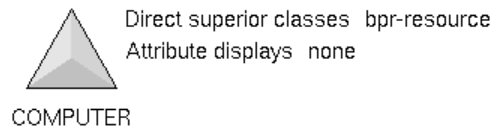
- 1 Create a resource pool from the Resources palette of the ReThink toolbox.
- 2 Create an object definition for a new class of resource, whose Direct Superior Classes is `bpr-resource`.

This object will serve as the resource the model will store dynamically in the pool.

By default, subclasses of `bpr-resource` automatically include an attribute display for their Label.

- 3 Configure the Attribute Displays of the class definition to be `none` so that the label will not display with the resource as it moves through the model.
- 4 Edit the icon for the resource, as needed.

This figure shows a class definition for a dynamically created resources named `computer`. The attribute displays show the attributes whose values are specified in the class definition. You can add computer resources to a pool dynamically.



Customizing Resources

You can customize how a Resource Manager allocates and deallocates resources from a pool. You can also customize how ReThink computes duration and cost metrics for resources.

For detailed information about how to customize these objects, see the *Customizing ReThink User's Guide*.

Using Work Objects

Describes how ReThink uses work objects, which are objects that blocks create, process, and delete when a model is running.

Introduction	345
Configuring Path Types	347
Comparing Work Objects and Resources	352
Understanding the Activities of Work Objects	353
Computing Utilization and Duration Metrics	354
Working with Work Object Costs	359
Customizing Work Objects	360



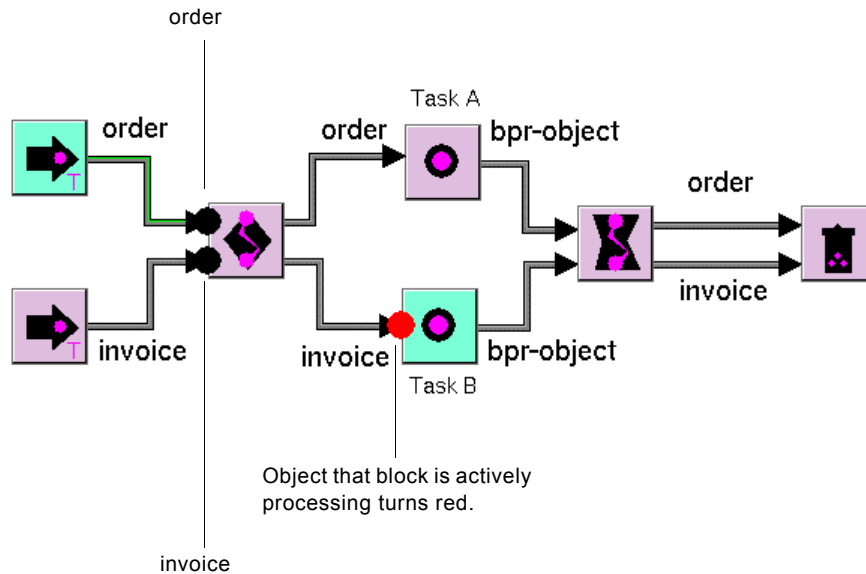
Introduction

The basic mechanics of a ReThink model involves creating, processing, and deleting objects in a business process. For example, a model of a sales process might:

- Generate leads.
- Create orders based on those leads.
- Generate invoices associated with the orders.

In such a model, the lead, order, and invoice are all **work objects**.

This running model shows several work objects. The active work objects are red and the work objects that are waiting are black. The model processes orders, generates invoices, then associates the orders and invoices.



Work objects, like other objects in ReThink, compute various metrics, which help you determine the overall performance of a model. For example, a work object keeps track of the total cost of all activities that the model has performed on it since it was created. By charting the total cost of a work object, you can get a sense of the total value-added that all the tasks in a process have contributed to a work object.

A work object also keeps track of the total amount of time that it is being worked on over the course of the simulation. By comparing these two numbers to compute the average utilization of the work object, ReThink can report on the overall performance of the model.

For example, if the average utilization of a work object is very low, you know that bottlenecks in the process are causing inefficiencies.

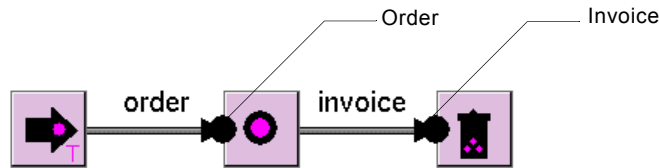
You obtain key performance metrics about your model by probing the work objects that the model processes.

For general information about using ReThink probes, see [Probing the Performance of Your Model](#).

For specific information about probing work objects, see [Probing the Performance of Work Objects](#).

Configuring Path Types

When you configure a model, you identify the kind of work the block processes by specifying the output path types of the block:



You can specify any one of three categories of work objects as the path type for a block:

- [The default work object type](#), which is bpr-object.
- [A container object](#), which is bpr-container, for use when you insert and batch objects into a container.
- [A user-defined work object](#), which inherits its definition from bpr-object or bpr-container.

Using the Default Path Type

- By default, the value of the Type parameter of a path is bpr-object, which allows any type of work object to flow on the path. If an upstream block specifies a user-defined object as the path type, the downstream block can generally use the default path type to process the user-defined object.

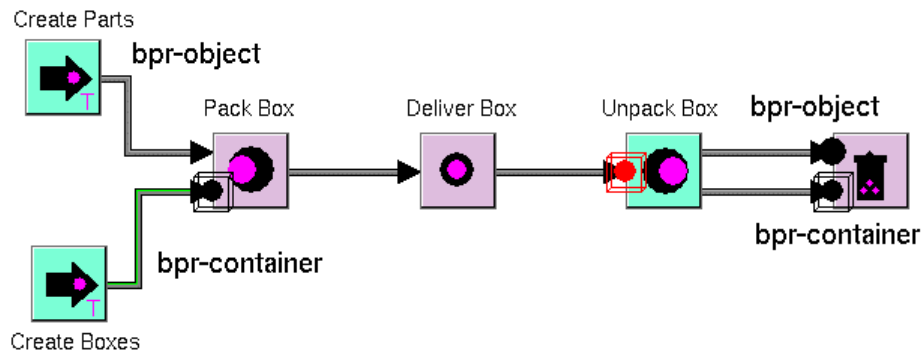
For examples of using the default path type, see [Configuring the Type of Work that Blocks Process](#).

Specifying a Container as the Path Type



Certain blocks operate on a container, which defines an item-list attribute that stores objects that the model processes.

For example, this running model shows how you would use containers to pack and unpack boxes, using the Insert and Remove blocks, respectively:



You also use containers with the Batch blocks to insert batches of objects into a container. For more information on these blocks, see the [Insert block](#), the [Remove block](#), and the [Batch block](#).

To use a container object in your model:

- ➔ Display the properties dialog for a path and configure the Type to be `bpr-container`.

The container object defines an attribute named `container-list`, which is an instance of an item-list. When you specify the Container List Attribute of an Insert, Remove, or Batch block, the default value is the `container-list` attribute of the container object.

To view the objects in the container:

- ➔ Choose Snapshot Container on the container.

ReThink displays a workspace with all the objects in the container.

You can create a subclass of `bpr-container`, as described in [Creating a New Class of Work Object](#).

Specifying a User-Defined Object as the Path Type

When you create a model, you can specify any type of work object as the value of the Type parameter of the path. For example, your model might process orders and invoices.

When you specify a user-defined object, you can either let ReThink create the class definition for you automatically, or you can create your own class definition, depending on the requirements of the model.

To use a user-defined object in your model:

- Display the properties dialog for a path and configure the Type to be any user-defined class name.

Automatically Generating the Work Object Class Definition

The first time ReThink encounters a user-defined work object as the path type that is not defined, it dynamically creates a class definition for the work object and places it on the model workspace. This new class definition is a subclass of `bpr-object`.

Once the user-defined class definition exists, ReThink uses that definition for the work objects that the model creates and processes. You can edit the class definition to specify user-defined attributes, as needed. For details, see [Creating a New Class of Work Object](#).

Typically, you transfer these class definitions to the detail of an Organizer to keep the definitions separate from the model.

Creating a New Class of Work Object

In some cases, the model requires that you create your own work object class definition or modify the default work object. You need to do this whenever you use a non-default attribute in a model, for example, when you use certain feeds to supply values to user-defined attributes of work objects or when you branch work objects based on a user-defined attribute of the model.

For an example of feeding values to user-defined attributes of a model, see [Updating User-Defined Attributes of a Work Object](#).

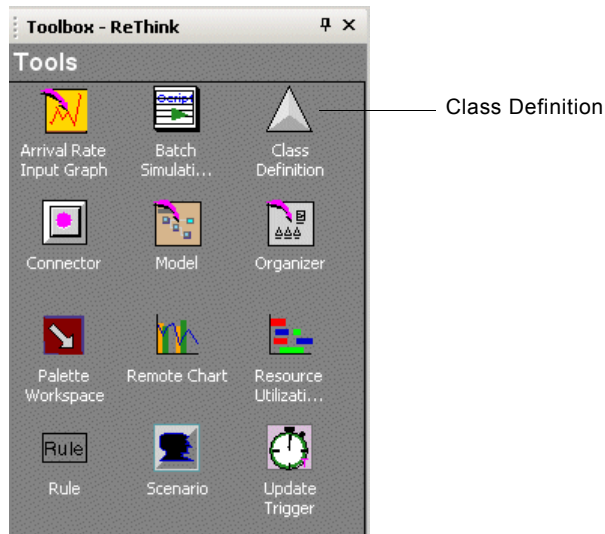
The classes you define for the ReThink model can have a rich set of attributes, substructure, relations to other classes, and other detailed complexity. Once you have created an alternative model of your process, you can use those same classes as the basis for implementing the information systems you will use to support your reengineered business process.

Typically, you edit the icon for the work object to distinguish it from the default work object icon.

For an example of the properties dialog for a user-defined class of work object, see [Viewing User-Defined Attributes of Work Objects](#).

To create a class definition for a work object:

- 1 Display the Tools palette of the ReThink toolbox:



- 2 Create a Class Definition and place it on a workspace.

Tip You typically place class definitions on the detail of an organizer.

- 3 Display the properties dialog for the Class Definition.
- 4 Configure the Class Name of the definition to be a unique symbol.

Note The class name of a definition cannot contain spaces; use hyphens in place of spaces, for example, **sales-order**.

- 5 Configure the Direct Superior Classes to specify the superior class.
The options are bpr-object or bpr-container or any subclass.
ReThink automatically fills in the table for the class, using inherited attributes.

- 6 Configure the Class Specific Attributes to create one or more user-defined attributes.

When you specify multiple attributes, separate each attribute specification with a semi-colon. You can specify any valid value type that G2 supports, including subobjects.

Here is an example of a Class Specific Attributes specification:

```
mileage is an integer, initially is 0;  
total-mileage is an integer, initially is 0
```


In general, you should specify a type and/or default value for class-specific attributes so ReThink can choose the correct control for the dialog.

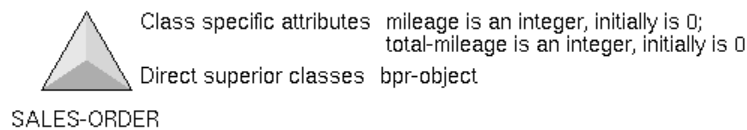
If the model is feeding values into a user-defined attribute or accessing values from an external file, it is not necessary to configure the attribute type; ReThink can determine the type from the external source. For example, you do not need to specify the type when sourcing objects from an external file, using a Source block, or when feeding a timestamp into a user-defined attribute of an object, using a Timestamp feed.

If the model is accessing values from a database for a database record, you should *not* configure the type; otherwise, ReThink will create an instance of the record with the attribute values you specify rather than accessing the record from the database. See [Creating a Work Object that Represents a Record](#).

7 Edit the icon for the work object, as needed.

For details on configuring class-specific attributes and icons, see the *G2 Reference Manual*.

Here is a fully specified ReThink work object class definition:



For general information on how to create G2 class definitions, see the *G2 Reference Manual*.

Viewing User-Defined Attributes of Work Objects

You can view class-specific attributes of a user-defined object in the properties dialog of the object as it moves through the model during a simulation.

By default, the user-defined attributes appear on the User tab of the dialog. The type of control that appears depends on the value type of the attribute, as follows:

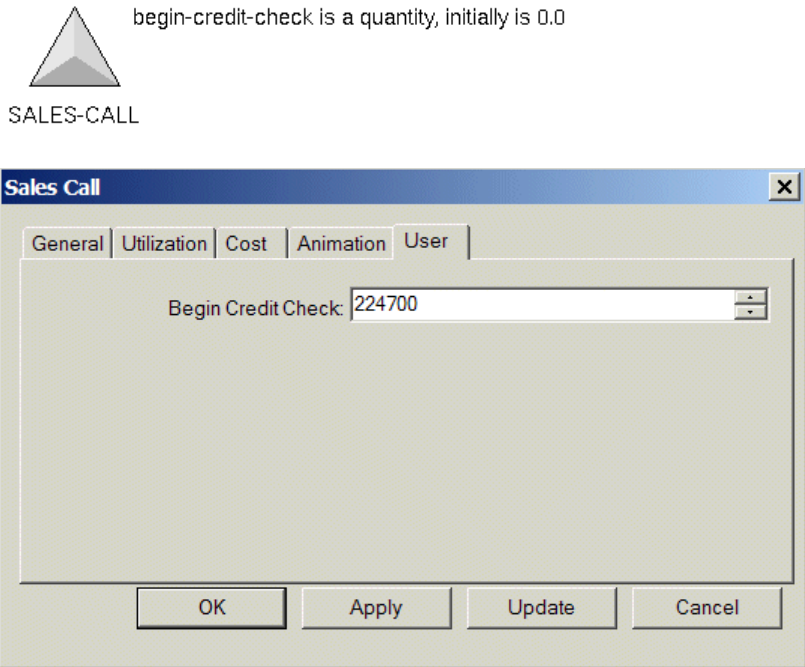
- Quantity, integer, and float values appear as spinner controls.
- Text values appear as scrollable text boxes.
- Symbolic values appear in type-in boxes.
- Truth values appear as check boxes.
- Subobject appear with an ellipses button, which the user can click to view the attributes of the subobject.

If the object has too many class-specific attributes, they appear on multiple User tabs, for example, User (1) and User (2).

To view user-defined attributes of a work object:

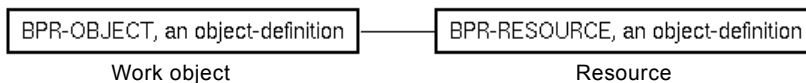
- 1 Run the simulation in step mode until you see a work object.
- 2 Display the properties dialog of the work object whose attributes you want to display and click the User Tab.

For example, here is the User tab for the **sales-call** class, which defines class-specific attribute:



Comparing Work Objects and Resources

In most ways, a work object is identical to a resource. Work objects have essentially the same attributes as a resource. In fact, in the ReThink class hierarchy, a resource is a subclass of a work object, as this hierarchy shows:



These differences exist between work objects and resources:

- Work objects are typically transient and resources are typically permanent.
- You can set the utilization and cost of a resource, whereas it does not make sense to set these values for a work object.

- A Resource Manager can only allocate and deallocate resources, not work objects.
- A resource computes the Not Available Time based on temporal constraints.

When you start a model running, ReThink generates and processes work objects. These objects are transient, which means they exist only during the simulation; when you reset the model, ReThink deletes all the work objects.

However, when you manually create a resource and place it on a workspace or in a pool, the resource remains in the model even when you reset. This is because resources are permanent objects in the model.

When you dynamically store resources in a pool for allocation and deallocation by a Resource Manager, you should only use objects that are a subclass of `bpr-resource`.

Note If you add resources to a pool dynamically and you reset the model, ReThink deletes the dynamically created resources.

Understanding the Activities of Work Objects

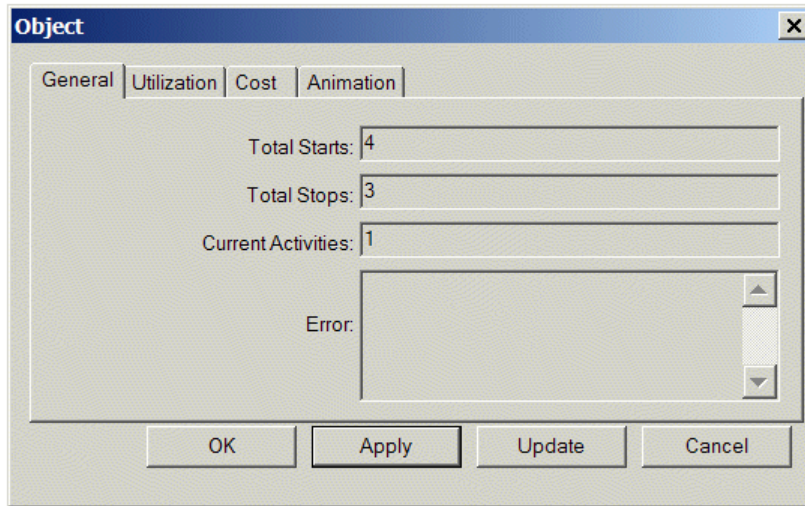
The General tab of the properties dialog shows these attributes related to activities of the work object:

Attribute	Description
Total Starts	The total number of activities that have been started for the work object since the start of the simulation.
Total Stops	The total number of activities that have finished for the work object since the start of the simulation.
Current Activities	The number of activities that are currently applied to the work object.

To show the attributes of a work object:

- 1 Run the simulation in step mode.
- 2 Display the properties dialog for the work object whose attributes you want to analyze and click the General tab.

ReThink displays the General tab of the properties dialog:



Computing Utilization and Duration Metrics

Work objects compute a number of metrics relating to utilization, which you can use to analyze the performance of your business model. You can also compute certain metrics, using instruments. For example, you can analyze the:

- Total work applied to a work object by all activities in the process.
- Average utilization of the work object, which is the amount of time that the work object has been active over the total life of the simulation.
- The difference in the creation of a work object and a timestamp downstream in the process, which is called **cycle time**.

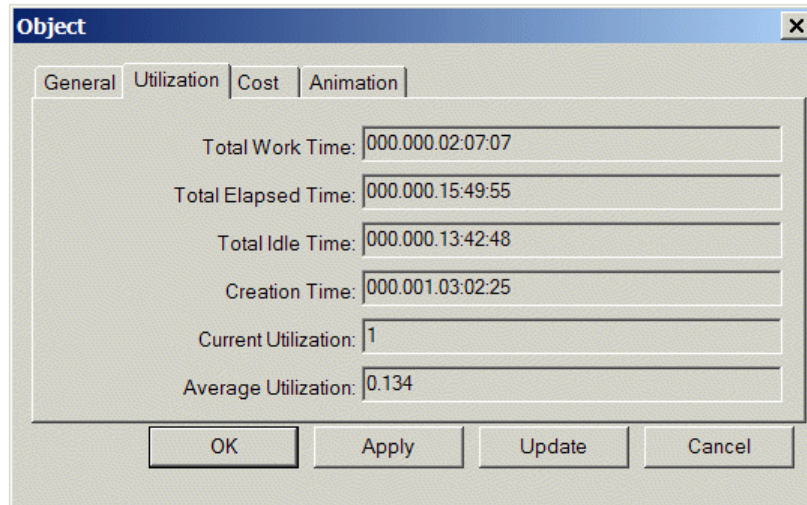
Computing Utilization Metrics

All information related to the utilization of a work object is contained on the Utilization tab of the properties dialog.

To display utilization metrics for a work object:

- 1 Run the simulation in step mode.
- 2 Display the properties dialog for the work object whose duration you want to analyze and click the Utilization tab.

Here is the Utilization tab of the properties dialog for a work object that is currently active:



The following headings explain these metrics and give examples under conditions of no constraints and under conditions of constraints.

Understanding the Duration Metrics of a Work Object

Each work object computes these duration metrics:

Attribute	Description
Total Work Time	The sum of all the work times for each activity that has processed the work object so far in the simulation.
Total Elapsed Time	The total amount of time that the work object has existed during the simulation.
Total Idle Time	The amount of time that the work object has been idle due to constraints on the model. The Total Idle Time is the difference between the Total Elapsed Time and the Total Work Time.
Creation Time	The current simulation time at which the work object was created.

If the model has no constraints, the Total Work Time is equal to the Total Elapsed Time, and the Total Idle Time is zero.

If the model has constraints, the Total Work Time can be less than the Total Elapsed Time due to work backups, and the Total Idle time is a positive number. If the Total Idle Time is high, the process is not very efficient because the work object is waiting for resources too much of the time.

For more information on how to constrain the model and detect work backups, see [Showing Work Backups on an Input Path](#).

Understanding the Utilization of a Work Object

The Utilization tab of the properties dialog of a work object computes these utilization metrics:

Attribute	Description
Current Utilization	The current status of the work object. If the work object is currently being worked on by a block, the value is 1; if the work object is waiting due to constraints on the model, the value is 0.
Average Utilization	The amount of time that the work object has been active compared to the amount of time that it has been idle, over the entire time that the work object has existed. Specifically, it is the ratio of the Total Work Time and the Total Elapsed Time.

If the model has no constraints, the Average Utilization is 1.0.

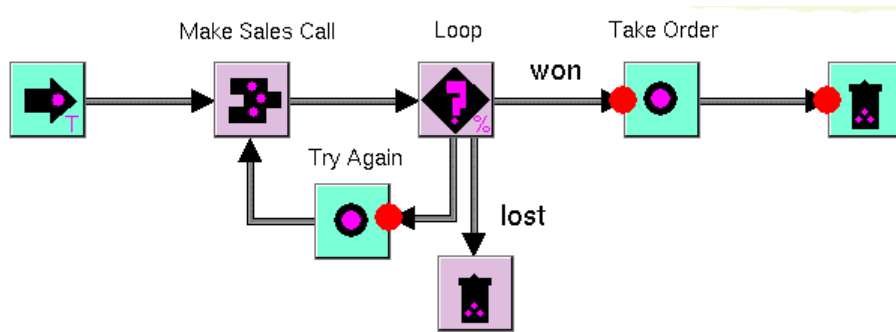
If the model has constraints, the Average Utilization can be a fraction when the Total Work Time is less than the Total Elapsed Time.

If the Average Utilization is low, this implies that work backups exist due to constraints on the model, which means that the process is not very efficient.

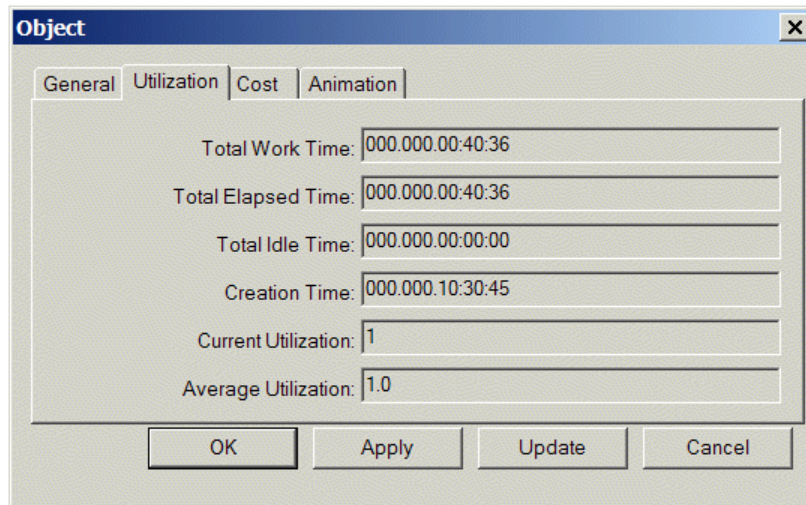
For more information on constraining the model and detecting work backups, see [Showing Work Backups on an Input Path](#).

Example of Computing Utilization Metrics With No Constraints

If the model has no constraints, the Total Work Time of a work object is always equal to the Total Elapsed Time, and the Total Idle Time is zero, as this example shows:



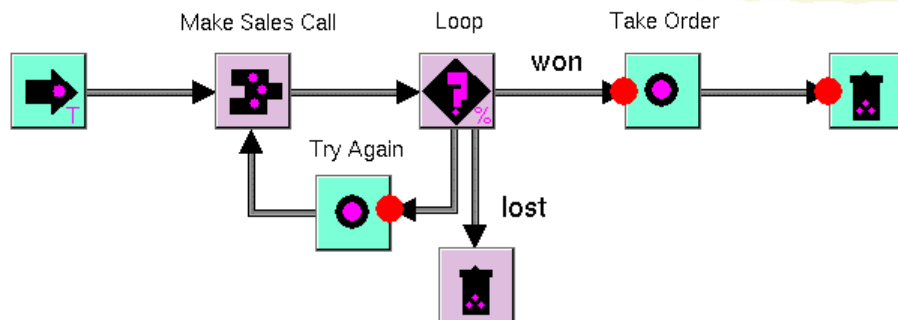
Here is the Utilization tab of the properties dialog for a sales call work object at the end of the process when the model has no constraints. Notice that the Total Work Time and the Total Elapsed Time are equal, and the Total Idle Time is 0. The Average Utilization is the ratio of Total Work Time to Total Elapsed Time, which is 1.0.



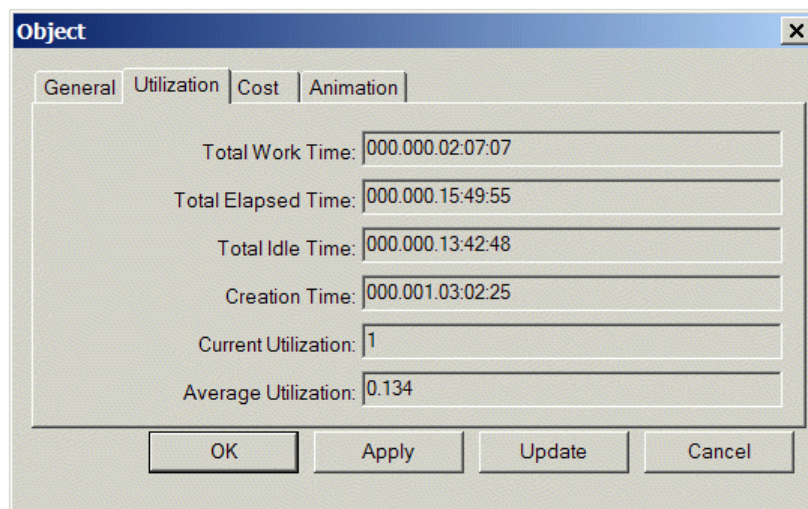
Example of Computing Utilization Metrics With Constraints

If a model has constraints and if the duration of the activities are such that a work object is required to wait on the input path to a block, the Total Work Time will be

less than the Total Elapsed Time for the work object, and the Total Idle Time will be a positive number as this model shows:



Here is the Utilization tab of the properties dialog for a sales call work object at the end of the process, when the Total Work Time is less than the Total Elapsed Time due to resource constraints, and the Total Idle Time is a positive number. Notice that now the Average Utilization is a fraction.



For more information on how to constrain the model and detect work backups, see [Showing Work Backups on an Input Path](#).

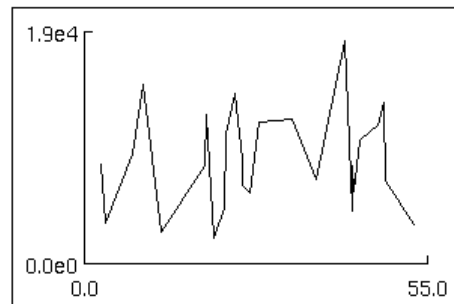
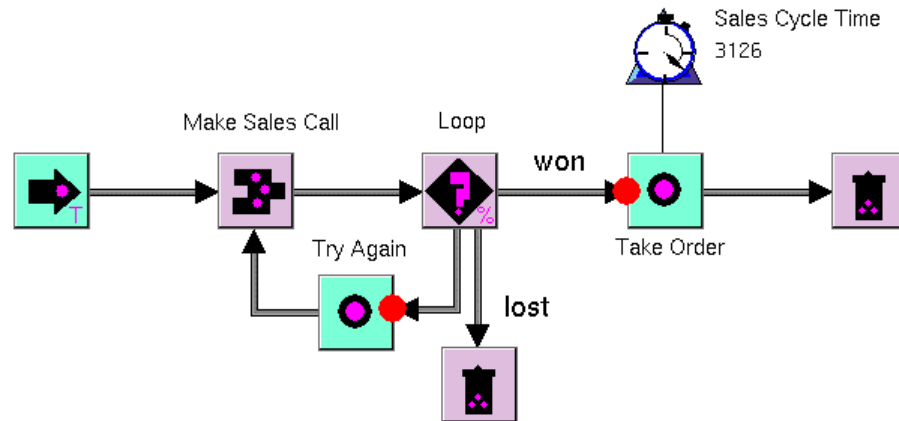
Computing the Cycle Time of a Work Object


One common measure of performance in a model is the cycle time from the creation of a work object to a point downstream in the process. You can easily compute the cycle time by probing the creation-time of a work object, using a Delta Time probe.

A Delta Time probe compares the creation time of a work object to a timestamp downstream in the process. For example, if you probe the work object at the end

of a process, you can compute the overall cycle time to accomplish all of the tasks applied to the work object in the model.

This model shows how you probe and chart the overall sales cycle time of each sales call in a sales process:



 Sales Cycle Time
3126.0

For information on how to probe and chart performance metrics, using ReThink instruments, see [Probing the Performance of Your Model](#) and [Charting Performance Metrics](#).

Working with Work Object Costs

Each work object keeps track of its total cost, which is the sum of the cost of each activity applied to the work object in the process. You assign fixed and variable costs to individual blocks or individual resources.

For information on specifying costs, see:

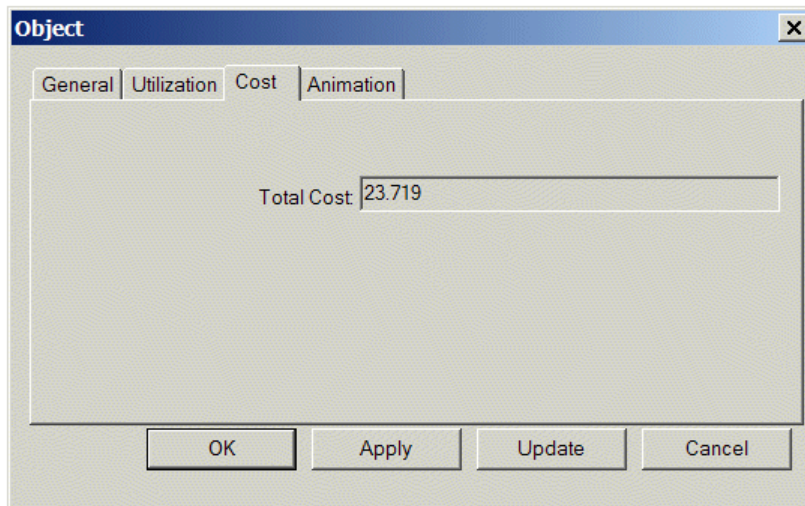
- [Working with Block Costs](#).
- [Working with Resource Costs](#).

All information related to cost is contained in the Cost tab of the properties dialog of the work object.

To display the total cost of a work object:

- 1 Run the simulation in step mode.
- 2 Display the properties dialog for the work object whose total cost you want to analyze and click the Cost tab.

ReThink displays the Cost tab of the properties dialog, which computes the total cost of the work object:



Customizing Work Objects

You can customize how work objects look, and you can customize how ReThink computes duration and cost metrics for work objects.

For detailed information about how to customize work objects, see the *Customizing ReThink User's Guide*.

Using Reports

Describes how to view metrics and enter parameter values through various types of reports.

- Introduction **362**
- Creating Reports **363**
- Configuring the Time Unit **370**
- Updating Output Reports at Regular Time Intervals **372**
- Keeping a History of Data Values **381**
- Charting Report Data **383**
- Configuring the Scope of the Report **384**
- Filtering Report Data **385**
- Configuring the Attributes to Appear in a Report **392**
- Creating Reports in Excel **394**
- Writing to and Reading from CSV Files **406**
- Writing to and Importing from Databases **408**
- Creating Specialized Reports **408**



Introduction

You determine the performance of your ReThink model by viewing **metrics**, which are attributes that the model computes, based on **parameters**, which are attributes that you configure. You can perform the following reporting tasks:

- [Create reports](#), including output reports for viewing metrics and input reports for configuring parameters.
- [Update output reports at regular time intervals](#).
- [Keep a history of data values](#).
- [Chart report data](#).
- [Filter the data to appear in a report](#).
- [Configure the attributes to appear in a report](#).
- [Create reports in Excel](#).
- [Write to and read from CSV files](#).
- [Write to and read from databases](#).
- [Create specialized reports](#):
 - [N-Dimensional Reports](#)
 - [Indexed Lookup Reports](#)
 - [Attribute Lookup Reports](#)
 - [Attribute Change Event Reports](#)

Note When creating reports in Excel, be sure you have Excel installed on your computer before you attempt to create a report.

For information on reports that you can create through the Navigator, see the *G2 Reporting Engine User's Guide*.

The toolbars appear in separate tabs at the bottom of the window.






Creating Reports








To create a report, you:




- [Create a report](#) for the desired type of [input or output report](#).
- Depending on the type of report:
 - [Generate output report data from the model.](#)
 - [Apply input report data to the model.](#)

Summary of Input and Output Reports

This table summarizes the input and output reports you can create:

Report	Description
Block Input Report 	Duration and cost parameters of blocks, and parameters relating to activities and animation.
Block Summary Report 	Duration and cost metrics of blocks, and metrics relating to activities.
Resource Input Report 	Resource priority, utilization, efficiency, cost, and animation parameters.
Resource Summary Report 	Duration, utilization, and cost metrics of resources, and metrics relating to resource activities.
Path Input Report 	Path type and parameters related to branching.

Report	Description
Path Summary Report 	Metrics relating to wait times and number of insertions of paths.
Object Input Report 	User-defined parameters of work objects.
Object Summary Report 	Duration, utilization, and cost metrics of work objects.
Probe Input Report 	Parameters for configuring the class to which the probe applies, the source attribute, and specific probe parameters.
Probe Summary Report 	Metrics that probes sample and compute.
N-Dimensional Input Report 	Individual parameters for any number of objects of any type in the model.
N-Dimensional Output Report 	Individual metrics for any number of objects of any type in the model.

Report	Description
Indexed Lookup Report 	Duration parameter values for a block whose Mode is Report Indexed Lookup.
Attribute Lookup Report 	Duration parameter values and index values for a block whose Mode is Report Lookup.
Attribute Change Event Report 	Durations and corresponding attribute names and values for one or more objects in the model, used to schedule attribute value changes during the simulation.

Creating a Report

The first step in creating a report is to determine where to place the report in your model. By default, reports include data for all objects of the specified type on the current workspace and all details. You can place reports in a number of locations in the model, as follows:

To include data for...	Place the report object on...
All objects of the specified type in the model	The model detail.
All objects of the specified type associated with a particular workspace and any details	The detail of an organizer and choose the root workspace for the report object.

You place report objects on an organizer when you have many reports and placing them on the model detail would make the detail too cluttered. For information on creating organizers, see [Creating an Organizer](#).

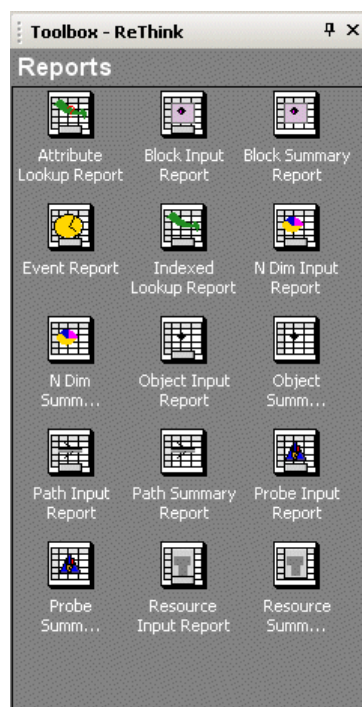
Once you place the report object in the appropriate location in your model, you can create the report. The report has a row for each object in the model that corresponds to the type of report. For example, a Block Summary Report has a row for each block in the model, and a Resource Summary Report has a row for each resource.

The report includes a column for each parameter or metric that the report defines, depending on whether it is an input or output report. For example, the Block Input Report contains columns for configuring the Maximum Activities, Mean, Standard Deviation, and other block parameters, and the Block Summary Report contains columns for viewing the Current Activities, Total Work Time, Total Elapsed Time, and other block metrics.

The report identifies each object by its label; therefore, be sure to configure labels for all objects in the model before you create the report.

To create a report:

- 1 Display the Reports palette of the ReThink toolbox:



- 2 Create an input or output report object, based on the type of data you want to enter or compute, and place it in the desired location in the model.

For information on the types of reports you can create, see [Summary of Input and Output Reports](#).

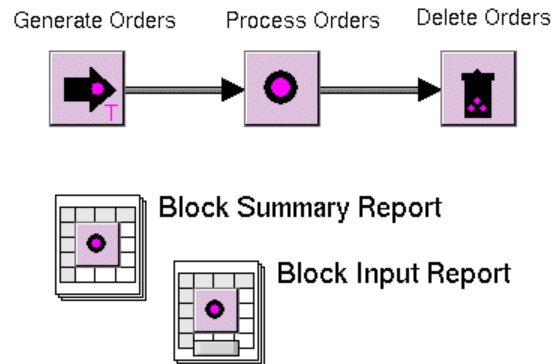
If you place the report object on the model detail or some other detail, you can skip the following step. Otherwise, if you place the report object on the detail of an organizer, you must choose the root workspace for the report object.

- 3 If necessary, choose Choose Root Workspace on the report object, then select the workspace to which the report object should apply and choose Select.

The report object applies to all objects of the specified type on the selected root workspace and all details.

- 4 To create the report, choose Show Report on the report object.

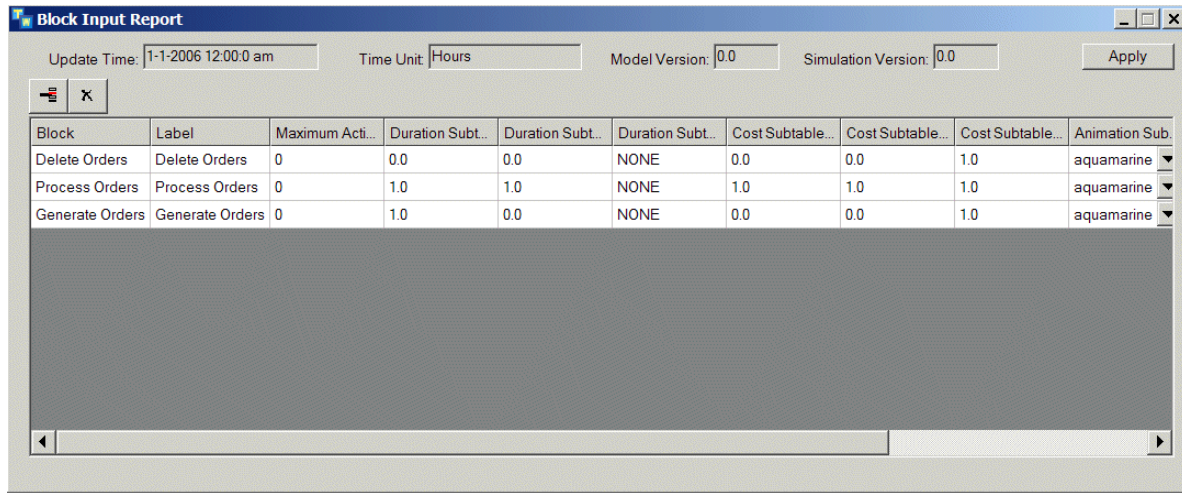
This figure shows a model with a Block Summary Report and a Block Input Report on the same workspace as the model:



Here is the report for the Block Summary Report:

Block	Total Starts	Total Stops	Current Activities	Duration Subtable.total Work Time	Duration Subtable.total Elapsed Time	Duration Subtable.creation Time
Delete Orders	0	0	0	0.0	0.0	0.0
Process Orders	0	0	0	0.0	0.0	0.0
Generate Orders	0	0	0	0.0	0.0	0.0

Here is the report for the Block Input Report:



Block	Label	Maximum Acti...	Duration Subt...	Duration Subt...	Duration Subt...	Cost Subtable...	Cost Subtable...	Cost Subtable...	Animation Sub.
Delete Orders	Delete Orders	0	0.0	0.0	NONE	0.0	0.0	1.0	aquamarine
Process Orders	Process Orders	0	1.0	1.0	NONE	1.0	1.0	1.0	aquamarine
Generate Orders	Generate Orders	0	1.0	0.0	NONE	0.0	0.0	1.0	aquamarine

Generating Output Report Data from the Model

To generate output report data, you simply run the simulation and update the report. Each time the report updates, new data appears in the report.

By default, output reports are configured to update manually and output static data.

You can configure the report to update automatically, as described in [Updating Output Reports at Regular Time Intervals](#).

You can also configure the report to output time-series data, as described in [Keeping a History of Data Values](#).

To generate output report data from the model:

- 1 Run the simulation.
For details, see [Controlling the Simulation](#).
- 2 Update the report manually, using one of these techniques:
 - Click the Update button at the top of the report.
 - or
 - Choose Update Report on the report object.

Here is a Block Summary Report after running the simulation for a period of time:

Block	Total Starts	Total Stops	Current Activities	Duration Subtable.total Work Time	Duration Subtable.total Elapsed Time	Duration Subtable.creation Time
Delete Orders	234	234	0	0.0	259.214	0.0
Process Orders	234	234	0	271.968	259.214	0.0
Generate Orders	235	235	0	259.214	259.214	0.0

Applying Input Report Data to the Model

To apply input report data, you enter data in the input report and apply the values to the model. Each time you apply new values, the parameters in the model update.

Configuring parameters through input reports provides an alternative to configuring the same parameters through properties dialogs and has these advantages:

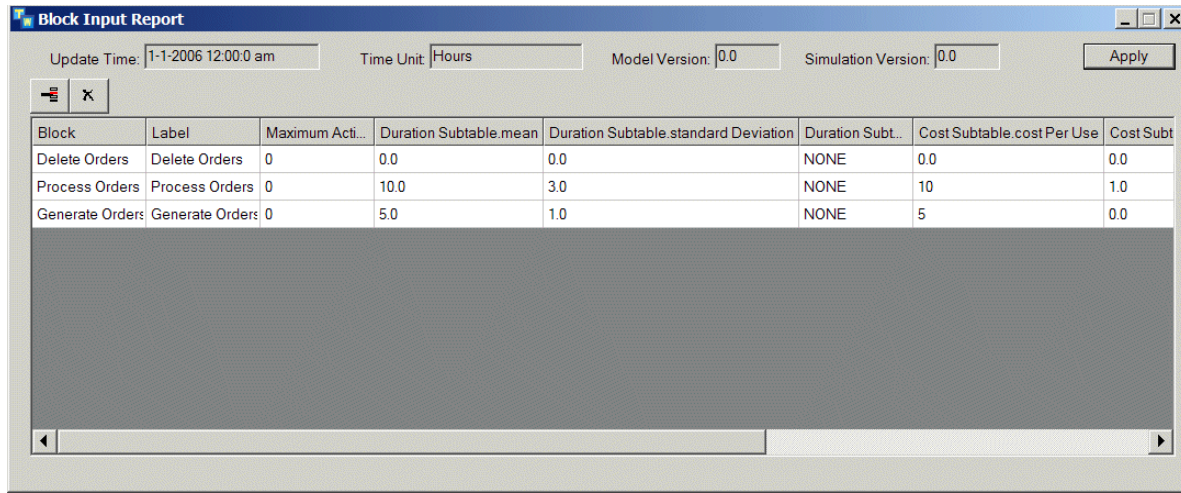
- Configuring parameters for the same types of items in a single spreadsheet, for example, all block durations or all resource costs.
- Running different configurations of the model, using different sets of input parameters and comparing the results.

To apply input report data to the model:

- 1 Configure the report data for the specified parameters of the report objects.
- 2 Click the Apply button at the top of the report to apply the data to the model.

ReThink applies the values from the report to the appropriate parameters in the model.

Here is a Block Input Report with values specified:



Block	Label	Maximum Acti...	Duration Subtable.mean	Duration Subtable.standard Deviation	Duration Subt...	Cost Subtable.cost Per Use	Cost Subt
Delete Orders	Delete Orders	0	0.0	0.0	NONE	0.0	0.0
Process Orders	Process Orders	0	10.0	3.0	NONE	10	1.0
Generate Orders	Generate Orders	0	5.0	1.0	NONE	5	0.0

Configuring the Time Unit

By default, reports display all time-based values in hours, which means:

- All time-based metrics display in units of an hour in output reports.
- You must enter all time-based parameter values in units of an hour in input reports.

For example, a value of 3 days displays in an output report as 72 hours, and you must enter 72 hours in an input report as the value for 3 days.

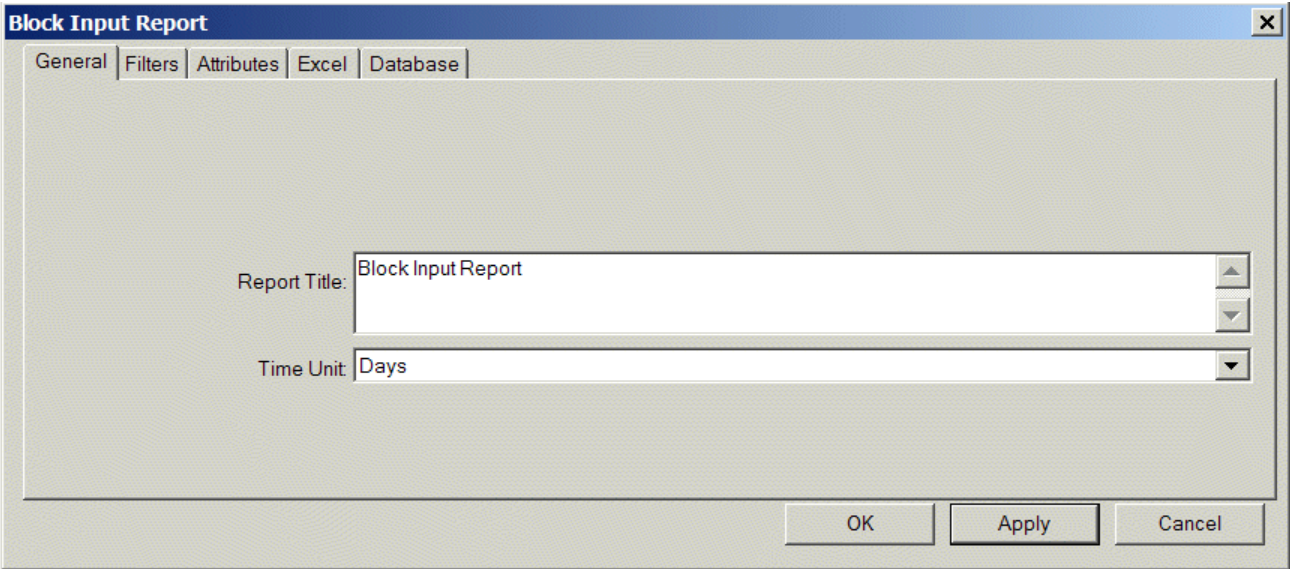
Depending on your model, you might want to use a different time unit, such as minutes, days, or weeks. Alternatively, you can configure the report to display all time-based parameters and metrics as durations, for example, one week, one day, one hour, one minute, and one second would be 001:001:001:001:001.

The report displays the current time unit at the top of the report.

To configure the time unit:

- ➔ Display the properties dialog for the report and, on the General tab, configure the Time Unit to be seconds, minutes, hours, days, or weeks, or configure the Time Unit to be none to use a duration.

The following figure shows a Block Input Report that is configured for entering time-based parameter values in units of a day, rather than in units of an hour, the default:



Here is the resulting report with values entered in units of a day:

The time unit is 1 day,

You enter time-based values in units of one day, rather than in units of one hour, the default.

The screenshot shows the "Block Input Report" application window. At the top, it displays "Update Time: 1-1-2006 12:00:0 am", "Time Unit: Days", "Model Version: 0.0", and "Simulation Version: 0.0". Below this is a table with the following data:

Block	Label	Maximum Acc...	Duration Subt...	Duration Subt...	Duration Subt...	Cost Subtable...	Cost Subtable...	Cost Subtable...	Animation Sub...
Delete Orders	Delete Orders	0	0.0	0.0	NONE	0.0	0.0	0.042	aquamarine
Process Orders	Process Orders	0	1	0.5	NONE	1.0	1.0	0.042	aquamarine
Generate Orders	Generate Orders	0	5	2	NONE	0.0	0.0	0.042	aquamarine

Updating Output Reports at Regular Time Intervals

By default, you update output report data manually. You can configure reports to update automatically at regular time intervals, based on:

- Simulation time.
- Clock time.

For example, you might want report data to update once a day or once a week, based on the amount of simulation time that has passed. Alternatively, you might want the report data to update once every five seconds of real time, based on the computer clock.

To ensure that the report includes data for the last update period, you should run the simulation for slightly longer than the last update interval. For example, to include data from four weeks of simulation time, you should run the simulation for 29 days, which is four weeks plus a day. You can also trigger updates by using an:

- Update Trigger tool, which allows you to configure in a single location the update interval for multiple reports.
- Update Trigger probe, which allows you to configure when a report updates, based on model events.

By default, output reports refresh their values each time the report updates. To improve performance, you can configure output reports to refresh at the end of the simulation and when you manually request an update only.

- [Trigger regular updates for multiple reports.](#)
- [Trigger updates based on model events.](#)

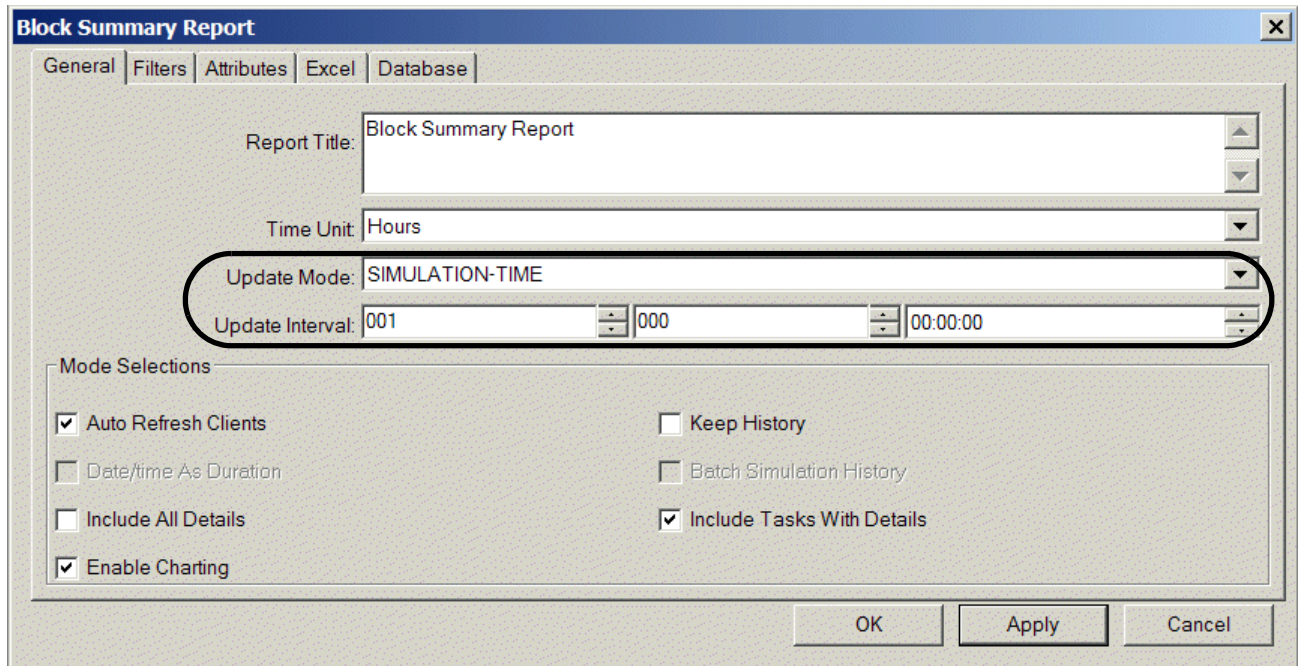
Configuring Output Reports to Update Regularly

The easiest way to trigger updates at regular time intervals is by configuring the report object.

To configure an output report to update regularly:

- 1 Display the properties dialog for the output report whose data you want to update regularly.
- 2 On the General tab, configure the Update Mode to be clock-time or simulation-time, depending on how you want to update the report.
- 3 Configure the Update Interval to be the time interval at which to update the report, based on the Update Mode.

This figure shows how to configure a Block Summary Report to update once a week, based on simulation time:



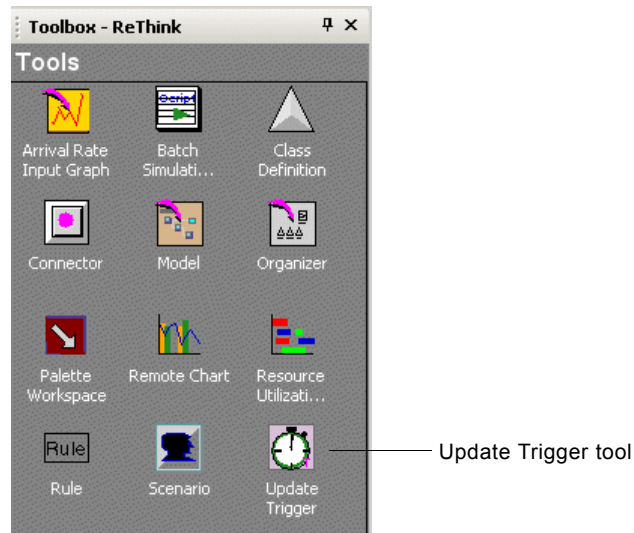
Triggering Regular Updates for Multiple Reports

You might want to trigger multiple reports to update at the same time. The easiest way to do this is to associate each report with an Update Trigger tool, which triggers updates for all associated reports.

Another advantage of using an Update Trigger tool is that you can configure a time delay before the first update.

To trigger regular updates for multiple reports:

- 1 Display the Tools palette of the ReThink toolbox:



- 2 Select an Update Trigger tool and place it on the workspace that contains the report object you want to update.
- 3 Display the properties dialog for the Update Trigger and on the General tab, configure the Block Label.

For example, you might label the Update Trigger “Hourly Update.”

By default, the Update Trigger tool triggers updates continuously.

- 4 Click the Block tab and configure the Maximum Starts to be the maximum number of times the trigger should update.

You might want to begin triggering updates after a time delay or stop triggering updates after a certain simulation time. Leaving the field blank means there is no limit.

- 5 Configure the Start Time and End Time to be the time at which the trigger tool should start and finish triggering updates.

Enter the value as a duration, such as 1 minute and 30 seconds, 1 hour, or 1 day. The value you enter is converted to seconds.

Tip To ensure that the report computes metrics for the current update interval, click the Block tab and configure the Start Time of the Update Trigger to be 1 second, which causes the report to trigger one second after the end of the update interval.

- 6 Click the Duration tab and configure the Period to be the frequency with which to trigger updates.

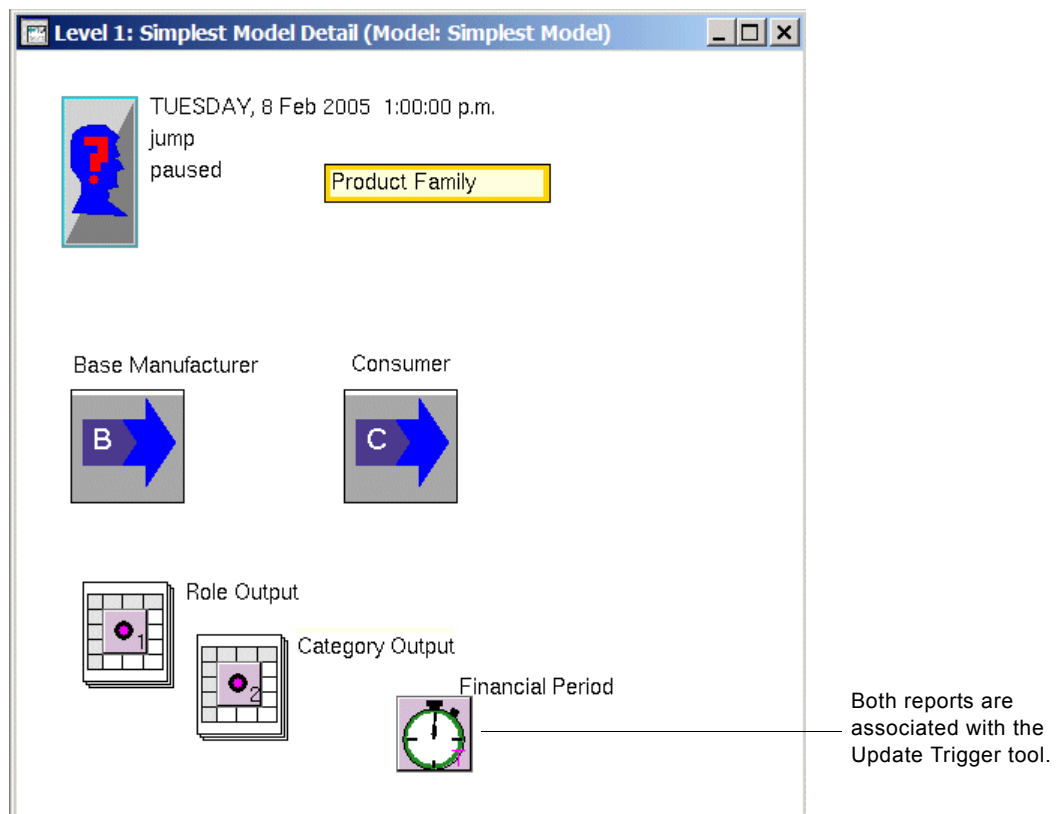
For example, to update the reports once an hour of simulation time, enter 1 hour.

- 7 Choose the Choose Update Trigger menu choice on a report object, then choose Select on the Update Trigger.

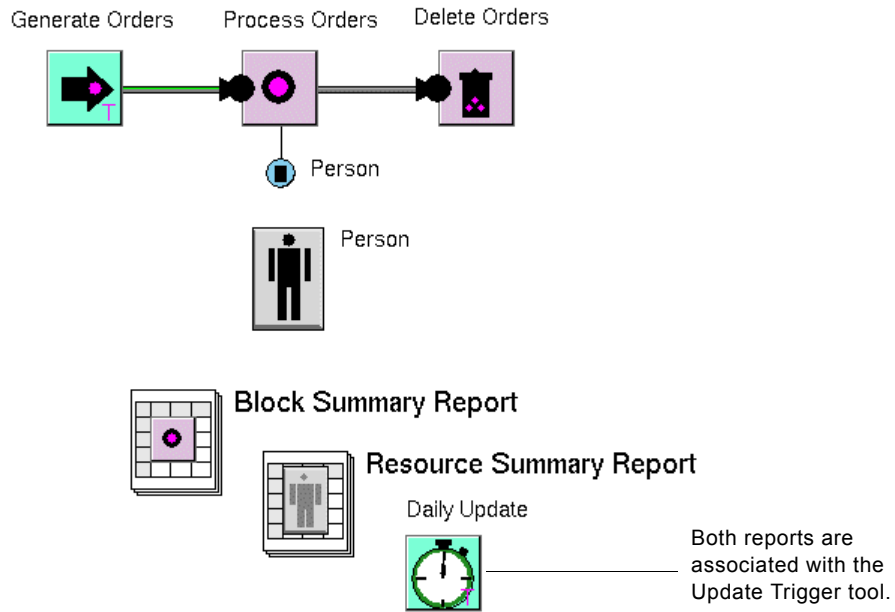
An indicator arrow appears indicating that you have selected the trigger.

- 8 Repeat Step 7. for each report whose data you want to update, based on the Update Trigger.

This figure shows how you use a single Update Trigger tool to update two reports. The Role Output Report and Category Output Report are both associated with the Financial Period Update Trigger, which is configured to update once per financial period. By configuring the Start Time of the trigger to be 1 second, the reports update one second after each update period, which means the report shows the most current values.



This figure shows how you use a single Update Trigger tool to update two reports. The Block Summary Report and the Resource Summary Report are both associated with the Daily Update trigger, which is configured to update its associated reports once a day. By configuring the Start Time of the trigger to be 1 second, the reports update one second after each update period, which means the report shows the most current values.



Here are both reports after several update cycles. Notice that Data Update Time of each report is 1-7-2006 12:01 am, which is one day and one second after the start of the simulation.

Report updates one second after the start of each day so values are always current.

Block Summary Report							
Update Time:		1-7-2006 12:00:1 am		Time Unit:		Hours	
						Model Version:	0.0
Block	Total Starts	Total Stops	Current Activities	Duration Subtable.total Work Time	Duration		
Process Orders	137	136	1	133.656	144.0		
Generate Orders	160	159	1	144.0	144.0		
Delete Orders	136	136	0	0.0	144.0		
Daily Update	7	6	1	144.0	144.0		

Resource Summary Report							
Update Time:		1-7-2006 12:00:1 am		Time Unit:		Hours	
						Model Version:	0.0
Resource	Total Starts	Total Stops	Current Activities	Total Waiting	Duration Subtable.total Work Time		
Person	137	136	1	1	133.656		

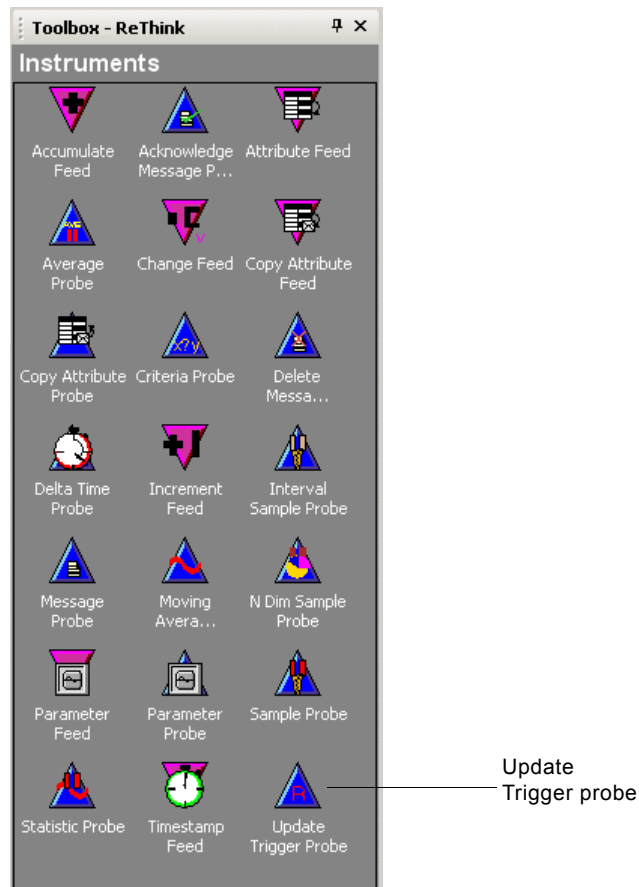
Triggering Updates Based on Model Events

You might want to trigger reports to update, based on model events, rather than based on a simulation time or real time. To do this, you connect an Update Trigger probe to an object in the model that should trigger updates, then you associate the report with the probe.

For example, you might want to update a Block Summary Report when a work object arrives at a block; you might want to update a Resource Summary Report when the resource is allocated; or you might want to update a Path Input Report when the block updates.

To trigger updates, based on model events:

- 1 Display the Instruments palette of the ReThink toolbox:



- 2 Select an Update Trigger probe and place it on the workspace that contains the report object.
- 3 Connect the Update Trigger probe to an object in the model that you want to trigger updates.

You can connect an Update Trigger probe to these types of objects:

- Block
 - Resource
 - Instrument
- 4 Display the properties dialog for the Update Trigger probe and configure the Label.

For example, you might label the Update Trigger probe “Update Block Report.”

- 5 Configure the Apply to Class Name attribute of the probe to be the object that should trigger updates.

The default value for Apply to Class Name is `bpr-object`. When the Update Trigger probe is connected to a block, this configuration means it triggers updates when a work object arrives at the block.

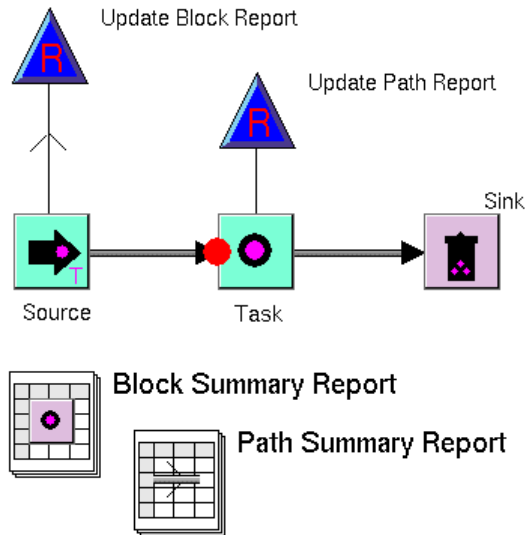
You can configure the Apply to Class Name to be any of these classes or a subclass, depending on the object to which the probe is attached:

If the probe is attached to a...	Configure Apply to Class Name to be...	To trigger updates when...
Block	<code>bpr-object</code>	A work object arrives at the block.
Block	<code>bpr-block</code>	The block updates.
Block	<code>bpr-path</code>	The input path updates.
Block	<code>bpr-activity</code>	An activity of the block is active.
Resource	<code>bpr-resource</code>	The resource is allocated.
Instrument	<code>bpr-instrument</code>	The instrument updates.

- 6 Choose the Choose Update Trigger menu choice on a report object, then choose Select on the Update Trigger probe.
- 7 Repeat Step 6. for each report whose data you wish to update, based on the model event.

When you run the simulation, the selected reports update when the model event occurs.

This figure shows how you use two Update Trigger probes to update two reports. The Update Block Report probe triggers updates of the Block Summary Report each time the Source block creates a work object by configuring the Apply to Class Name to be `bpr-object`. The Update Path Report triggers updates of the Path Summary Report each time the Task block updates by configuring the Apply to Class Name to be `bpr-block`.



Triggering Updates Manually

When using an Update Trigger tool or Update Trigger probe, you can trigger the update of all associated reports manually. You can also show all the items to update.

To trigger the update of all associated items manually:

➔ Choose the Update All Related Items menu choice on the Update Trigger.

To show items associated with the Update Trigger:

➔ Choose the Show Items to Update menu choice on the Update Trigger.

Configuring When Clients Refresh Their Data

By default, clients refresh their data each time the report updates. This means that reports that appear within the client, reports that are generated in CSV files and Excel, and reports that output their data to a database all refresh their data automatically each time new data is collected in the server.

Depending on the number of reports in your model, the number of clients of the server's data, and the interval at which reports update, performance can be degraded. To improve performance, you can disable the automatic refreshing of

client data. When automatic refreshing is disabled, ReThink refreshes client data only when explicitly requested, using the Update menu choice or button, and automatically at the end of the simulation.

To disable automatic refreshing of client data:

- 1 Display the properties dialog for the output report object for which you want to disable client refreshing.
- 2 On the General tab, disable the Auto Refresh Clients option.

Keeping a History of Data Values

By default, output reports generate static data. To generate time-series data for any type of output report, you configure the report to keep a history of data values. Typically, you configure reports that keep a history to update at regular time intervals. Each time the report updates, ReThink outputs new values to the report for each metric so you can compare values over time. You can then output the data to a CSV file, to Excel, or to a database to perform analysis on the time-series data. For example, you might configure a Resource Summary Report to output cost data once per month to track monthly salary expenses.

Each data value in the history has an associated timestamp that indicates when the value was generated. You can format the timestamp as an absolute date and time or as a relative duration.

When using the Scenario Manager to perform multiple simulations for the same model, you can configure the report to keep a history across all simulations. When keeping a history across multiple simulations, the output report has an additional column named Simulation Counter, which indicates the number of the simulation run. For details, see [Using Batch Simulation](#).

To keep a history of data values:

- 1 Display the properties dialog for the output report object for which you want to keep a history.
- 2 Configure the report to update at regular time intervals.
For details, see [Updating Output Reports at Regular Time Intervals](#).
- 3 On the General tab, enable the Keep History option.

By default, the timestamp the report generates uses an absolute time and date, based on simulation time. For example:

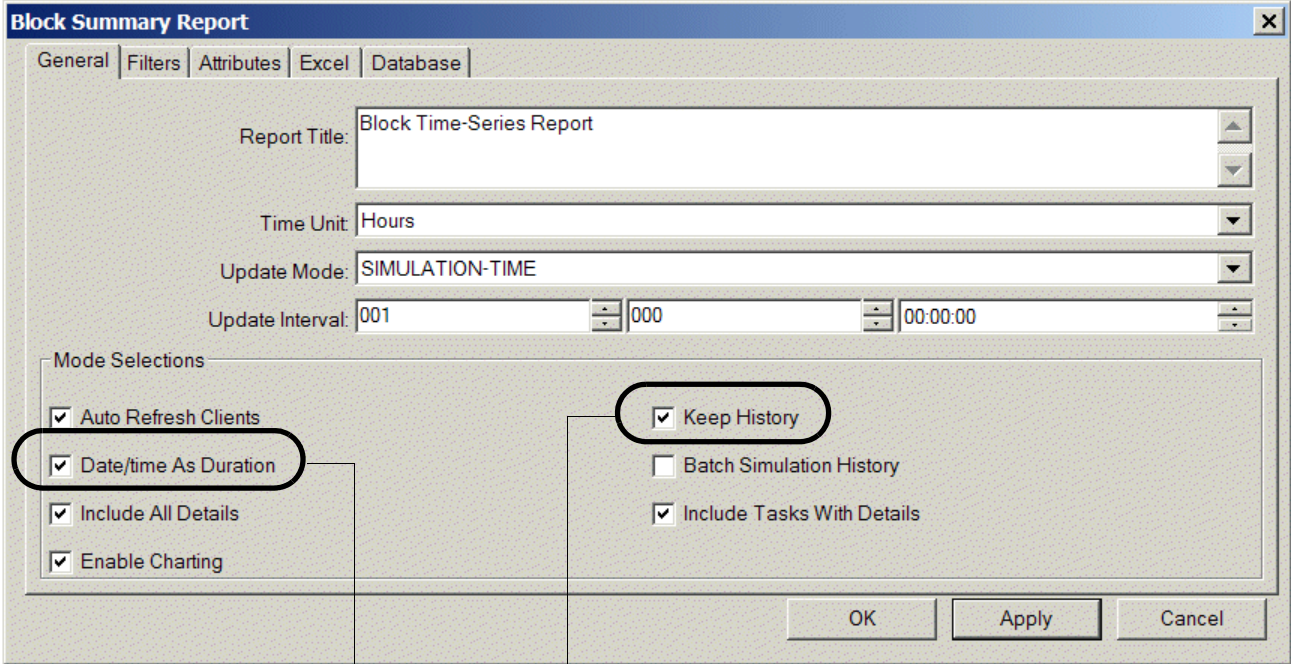
1/1/02 0:00

You can configure the timestamp to use a relative duration from the start of the simulation. For example:

- 1 weeks, 2 days, 1 hour, 30 minutes, and 10 seconds
- 2 weeks, 1 day, 30 minutes, and 30 seconds
- 3 weeks and 1 hour
- etc.

- 4 Enable the Date/Time as Duration option to use relative timestamps, if desired.
- 5 When running batch simulations on the same model, using the Batch Simulation object, enable the Batch Simulation History option to keep a history across multiple models, if desired.

This figure shows how to configure a Block Summary Report to generate time-series data once every week of simulation time, where the timestamps appear as durations:



Report displays timestamps as relative durations.

Report generates time-series data each Update Interval, based on the Update Mode.

Here is the resulting report after running the simulation for 15 days, which includes data for two weekly time periods:

Block Time-Series Report							
Update Time: 1-15-2006 12:00:0 am		Time Unit: Hours		Model Version: 0.0		Simulat	
Simulation Time	Block	Delete Orders	Delete Orders	Delete Orders	Delete Orders	Delete Orders	Delete Orders
1 week		184	184	0	0.0	168.0	0.0
2 weeks		374	374	0	0.0	336.0	0.0

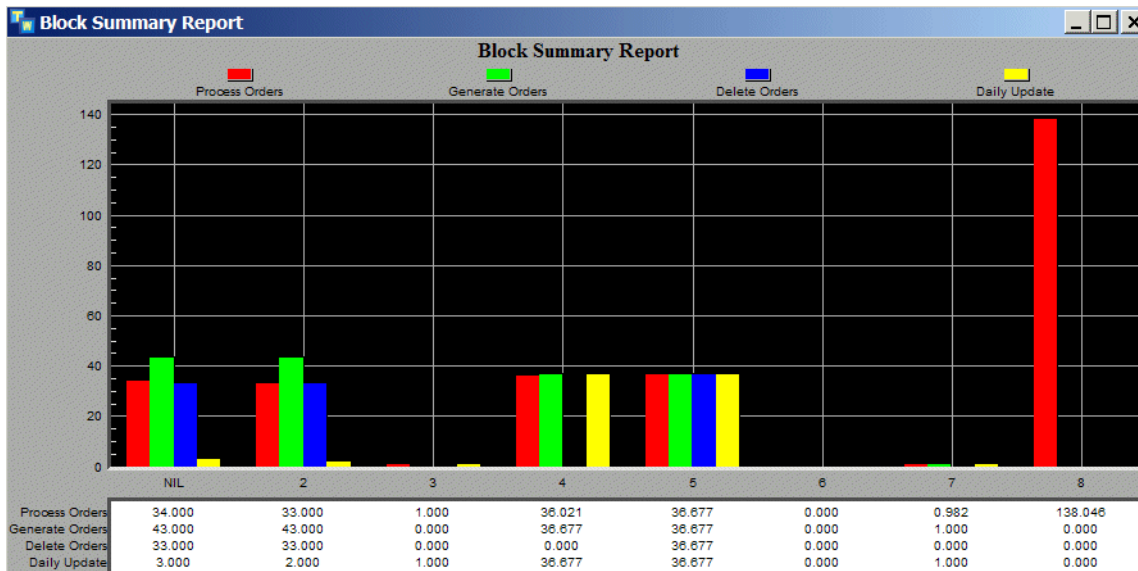
Charting Report Data

You can chart the data in a report in various types of charts. The chart updates according to the update setting of the report.

To chart report data:

- 1 Create and configure an output report.
For details, see [Creating a Report](#).
- 2 Configure the update interval for the report.
For details, see [Configuring Output Reports to Update Regularly](#).
- 3 Ensure that the Enable Charting option on the General tab of the properties dialog for the report is enabled, the default.
- 4 Enable the Update Charts option in the Scenario.
For details, see [Configuring the Computation Behavior](#).
- 5 Choose Show Report on the report to create the report.
- 6 Choose Show Chart on the report to show a chart of the report data.

For example, here is a chart for a Block Summary Report:



Configuring the Scope of the Report

By default, the Block Summary Report and the Block Input Report include objects on details, as well as the Task block with details itself. Depending on the model, you might want to limit the scope of the report to include only the objects on details or only the Task block with details.

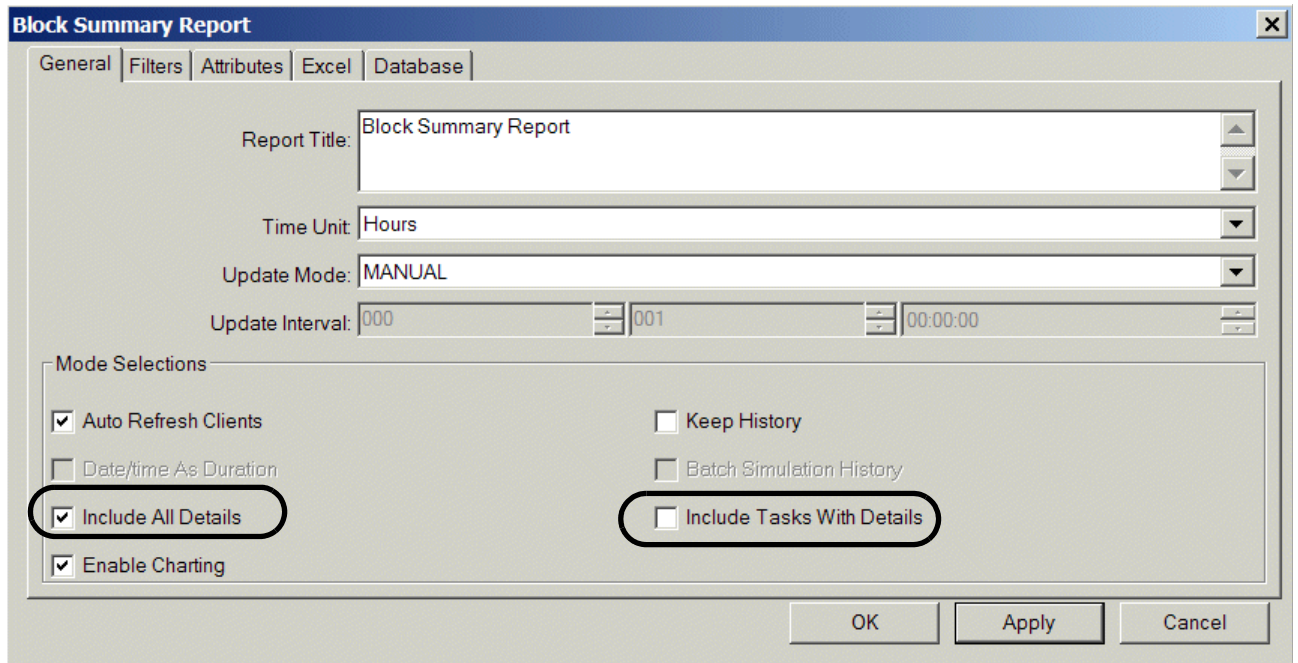
To limit the scope of the report include Task blocks with detail only:

- 1 Display the properties dialog for the report whose scope you want to configure.
- 2 On the General tab, disable the Include All Details option and enable the Include Tasks with Detail option, the default.

To limit the scope of the report include blocks on details only:

- 1 Display the properties dialog for the report whose scope you want to configure.
- 2 Disable the Include Tasks with Detail option and enable the Include All Details option, the default.

This figure shows how to configure a Block Summary Report to include blocks on details only:



Filtering Report Data

The default report include data for all types of objects included in the report. For example, a Block Summary Report includes data for all categories of blocks, an Object Summary Report includes data for all types of work objects, and a Resource Summary Report includes data for all types of resources.

You can filter the data that appears in a report. For example, you might want to generate separate reports for each category of block or resource, or you might prefer to configure input reports separately for each category of object of a particular type.

To filter report data, you configure the class or classes of objects to which the report applies. You can configure a specific class or a superior class, which includes all classes below it in the hierarchy. For example, you could filter the

report data, based on the `bpr-task` class to include data for Task blocks only or you could filter it based on `bpr-feed` to include all feeds.

You can also filter report data, based on user-defined classes, such as user-defined work objects that you specify as the path type for a block. For more information, see [Specifying a User-Defined Object as the Path Type](#).

The following table shows the class names for all the built-in classes of each type:

Block Classes

Class Name	Description
<code>bpr-block</code>	Superior class for all blocks
<code>bpr-source</code>	Source block
<code>bpr-sink</code>	Sink block
<code>bpr-task</code>	Task block
<code>bpr-copy</code>	Copy block
<code>bpr-merge</code>	Merge block
<code>bpr-branch</code>	Branch block
<code>bpr-batch</code>	Batch block
<code>bpr-associate</code>	Associate block
<code>bpr-reconcile</code>	Reconcile block
<code>bpr-store</code>	Store block
<code>bpr-retrieve</code>	Retrieve block
<code>bpr-insert</code>	Insert block
<code>bpr-remove</code>	Remove block
<code>bpr-copy-attributes</code>	Copy Attributes block
<code>bpr-yield</code>	Yield block

Probe Classes

Class Name	Description
bpr-instrument	Superior class for all instruments
bpr-probe	Superior class for all probes
bpr-delta-time-probe	Delta Time probe
bpr-sample-probe	Sample Value probe
bpr-average-probe	Average probe
bpr-moving-average-probe	Moving Average probe
bpr-interval-sample-probe	Interval Sample probe
bpr-parameter-probe	Parameter probe
bpr-copy-attribute-probe	Copy Attributes probe
bpr-statistic-probe	Statistic probe
bpr-criteria-probe	Criteria probe
bpr-update-trigger-probe	Update Trigger probe
bpr-n-dim-sample-probe	N-Dimensional Sample probe
bpr-message-probe	Message probe
bpr-acknowledge-message-probe	Acknowledge Message probe
bpr-delete-message-probe	Delete Message probe

Work Object and Resource Classes

Class Name	Description
bpr-object	Superior class for all work objects
bpr-resource	Superior class for all resources
person	Person resource
truck	Truck resource
computer	Computer resource

Work Object and Resource Classes

Class Name	Description
machine	Machine resource
bpr-pool	Pool resource
bpr-container	Container work object that defines a container list attribute that is an item-list
bpr-surrogate	A resource surrogate

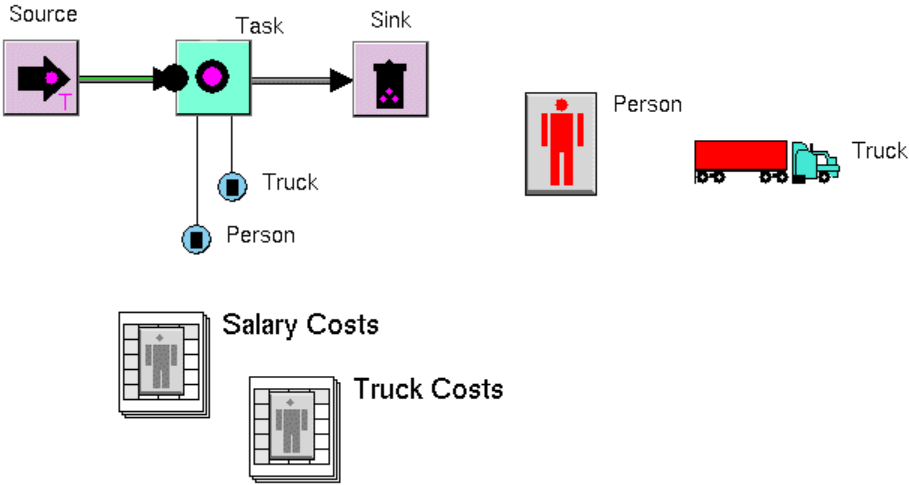
Path Class

Class Name	Description
bpr-path	Superior class for all paths

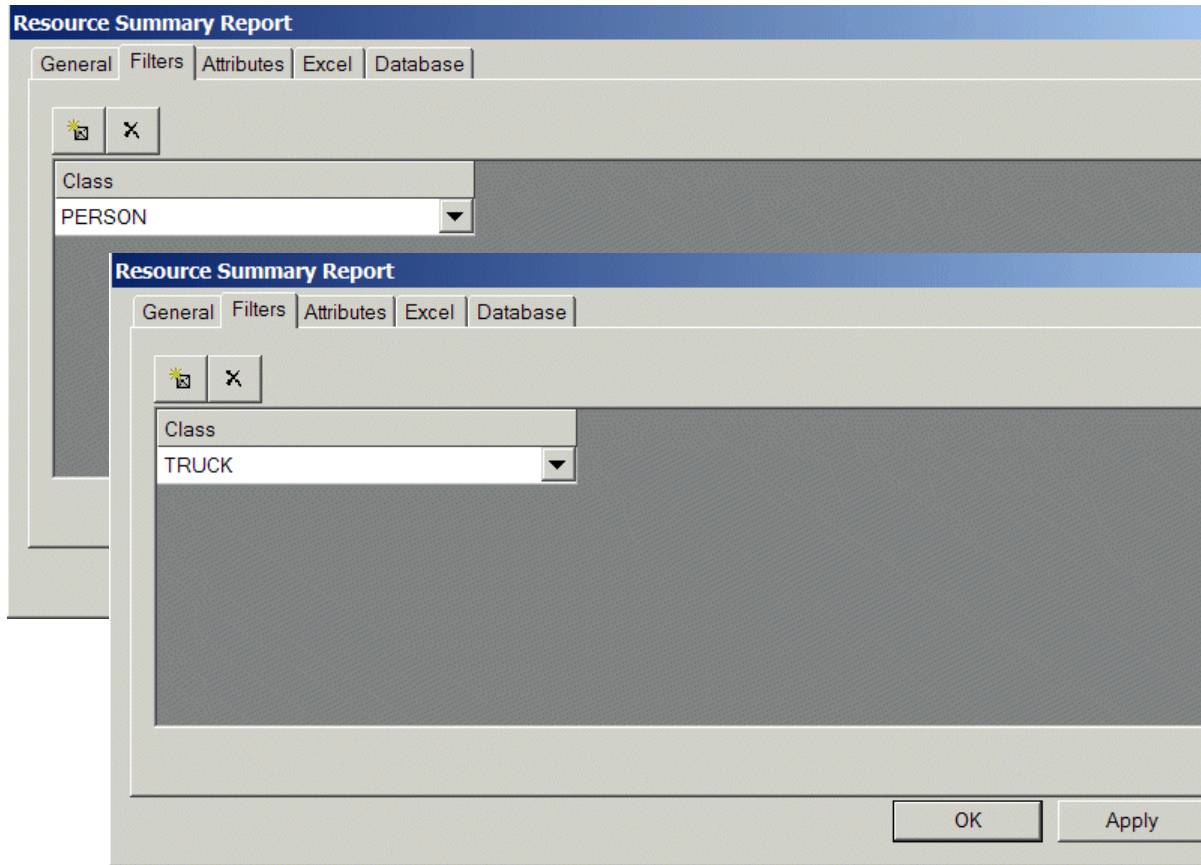
To filter report data by object class:

- 1 Create as many report objects as needed, depending on the classes of objects to which each report should apply.
- 2 Display the properties dialog of each report object and click the Filters tab.
- 3 Configure the class to which the report data should apply.
- 4 Add and remove classes to and from the filter, as needed:
 - Click Add Row to add a row to the end of the list.
 - Click Delete Row to delete the selected row or rows.

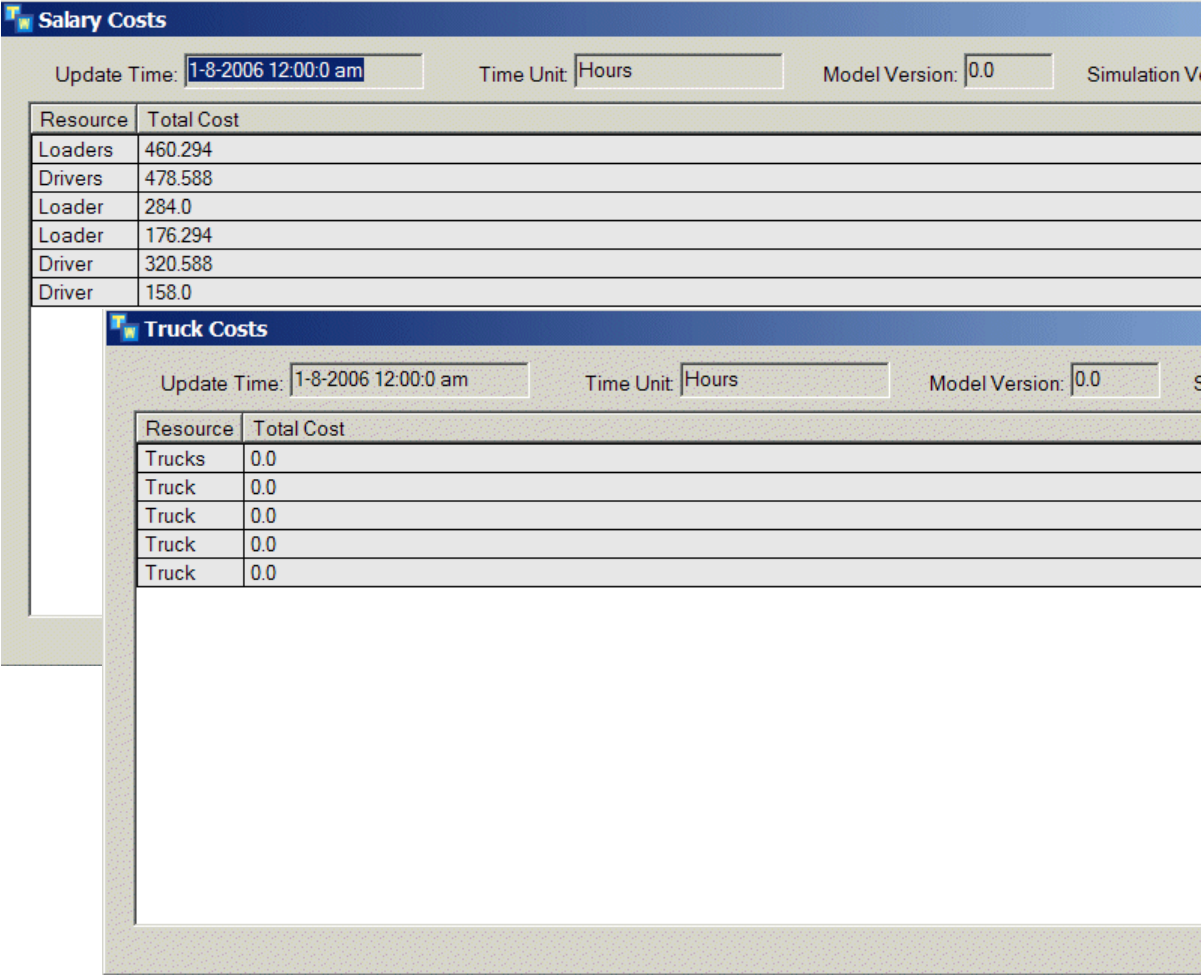
This example shows a model with separate output reports that report on weekly Salary Costs for people resources and weekly Truck Costs for truck resources:



Here are the Filter tabs of the properties dialogs for the Salary Cost report and the Truck Cost report, which filter report data based on the person and truck classes, respectively:



This figure shows the Salary Costs and Truck Costs reports after a week of simulation time has past, where only the Total Cost attribute is visible:



Configuring the Attributes to Appear in a Report

Each type of report includes a default set of parameters and/or metrics, depending on the type of report. You can configure the list of visible attributes through the report object.

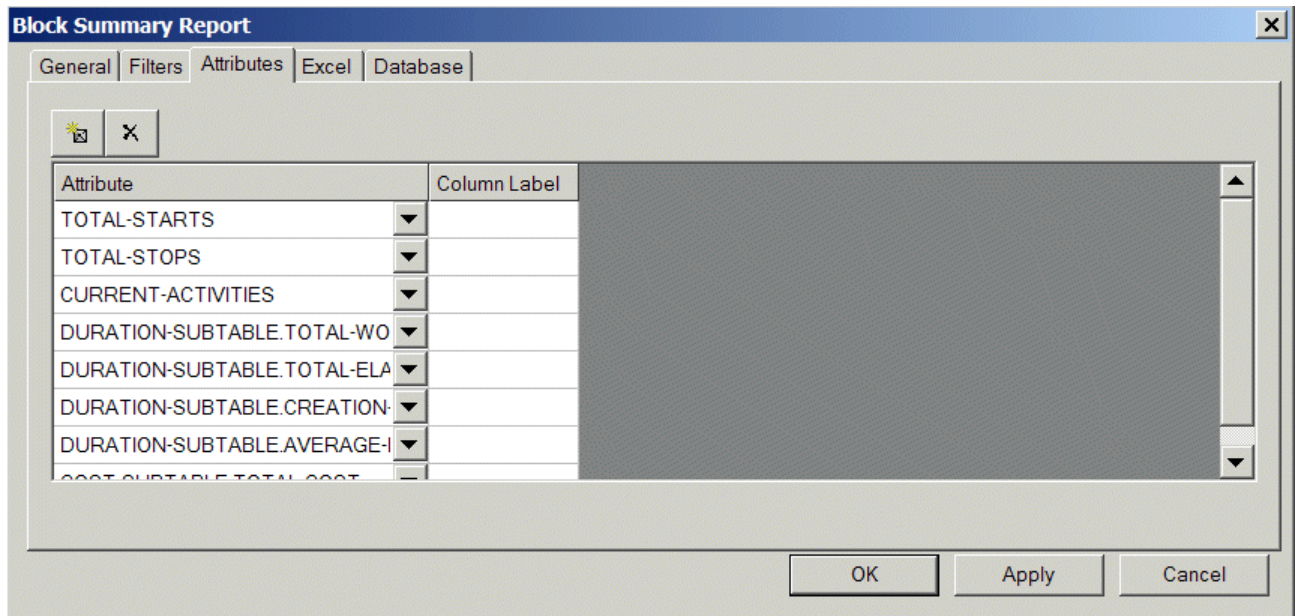
Note In addition to the default set of attributes, all input report include the GFR-UUID attribute, which uniquely identifies the object. This attribute must appear when creating reports in `.csv` files; otherwise, you cannot apply values to the model. When you create reports locally, this attribute is only required if the object labels are not unique.

To configure the list of visible attributes, you choose from a list of available attributes within a group. The groups correspond to the tab pages in the properties dialog for each type of object. For example, the Block Summary Report contains groups named `duration-subtable`, `cost-subtable`, and `animation-subtable`, where the attributes within each group correspond to the parameters or metrics on the Duration, Cost, and Animation tabs, respectively. The column headers that appear in the report refer to both the group name and the attribute name, for example, `duration-subtable.mean` and `cost-subtable.total-cost`.

To configure the list of visible attributes in a report:

- 1 Display the properties dialog for the report object and click the Attributes tab.
The dialog shows the attribute in the left column and the column label in the right column.

Here is the Attributes tab of the properties dialog for the default Block Summary Report:



- 2 Add and remove attributes to and from the list, as needed:
 - Click the Add Row button to add a row above the currently selected row.
 - Click Delete Row to delete the selected row or rows.

Tip You can use the Shift key to select multiple rows.

- 3 To configure the attribute to appear in the report, double-click the attribute name in the left column to display a dropdown list of available attributes and their associated groups, where relevant, then choose an attribute.

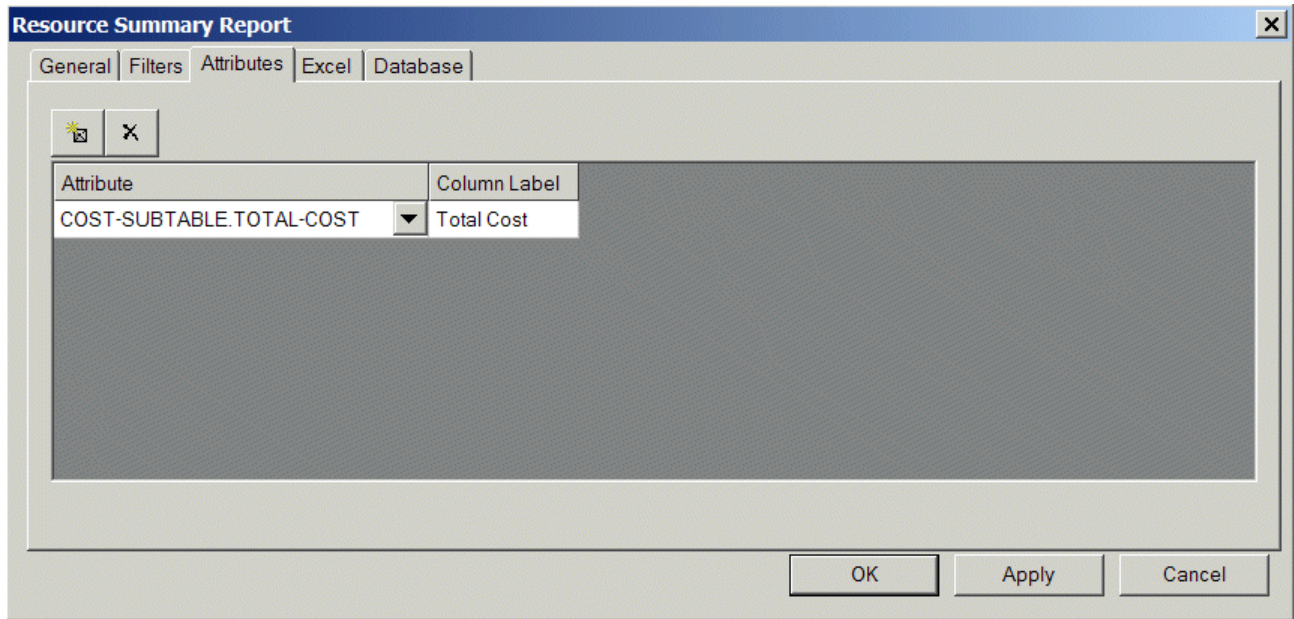
Tip Double-click the border between the column headers to expand the column width to just fit the longest attribute name.

By default, the attribute name and its associated group appear as the column header when you display the report. You can also enter a label for the column header.

- 4 Enter a Column Label for the attribute, if desired.

The report contains the attributes and column labels you configured.

For example, this figure shows a Resource Summary Report with just the Total Cost attribute visible and the column label configured:



Here is the resulting report for a model with a number of resources:

The screenshot shows a window titled "Salary Costs" with a blue header bar. Below the header are several fields: "Update Time: 1-8-2006 12:00:0 am", "Time Unit: Hours", "Model Version: 0.0", "Simulation Version: 0.0", and an "Update" button. Below these fields is a table with two columns: "Resource" and "Total Cost". The data rows are:

Resource	Total Cost
Loaders	460.294
Drivers	478.588
Loader	284.0
Loader	176.294
Driver	320.588
Driver	158.0

Creating Reports in Excel

You might want to create reports in Excel so you can perform further analysis on the data and generate graphics. Creating a report in Excel is similar to creating a report in the client, as follows:

- [Create a report](#) in Excel for the desired type of input or output report.

- Depending on the type of report, you:
 - [Generate output report data from the model to Excel.](#)
 - [Apply input report data to the model from Excel.](#)

When creating reports in Excel, you can:

- [Filter report data in Excel.](#)
- [Control the simulation from Excel.](#)
- [Connect to and disconnect from the server from Excel.](#)

Creating a Report in Excel

Before you can generate output report data or apply input report data, you must create a report in Excel. To do this, first, you create a report object in the model, then you create the report from the report object. You create the report in the following default Excel spreadsheet:

```
\\rethink\data\ReThink-Summary-Reports.xls
```

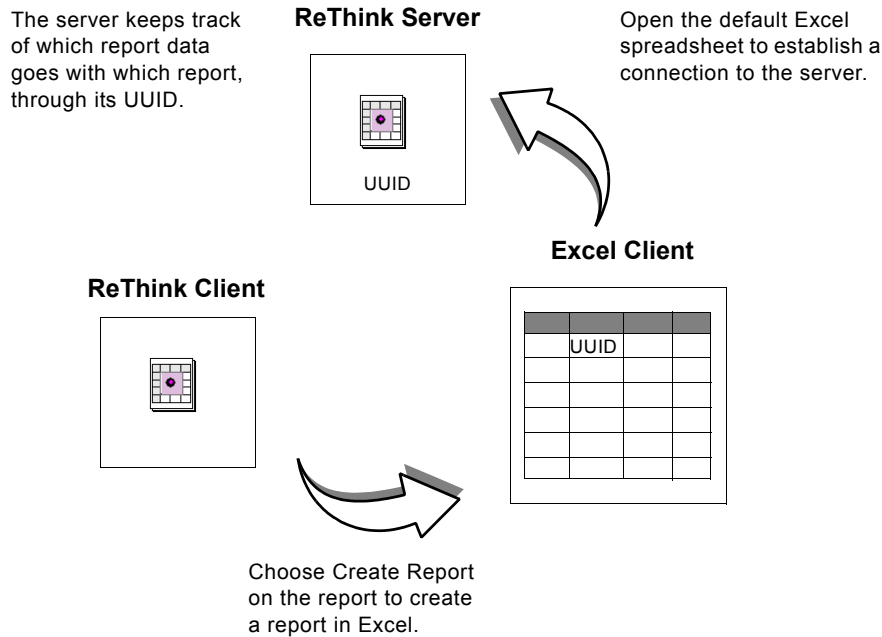
Each time you open the default Excel spreadsheet or a spreadsheet that is based on this default, the Excel client automatically attempts to establish a connection with the ReThink server. Excel can be running on the same computer as the server or on a different computer. By default, you must configure the location of the server each time you open the spreadsheet.

The ReThink server keeps track of which Excel spreadsheet corresponds to which report object, based on a unique identifier, called a UUID. The Excel spreadsheet need not be open to update an existing spreadsheet. The spreadsheet updates the next time you open it, based on changes cached in the server.

When you create a report, Excel formats the tab page associated with the report object to include the appropriate columns and rows for the particular report object and model. For example, if you create a Block Summary Report, the spreadsheet includes columns for all the block, duration, and cost metrics associated with a block, and rows for each block in the model.

Once you have created the report, you typically format the rows and columns manually to suit your needs, then save the default spreadsheet to a new name. When you are ready to generate output report data or apply input report data, you simply open the spreadsheet you saved, which is automatically configured to communicate with the correct report in the server.

The following figure illustrates the process of creating a report in Excel:



To create a report in Excel:

- 1 Create an input or output report, based on the type of data you want to enter or compute, and place it in the desired location in the model.

For details, see [Creating Reports](#).

If you place the report object on the model detail or some other detail, you can skip the following step. Otherwise, if you place the report object on the detail of an organizer, you must choose the root workspace for the report object, as the next step describes.

- 2 If necessary, choose Select Root Workspace on the report object, then select the workspace to which the report object should apply and choose Select.

The report object applies to all objects of the specified type on the selected root workspace and all details.

- 3 Display the properties dialog for the report object and configure the Report Title on the General tab to be a unique name.
- 4 In Excel, open *ReThink-Summary-Reports.xls*, located in the *rethink\data* directory of your installation directory.

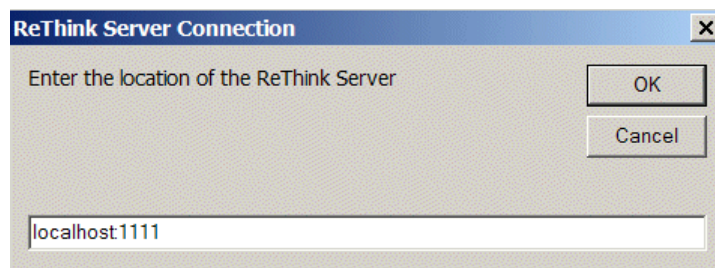
Shortcut You can open the default report by choosing Start > Programs > Gensym G2 2011 > Examples > G2 ReThink > ReThink Default Summary Reports.

The report uses a default macro to format the cells. You must enable macros each time you open the default spreadsheet.

Tip To configure Excel to enable macros automatically, choose Options from the Excel Tools menu, click the General tab, and click the Macro Virus Protection option off.

- 5** Click the Enable Macros button in the confirmation dialog that appears.

The Excel client attempts to establish a connection to the ReThink server. By default, ReThink prompts you for the location of the server by displaying this dialog:



- 6** Enter the location of the computer on which the server is running, using the following syntax:

`<host>:<port>`

You only need to edit the value if you are running the server on a computer other than localhost:1111, such as my-host:1112.

The Excel client is now connected to the server.

- 7** In ReThink, choose Show Report on the report to create the report in the default Excel spreadsheet.

Excel formats the tab page associated with the report for the particular model by creating the appropriate rows and columns.

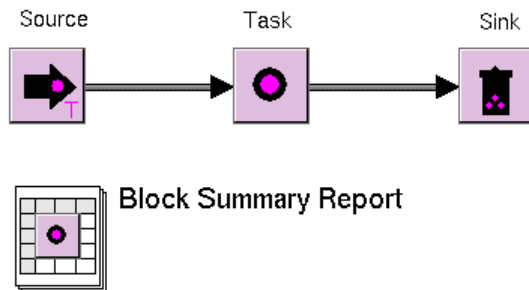
- 8** Reset the scenario.
- 9** In Excel, format the rows and columns of the report to suit your needs.

For example:

- Choose Format > Row > Hide and Format > Column > Hide to hide rows and columns whose data you do not need to see.
- Choose Format > Column > Autofit Selection to adjust the column width to match the column headers.
- Click a cell and choose Window > Freeze Panes to define column and row borders that always remain visible, even when scrolling.

10 In Excel, save the report to a new file name, such as *My-Model-Summary-Report.xls*.

This figure shows how to create a Block Summary Report in Excel:



This figure shows the resulting in the default Excel report:

The screenshot shows an Excel spreadsheet with the following data:

	A	B	C	D	E
1			Simulation Time:	12/31/2005 23:59	Model Version:
2	Block Summary Report		Simulation State:	RESETTING	Simulation Version
3	(Press Ctrl + U to update the report, Ctrl + A to apply input Reports, Ctrl + Shift + C to connect to server and Ctrl + Shift + D to				
4	Block	Total Starts	Total Stops	Current Activities	Duration Subtable.to
5					
6					
7					
8					
9					
10					
11					
12					
13					

A 'ReThink' dialog box is visible over the spreadsheet, containing a dropdown menu with 'Server', 'Report', and 'Simulation' options.

Generating Output Report Data from the Model to Excel

Before you can generate output report data from the model, you must create a report in Excel for the output report whose data you want to generate.

To generate output report data, you simply run the simulation and update the report. Each time the report updates, new data appears in the report.

By default, output reports are configured to update manually and output static data.

You can configure the report to update automatically, as described in [Updating Output Reports at Regular Time Intervals](#).

You can also configure the report to output time-series data, as described in [Keeping a History of Data Values](#).

By default, Excel does not format the data when the report updates. You can create a macro in Excel to format report data. For details, see the *Customizing ReThink User's Guide*.

For information about reducing the amount of data that appears in the report, see [Filtering Report Data in Excel](#).

To generate output report data from the model to Excel:

- 1 Open an Excel report that is based on the *ReThink-Summary-Reports.xls* spreadsheet.

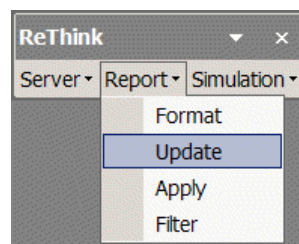
For details, see [Creating a Report in Excel](#).

- 2 Run the simulation.

For details, see [Controlling the Simulation](#).

- 3 Update the report manually, using one of these techniques:

➔ In Excel, choose Report > Update from the ReThink floating toolbar:



or

➔ In Excel, enter Ctrl + U.

or

➔ In ReThink, choose Update Report on the report.

or

➔ In ReThink, select the report whose values you want to update and choose Reports > Update.

- 4 When you have finished running the simulation, sort the report data, as needed, by choosing Data > Sort in Excel.

Note Do not attempt to sort report data before the simulation has finished, because updating the report reverts to the default sort order.

Here is a Block Summary Report for a model with three blocks:

My-Model-Summary-Reports.xls					
	A	B	C	D	E
1			Simulation Time:	1/28/2006 23:14	Model Version:
2	Block Summary Report		Simulation State:	STOPPED	Simulation Version
3	<i>(Press Ctrl + U to update the report, Ctrl + A to apply input Reports, Ctrl + Shift + C to connect to server and Ctrl + Shift + D to disconnect from server)</i>				
4	Block	Total Starts	Total Stops	Current Activities	Duration Subtable.to
5	Sink	651	651	0	0
6	Task	652	651	1	695.5480556
7	Source	653	652	1	671.2402778

Applying Input Report Data to the Model from Excel

Before you can apply input report data to the model, you must create a report in Excel for the input report whose data you want to apply.

To apply input report data, you simply enter data into the input report and apply the values to the model. Each time you apply new values, the parameters in the model update.

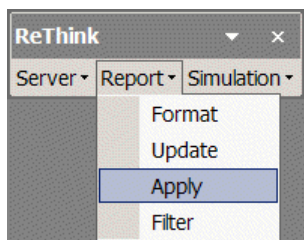
To apply input report data to the model from Excel:

- 1 Open an Excel report that is based on the *ReThink-Summary-Reports.xls* spreadsheet.

For details, see [Creating a Report in Excel](#).

- 2 Edit the spreadsheet cells for the specified parameters of the specified objects.
- 3 Apply the report data to the model, using one of these techniques:

➔ In Excel, choose Report > Apply from the ReThink floating toolbar:



or

➔ In Excel, enter Ctrl + A.

or

➔ In ReThink, select the report whose values you want to apply and choose Reports > Apply.

Excel applies the parameter values from the spreadsheet to the appropriate parameters of the appropriate objects in the model.

For example, here is the Block Input Report for a model with three blocks:

My-Model-Summary-Reports.xls					
	A	B	C	D	E
1			Simulation Time:	1/1/2006 0:00	Model Version:
2	Block Input Report		Simulation State:	STOPPED	Simulation Version
3	<i>(Press Ctrl + U to update the report, Ctrl + A to apply input Reports, Ctrl + Shift + C to connect to server and Ctrl + Shift + D to disconnect from server)</i>				
4	Block	Label	Maximum Activities	Duration Subtable.meas	Duration Subtable.standards
5	Sink	Sink		0	0
6	Task	Task		1	1
7	Source	Source		1	0

Filtering Report Data in Excel

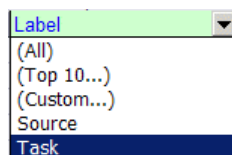
If you have many objects in a model, you might want to filter report data to view or configure a subset of the data. You can filter the report data, based on values in any column of the report. For example, you might want to show the rows associated with particular resources or the rows with the Total Cost greater than a certain number.

Note Unlike filtering report data by object class, filtering data in Excel simply hides certain data from view; the data still exists and can be displayed at any time.

To filter report data in Excel:

- ➔ Click the dropdown list for a column header and choose the filter criterion, based on available column data.

Here is the column header dropdown list for the Blocks column of a Block Summary Report:



To show all report data:

- ➔ Click the dropdown list for a column header and choose All.

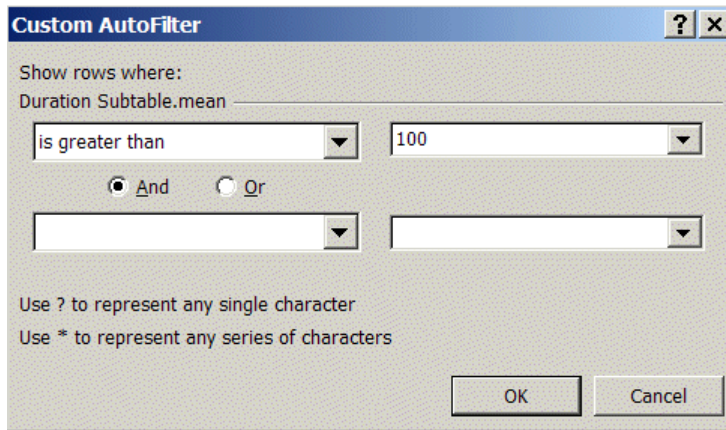
To show only the top ten rows of data:

- ➔ Click the dropdown list for a column header and choose Top 10.

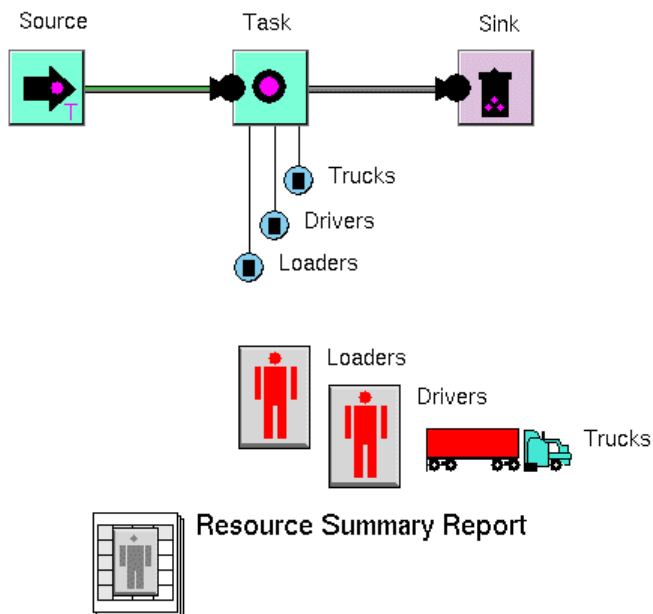
To filter report data based on custom criteria:

- ➔ Click the dropdown list for a column header, choose Custom, and configure the custom filter criteria.

For example, here is how you would configure the report data to show only those rows whose Total Cost is greater than 100:



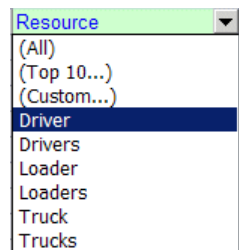
Here is a model that contains three resource pools with multiple individual resources of each type in each pool:



By default, the Resource Summary Report would look like this, with all resources visible:

My-Model-Summary-Reports.xls				
	A	B	C	D
1			Simulation Time:	1/6/2006 22:48
2	Resource Summary Report		Simulation State:	PAUSED
3	<i>(Press Ctrl + U to update the report, Ctrl + A to apply input Reports, Ctrl + Shift + C to connect to se</i>			
4	Resource	Total Starts	Total Stops	Current Activities
5	Trucks	32	31	1
6	Drivers	32	31	1
7	Loaders	32	31	1
8	Truck	17	16	1
9	Truck	15	15	0
10	Driver	17	17	0
11	Driver	15	14	1
12	Loader	13	12	1
13	Loader	19	19	0

To show report data for individual drivers, you would click the dropdown list for the Resource column header and choose Driver:

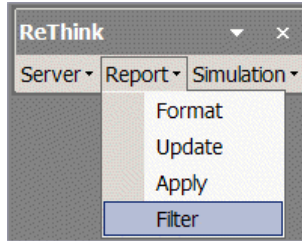


The resulting report shows report data for Driver resources only:

My-Model-Summary-Reports.xls				
	A	B	C	D
1			Simulation Time:	1/6/2006 22:48
2	Resource Summary Report		Simulation State:	PAUSED
3	<i>(Press Ctrl + U to update the report, Ctrl + A to apply input Reports, Ctrl + Shift + C to connect to se</i>			
4	Resource	Total Starts	Total Stops	Current Activities
10	Driver	17	17	0
11	Driver	15	14	1

To toggle the dropdown list buttons:

➔ Choose Report > Filter from the ReThink floating toolbar:



Here is a Block Input Report with the dropdown list buttons hidden:

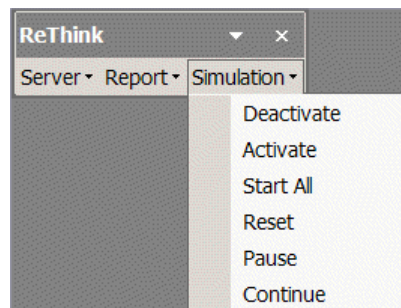
My-Model-Summary-Reports.xls						
	A	B	C	D	E	F
1	Resource		Simulatic	1/6/2006 22:48	Model Ver	0
2	Summary		Simulatic	PAUSED	Simulation	0
3	<i>(Press Ctrl + U to update the report, Ctrl + A to apply input Reports, Ctrl + Shift + C to connect to serve</i>					
4	Resource	Total Starts	Total Stops	Current Activities	Total Waiting	Duration Subtable.total Work Time
5	Trucks	32	31	1	1	152.5363889
6	Drivers	32	31	1	1	152.5363889
7	Loaders	32	31	1	0	152.5363889
8	Truck	17	16	1	0	77.34138889
9	Truck	15	15	0	0	75.195
10	Driver	17	17	0	0	80.13916667
11	Driver	15	14	1	0	72.39722222
12	Loader	13	12	1	0	65.42694444
13	Loader	19	19	0	0	87.10944444

Controlling the Simulation from Excel

When creating Excel reports, you might want to control the simulation from Excel, rather than switching back to ReThink. You can activate, deactivate, start, reset, pause, or continue the simulation from Excel.

To control the simulation from Excel:

- ➔ In Excel, choose Simulation from the ReThink floating toolbar, then choose the command to control the simulation:



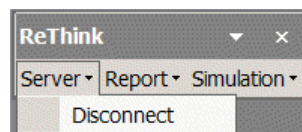
Connecting to and Disconnecting from the Server from Excel

When you first open the default report, ReThink automatically attempts to connect to the server. Exiting Excel automatically disconnects from the server.

You can manually connect and disconnect to the server, as well, for example, if for some reason you need to shut down ReThink while Excel is still open.

To connect to and disconnect from the server from Excel:

- ➔ In Excel, choose Server > Connect or Disconnect from the ReThink floating toolbar:



or

- ➔ Enter Ctrl + Shift + C to connect to the server.

Writing to and Reading from CSV Files

For any output or input report in the model, you can:

- [Write output report data to an CSV file.](#)
- [Import input report data from an CSV file.](#)

You use CSV (comma separated values) files to import report data into a graphics program or to perform further analysis on the data.

You also use CSV files when using the Batch Simulation object to run multiple simulations from a script. For more information, see [Using Batch Simulation.](#)

Writing Output Report Data to CSV Files

You can create a CSV file for any output report.

Caution Do not attempt to edit the CSV file while the simulation is running; otherwise, the model cannot write the data to the file.

To write output report data to an CSV file:

- 1 Display the properties dialog for the output report whose data you want to write to a CSV file and click the Excel tab.
- 2 Enable the Excel Report Enabled option.
- 3 Configure the report to update at the desired intervals.
For details, see [Updating Output Reports at Regular Time Intervals.](#)
- 4 Configure the report to keep a history, if desired.
For details, see [Keeping a History of Data Values.](#)
- 5 Configure the attributes to appear in the report.
For details, see [Configuring the Attributes to Appear in a Report.](#)
- 6 Configure the time unit, as needed.
For details, see [Configuring the Time Unit.](#)
- 7 Generate the output report data.
For details, see [Generating Output Report Data from the Model.](#)

For details, see [Generating Output Report Data from the Model to Excel.](#)

When you run the simulation and the report updates, ReThink writes the report data to a CSV file located in the *Output* directory of your installation directory with this format:

model-label vmodel-version - report-title vscenario-version .csv

For example, for the model named My Model, the default CSV file name for a Block Summary Report report is:

My Model V0.0 - Block Summary Report V0.0.csv

The *model-version* is the Model Version of the Model object, and the *scenario-version* is the Scenario Version of the Scenario object.

ReThink creates this file in the *Output* directory of your product installation directory.

Importing Input Report Data from CSV Files

You can import data into the model from a CSV file.

To import data into the model from an CSV file:

- 1 Display the properties dialog for the input report whose data you want to import from a CSV file and click the Excel tab.
- 2 Enable the Excel Report Enabled option.
- 3 Configure the attributes to appear in the report.
For details, see [Configuring the Attributes to Appear in a Report](#).
- 4 Configure the time unit, as needed.
For details, see [Configuring the Time Unit](#).
- 5 Create the input report, which also creates the CSV file.
For details, see [Creating a Report in Excel](#).

Tip To create a new CSV file, you must first delete the existing CSV file by choosing Delete CSV Report File on the report.

ReThink creates a CSV file located in the *Output* directory with this format:

model-label vmodel-version - report-title vscenario-version .csv

For example, for the model named My Model, the default CSV file name for a Block Input Report report is:

My Model V0.0 - Block Input Report V0.0.csv

ReThink creates this file in the *Input* directory of your product installation directory.

The report contains a header row that identifies each input parameter to configure and rows for each object in the report. Each row contains default values for each parameter in the report.

- 6 Open the CSV file, edit the input report data, and save the data in CSV file format.

Note Be sure to save the file in CSV format; do not save the file in Excel format. Also, do not edit the first row of the report or the object labels that identify each row of the report.

- 7 Choose Import Data from File on the report.

Importing data from the CSV file automatically applies the data to the model.

Writing to and Importing from Databases

For any output or input report in the model, you can:

- Write output report data to a database.
- Import input report data from a database.

For details, see [Using Reports to Access External Databases](#).

Creating Specialized Reports

You can create the following specialized reports:

- [N-Dimensional Report](#), which includes individual attributes for any number of objects of any type.
- [Indexed Lookup Report](#), which allows you to configure the duration of a block in a report.
- [Attribute Lookup Report](#), which allows you to configure the duration of a block, based on the attribute value of a work object that the block uses as an index to look up the duration in a report.
- [Attribute Change Event Report](#), which allows you to schedule parameter value changes during the simulation, similar to a Scenario Manager.

Creating N-Dimensional Reports

To facilitate data entry and analysis, you can create an N-Dimensional Input Report that includes the key parameters you need to configure and an N-Dimensional Output Report that includes the key metrics you want to analyze.

An N-Dimensional Input or Output Report includes a single attribute value for any number of objects of any type in the model. You might need to create such a report when the attributes you want to configure or output exist in individual objects of different types, rather than in objects of the same type.

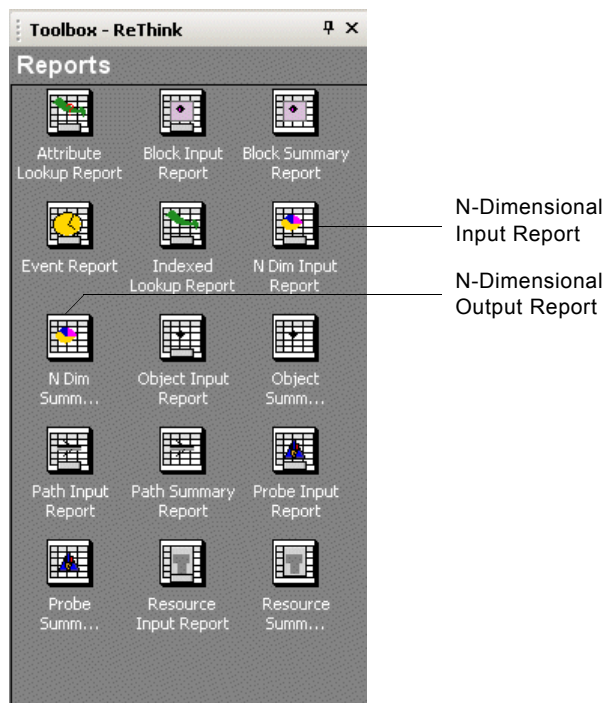
For example, you might create an N-Dimensional Output Report that includes the Average in Process for several key blocks in the model and the Average Utilization of each resource pool. You might create an N-Dimensional Input Report that includes the Mean and Standard Deviation for key blocks in the model and the Cost per Time Unit of key resources.

When configuring the attributes to appear in an N-Dimensional Input or Output Report, you must uniquely identify the object by its label or by its UUID. If you are entering labels, the labels must be unique and cannot contain carriage returns.

For example, to include the Mean parameters for several Task blocks, you must configure the labels to be unique, for example, Pack Boxes, Load Boxes, and Load Trucks.

To create an N-Dimensional Input or N-Dimensional Output Report:

- 1 Display the Reports palette of the ReThink toolbox:



- 2 Select an N-Dimensional Input or Output Report, and place it on the model detail.
- 3 Display the properties dialog for the report and configure the parameters on the General tab.

For details, see:

- [Updating Output Reports at Regular Time Intervals.](#)
- [Keeping a History of Data Values.](#)

If you are creating reports in Excel, see [Creating Reports in Excel](#).

4 Configure the attributes to appear in the report.

You configure the subtable and attribute names separately. For example, to include the Label attribute, the subtable is **general** and the attribute is **label**. To include the Average Utilization attribute of a resource, the subtable is **duration-subtable** and the attribute is **average-utilization**. To include the Total Cost of a block, the subtable is **cost-subtable** and the attribute is **total-cost**.

For more information, see [Configuring the Attributes to Appear in a Report](#).

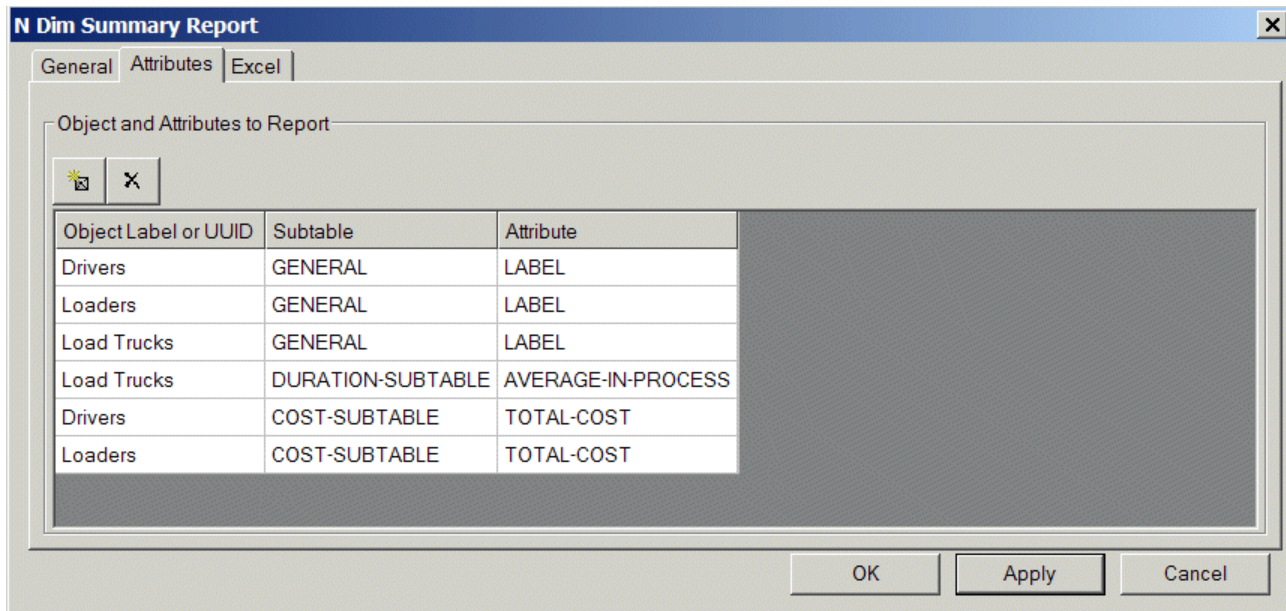
Tip You can use Ctrl + X, Ctrl + C, and Ctrl + V, respectively, to cut, copy, and paste values, as needed.

5 Create the report, then generate the output report data or apply input report data to the model.

For details, see one of the following, depending on where you are creating the report:

- [Creating Reports](#).
- [Creating Reports in Excel](#).

Here is the Attributes tab for an N-Dimensional Output Report that includes the labels for all objects, the Average in Process of the Load Trucks task, the Total Cost for the Drivers and Loaders resources.



Here is a sample N-Dimensional Output Report after running the simulation for a period of time:

N Dimensional Output Report			
Update Time: 1-4-2006 12:00:0 am		Time Unit: Hours	Model Version: 0.0
Object	General.label	Duration Subtable.average In Process	Cost Subtable.total Cost
PERSON	Drivers		162.0
PERSON	Loaders		166.0
BPR-TASK	Load Trucks	0.764	328.0

Creating Indexed Lookup Reports

You might know exactly when an activity occurs in your model, based on durations in a report. To do this, you create an Indexed Lookup Report with a column that defines the durations, then you configure the duration of the block to use the data in the report.

For details, see [Specifying Duration Based on an Indexed Report Lookup](#).

Creating Attribute Lookup Reports

You might have a block that processes several different types of objects, where an attribute of the work object is an index that the block uses for looking up the duration in a report. To do this, you create an Attribute Lookup Report with columns that define the attribute to use as the index and the durations, then you configure the duration of the block to use the data in the report.

For details, see [Specifying Duration Based on an Attribute Report Lookup](#).

Creating Attribute Change Event Reports

You might want to schedule parameter value changes to the model while the simulation is running. For example, over the course of the simulation, the

manufacturing time might decrease as the process becomes more efficient, or the frequency of orders might decrease over the life cycle of a product.

If you have a relatively large number of parameter changes, you can use an Attribute Change Event Report to configure those changes. Alternatively, if you have a relatively small number of parameter changes, you can use a Scenario Manager.

An Attribute Change Event Report includes these predefined columns:

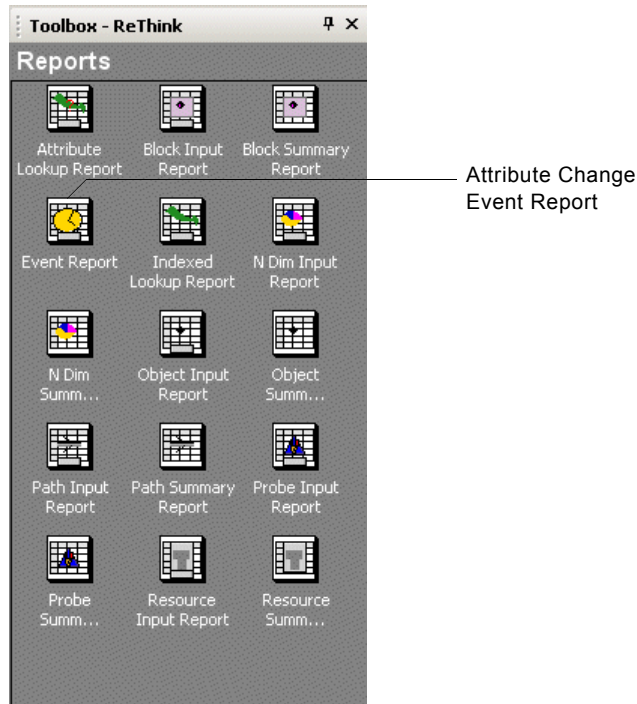
Column	Description
Duration	The time at which the scheduled parameter value change should occur, from the beginning of the simulation. Configure the duration in the time unit of the report, which is hours, by default.
Object Label	The value of the Label for the object whose parameter values should change.
Object UUID	The unique ID for the object. This value is required if the Object Label is not unique.
Subtable Name	<p>The name of the subtable in which the attribute appears, if any. The subtable names correspond with the tabs in the properties dialogs, as follows:</p> <ul style="list-style-type: none">• duration-subtable includes attributes on the Duration tab.• cost-subtable includes attributes on the Cost tab.• animation-subtable includes attributes on the Animation tab. <p>If the attribute does not appear on the Duration, Cost, or Animation tabs of the properties dialog, the Subtable Name is none.</p>

Column	Description
Attribute Name	The name of the attribute whose value should change, as a symbol. For example, mean and standard-deviation .
Attribute Value	<p>The value of the attribute to change. If the attribute value is a duration, you must specify the value in seconds or using an expression with this format:</p> <p style="text-align: center;"><i>ww</i> weeks, <i>dd</i> days, <i>hh</i> hours, <i>mm</i> minutes, and <i>ss</i> seconds</p> <p>For example, 4 weeks, 2 days, or 1 hour and 30 minutes.</p>

You can enter values in the report manually, or you can add attribute values to the report through the dialogs. If you are entering many values, it is typically easier to add attributes initially through the dialogs. If the object labels are not unique, you must add the attributes through the dialogs to enter the UUID in the report. You can enter values for any number of attributes and any number of objects.

To create and use an Attribute Change Event rePort:

- 1 Display the Reports palette of the ReThink toolbox:



- 2 Create an Attribute Change Event Report and place it on the model detail or organizer detail.
- 3 Configure the Time Unit of the report to be convenient for entering values, as needed.

For example, if you are scheduling updates over a period of days or weeks, configure the Time Unit of the report to be 1 day or 1 week. Remember, you must enter all durations in the time period of the report.

For details, see [Configuring the Time Unit](#)

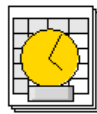
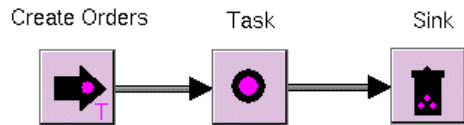
- 4 Create the report.
For details, see one of the following, depending on where you are creating the report:
 - [Creating Reports](#).
 - [Creating Reports in Excel](#).
- 5 Click Add Row to add as many rows as you need to configure the report.
- 6 Enter values in each column in the report to schedule attribute value changes, as needed.

Enter values directly into the report manually, then edit the Duration and Attribute Value columns to schedule attribute value changes. If you add attributes to the report through the dialogs, you must update the report to see the new attributes. Once the attributes exist in the report, you can copy and paste rows in the report, then edit the Durations and Attribute Value columns to schedule attribute changes for the same object.

7 Run the simulation.

The model uses the specified attribute values during the simulation, based on the durations.

Here is a model that uses an Attribute Change Event Report to schedule attribute value changes during the simulation:



Attribute Change Event Report

Here is the report that schedules changes to the Mean attribute of the Orders block over a period of 10 days. The value increases, remains high for a period of days, then decreases to its original level.

Attribute Change Event Report					
Update Time: 1-6-2006 6:16:12 am		Time Unit: Hours		Model Version: 0.0	
Duration	Object Label	Object Uuid	Subtable Name	Attribute Name	Attribute Value
1 day	Create Orders		duration-subtable	mean	8 hours
1 week	Create Orders		duration-subtable	mean	2 hours
2 weeks	Create Orders		duration-subtable	mean	1 hour
3 weeks	Create Orders		duration-subtable	mean	2 hours
4 weeks	Create Orders		duration-subtable	mean	8 hours

Accessing External Databases

Describes how to access databases.

Introduction	417
Configuring ReThink for Database Access	418
Creating a Work Object that Represents a Record	425
Creating an SQL Query for Accessing the Data	427
Sourcing Records from a Database	428
Retrieving Records from a Database	432
Storing Work Objects to a Database Table	435
Using Reports to Access External Databases	438



Introduction

You can access external databases directly through ReThink to:

- Generate work objects, using data from an external database.
- Retrieve data from an external database.
- Store work objects to a database table.
- Write report data to a database.
- Read report data from a database.

To access external databases, you must create a database and configure the ODBC data source. You then create and configure a Database Interface object as the link between the database and ReThink.

These blocks support database access:

- The Source block, which generates work objects from an external database.
- The Retrieve block, which retrieves work objects from an external database.
- The Store block, which stores work objects to an external database or updates existing records.

When you run the simulation, ReThink accesses the external database dynamically, as follows:

- When you generate or retrieve data from a database, ReThink dynamically creates or retrieves one work object for each database record that matches the SQL query you specify.
- When you store data to a database, ReThink dynamically adds new records to or updates existing records of the specified table of the external database.

To configure a report for database access, you specify the database table and enable database reporting. When you run the simulation, ReThink writes output report data to a database when the report updates. To read input report data from a database, you must explicitly import the data.

Note To configure ReThink for database access, you must be in Developer mode.

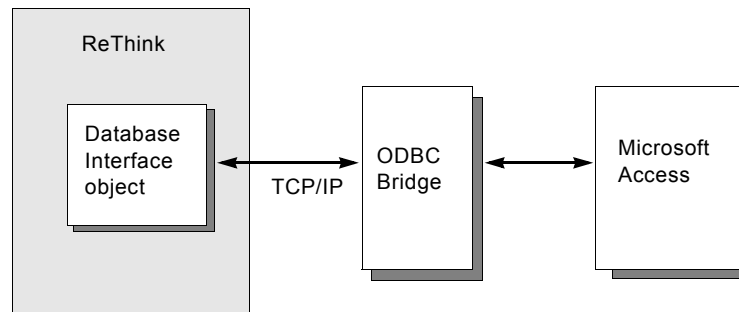
- ReThinkCreate a work object that represents a record.
- Create an SQL query for accessing the data.
- Source records from a database.
- Retrieve records from a database.
- Store work objects to a database table.

Configuring ReThink for Database Access

To access an external database, you must first start the ODBC Bridge and configure your computer to use the bridge. You can access any external database that the ODBC Bridge supports, including, Microsoft Access, Oracle, and SQL2000.

The ODBC Bridge allows ReThink to communicate with external databases via a Database Interface object and a TCP/IP connection.

This figure shows how ReThink communicates with a Microsoft Access database through the ODBC Bridge:



To configure ReThink for database access, you:

- [Create the database.](#)
- [Configure the ODBC data source.](#)
- [Start the ODBC Bridge process.](#)
- [Create and configure a Database Interface object.](#)
- [Connect to the database.](#)

Creating the Database

You create the database by using any database that the ODBC Bridge supports.

If your model is generating or retrieving work objects from a database, using the Source block or Retrieve block, respectively, or if your model is storing work objects to a database, using the Store block, you must create the database table manually. Each record field corresponds to an attribute of a work object that the model generates, retrieves, or stores.

If you are sourcing or retrieving work objects from a database, you must also populate the database with data. You can populate the database manually, or you can populate by using the Store block, as described in [Storing Work Objects to a Database Table](#).

The examples use the sample database named *Orders.mdb*.

If your model is writing report data to a database or reading report data from a database, you create the database manually, then you can create the database table directly from the report object.

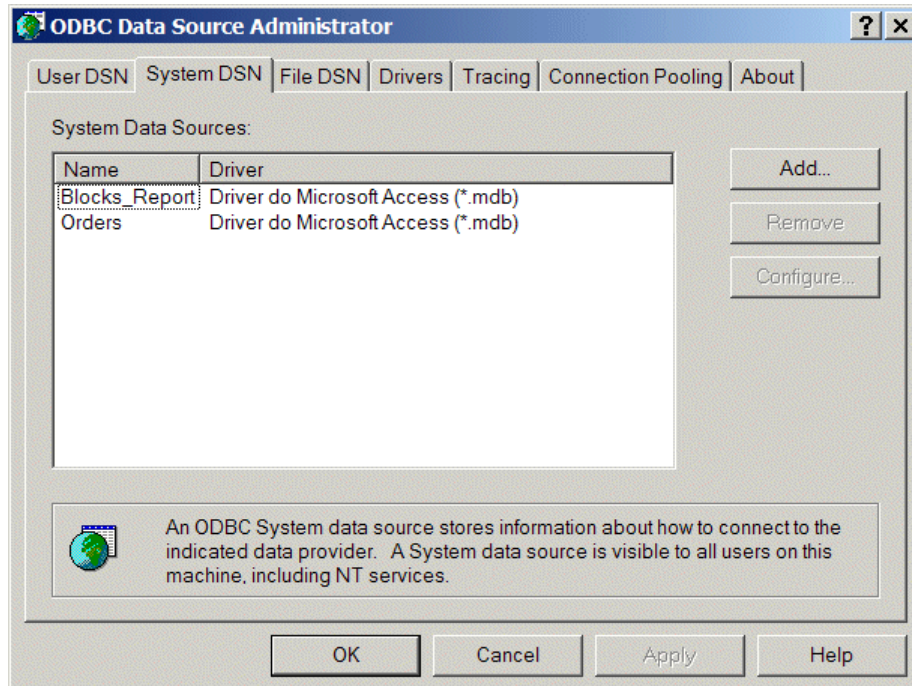
Configuring the ODBC Data Source

Once you have created your database, you must configure the ODBC data source on your computer to name the data source and point to your database. The following example shows how to configure the ODBC data source for a Microsoft Access database named *Blocks_Report.mdb*.

To configure the ODBC data source:

- 1 Display the ODBC Data Source Administrator dialog by choosing Start > Programs > Administrative Tools > Data Sources (ODBC).
- 2 Click the System DSN tab.
- 3 Click Add to add a new ODBC data source.
- 4 Choose the appropriate driver for the database you created, for example, Microsoft Access Driver (*.mdb) and click Finish.
- 5 Configure the Data Source Name to be any name, for example, *Orders* or *Blocks_Report*.
- 6 Click the Select button, navigate to your database, and click OK.
- 7 Click OK in the ODBC Microsoft Access Setup dialog to create the new data source.

The ODBC Data Source dialog should have an entry for each data source. For example, the following dialog shows the *Orders* and *Blocks_Report* data sources:



Starting the ODBC Bridge Process

Once you have configured the ODBC data source, you can start the ODBC Bridge process. You must identify the host and port to which the bridge is connected for configuring in the Database Interface object.

To start the ODBC Bridge process:

→ Choose Start > Programs > Gensym G2 2011> Bridges > G2 ODBC Bridge.

The ODBC Bridge process appears in the command window.

To determine the bridge port:

→ Open the command window for the bridge process.

The last line indicates the TPC/IP host and port number, for example:

```
TCP_IP:NSALVO-1165:22041
```

Creating and Configuring the Database Interface Object

The Database Interface object specifies:

- A name, which the reports and block use to connect to the database.
- The ODBC source as a connect string.
- The host and port of the machine running the database bridge.

If the bridge process is running on the local machine, the host is `localhost`. The port number is `22041`, or `22042`, or `22043`, and so on, depending on the number of clients that are currently connected on that port.

You create a Database Interface object for each database you want your model to access. Typically, you write data to one database and read data from another database, which means you must create two Database Interface objects.

Note To configure a Database Interface object, you must be in Developer mode.

To create and configure a database interface object:

- 1 Switch to Developer mode.
For details, see [Switching User Modes](#).
- 2 Choose Project > System Settings > Interfaces > SQL > Manage and click the New button to create a new Database Interface object.
Alternatively, you can choose View > Toolbox - G2, click the Network Interfaces tab, and create a Database Interface object.
- 3 In the properties dialog for the Database Interface object, configure the Interface Name attribute to be any symbol, for example, `orders-database-interface`.

Tip This is the Database Interface Name you specify when you configure the report or block for database access.

- 4 Configure the Type of Database to be Access-ODBC.
- 5 Configure the Bridge Host and Bridge Port attributes to match the host and port of the machine running the bridge, `localhost` and `22041`, by default.
- 6 Configure the Connect String attribute to be the name of the ODBC data source, for example: `orders`.
- 7 Click Apply to apply these values.
- 8 Choose Tools > User Mode > Modeler to return to Modeler mode.

Here is a the Database Interface object named orders-database-interface and its properties dialog:



ORDERS-DATABASE-INTERFACE

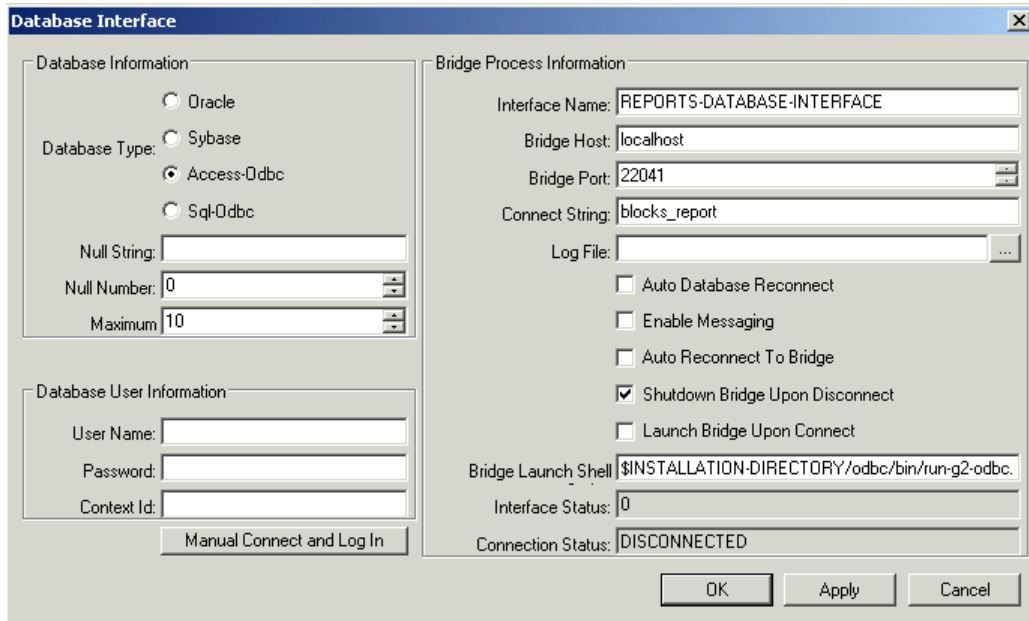
Database Interface [X]

<p>Database Information</p> <p> <input type="radio"/> Oracle <input type="radio"/> Sybase <input checked="" type="radio"/> Access-Odbc <input type="radio"/> Sql-Odbc </p> <p>Null String: <input type="text"/></p> <p>Null Number: <input type="text" value="0"/></p> <p>Maximum: <input type="text" value="10"/></p> <hr/> <p>Database User Information</p> <p>User Name: <input type="text"/></p> <p>Password: <input type="text"/></p> <p>Context Id: <input type="text"/></p> <p><input type="button" value="Manual Connect and Log In"/></p>	<p>Bridge Process Information</p> <p>Interface Name: <input type="text" value="ORDERS-DATABASE-INTERFACE"/></p> <p>Bridge Host: <input type="text" value="localhost"/></p> <p>Bridge Port: <input type="text" value="22041"/></p> <p>Connect String: <input type="text" value="orders"/></p> <p>Log File: <input type="text"/></p> <p> <input type="checkbox"/> Auto Database Reconnect <input type="checkbox"/> Enable Messaging <input type="checkbox"/> Auto Reconnect To Bridge <input checked="" type="checkbox"/> Shutdown Bridge Upon Disconnect <input type="checkbox"/> Launch Bridge Upon Connect </p> <p>Bridge Launch Shell: <input type="text" value="\$INSTALLATION-DIRECTORY/odbc/bin/run-g2-odbc."/></p> <p>Interface Status: <input type="text" value="0"/></p> <p>Connection Status: <input type="text" value="DISCONNECTED"/></p> <p> <input type="button" value="OK"/> <input type="button" value="Apply"/> <input type="button" value="Cancel"/> </p>
---	---

Here is a Database Interface object named reports-database-interface and its properties dialog:



REPORTS-DATABASE-INTERFACE



Connecting to the Database

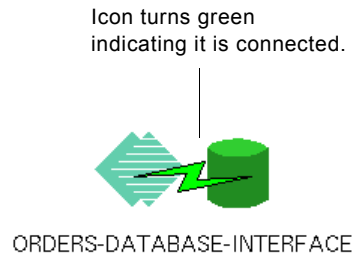
Once you have started the ODBC bridge and configured the Database Interface object, you can connect to the database via the Database Interface object. You must be connected to write records to or read records from the database.

To connect to the database:

- 1 Switch to Developer mode.
For details, see [Switching User Modes](#).
- 2 Choose Connect on the Database Interface object or click the Manual Connect and Log In button in the properties dialog.

The color of the Database Interface object turns to green and the Interface Status in the properties dialog becomes 2 to indicate it is connected.

Here is a Database Interface object that is currently connected to the database:



Creating a Work Object that Represents a Record

If you are generating data from a database, using the Source block, or if you are retrieving data from a database, using the Retrieve block, you must create a work object that is a type of database record. To create a record, you create an object definition that inherits from the `bpr-object` class and from the `db-qo-record` class.

The class definition must declare class-specific attributes for all the database fields that the model generates or retrieve through an SQL query. Each work object that the model generates or retrieves corresponds to a record in the database.

To use the work object in a model, you configure the input path type of the block to be a kind of database record.

This topic describes how to:

- [Create a class definition for a query object.](#)
- [Use a query object in a model.](#)

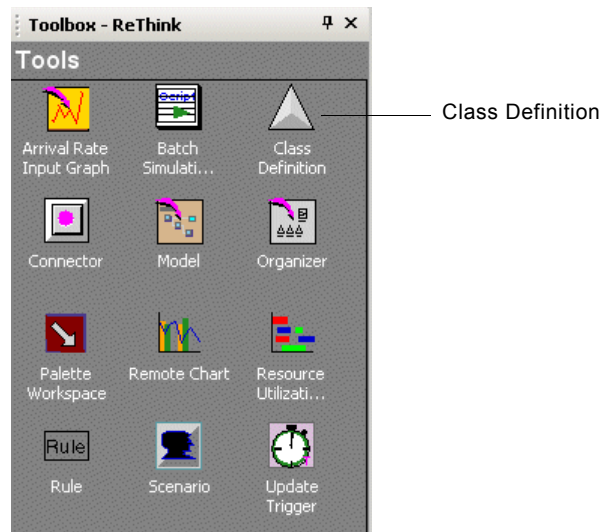
Creating a Class Definition for a Query Object

When creating a class definition for the query object, the order of the attributes does not matter; however, the names must match exactly. Also, it is not necessary to create attributes for every field in the database record; it is only necessary to create attributes for those database fields that match the SQL query. Finally, some databases do not allow hyphens in field names; therefore, you should not use them in attribute names. Use underscores instead.

For information on specifying the SQL query, see [Creating an SQL Query for Accessing the Data.](#)

To create a class definition for a query object:

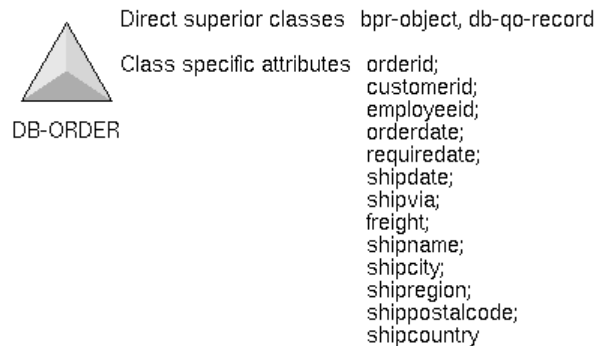
- 1 Display the ReThink toolbox and display the Tools palette:



- 2 Select a Class Definition and place it on the model detail or organizer detail.
- 3 Display the properties dialog for the class definition.
- 4 Configure the Names to be a unique class name, for example, db-order.
- 5 Configure the Direct Superior Classes to be:
bpr-object, db-qo-record
- 6 Configure the Class Specific Attributes attribute to name all the fields in the database record that the Source block or Retrieve block will access.

Note Do not provide types and/or default values for the class-specific attributes of the record; otherwise, ReThink will create the record rather than obtaining it from the database.

This example shows a class definition for a work object named `db-order` that stores data from the Orders database:



Using a Query Object in a Model

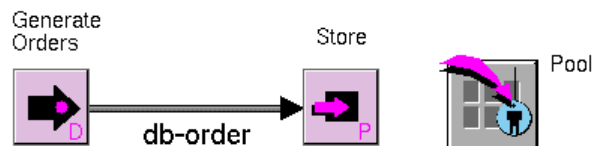
You use query objects in models that generate or retrieve data from a database, using the Source block or Retrieve block.

To use the database record in a model:

- ➔ Display the properties dialog for the output path of a Source block or a Retrieve block, and configure the Type to be the name of the database record.

See [Retrieving Records from a Database](#) and [Sourcing Records from a Database](#) for specific information on how to configure these blocks for database access.

Here is a simple model that generates `db-orders` from a database and stores them in a pool, where `db-order` is a database record:



Creating an SQL Query for Accessing the Data

When you generate or retrieve data from a database, using the Source or Retrieve block, respectively, you must specify an SQL query to extract from the database. You can also specify an SQL query when storing data to a database. The SQL query you specify uses standard syntax for retrieving records from a database.

Note The SQL query should not contain any carriage returns. Also, the SQL query must use single quotes around text values, not double quotes.

The class definition for the database record must contain class-specific attributes for all fields in the query. For information on creating a class definition for a record, see [Creating a Work Object that Represents a Record](#).

For example, here is an SQL query that retrieves all of the records in the Orders table where the employee name is Laura Callahan. The employee first and last name are not actually in the Orders table; they are in the Employees table. This query does a join operation by combining the Orders and Employees tables, using the EmployeeID field. The `Orders.*` returns all of the fields in the Orders table, without having to specify them explicitly.

```
Select Orders.* from Orders, Employees where Employees.FirstName =  
'Laura' and Employees.LastName = 'Callahan' and Orders.EmployeeID =  
Employees.EmployeeID
```

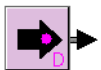
The SQL query can include expressions to be evaluated at runtime by delimiting the expression in square brackets ([]). The expression can refer to the input work object of the Retrieve block by using the variable named `InputObject`.

For example, the following query retrieves all the records in the Orders table in which the OrderID field is between the current value of the MinimumOrderID and the MaximumOrderID of the input work object:

```
“Select * from Orders where OrderID >= [the MinimumOrderID of InputObject]  
and OrderID <= [the MaximumOrderID of InputObject]”
```

ReThink queries the external database and generates or retrieves one work object for each record that matches the query, depending on whether you are using a Source block or Retrieve block. The attributes of the work object correspond to the fields in each record.

Sourcing Records from a Database



You can generate work objects directly from an external database, using the Source block. When you generate data from a database, you configure the output path type of the Source block to determine the type of work object. The work object you specify must define attributes for each field in the database record that match the query. You can create work objects that contain all the database record fields or a subset of fields, depending on the query you specify.

By default, the Source block generates work objects continuously by looping back to the beginning of the database table when it reaches the end. You can control whether the block stops when it reaches the end of the table or continues to generate records.

To source records from a database, you must first populate the database.

The following example uses a sample database called `Orders.mdb`.

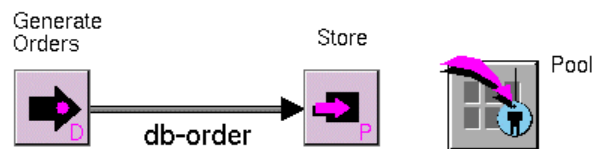
To source records from a database:

- 1 Configure ReThink for database access.
For details, see [Configuring ReThink for Database Access](#).
- 2 Create a class definition that is a record.
For details, see [Creating a Work Object that Represents a Record](#).
- 3 Display the properties dialog for the output path of the Source block and configure the Type to be the name of the database record whose attributes correspond to the database fields.
- 4 Click the Block tab and configure the Source Mode to be Database.
- 5 Click the Database tab and configure the Database Interface Name to be the Database Interface object that provides access to the database.

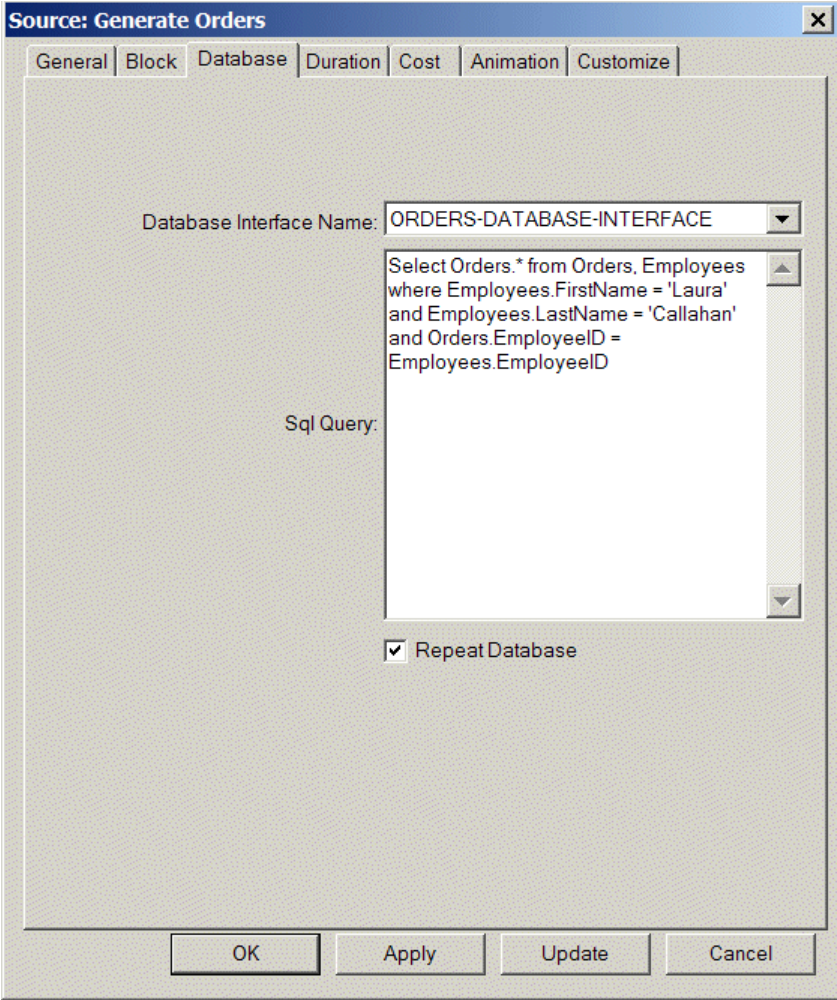
Tip If you have populated the external database by using the Store block, the Database Interface Name attribute is the same as the Database Interface Name you specify in the Store block.

- 6 Configure the SQL Query to access records from the database.
For details, see [Creating an SQL Query for Accessing the Data](#).
- 7 Configure the Repeat Database option to determine whether to generate work objects continuously or stop when it reaches the end of the table.

This model shows how to generate orders from an orders database:



Here is the Database tab of the properties dialog for the Source block that uses the SQL Query shown earlier:



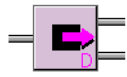
Here is the User tab of the properties dialog for an order generated from the database:

The screenshot shows a dialog box titled "Db Order" with a close button (X) in the top right corner. The dialog has several tabs: "General", "Utilization", "Cost", "Animation", "User (1)", and "User (2)". The "User (1)" tab is currently selected. The dialog contains the following fields and values:

- Customerid: RATTC
- Effective Data Type: SYMBOL
- Employeeid: 8
- Evaluation Attributes: structure (MAY-REFER-TO-INACTIVE-ITEMS: false, MAY-RUN-WHILE-INACTIVE: false)
- Freight: 48.29
- Initial Value: GSI
- Last Recorded Value Text: GSI
- Orderdate: 1993-07-19 00:00:00
- Orderid: 10262
- Shipcity: Albuquerque

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Retrieving Records from a Database



You can retrieve records from an external database, using the Retrieve block. When you retrieve data from a database, you configure the output path type of the Retrieve block to determine the type of work object. The work object must define attributes for each field in the database record that you want to retrieve. You can retrieve work objects that contain all the database record fields or a subset of fields, depending on the query you specify.

To retrieve work objects from a database:

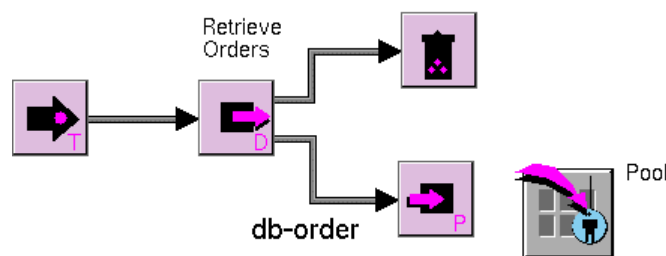
- 1 Configure ReThink to retrieve data from an external database.
For details, see [Configuring ReThink for Database Access](#).
- 2 Create a class definition that is a record.
For details, see [Creating a Work Object that Represents a Record](#).
- 3 Display the properties dialog for the output path of the Retrieve block and configure the Type to be the name of the database record object whose attributes correspond to the database fields.
- 4 Click the Block tab and configure the Retrieve Mode to be Database.
- 5 Click the Database tab and configure the Database Interface Name attribute to be the database interface object that allows access to the external database.

Tip If you have populated the external database by using the Store block, the Database Interface Name attribute is the same as the Database Interface Name you specify in the Store block.

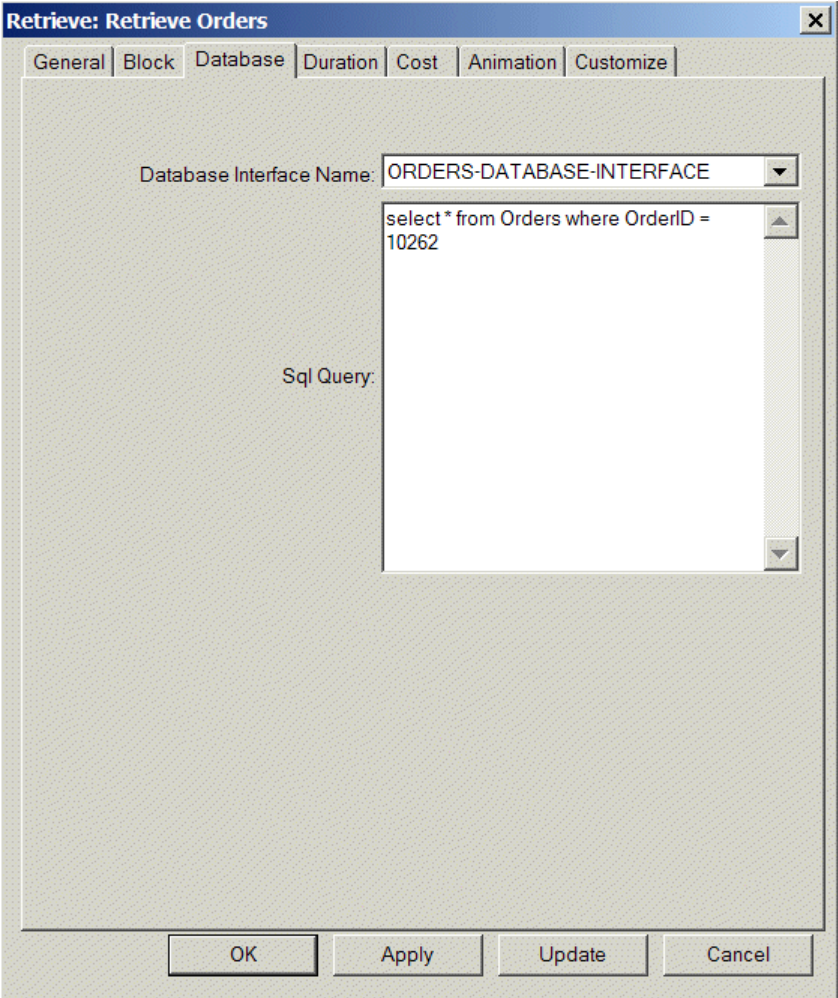
- 6 Configure the SQL Query to access records from the database.

For details, see [Creating an SQL Query for Accessing the Data](#).

This model shows how to retrieve orders from the orders database:



Here is the Database tab of the properties dialog for the Retrieve block:



Here is the User tab of the properties dialog for the order retrieved from the database:

The screenshot shows a dialog box titled "Db Order" with a close button (X) in the top right corner. The dialog has several tabs: "General", "Utilization", "Cost", "Animation", "User (1)", and "User (2)". The "User (1)" tab is currently selected. The dialog contains the following fields and values:

- Customerid: RATTC
- Effective Data Type: SYMBOL
- Employeeid: 8
- Evaluation Attributes: structure (MAY-REFER-TO-INACTIVE-ITEMS: false, MAY-RUN-WHILE-INACTIVE: false)
- Freight: 48.29
- Initial Value: GSI
- Last Recorded Value Text: GSI
- Orderdate: 1993-07-19 00:00:00
- Orderid: 10262
- Shipcity: Albuquerque

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Storing Work Objects to a Database Table



You can store work objects to a database table, using a Store block. When you store objects to a database table, ReThink creates one record for each work object that the Store block processes. The fields of each record correspond to the user-defined attributes of each work object. The database table must initially contain fields for each user-defined attribute of the work object that the Store block receives. The database updates dynamically when you run the ReThink model.

You can use the database that the Store block creates as input to a Source block to generate work objects in a model. In this way, you can create a reproducible model that uses the same values each time you run the simulation. For more information, see [Sourcing Records from a Database](#).

You can also use the Store block to update existing database records, based on an attribute of the work objects you are storing.

This topic describes how to:

- [Store new objects in a database.](#)
- [Update existing records in a database.](#)

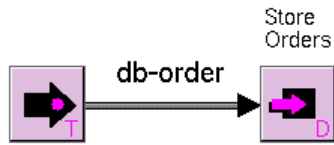
Storing New Objects in a Database

When you store user-defined attributes to a database, ReThink only stores attributes that have a value. If the attribute does not have a value, ReThink ignores the attribute.

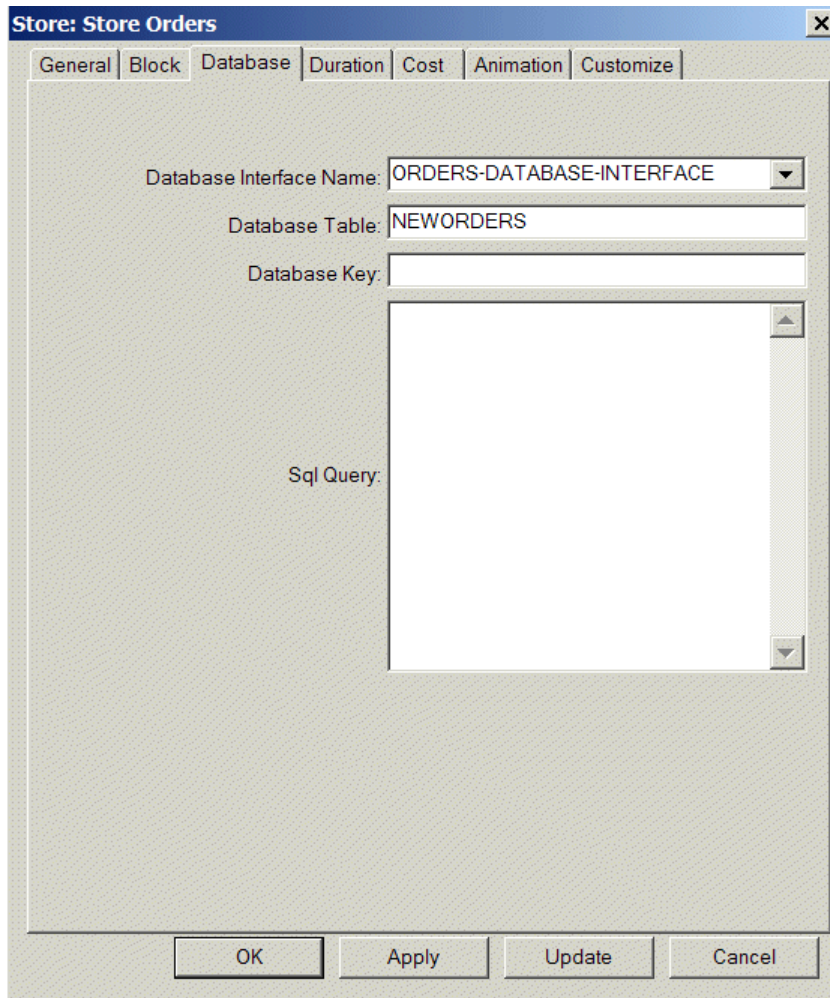
To store new objects in a database:

- 1 Configure ReThink for database access.
For details, see [Configuring ReThink for Database Access](#).
- 2 Generate an external database table.
For details, see [Creating the Database](#).
- 3 Display the properties dialog for the Store block, click the Block tab, and configure the Store Mode to be Database.
- 4 Click the Database tab and configure the Database Interface Name attribute to be the database interface object that allows access to the external database.
- 5 Configure the Database Table attribute to be the name of the table within the database in which the records will be written.

This model shows how to store orders to the orders database:



Here is the Database tab of the properties dialog for the Store block that stores data to a database table named `neworders`:



Updating Existing Records in a Database

You might want to update existing database records, using work objects that the ReThink model processes. You use the Database Key attribute of the Store block to update existing records, as opposed to storing new records.

If the value of the attribute of the work object that is named by Database Key matches the value of the corresponding field in an existing database record, ReThink updates the record, rather than creating a new record.

If the Database Key attribute is unspecified, ReThink creates new records for all work objects it receives.

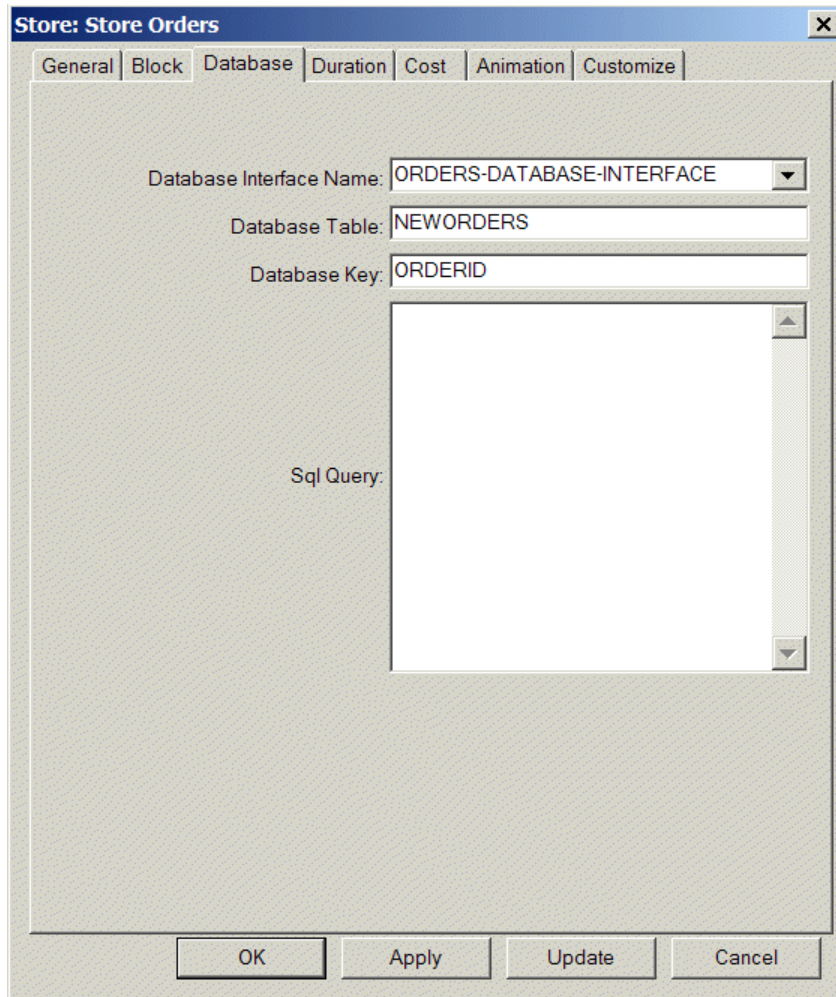
By default, you specify the name of a database table, and ReThink creates its own query for creating or updating the records. If you do not specify a table, the Store block uses the SQL Statement that you configure. ReThink stores one work object in each record that matches the query, which can refer to attributes of the work object. The attributes of each work object correspond to the fields of each record.

To update existing records in a database:

- 1 Follow the steps for storing new work objects to a database table.
For details, see [Storing New Objects in a Database](#).
- 2 On the Database tab of the properties dialog for the Store block, configure the Database Key to be an attribute of the work object that the block uses to determine whether the record exists.
- 3 Configure the Database Table or SQL Query, depending on whether you want ReThink to configure the query for you or whether you want to configure it yourself.

For details, see [Creating an SQL Query for Accessing the Data](#).

Here is the Database tab of the properties dialog for a Store block that updates existing records whose orderid attribute value matches the Orderid field of an existing record in the database:



Using Reports to Access External Databases

For any output or input report in the model, you can:

- [Configure the report object for database access.](#)
- [Write output report data to a database.](#)
- [Import input report data from a database.](#)

Configuring Report Objects for Database Access

To configure a report object for database access, you must identify the Database Interface object that connects to the database. You must also identify the database table to access. You can create the database table from the report object if it does not exist.

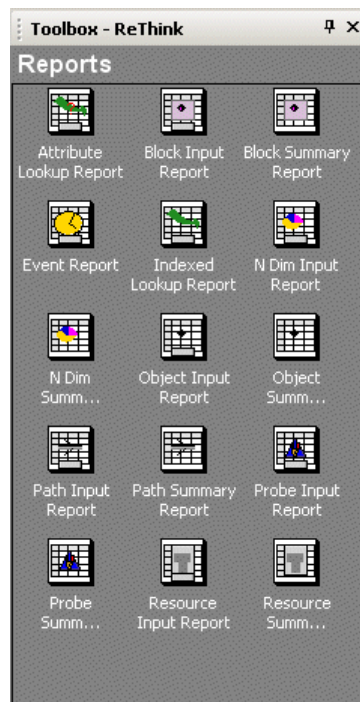
The resulting database provides columns for each attribute configured in the report. The resulting database also provides these standard columns, which you can use for querying data:

- Row_ID, which is a unique ID for the row.
- Model_Version, which is the value of the Model Version attribute of the associated Model tool.
- Simulation_Version, which is the value of the Simulation Version attribute of the associated Scenario tool.

Note To use reports to access external databases, you must be in System-Administrator mode.

To configure a report object for database access:

- 1 Display the ReThink toolbar and click the Reports tab:



2 Create and configure the report:

For details, see:

- [Creating Reports](#).
- [Configuring the Time Unit](#).
- [Updating Output Reports at Regular Time Intervals](#).
- [Keeping a History of Data Values](#).
- [Configuring the Scope of the Report](#).
- [Filtering Report Data](#).
- “Configuring the Attributes to Appear in a Report” on page 372.

3 Click the Database tab of the properties dialog for the report object, then click the Database Reporting Enabled option on.

4 Configure the Database Interface Name to be the name of an existing database interface object.

For details, see [Configuring ReThink for Database Access](#).

5 Configure the Database Table Name to be the name of an existing table in the database or a new name.

The table name must be a legal database table name, with no spaces or hyphens and no more than 32 characters, for example, `block_output_report`.

6 Click Apply to apply the values configured above.

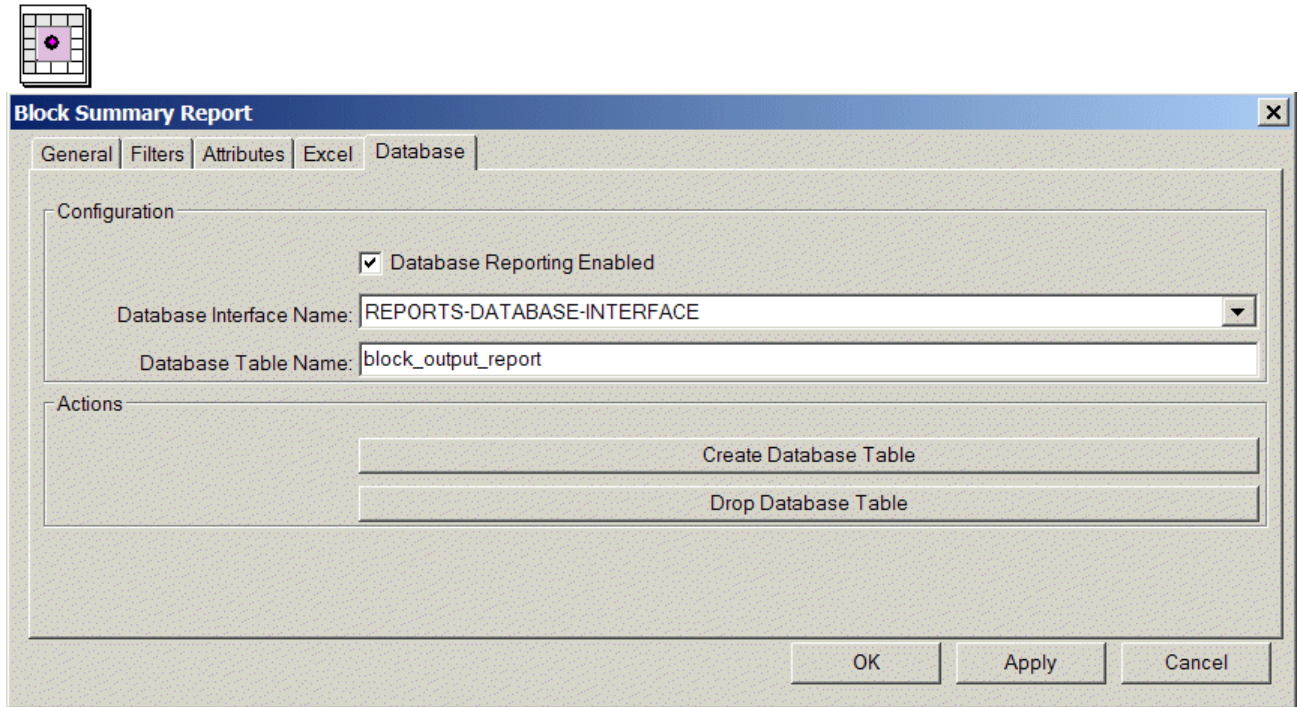
7 If you entered the name of a database table that does not exist, click the Create Database Table button to create the table.

Note The report must already be configured for database reporting before you can create a table.

If you are using Microsoft Access, ensure that the database is closed before attempting to create the table.

The database table now exists in the database with database fields for each attribute defined in the report object. You can click Drop Database Table to drop the table, as needed.

Here is the Database tab for a Block Summary Report that is configured to access the ODBC data source defined by the reports-database-interface Database Interface object. The report writes data to the table named `block_output_report`.



Block Summary Report

General | Filters | Attributes | Excel | Database

Configuration

Database Reporting Enabled

Database Interface Name: REPORTS-DATABASE-INTERFACE

Database Table Name: block_output_report

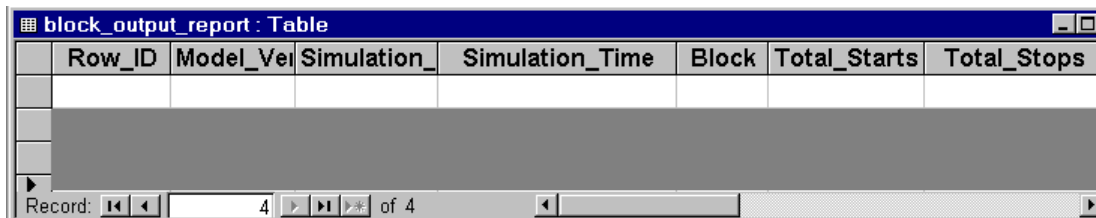
Actions

Create Database Table

Drop Database Table

OK Apply Cancel

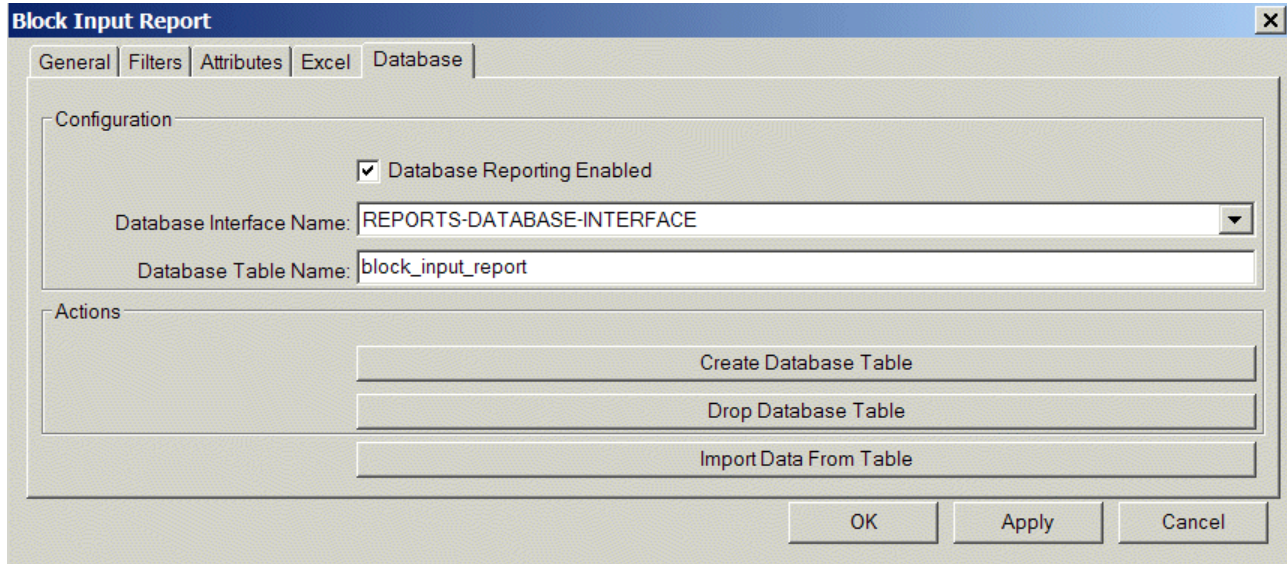
Here is the resulting Microsoft Access database table that is created for the default attributes of the Block Summary Report *before* creating or updating the report:



Row_ID	Model_Vel	Simulation_	Simulation_Time	Block	Total_Starts	Total_Stops

Record: 4 of 4

Here is the Database tab for a Block Input Report that is configured to access the ODBC data source defined by the input-report-database-interface Database Interface object. The report imports data from the table named block_input_report.



Here is the resulting Microsoft Access database table that is created for the default attributes of the Block Input Report before entering any data:

Row_ID	Model_Version	Simulation_Ver	Simulation_Tirr	Block	Label

Record: 1 of 3

Writing Output Report Data to a Database

To write output report data to a database, you simply create the report and run the simulation. ReThink writes the report data to the database each time the report updates, either manually or based on clock time or simulation time, and at the end of the simulation.

To write output report data to a database:

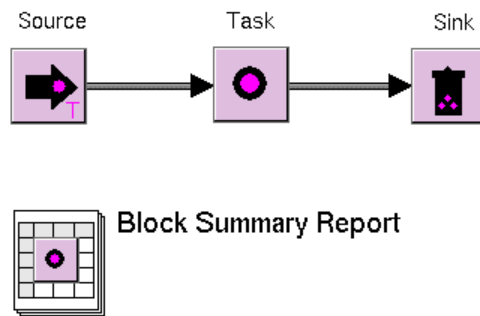
- 1 Configure ReThink for database access.
For details, see [Configuring ReThink for Database Access](#).
- 2 Configure the report object for database access.
For details, see [Configuring Report Objects for Database Access](#).

Note If you are using Microsoft Access, ensure that the database is closed before you run the simulation.

- 3 Run the simulation.

Typically, when writing report data to a database, you run the simulation for a fixed duration. For details, see “Configuring the Duration of the Simulation” on page 15.

This figure shows a model that is configured to write Block Summary Report data to a database:



Here is the resulting Microsoft Access database table that is created for three blocks after creating or updating the report:

block_output_report : Table							
Row_ID	Model_	Simulation_	Simulation_Time	Block	Total_Starts	Total_Stops	
0.0_0.0_0	0	0	11/15/01 9:56:57 AM	Sink	171	171	
0.0_0.0_1	0	0	11/15/01 9:56:57 AM	Task	171	171	
0.0_0.0_2	0	0	11/15/01 9:56:57 AM	Source	172	172	
*							

Record: 3 of 3

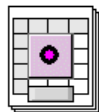
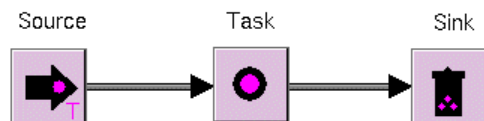
Importing Input Report Data from a Database

To import input report data from a database, you must populate the database table with data that corresponds to the report you create, then you import the data. Importing the data updates values in the model.

To import input report data from a database:

- 1 Configure ReThink for database access.
For details, see [Configuring ReThink for Database Access](#).
- 2 Configure the report object for database access.
For details, see [Configuring Report Objects for Database Access](#).
- 3 Choose Import Data from Database on the report object, or on the Database tab of the Report object, click the Import Data from Table button.

This figure shows a model that is configured to import Block Input Report data from a database:



Block Input Report

Here is the Microsoft Access database table from which the data is to be imported:

Block	Label	Maximum_Activities	Duration_Mean	Duration_STD
Source	Source	5	10	5
Task	Task	3	5	2
Sink	Sink	2		
*				

Record: 3 of 3

Using Batch Simulation

Describes how to use run multiple simulations from a script.

Introduction **445**

Using the Batch Simulation Object to Run Simulations **446**

Simulation Keywords **451**

Report Keywords **452**

Setting Attribute Values **453**



Introduction

You use the Batch Simulation object to:

- Run multiple simulations from a script.
- Change the value of any parameter in the model while the simulation is running.

For example, you can use the Batch Simulation object to optimize key parameters and metrics by running multiple simulations, using different parameter values for each simulation. You can then save the results to separate reports to analyze the results. You can also use the Batch Simulation object to determine the impact on the current model of changes in key parameters over time, such as those that determine the frequency of work objects that the model generates.

To use the Batch Simulation object, you create a script that consists of a number of keywords. The keywords identify the model to run and the various parameter values to set. You can set parameter values for any type of object in the model.

You can save the results of a simulation to an Excel report, CSV file, or database associated with an output report.

The keywords and the arguments to keywords are not case-sensitive in the script. The script ignores extra spaces and carriage returns in labels. When entering time values, you may enter *hour* or *hours*, *minute* or *minutes*, and so on.

The script executes in the order in which the keywords appear.

This topic describes how to:

- [Use the Batch Simulation object to run simulations.](#)
- [Use simulation keywords.](#)
- [Use report keywords.](#)
- [Set attribute values.](#)

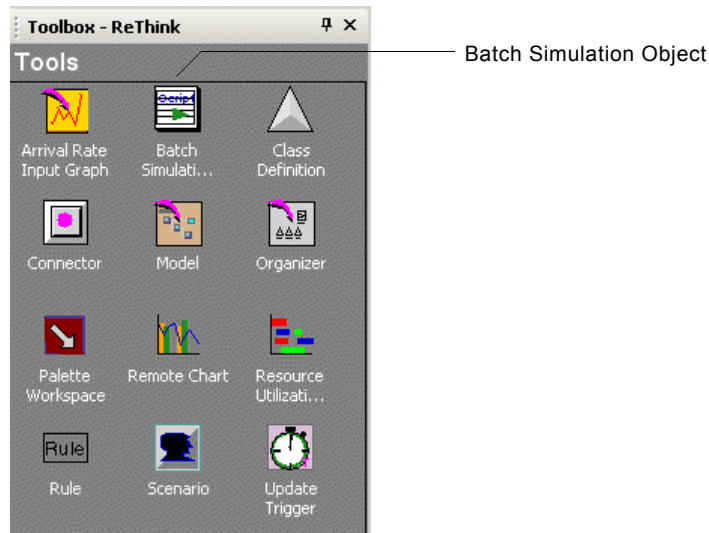
Using the Batch Simulation Object to Run Simulations

You might want to run several simulations for the same model, each with different values for the mean parameter. Alternatively, you might want to run a single simulation in which the value of the Mean parameter decreases over time, which is common in many manufacturing processes.

You identify the simulation to run by referring to a unique scenario.

To use the Batch Simulation object to run simulations:

- 1 Display the properties dialog for the Scenario and configure the Label parameter to be a unique name.
- 2 Choose Toolbox - ReThink to display the ReThink toolbox, then click the Tools palette:



- 3 Select the Batch Simulation object and place it on the model detail or organizer detail.
- 4 Display the properties dialog for the Batch Simulation object and configure the script.


To do this, enter keywords and arguments in the Script field.

For a description of the available keywords, see:

- [Simulation Keywords](#).
- [Report Keywords](#).
- [Setting Attribute Values](#).


- 5 Add comments to the script, as needed, using the following syntax:

```
/* this is a comment */
```


- 6 To verify that you have entered the keywords in the script correctly, click the Check Script button: 


ReThink displays error messages and warnings in red and yellow, respectively, in the Log Book area.


For example, the Batch Simulation object generates an error if you have not named the scenario, if an object does not exist, or if you have entered a keyword that does not exist.

- 7 Click the Start button to start running the script: 

ReThink displays messages in green in the Log Book each time a keyword completes its execution, so you can follow the progress of the simulation.

- 8 To pause the simulation, click the Pause button: 

- 9 To resume the simulation, click the Resume button: 

- 10 To stop the simulation, click the Stop button: 

ReThink completes the execution of the currently active keyword before pausing or stopping the simulation. Normally, you let the script run by itself until it finishes.

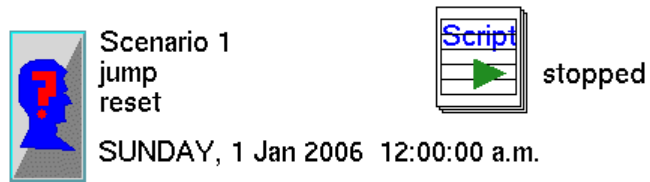
When the simulation finishes, it is as if you had run the simulation manually; metrics compute normally and reports update at the specified time intervals. If the script sets parameter values, the values update at the specified time and the simulation continues.

Typically, if you are running multiple simulations, you save report data at the end of one simulation, before starting the next one.

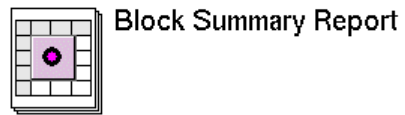
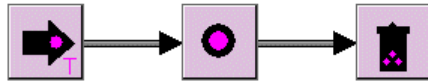
For example, this script runs the simulation associated with Scenario-1 for 91 days, saves a report, then sets the Mean parameters to different values, runs the simulation again, then saves the report to a new file name. Notice that the keyword that sets the Mean parameter comes before the *start-simulation* keyword.

```
select-scenario(scenario 1)
set-attribute-value(Create Order,duration-subtable,mean,1 hour,0)
start-simulation(9 days)
wait-simulation
set-file-reporting(block summary report,true,Block Summary Report 1.csv)
save-report(block summary)
set-attribute-value(Create Order,duration-subtable,mean,2 hours,0)
start-simulation(9 days)
wait-simulation
set-file-reporting(block summary report,true,Block Summary Report 2.csv)
save-report(role-output)
```

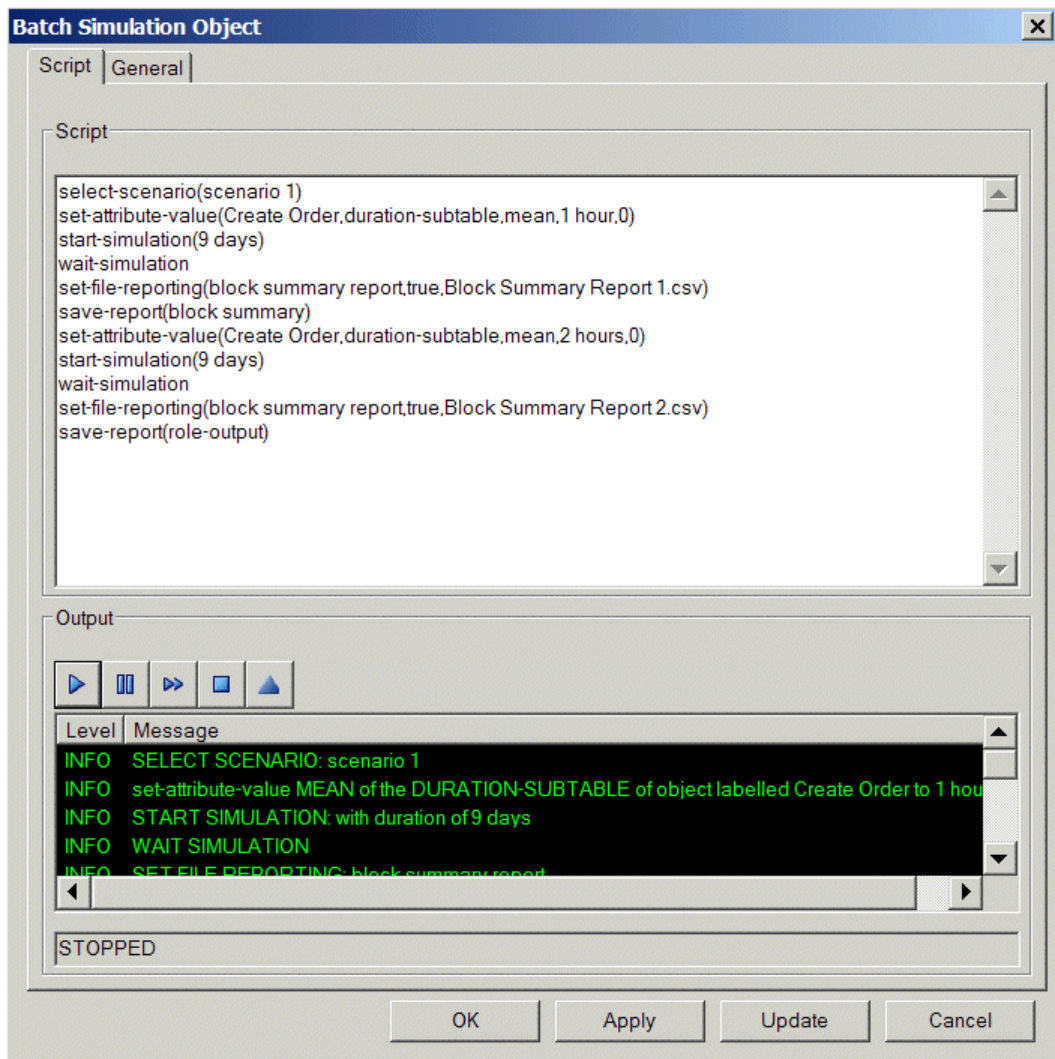
Here is a model that uses a Batch Simulation object to run multiple simulations:



Create Order



Here is the Batch Simulation object dialog that results while running the simulation:



Simulation Keywords

You use the Batch Simulation object to run a simulation for one or more scenarios and for a specified duration for each. To run multiple simulations, you must wait until the first simulation is finished running before starting another.

SELECT-SCENARIO (<scenario-label>)

Determines the scenario to use for all operations in the simulation, until a new scenario is selected. You must configure the Label parameter of the scenario to be a unique name.

Example:

SELECT-SCENARIO (Scenario 1)

START-SIMULATION (<duration>)

Starts the simulation and determines the duration of the simulation.

Note Always set parameter values *before* you start the simulation, even if you schedule the parameter values to be set during the simulation.

Example:

START-SIMULATION (52 weeks)

WAIT-SIMULATION

Waits until the scenario is finished running. Use this keyword to run multiple simulations for the same model, using different parameter values. This keyword has no arguments.

CREATE-NEW-SEED (<truth-value>)

Whether to create a new value for the simulation.

SET-NEW-SEED (<seed>)

Sets a new seed value for the simulation.

Report Keywords

The following keywords control various reporting functions that can occur during the simulation. All keywords specify the *<report-title>* argument to identify the report object. When you save a report, it writes the report data to an Excel report, CSV file, or database, depending on how the report is configured.

SET-FILE-REPORTING(*<report-title>*,*<excel-report-enabled>*,
<excel-report-filename>)

Enables the creation of CSV files for reports and specifies the *.csv* file name to create, as if you had clicked the Excel Report Enabled option and specified the file name in the report dialog. This keyword sets the file name to save when you use the *SAVE-REPORT* keyword. Typically, you save report data to separate CSV files; otherwise, saving report data overwrites the file data.

Example:

```
SET-FILE-REPORTING(block summary report,true,  
block summary report.csv)
```

UPDATE-REPORT(*<report-title>*, *<time>*)

Updates a report at the specified simulation time interval, as if you had manually updated the report while the simulation was running.

Example:

```
UPDATE-REPORT(block summary report, 1 hour)
```

SAVE-REPORT(*<report-title>*)

Saves report data to the *.csv* file associated with the report that you set by using the *SET-FILE-REPORTING* keyword. To save reports to different filenames, use the *SET-FILE-REPORTING* keyword each time you use *SAVE-REPORT*.

If an Excel client is currently connected or if a report view is currently visible in the client, this keyword updates the report in Excel or the client.

If the report object is configured to output data to a database, this keyword saves the report data to the specified database. The database table includes columns for the *Model_Version* and *Simulation_Version* to uniquely identify data for each simulation run of each model.

Example:

```
SAVE-REPORT(block summary report)
```

LOAD-REPORT(*<report-title>*)

Loads the *.csv* file associated with an input report, as if you had clicked the Import Data from File button in the report dialog.

Example:

LOAD-REPORT(block input report)

LOAD-REPORT-FROM-DATABASE (*<report-title>*)

Loads the data from the database associated with an input report, as if you had clicked the Import Data from Table button.

Example:

LOAD-REPORT-FROM-DATABASE(block input report)

Setting Attribute Values

To set attribute values for any object, use the following keyword:

SET-ATTRIBUTE-VALUE (*<object-label>*, *<subtable-name>*,
<attribute-name>, *<value>*, *<simulation-time>*)

<object-label> is the text of the Label attribute.

<subtable-name> is the name of the subtable in which the attribute appears, if any. The options are: *duration-subtable*, *cost-subtable*, and *animation-subtable*. If the attribute appears on any tab other than the Duration tab, the Cost tab, or the Animation tab, the *<subtable>* has no value.

<attribute-name> is a symbol that names the attribute whose value you want to set. For example, *mean*, *standard-deviation*, *maximum-activities*, and so on.

<value> is the value of the attribute to set, for example, *1 hour* or *6*.

<simulation-time> is the simulation time at which to set the attribute value, for example, *0* or *4 weeks*.

This example shows how to set the value of the Mean attribute of the Create Order block to *1 hour* at the start of the simulation, then change it to *50 minutes* 1 week into the simulation:

set-attribute-value(Create Order,duration-subtable,mean,1 hour,0)

*set-attribute-value(Create Order,duration-subtable,mean,50 minutes,
1 week)*

Using ReThink in Online Mode

Describes how to use ReThink in online mode.

Introduction	455
Using ReThink in Online Mode	456
How Online Mode Works	457
Using Interface Pools	458
Using Online Blocks	463
Sending Email	468
Using JMS Messaging	470



Introduction

ReThink supports online transaction processing, where ReThink works as a workflow engine, managing decisions and coordinating activities.

Using ReThink in online mode allows you to:

- Define hierarchical models and metrics to organize, manage, and display information and best practices.
- Use simulation to do what-if analysis.
- Switch to operational mode to manage your processes.

All ReThink features such as instruments, resources, and reports are functional in online mode.

To support online mode, ReThink provides:

- An online mode for scenarios, where the simulation clock is synchronized with the current actual time.
- A framework for performing parallel and distributed processing that the ReThink model coordinates and manages.
- A number of out-of-the-box blocks for interaction with databases, JMail, and JMS Message servers.

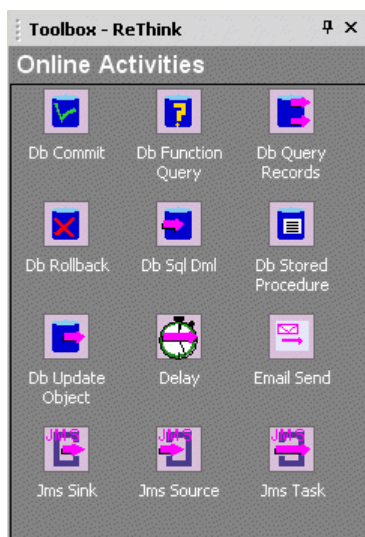
Using ReThink in Online Mode

When using ReThink in online mode, keep the following considerations in mind:

- Except for the Remote Process Source block and the Delay block, all block durations are disregarded, and the Work Time and Elapsed Time of all activities are set to zero.
- For the Remote Process Source block and the Delay block, the duration specified in the block represents the delay in actual time, which the block uses to schedule events.
- In jump mode, the blocks and durations work as before, including the Delay block, where the delay is in simulation time, not in actual time.

Note We highly recommend that you understand how ReThink processes events before using ReThink in online mode. For a detailed description of internal block processing, see the *Customizing ReThink User's Guide*.

The ReThink toolbox includes the Online Activities palette:



To build a new online model:

- Create a new project with ReThink as the selected library.

For details, see [Working with Projects](#).

To switch to online mode:

- On the General tab of the Scenario properties dialog, choose **Online** as the Mode or choose Simulation > Online Mode.

To view examples of online mode:

- Load *rethink-40-online-examples.kb* from the *rethink/examples* directory.

How Online Mode Works

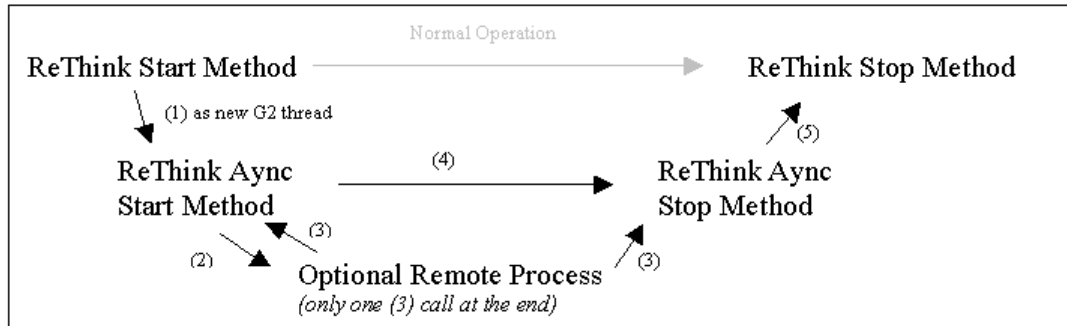
In online mode, ReThink advances the simulation clock as the wall clock changes, in units of second. Events are scheduled for the current second, including new events that have been scheduled within the current second due to ReThink internals or received from external sources.

When using the basic blocks found on the Basic Activities tab of the ReThink toolbox, ReThink processes work objects synchronously within each thread, which means the phase completes before ReThink processes another work object in the same or in a different block. When using the blocks found on the Online Blocks tab of the ReThink palette, ReThink processes work objects in parallel. It does this by offloading the processing to another G2 thread, to a thread in a remote program, or to a bridge processes, such as a database bridge; it does not wait until the thread completes before processing other work objects.

ReThink contains two categories of online blocks:

- One set receives events from external programs, and requests external programs or G2 threads to process work objects or to send work objects to another program or ReThink model.
- One set executes statements in the external system, such as SQL statements that delete rows, insert rows, update rows, update attributes of work objects, query database records, and execute stored procedures.

This figure shows a diagram of normal block processing compared with online block processing:



- 1 For each work object that an online block processes, the block spawns an asynchronous start method as a new thread during the start phase of the block. ReThink does not call the stop method of the block as it would for standard blocks, but instead continues processing other events in the model.
- 2 The asynchronous start method can perform processing in the same G2 process in a different thread, or it can offload the processing to another G2, ReThink, C, VisualBasic, or Java program.
- 3 If the processing is performed outside of the local process, once processing is complete, the remote process calls the asynchronous stop method.
- 4 If the processing is performed within the local process, once processing is complete, the local process calls the asynchronous stop method directly.
- 5 When the asynchronous stop method executes, ReThink then schedules the stop phase of the block, which dispatches the work object to downstream blocks in the model.

To optimize online processing, you can extend and customize the online blocks to call remote programs, as needed. To customize these blocks, instead of customizing the `bpr-stop-method`, you customize the method named `bpr-async-start-method`. For examples, see the blocks in *methods-online.kb*.

Using Interface Pools

One issue when using online blocks for workflow operations, and especially when interacting with databases, is the IO throughput. For example, when many work objects are being processed, querying a database or calling a remote program to perform processing-intensive tasks may not significantly improve performance or scale well, even when processing work objects from different parts of your ReThink model. This is because the remote program may be busy processing one work object at the time, or the same IO channel to the database may be used in different part of the model, which blocks other queries or updates when one is already processing.

For scalability, ReThink supports pools of interfaces to communicate with databases or other remote programs. The pool uses parallel bridges and communication channels to interact with external databases and improve performance. This means that ReThink can perform multiple database queries or updates in parallel and use multiple instances of remote programs performing processing-intensive tasks. For each request, ReThink selects an available or the least-used network object from the pool at run time.

ReThink provides several types of interface pools, which are configured to create network objects that connect to a database, JMail bridge, JMS Message server, or another ReThink process.

To configure multiple communication channels, you configure the initial interface count to be the desired number of connections. When the model resets, ReThink automatically creates and configures the network objects to connect to the remote process or bridge. Alternatively, you can configure the network communication pool manually, and when the model resets, ReThink launches the remote programs or bridges.

If you use the default procedures to launch the remote process or bridge, the command line to launch it must accept one argument, which is the TCP/IP port name to use for listening for connections.

To write your own procedures to launch or kill processes, the signatures of the procedures are:

```

bpr-launch-process
  (cmd: text, host: text, port: integer,
   network-pool: class bpr-network-connection-pool)
  -> return: float

bpr-kill-process
  (io: class network-interface,
   network-pool: class bpr-network-connection-pool)

```

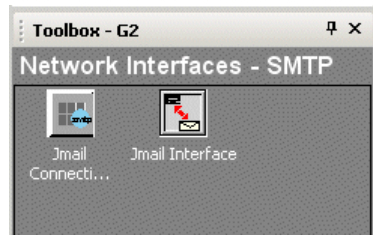
The launch process returns the Process ID of the launched process or -1 if an error occurred.

To configure an interface pool:

- 1 Choose View > Toolbox - G2 and choose the desired Network Interfaces palette.

Alternatively, choose Project > System Settings > Interface Pools, choose the desired type of interface, and choose Manage to create a new interface pool.

For example, here is the Network Interfaces - SMTP palette:



- 2 Create a network interface pool from the palette, display its properties dialog, the interface pool and configure these attribute on the General tab:

Attribute	Description
Label	Any label for the object.
Comments	Any comment for the object.
User Name	User name to connect to interfaces.
Password	Password used to connect to interfaces.
Initial Network Interface Count	The default number of interfaces created in the pool at reset time.
Network Connection Timeout	The timeout of the network connection.
Enable Initialization During Reset	Enables and disables the initialization of the pool at reset time of the model.

- 3 Click the Network Interface tab and configure these attributes:

Attribute	Description
Interface Default Host	The host where the interface to connect to is running.
Interface Base Port Number	The TCP/IP port to which the remote interfaces connect. For every additional interface automatically added at reset time, this port number is incremented by 1, except for G2-to-G2 interfaces.
Network Interface Timeout	The timeout for the network interface.

Attribute	Description
Network Interface Initialization String	The initialization string for the network interface.
Remote Process Launch Arguments	Arguments to the Bridge Process Launch Command.
Bridge Process Launch Command	The command line to launch the bridge or process. If you use <code>bpr-launch-process</code> as the procedure to launch the process, ReThink expects the command line to take one argument, which is the TCP/IP port number the bridge should use to listen for G2 connections. Standard G2 bridges such as database bridges offered by Gensym comply with this requirement.
Bridge Process Launch Procedure	The name of a procedure to launch a bridge or process. This procedure can be a executable (<code>.exe</code>) or a batch file. The procedure should return the process ID of the process it launched or <code>-1</code> if it failed. The default procedure is <code>gdsm-launch-bridge-process</code> .
Bridge Process Kill Procedure	The name of a procedure to kill the bridge to which an interface is connected. The default procedure is <code>gdsm-kill-bridge-process</code> .
Auto Connect to Bridge	Whether to automatically connect to the bridge when required.
Shutdown Bridge on Disconnect	Whether to automatically shutdown the bridge when disconnecting.
Launch Bridge Upon Connect	Whether to automatically launch the bridge when attempting a connection.

- 4 For Database Interface Pools, click the Database tab and configure these attributes:

Attribute	Description
Database Connect String	The database name to connect to or the ODBC DSN to use to connect to the database.
Maximum Number of Cursors	The maximum number of cursors to manage in each database bridge. The default value is 100.
Bind Variable Prefix	The prefix used by the database to tag arguments as being bind variables. The default value is colon (:).

For details, see the *G2 Database Bridge User's Guide*.

- 5 For JMail Interface Pools, click the JMail tab and configure these attributes:

Attribute	Description
Incoming Host	The name of the host computer used for incoming email.
Incoming Port	The port number of the host used for incoming email.
Incoming Protocol	The protocol used for incoming email.
Incoming Folder	The name of the folder for incoming email.
Delete Messages on Server	Whether to delete messages on the server when sent.
Outgoing Host	The name of the host computer used for outgoing email.
Outgoing Port	The port number of the outgoing host.
Outgoing From	The email address to use as the From address when the email message is sent.

For details, see the *G2 JMail Bridge User's Guide*.

- 6 For JMS Interface Pools, click the JMS tab and configure the attribute.

For details, see the *G2 JMSLink User's Guide*.

Using Online Blocks

This section describes the set of blocks that are based on asynchronous operation and used to interact with remote processes and databases. To communicate with the remote process, these blocks rely on network interface pools.

Remember, none of the online blocks except the Delay block uses durations. Instead, the delay between the start and the stop phases depend on the remote process.

The two types of models that you typically build, using the online blocks are:

- Distributed workflow applications.
- Database interaction applications.

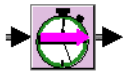
Handling Errors

If an error occurs during processing within an online block, the error context is stored in the work object; therefore, your model can reason and take actions, based on the error condition. In addition, you can specify an error path on these blocks, where work objects are sent to this path if an error occurs.

Errors can be due to communication errors, errors in the remote program, or timeout conditions. These attributes on work objects identify error conditions:

Attribute	Description
Error Status	The default value is <code>no-error</code> . The status could also contain an error symbol such as <code>timeout-error</code> .
Error Code	The error code, such as the error code from the database bridge.
Error	The error message, such as the error message returned from the database.

Introduce Delays into the Process



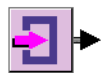
Use this block to introduce delays in your model. You can use the Delay block in simulation or online mode.

Modeling Distributed Workflow Applications

To model distributed workflow applications, you use the:

- [Remote Process Source Block](#)
- [Remote Process Task Block](#)
- [Remote Process Sink Block](#)

Remote Process Source Block



The Remote Process Source block receives work objects from a remote process and generates a new work object. The block should have a single output path. The block is configured to receive work objects from the Remote Process Sink block.

The external process generates the event that determines when the object is transferred. It transfers one work object per event, along with sender identification information, such as the host, port, scenario label, and block label. The event itself is in the form of an RPC, which passes in the `bpr-object`. Once the block receives the object, the block processes the object and inserts it into the model.

Remote processes call the procedure `bpr_receive_work_object` or `bpr_receive_serialized_work_object` to post new objects to the model. The remote process identifies the source block into which it inserts the object, based on the Remote Source Key of the source block, which contains any symbol.

By default, when the Remote Process Source block receives a work object from the remote program, it generates a work object. You can also configure the block to execute in batch mode, in which case it generates the work object when the block executes, based on its duration.

The remote system transfers one object per event. The object contains sender identification information such as the host, port, scenario label, and block label. The event itself is in the form of a remote procedure call (RPC) that passes in an instance of `bpr-object`. Once the block receives the object, it generates a work object.

The class of the object that gets passed in is mapped to a corresponding class in ReThink. The local class must be a subclass of `bpr-object`. This procedure passes in the class name as a symbol, where the object's attributes and values are passed in as sequences. The procedure to call from the external process has this signature:

```
bpr_receive_work_object(from-host: text, from-port: text, from-scenario: text,
from-block: text, object: class bpr-object) = (symbol, integer, text)
```

The procedure returns the symbol **success** if it succeeds in posting the object to the block. Otherwise, it returns a symbol specifying the error class, an error code, and an error text.

Alternatively, and especially important when integrating with environments that cannot rely on object passing such VisualBasic via the ActiveX control, the external process can call a procedure with this signature. This procedure passes in the class name as a symbol, where the object's attributes and values are passed in as sequences.

```
bpr_receive_serialized_work_object(from-host: text, from-port: text,
    from-scenario: text, from-block: text, dest-name: symbol,
    class-name: symbol, attribute-names: sequence,
    attribute-values: sequence) = (symbol, integer, text)
```

Remote Process Task Block



The Remote Process Task block offloads processing of activities to a separate G2 thread, in a remote G2 or ReThink process, VB program, C/C++ program, or Java program. In the remote process procedure, you either specify the name of a local procedure or remote procedure. The task calls this procedure either by:

- Passing the work object by reference, when Use Object Passing is enabled.
- Passing it via serialization, whereby attribute/value pairs are passed in and returned by the remote procedure as sequences.

The remote procedure should specify a timeout so that if the procedure does not complete within the timeout period, the call to the remote procedure is aborted.

Remote Process Sink Block



The Remote Process Sink block sends work objects to a remote procedure. The model performs no further processing of the work object.

If the destination attribute of the work object has the default value of none, it receives a new value from the Remote Destination attribute of the block.

You can also configure the Remote Process Interface Label to be a **bpr-network-connection-pool** that contains 1 - n connection interfaces to be used. The block uses the next least-used available connection. You use pools of multiple connections that can be used in parallel to improve performance when the block needs to make multiple calls to remote systems. If the block does not configure a remote process interface, it uses a default interface to make a local connection.

If the block has multiple inputs, it sends work objects from each input path. By default, it sends the objects as they arrive. You can also enable the Needs All Inputs option to send the objects when the block has an object waiting on each input path.

The remote process must define the following procedure:

```
bpr_receive_work_object(from-host: text, from-port: text, from-scenario: text,  
from-block: text, object: class bpr-object) = (symbol, integer, text)
```

The block sends the work object with all of its non-system attributes, except the `animation-subtable`. The procedure should return the symbol `success` if the operation is successful, or an integer representing the error code and a text explaining the error, otherwise.

Interacting with Databases

The following set of blocks provide out-of-the-box support for interacting with databases. You use them to insert or query an external database. To use these blocks, you must be familiar with writing SQL statements.

The timeout of these blocks limits the time that a query, insert, or update should take. If a timeout error occurs, the Error Status of the work object is set to `timeout-error` to enable the model to detect abnormal database connectivity conditions.

If a timeout error occurs, the query can be automatically aborted or not. If it is not aborted, when the query completes, the results of the query are discarded because the work object will already have been processed and sent to other blocks.

All of the online database blocks, with the exception of the Database Commit and Database Rollback blocks, take an SQL statement. The SQL query may refer to attributes of the work object by specifying the attribute name preceded with a `$`. For example, to refer to the Total Starts of a work object, use `$TOTAL-STARTS` in the query. At run time, this variable is replaced by the value of the `total-starts` attribute of the work object on the input path of the block.

Several SQL statements also support bind variables, which generally increases performance. If the statement uses bind variables, the bind variable names should correspond to the work object attribute names, without the bind variable prefix. For example, suppose you define the following SQL statement to insert a work object into a database:

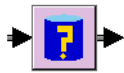
```
insert into emp (empno, ename) values (:nr,:name)
```

In this query statement, `nr` and `name` need to be defined as attributes of your work object, and the character `:"` needs to be defined as the bind variable prefix in the Network Interface Pool that contains the connection to your database. At run time, the code collects the list of bind variables, fetches the corresponding attributes from the work object, and executes the statement. The code also maps any hyphen or space character specified in a bind variable to an underscore before sending the SQL statement to the database. This means you can specify attribute names that include hyphens without compromising requirements for the database.

To interact with databases, you use the:

- [DB Function Query Block](#)
- [Database Stored Procedure Block](#)
- [Database Update Object Block](#)
- [Database SQL DML Block](#)
- [Database Query Block](#)
- [Database Commit Block](#)
- [Database Rollback Block](#)

DB Function Query Block



The DB Function Query block performs any SQL function query, such as a count, and stores the result into an attribute of the work object.

You can use one of two mechanisms to replace values in the SQL statement with values from the input work object:

- "[the <attribute> of InputObject]"

You use this expression when you need to extract attributes from the duration or cost subtable.

- \$<attribute>

You use this expression to create easier-to-read SQL statements when referring to attributes that do not appear in subtables.

Bind variables in SQL statements can only use the syntax :<attribute> where : is defined as the bind variable prefix. At run time, bind variables are created based on the attribute name by replacing hyphens with underscores and replaced with the actual value of the attributes as rows are inserted.

Database Stored Procedure Block



The Database Stored Procedure block executes stored procedures in the database. You provide an SQL statement that is the name of a stored procedure to call, including the arguments to pass. Similar to the SQL Function block, the statement can refer to \$<attribute-name> or [the <attribute> of InputObject] to pass arguments of the work object.

Database Update Object Block



The Database Update Object block uses an SQL statement to update attributes of the work object with values from the database, using a *SELECT* statement. Note that the *SELECT* statement should select a single row only. The block uses the values of the selected row to update the attributes of the work object.

Database SQL DML Block



The Database SQL DML block uses an SQL statement to delete, insert, or update values in the database, using a *DELETE*, *INSERT*, or *UPDATE* statement. You can configure the block to automatically commit the update or not. If you do not automatically commit the statement, your model needs to include a Database Commit and/or Database Rollback block. Once the commit or rollback block is executed, the SQL statement is committed or rolled back.

Database Query Block



The Database Query block uses an SQL statement to query rows from the database, using a *SELECT* statement. It queries the database for multiple rows. For each row, the block creates an object of the class specified in the Object Type of the block and updates the object's attributes with values from the row. The object is then inserted into the attribute of the work object specified by the Container List Attribute of the block.

Database Commit Block



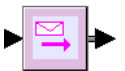
The Database Commit block commits a series of SQL statements that have been posted and releases the database interface connection. It commits all transactions for all interfaces in a network interface pool.

Database Rollback Block



The Database Rollback block rolls back a series of SQL statements that have been posted and releases the database interface connection. It rolls back all transactions for all interfaces in a network interface pool.

Sending Email



You use the Email Send block and accompanying JMail Interface Pool for sending email as part of online transaction processing. For example, you might create a distributed workflow application or database interaction application that sends email when a certain event occurs, thereby providing real-time communication as part of your online application.

To use the Email Send block, you must first create and configure a JMail Connection Pool, then you must start the G2 JMail Bridge. The bridge provides communication between your ReThink application and the email server specified in your JMail Connection Pool. When a work object arrives at the Email Send block in your online application, ReThink sends the email message via the bridge.

The Email Send block supports sending email only; it does not support receiving email.

To send email as part of online transaction processing:

- 1 Create a JMail Connection Pool and configure its attributes for connecting to an email server.

For details, see [Using Interface Pools](#).

- 2 Use the blocks on the Online palette of the ReThink toolbox to create an online application that interacts with databases or other external systems.
- 3 Configure the application to use the Email Send block by connecting it to a block in the model where you want email to be sent.
- 4 You must configure the application such that a work object arrives at the block, which triggers the email to be sent.
- 5 Configure the standard attributes on the General tab of the Email Send block.
- 6 Click the Block tab and configure the following attributes:

Attribute	Description
Email To	A text string that includes a single email address. To send email to multiple recipients, create an alias on your mail server. The text string can include an expression that refers to an attribute of object that is the email address, as described in Email Subject.
Email CC	A text string that includes a single email address. To send email to multiple recipients, create an alias on your mail server. The text string can include an expression that refers to an attribute of object that is the email address, as described in Email Subject.
Email Subject	A text string that is the subject of the email. The text string can include expressions to be evaluated at runtime by delimiting the expression in square brackets ([]) or by using "\$<attribute-name>" when referring to attributes in subtables. The expression can refer to the input work object of the block by using the variable named InputObject.

Attribute	Description
Email Message	A text string that is the email message. The text string can include an expression, as described in Email Subject. For an example, see "Creating an SQL Query for Accessing the Data" in Chapter 8, "Accessing External Databases" in the <i>ReThink User's Guide</i> .
Email Connection Pool	The name of an existing JMail Connection Pool. The dropdown list shows all instances of this type of pools associated with the scenario.

When you are ready to run your application in online mode, you must start the G2 JMail Bridge.

- To start the G2 JMail Bridge, choose Start > Programs > Gensym G2 2011 > Bridges > G2 JMail Bridge.

This command launches a bridge process that establishes communication between your ReThink application and the email server you specified in the JMail Connection Pool.

- Now, run your application in online mode.

When a work object arrives at the Email Send block, ReThink sends an email message to the specified recipients via the bridge.

Using JMS Messaging



JMS, an acronym for Java Message Service, is an industry-standard API for Java-based clients to interact with native message-oriented middleware (MOM) systems, which are designed especially for enterprise messaging applications.



Some JMS-compliant MOM systems include IBM's WebSphereMQ, Sun JMQ, FioranoMQ, BEA Weblogic, and JBoss 3.2.6 on Linux and HP, Open JMS 0.7.6.1 on Windows, Linux, and HP, Java 2 Platform, Enterprise Edition (J2EE) 1.3.1 on Linux.



JMS supports two types of messaging models:

- Point-to-point (PTP) is a one-to-one message delivery system that allows only two JMS clients to send and receive messages, both synchronously and asynchronously, via a virtual channel known as a queue.
- Publish-and-subscribe (pub/sub) is a one-to-many message delivery system in which one JMS client is a message publisher that can send a message to many JMS clients as message subscribers through a virtual channel known as a topic.

You use the JMS Source, JMS Task, and JMS Sink blocks and accompanying JMS Interface Pool to send and receive messages, using the JMS Message Service. For example, you might create a distributed workflow application or database interaction application that sends messages when a certain event occurs, thereby providing real-time communication as part of your online application.

To use the JMS blocks, you must first create and configure a JMS Connection Pool, then you must start the G2 JMS Bridge. The bridge provides communication between your ReThink application and the email server specified in your JMS Connection Pool. When a work object arrives at the JMS blocks in your online application, ReThink communicates with the JMS message server via the bridge.

For details about configuring the JMS Connection Pool and JMS blocks, see the *G2 JMSLink User's Guide*.

ReThink Reference

Chapter 12: Blocks Reference

Provides a description and example of each ReThink block.

Chapter 13: Instruments Reference

Provides a description and example of each ReThink feed and probe.

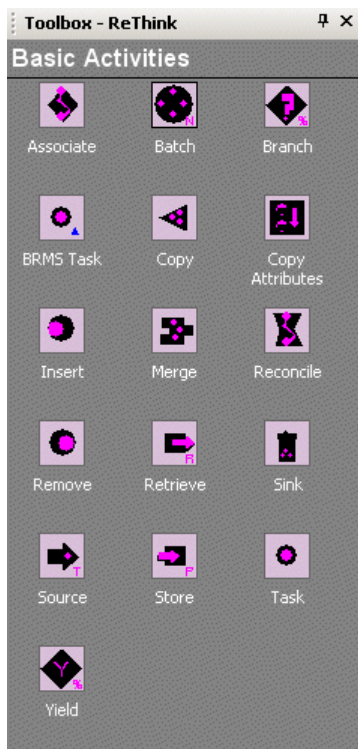
Blocks Reference

Provides a description and example of each ReThink block.

Introduction	476
Common Attributes of Blocks	477
Common Menu Choices for Blocks	486
Common Attributes of Paths	487
Common Menu Choices for Paths	491
Associate	492
Batch	498
Branch	511
BRMS Task	535
Copy	537
Copy Attributes	542
Insert	548
Merge	554
Reconcile	556
Remove	562
Retrieve	570
Sink	585
Source	587
Store	600
Task	608

Introduction

This chapter describes each block in the Basic Activities palette of the ReThink toolbox:



The chapter begins with sections describing the attributes and menu choices that are common for all blocks and their paths. It then describes each block in detail, including:

- A general description of the block.
- Specific uses of the block.
- An example.
- Specific attributes and menu choices.
- Customization attributes, where relevant.

For general information about how to use blocks to create a model, see [Using Blocks](#).

Common Attributes of Blocks

You access the common attributes of blocks through the properties dialog. The properties dialog has tabs, which allow you to edit and view general block attributes, block-specific attributes, duration attributes, cost attributes, and animation attributes.

This table describes the various tabs that are available in the properties dialog for blocks:

Tab	Description
General	Parameters that are common to all blocks, such as the Label and Comments. Metrics related to the number of activities the block processes.
Block	Parameters and metrics that are specific to the particular block, for example, the mode in which the block operates. For a description of the attributes on this tab, see the individual block listings.
Database	Parameters for accessing records in a database when the block is configured for database access. This tab is only available for certain blocks in database mode. For details, see Accessing External Databases .
Duration	Parameters that determine how the block computes the duration of each activity. Metrics related to the timing of the block.
Cost	Parameters for configuring fixed and variable costs for the block, and the total cost metric for the block.
Animation	Parameters for specifying animation colors for the block.

Following are examples for a Task block of the tab pages of the properties dialog that are common to all blocks: General, Duration, Cost, and Animation. Following each dialog is a description of the attributes that appear in Modeler mode. Each attribute indicates whether it is a parameter (P) or metric (M).

For information on the attributes that are available in Developer mode, see the *Customizing ReThink User's Guide*.

General Tab

The screenshot shows a dialog box titled "Task: Load Trucks" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "General" tab is selected. The fields and their values are as follows:

- Block Label: Load Trucks
- Comments: Requires a loader and a truck resource
- Maximum Activities: 5
- Uri: file:c:/docs/load.html
- Total Starts: 17
- Total Stops: 16
- Current Activities: 1
- Error: (empty)

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Attribute	P/M	Description
Block Label	P	A label for the block, which appears next to the block in quotes. For information on hiding the label, see Using Attribute Displays .
Comments	P	An area for entering a description of the purpose of the block and whatever other information you want to keep.

Attribute	P/M	Description
Maximum Activities	P	The maximum number of activities that the block can process concurrently. Specify a positive integer or leave empty. The default value is empty, which means the block can process any number of activities concurrently, depending on other constraints. See Limiting the Number of Concurrent Activities .
URL	P	A URL to an HTML file, either on the World Wide Web or on the file system, or a URL to an RTF file on the file system, which explains the block. When this attribute is configured, choosing Show URL or clicking the block displays the file in its own window.
Total Starts	M	The total number of activities that the block has started processing since the start of the simulation.
Total Stops	M	The total number of activities that the block has finished processing since the start of the simulation.
Current Activities	M	The total number of activities that the block is currently processing. The sum of the Current Activities and Total Stops equals the Total Starts for the block.
Error	M	A description of any error for the block. See Debugging Blocks .

For information on the metrics related to activities, see [Determining the Current Activities](#).

Duration Tab

Task: Load Trucks [X]

General | Block | **Duration** | Cost | Animation

Duration

Distribution Mode: Random Triangular

Min: 000 000 05:00:00

Max: 000 000 07:00:00

Mode: 000 000 06:00:00

Miscellaneous

Time Per Unit Attribute: []

Total Work Time: 001.000.03:59:34

Total Elapsed Time: 000.003.00:51:45

Creation Time: 000.000.00:00:00

Average In Process: 000.000.00:00:02

OK Apply Update Cancel

Attribute	P/M	Description
Distribution Mode	P	<p>How the block computes duration. The default for all blocks (except the Source block) is Random Normal, which computes a random duration, based on a normal distribution, where you configure a Mean and Standard Deviation.</p> <p>The default value for a Source block is Random Exponential, which computes a random duration, based on an exponential distribution, where you configure the Mean.</p> <p>You can also choose from the following random distributions: Fixed, Uniform, Triangular, Erlang, Weibull, Lognormal, Gamma, and Beta. Choosing one of these distributions displays additional parameters for specifying the particular function.</p> <p>For details on configuring these distributions, see Specifying a Fixed Duration and Specifying a Random Duration.</p> <hr/> <p>You can also configure the Distribution Mode to be:</p> <ul style="list-style-type: none"> • Work Object Duration (See Specifying Duration Based on an Attribute of a Work Object) • Report Indexed Lookup (See Specifying Duration Based on an Indexed Report Lookup) • Report Lookup (See Specifying Duration Based on an Attribute Report Lookup) • Duration File (See Specifying Duration from a File) • Arrival Rate Input Graph (See Using a Graph to Specify Duration.) • Custom (See Specifying a Custom Duration)

Attribute	P/M	Description
Time per Unit Attribute	P	<p>The name of an attribute of a work object that specifies the number of units of work. The block multiplies the value of the attribute of the work object by the duration to compute the duration of the activity.</p> <p>You can use dot notation to refer to the attribute of a subobject, for example, <code>my-subtable.my-attr</code>.</p> <p>See Computing Duration for Multiple Units of Work.</p>
Total Work Time	M	<p>The sum of all the work times for each activity that the block performs as of the current time. See Understanding Total Work Time and Total Elapsed Time.</p>
Total Elapsed Time	M	<p>The amount of time that has elapsed since the simulation began or since you created the block in a running simulation. See Understanding Total Work Time and Total Elapsed Time.</p>
Creation Time	M	<p>The simulation time when the block was created. See Understanding Total Work Time and Total Elapsed Time.</p>
Average in Process	M	<p>The average number of concurrent activities for the block, which is the Total Work Time divided by the Total Elapsed Time. See How the Block Uses Total Work Time and Total Elapsed Time.</p>

For general information on block duration, see [Working with the Duration of Blocks](#).

Cost Tab

The screenshot shows a dialog box titled "Task: Load Trucks" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Cost" tab is selected. Inside the dialog, there are four rows of input fields:

- Cost Per Use: 5
- Cost Per Time Unit: 10
- Time Unit: 000, 000, 01:00:00
- Total Cost: 1276.146

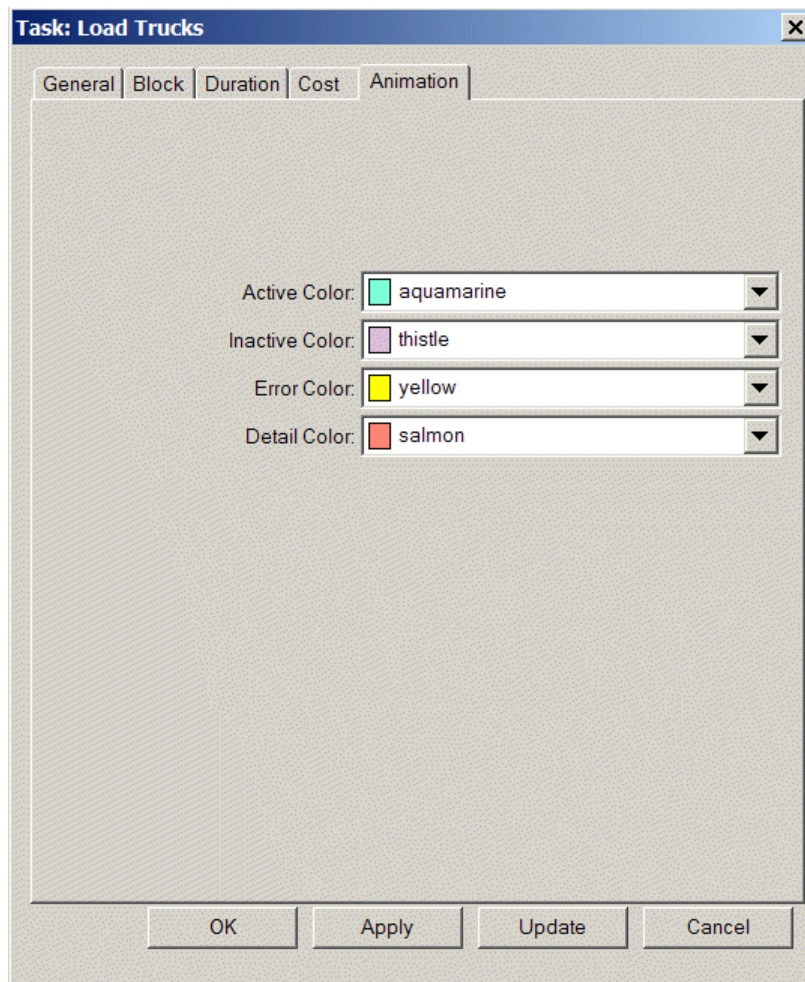
At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Attribute	P/M	Description
Cost Per Use	P	The fixed cost of each activity the block processes.
Cost Per Time Unit	P	The variable cost of each activity, which the block computes based on the specified Time Unit and the duration of the block.

Attribute	P/M	Description
Time Unit	P	The time unit the block uses to interpret the Cost Per Time Unit attribute. By default, the Time Unit is 1 hour, which means variable costs is computed on an hourly basis.
Total Cost	M	The sum of the costs of each individual activity performed by the block, which includes fixed and variable costs assigned to the block itself, as well as fixed and variable costs assigned to the resources allocated to each activity.

For information on configuring and computing costs for a block, see [Working with Block Costs](#).

Animation Tab



Attribute	P/M	Description
Active Color	P	The color of the block when it is currently processing work objects.
Inactive Color	P	The color of the block when it is idle.
Error Color	P	The color of the block when it is in an error state.
Detail Color	P	The color of a Task block when it has detail. This attribute is only available for the Task block.

Common Menu Choices for Blocks

Menu Choices	Description
Delete	Permanently deletes the block.
Transfer Clone	Cuts and copies (transfers) the block from one workspace to another, or copies (clones) the block to a workspace. ReThink copies all of the block's configured attributes and detail, if one exists.
Order	Lifts the object to the top or drops to the bottom.
Nudge	Moves the object up, down, left, or right by one pixel.
Align or Distribute	Align the left, center, right, top, middle, or bottoms of two or more selected objects. Distributes three or more selected objects vertically or horizontally.
Rotate or Flip	Rotates the object 90 degrees clockwise or counterclockwise or 180 degrees. Flips the object vertically or horizontally.
Properties	Displays the properties dialog for the block for configuring parameters and viewing metrics.
Create Input Create Output	Creates an input or output path on the block, depending on the type of block.
Snapshot Activities	Displays a temporary workspace that shows the current activities of the block. The number of activities on the workspace corresponds to the Current Activities of the block. Each activity object computes metrics about the specific activity. When the block finishes processing an activity, ReThink deletes the activity from the workspace; however, it does not dynamically add activities to the workspace. See Determining the Current Activities .
Disconnect	Breaks all connections between the block and other blocks in the model and adds junction blocks to the ends of the stubs that remain on the block. See Inserting a Block Between Two Connected Blocks .

Menu Choices	Description
Set Break	Causes ReThink to pause the simulation when a work object arrives at an input path of the block. ReThink indicates the break point by using an indicator arrow. You set break points to help in debugging. To continue the simulation, choose Continue or hide the indicator. See Debugging Blocks .
Clear Break	When you have set a breakpoint for a block, this menu choice appears allowing you to clear the breakpoint. Clearing the breakpoint causes work objects to continue flowing through the block normally. To clear all break points in a model, click the Clear Breaks button on the toolbar.
Show Scenario	Displays an indicator arrow next to the scenario that controls this block. This menu choice is only available when the scenario is active.
Update	Updates all the metrics for the block. See Updating Duration Metrics for Blocks .

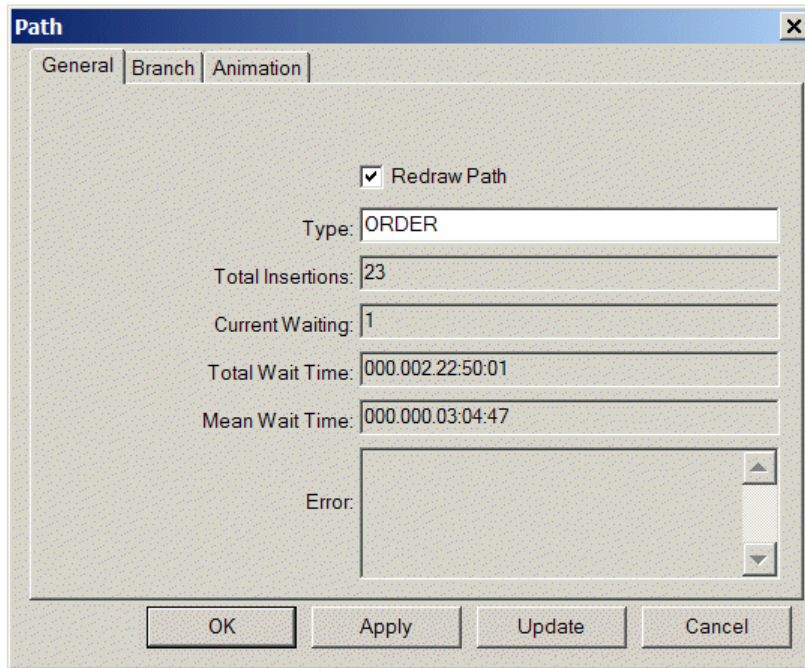
Common Attributes of Paths

The properties dialog of a path has these tabs:

Tab	Description
General	Parameter for specifying the type of work object that the path carries. Metrics that compute metrics related to the number of work objects the path carries and the amount of time the path has been waiting.
Branch	Parameters related to configuring the path for a Branch block and Yield block.
Animation	Parameters for configuring the animation color for paths.

The following sections describe the attributes that appear in Modeler mode. For information on the attributes that are available in Developer mode, see the *Customizing ReThink User's Guide*.

General Tab



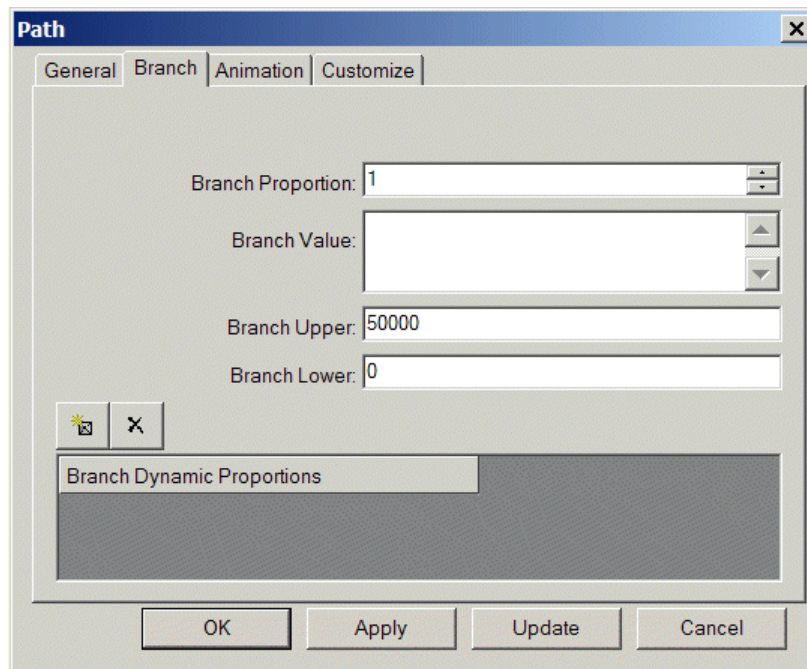
Attribute	P/M	Description
Redraw Path	P	Determines whether or not the path reconfigures itself when you move the connected blocks. See Disabling Path Redrawing .
Type	P	Specifies the class name of the work object that the block processes or creates, depending on the type of block. The default value is <code>bpr-object</code> , which allows work objects of all types to pass on the path. See Configuring the Type of Work that Blocks Process .
Total Insertions	M	The total number of work objects that have ever been on the path.
Current Waiting	M	The current number of work objects that are waiting in the path queue.

Attribute	P/M	Description
Total Wait Time	M	The total amount of time that all work objects have spent waiting in the path queue since the start of the simulation.
Mean Wait Time	M	The total amount of time that all work objects have spent waiting in the path queue (Total Wait Time) divided by the total number of work objects that have ever been in the path queue (Total Insertions).
Error	M	A description of any error for the path. See Debugging Blocks .

For information on path metrics, see [Analyzing the Wait Time Due to Work Backups](#).

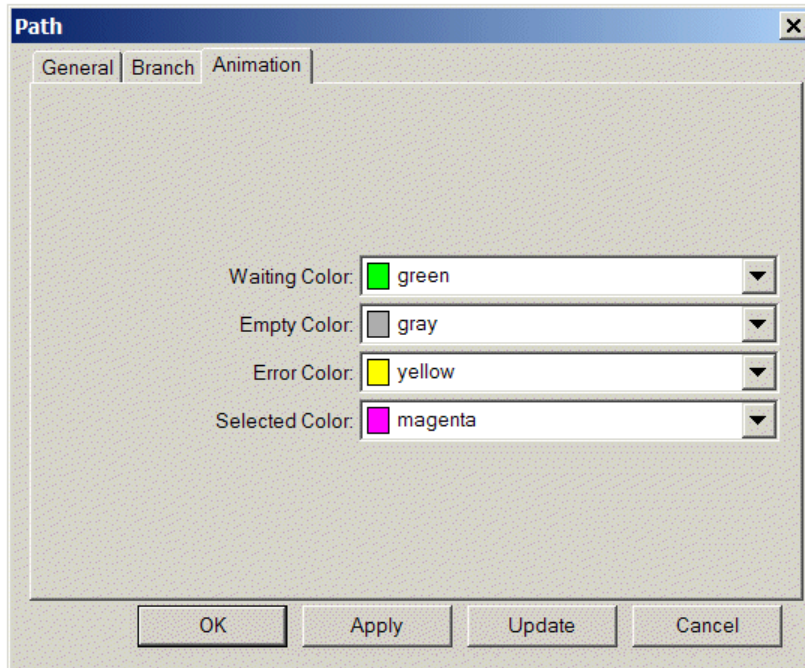
For information on menu choices related to the path queue, see [Showing Work Backups Interactively](#).

Branch Tab



For information on the attributes on the Branch tab, see [Path Attributes that Pertain Only to Branching](#).

Animation Tab



Attribute	P/M	Description
Waiting Color	P	The color of the path when work objects are in the path queue due to work backups.
Empty Color	P	The color of the path when no work objects are in the path queue.
Error Color	P	The color of the path when it is in an error state.
Selected Color	P	The color of the path when you select it for certain operations, such as when you use the Choose Original Output Path menu choice of a Copy block.

For more information, see [Configuring the Animation of Paths](#).

Common Menu Choices for Paths

Menu Choices	Description
Delete	Breaks the connection between two blocks and adds junction blocks to the ends of the stubs that remain on the block. See Inserting a Block Between Two Connected Blocks .
Properties	Displays the properties dialog of the path.
Snapshot Queue	Displays a workspace that shows the work objects currently waiting on the path. See Showing Work Backups on an Input Path .

Associate



An Associate block relates two or more work objects that get out of sequence in a process. For example, you use an Associate block to relate an order and its invoice so that you can match them again later in the process. You match associated items, using the [Reconcile block](#). You can also store associated objects in a pool, using the [Store block](#), then retrieve them later in the process, using the [Retrieve block](#).

The Associate block synchronizes its inputs by waiting to process until all of its inputs have arrived at the block. You can add input paths to the block to associate more than two work objects.

You can use the Associate block to create a new **association**, which is a name that links the associated work objects, or to add work objects to an existing association, thereby associating multiple work objects, using the same association. When reconciling multiple associated work objects, you can reconcile individual associated work objects or all associated work objects. Similarly, if multiple work objects are waiting on the input path of the Associate block, you can choose to associate individual work objects or all work objects at once.

Occasionally, you might need to use the Associate block to remove work objects from an association. For example, you might want to remove work objects from an existing association, then create a new association between one of the work objects and some other work object.

Configuring the Association Mode

The Associate block provides three operating modes:

Mode	Description
New	Creates a new association object between the input work objects.
Add	Adds an input work object to an existing association of another input work object, thereby creating multiple associated work objects.
Remove	Removes an input work object from an existing association of another input work object. If no associated work objects exist when you remove work objects from the association, the Associate block also deletes the association.

Creating New Associations

When you use the block to create a new association, the block associates the work objects on each input path and passes the associated objects together downstream.

To create a new association between two objects:

- 1 On the Block tab of the properties dialog, name the association in the Association Name attribute, using a symbol.

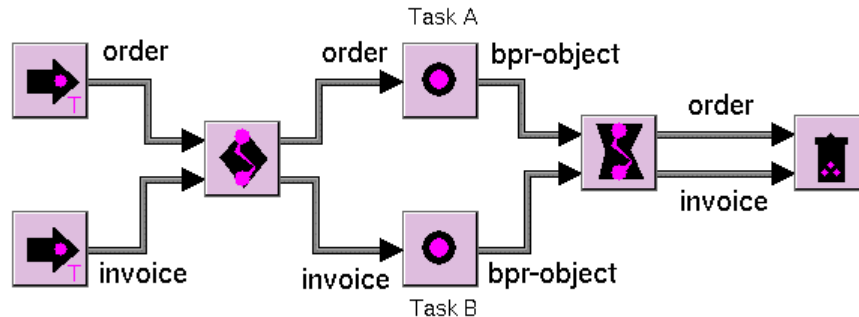
This is the association name that you use downstream in a process to reconcile the two objects.
- 2 Configure the Mode to be **New**.
- 3 Click the Associate All option on to create a new association between all work objects as they arrive at the block; otherwise, use the default to create new associations between individual work objects only.
- 4 Configure the output path types of the block.

Note If you use the default path type for the output paths of the block, ReThink places both input work objects on the same output path.

For a general explanation of how ReThink determines the output path type for blocks with multiple output paths, see [Determining the Output Path Based on Its Type](#).

For example, you might associate an order and an invoice upstream in a process, then pass each separately to Task A and Task B, which might have different

durations. The Association Name might be `order-invoice`. Once the individual processing is complete, the Reconcile block reconciles the order and invoice again, matching associated objects based on the association name.



For information on reconciling associated objects, see the [Reconcile block](#).

Adding Work Objects to Existing Associations

Suppose you have several work objects that are all associated with a single work object. For example, you might have a software release object that is associated with both a software build and its documentation. To model such a process, you might create an initial association between the software release and the software, then add the documentation to the existing association downstream in the process. You could then reconcile both the software and the documentation with the software release when both tasks are complete.

To add work objects to an existing association:

- 1 Create a model in which you create a new association between work object A and work object B, as described in [Creating New Associations](#).
- 2 On the Block tab of the properties dialog, click the Associate All option on to add work objects to existing associations as they arrive at the block; otherwise, use the default to add individual work objects to existing associations.
- 3 Downstream in the model, create an association between work object C and work object B, where:
 - Association Name is the same as the Association Name between work object A and work object B.
 - The Mode attribute of the Associate block is **Add**.

For a complete example that adds work objects to an existing association, then reconciles all of the work objects, see [Reconciling All Objects](#).

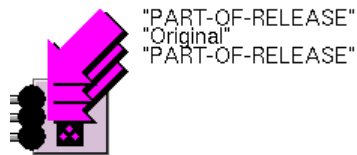
Showing Associated Work Objects

To show associated work objects in a running model:

- Run the simulation in step mode and choose Show Associations on a work object.

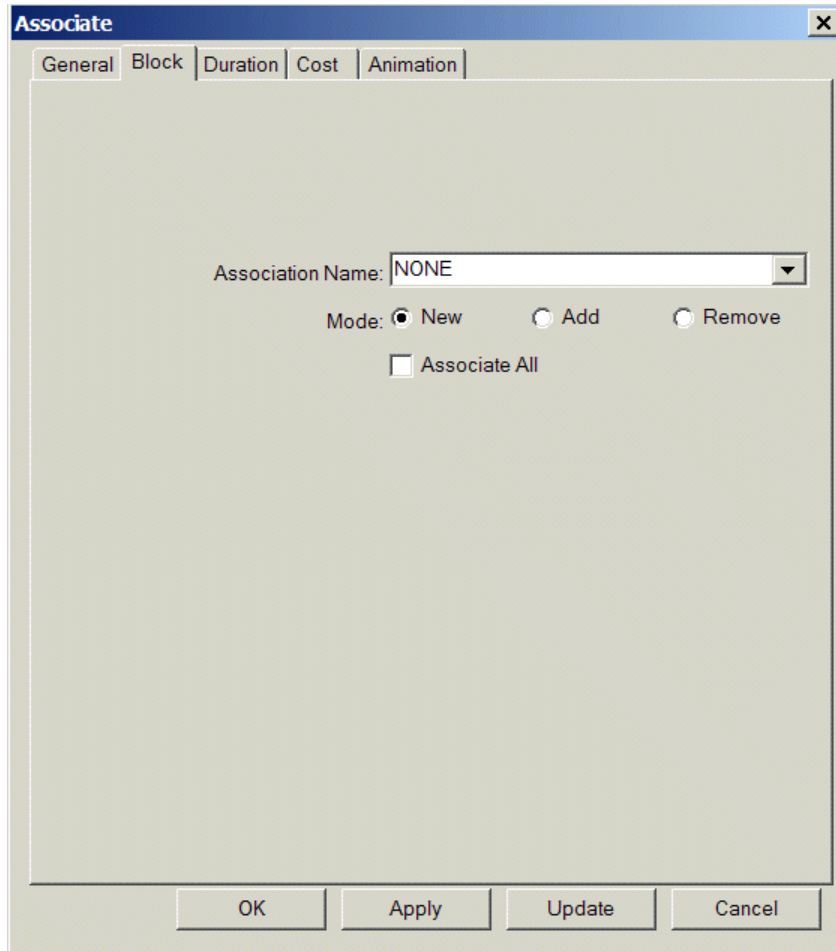
ReThink displays an indicator arrow next to all the associated work objects in the model. The label of the associated objects displays the association name, while the label of the selected object is "Original."

For example, here is the Sink block in the example shown in [Reconciling All Objects](#) after choosing Show Associations on the software release object:



Specific Attributes

Here is the Block tab of the properties dialog for the Associate block:



The specific attributes of the Associate block are:

Attribute	Description
Association Name	The name of the association that the block creates between the associated objects. The Reconcile block uses this association name when it reconciles the associated objects.
Mode	Whether the block creates a new association object between the input work objects, adds an input work object to an existing association of another input work object, or removes an input work object from an existing association of another input work object.
Associate All	If multiple work objects arrive on one of the input paths while the other path is waiting for an object to associate, enabling the Associate All option associates all input work objects, according to the Mode. If Associate All is off, the block associates objects one at a time.

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The Associate block has no specific menu choices.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Batch



A Batch block gathers a group of objects into a **batch** before passing them together to the downstream block. By default, ReThink waits for a critical threshold of objects to arrive before passing them downstream. The block can also pass the batch downstream based on the sum of a value of an attribute of an input work object, a particular work object arriving at the block, or a specified time interval.

You can batch objects into a group and pass the group of objects simultaneously, or you can batch objects into a container and pass the container.

The Batch block synchronizes its inputs by waiting until it receives all the objects in the batch before it passes them downstream together.

The Batch block creates a single activity for each object it processes. For example, if three objects are in the batch, the block creates three activities.

Configuring the Batch Mode

A Batch block provides the following operating modes:

Batch Mode	Description
Number	Passes the batch when a critical threshold or number of objects arrive at the block.
Sum	Passes the batch when the sum of the values of an attribute of the input work objects exceeds a minimum threshold.
Trigger	Passes the batch when a triggering work object arrives at the block on the specified input path.
Interval	Passes the batch at a specified start and end time, at a specified frequency, and on specified days.
Custom	Allows you to specify a custom procedure for determining the batch.

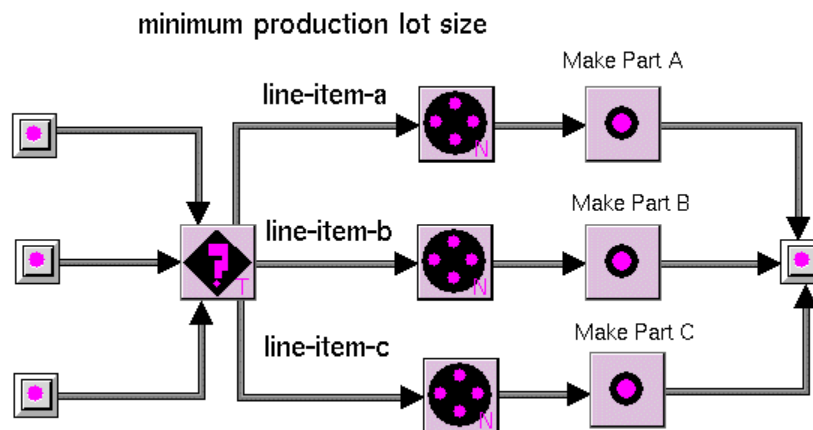
Batching Objects in a Group

To batch objects in a group, specify the number of input work objects in the batch. The block passes the objects together to the downstream block when the number of input work objects meets the threshold.

To batch objects in a group:

- 1 On the Block tab of the properties dialog, configure the Batch Mode as Number.
- 2 Configure the Threshold attribute to be the number of work objects in the batch.

This example uses number mode to model the replenishment detail of a manufacturing process. Line items of different types come into the Branch block, which separates them by type. The Batch blocks require a minimum production lot before the line items are manufactured and stored in inventory.



Batching Objects By Summing an Attribute of a Work Object

You might have a batch process that determines when to pass the batch by summing the value of an attribute of the input work object and comparing this sum to a threshold. For example, you might batch boxes to load on a truck based on the total weight.

To batch objects by summing an attribute of a work object:

- 1 Define a work object class definition with an attribute whose value the Batch block will sum.

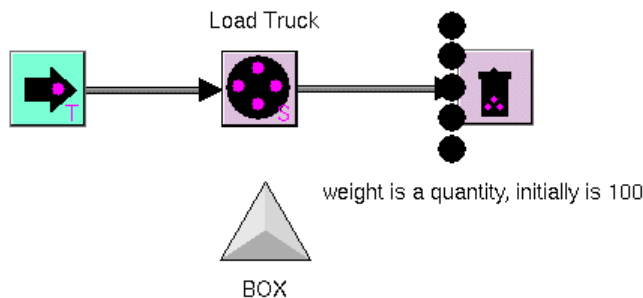
Configure the attribute type to be a quantity and provide an initial value, for example:

weight is a quantity, initially is 100

For details, see [Creating a New Class of Work Object](#).

- 2 On the Block tab of the properties dialog, configure the Batch Mode as Sum.
- 3 Configure the Attribute Name as the attribute of the work object whose value the Batch block will sum.
- 4 Configure the Minimum Threshold as a numerical value that represents the threshold the block will use to determine when to pass the batch.
When the sum exceeds the threshold, the block passes the batch.
- 5 Configure the input path type of the Batch block to be the class you defined.

This example shows a before and after view of a simple model that batches boxes, based on the weight of each box, which is 100 pounds. When the Batch block receives enough boxes such that the total weight equals 500 pounds, the block passes all the work objects downstream.



This figure shows the Block tab of the properties dialog, which specifies the attribute to sum and the threshold:

The screenshot shows the 'Batch: Load Truck' dialog box with the 'Block' tab selected. The 'Batch Mode' is set to 'Sum'. The 'Container List Attribute' is 'CONTAINER-LIST'. The 'Threshold' is '1'. The 'Attribute Name' is 'WEIGHT'. The 'Minimum Threshold' is '500'. The 'Start Time', 'End Time', and 'Period' are all set to '000:000:00:00:00'. The 'Days' section has checkboxes for Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday, all of which are unchecked. The buttons at the bottom are 'OK', 'Apply', 'Update', and 'Cancel'.

Batching Objects Based on a Triggering Work Object

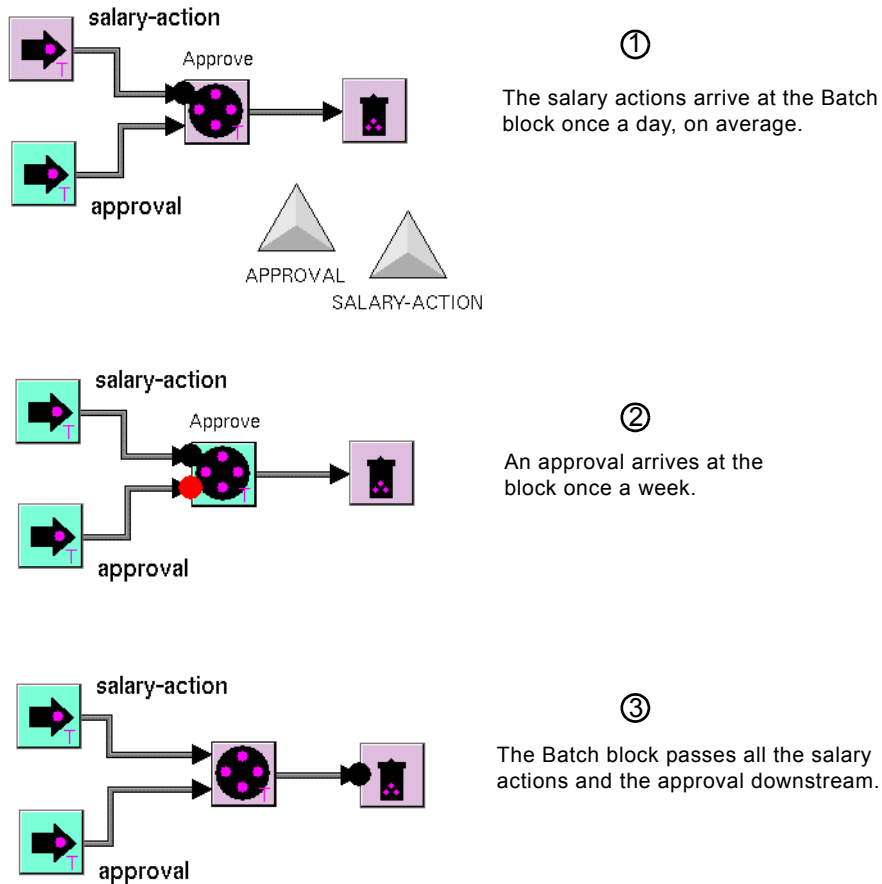
You might have a batch process that passes a batch downstream based on the arrival at the block of a particular type of work object. Typically, the triggering work object arrives at the block less frequently than the other work. For example, you might use this mode to model an approval process for salary actions, where the salary action requests arrive once a day, on average, and approvals are given once a week.

When batching work objects based on a trigger object, you can configure the block so it processes the trigger object or deletes it. If you do not identify the trigger output path, the Batch block deletes the trigger object.

To batch objects based on a triggering work object:

- 1 Create a Batch block with at least two input paths and configure different types of work objects.
- 2 On the Block tab of the properties dialog, configure the Batch Mode to be Trigger.
- 3 Choose the Choose Trigger Input Path menu choice on the Batch block, then choose Select on the input path whose work object type triggers the Batch block.
- 4 If you want the Batch block to process the trigger object, choose the Choose Trigger Output Path menu choice and choose Select on the output path that should carry the trigger object.

This example shows three stages in a running process that models an approval process for salary actions. The trigger input path is the path that sends approvals into the Batch block. Because the output path has been identified as the trigger output path, the block passes both the salary actions and approval processes onto the output path.



Batching Objects At Specified Time Intervals

Your process might need to batch work according to a schedule, beginning and ending at a particular time. For example, you might process work once every three hours between the hours of 9:00 and 5:00, Monday through Friday. If work arrives at the Batch block approximately once an hour, on average, the block will emit three work objects at each time interval between the designated hours.

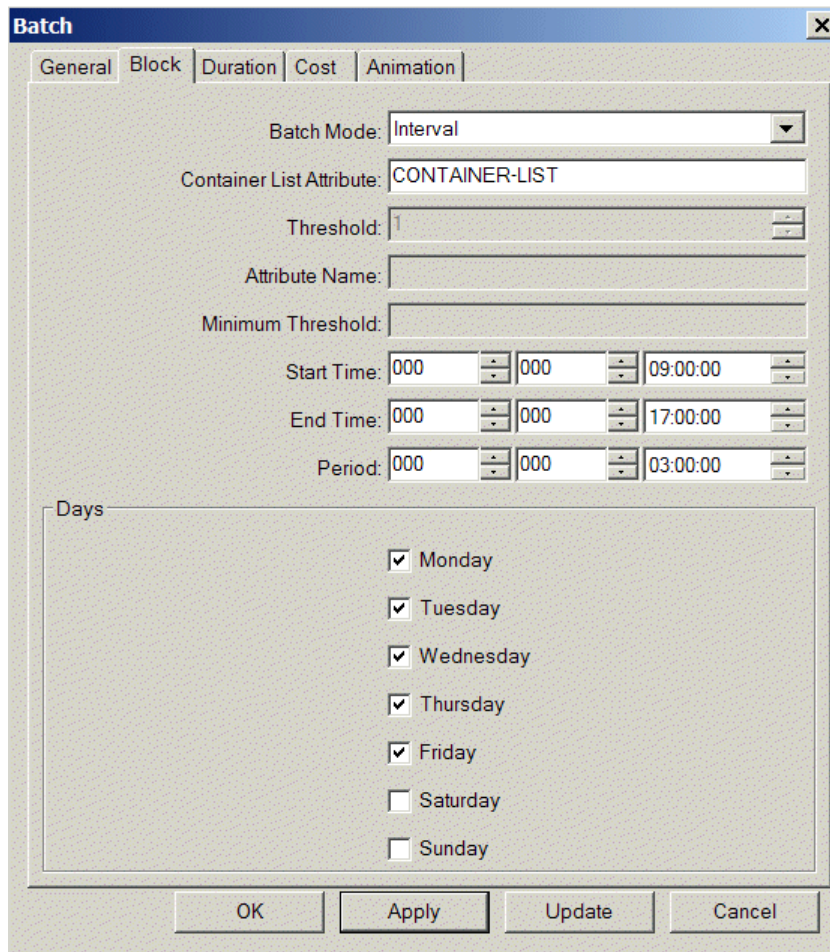
Note If work arrives at the block at the exact same time that the batch is emitted, it is indeterminate as to whether the work object is emitted with the current batch or the next batch.

When you use the Batch block to batch work at specified time intervals, you cannot specify the duration of the block; the block computes the duration for you.

To batch objects at specified time intervals:

- 1 On the Block tab of the properties dialog, configure the Batch Mode as Interval.
- 2 Configure the Start Time and End Time as time intervals, which determine the time at which the block emits its first and last batch on each of the specified days.
- 3 Configure the Period as a time interval, which determines the times at which the block emits each additional batch during the day.
- 4 Configure the Days by selecting the desired day of the week, during which the Batch block processes batches according to the Start Time, End Time, and Period.

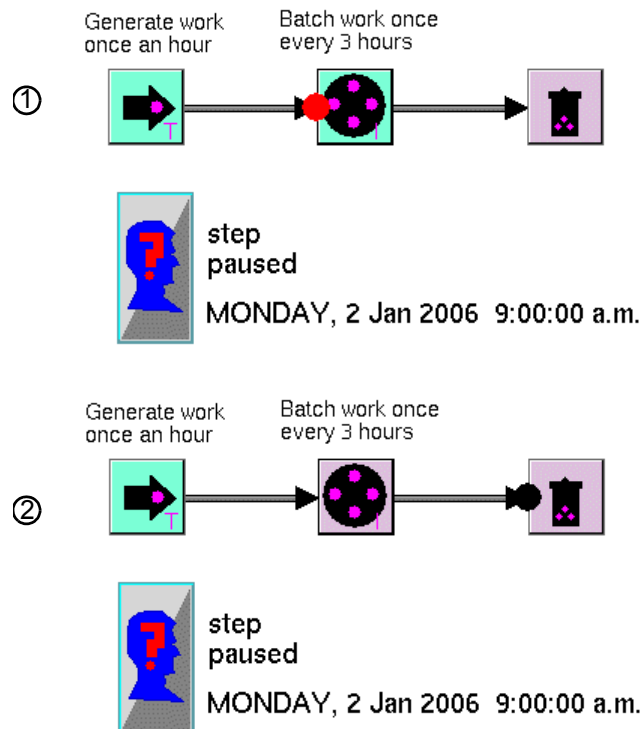
Here is Block tab of the properties dialog for a Batch block that batches work once every three hours on weekdays:



The following example shows two steps in a model that batches work once every three hours during normal business hours, Monday through Friday. The Source block generates work once an hour, 24 hours a day.

The first model shows work objects sitting on the input path of the Batch block, waiting for the first batch. The simulation time is 9:00, which is the start time of the Batch block. The next model shows the next step in the process when the

Batch block passes the batch downstream. The work objects in the batch have been moved to show all the objects.



Batching Objects into a Container

You can use the Batch block in any mode to batch objects into a container. A container is a work object that defines an item-list attribute in which the batch is stored. The Batch block passes the container to its output path, rather than passing the individual objects.

Using the Batch block to batch items into a container is similar to using the Insert block to insert all its objects into an item-list attribute of another object.

Once you have inserted the batch into the container, you can remove the batch, either one at a time or all at once, using the Remove block.

The Batch block creates one activity for each item in the container.

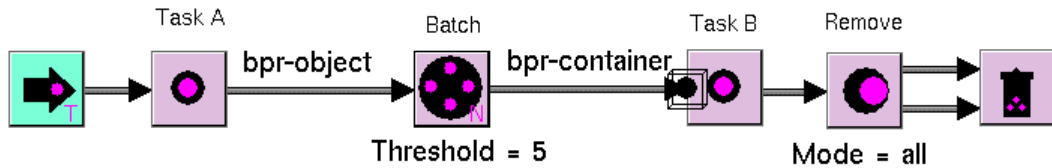
To batch items into a container:

- 1 Configure the output path type of the Batch block to be `bpr-container`, any subclass of `bpr-container`, or any user-defined object that has an item-list attribute.

The `bpr-container` class defines an item-list attribute named `container-list`. The block stores the objects in the `container-list` attribute of the container.

- 2 On the Block tab of the properties dialog, configure the Batch Mode and configure the appropriate mode attributes and paths for the particular mode.
- 3 Configure the Container List Attribute to be the item-list attribute of the container, which is `container-list`, by default.

For example, the following figure shows a simple model that batches five objects into a container, using number mode, processes the container, then removes the objects all at once, using a Remove block.



For information on removing objects from a container, see the [Remove block](#).

Showing Work Objects in the Container

To show work objects in the container:

- ➔ Run the simulation in step mode and choose Snapshot Container on a container.

ReThink displays a workspace with all the work objects in the container, similar to the workspace you see when you use Snapshot Queue on a path with work backups.

Specific Attributes

Batch [X]

General | **Block** | Duration | Cost | Animation

Batch Mode: Number

Container List Attribute: CONTAINER-LIST

Threshold: 1

Attribute Name:

Minimum Threshold:

Start Time: 000 000 00:00:00

End Time: 000 000 00:00:00

Period: 000 000 00:00:00

Days

- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday
- Sunday

OK Apply Update Cancel

The specific attributes of the Batch block are:

Attribute	Description
Batch Mode	The criteria the block uses to pass the batch to the downstream block. The options are: Number , Sum , Interval , Trigger , and Custom . The default value is Number .
Container List Attribute	An attribute of the container whose value is an item-list. You specify this attribute when you batch objects into a container. The default value is the container-list attribute of a bpr-container . You can use dot notation to refer to the attribute of a subobject, for example, my-subtable.my-attr .

The mode-specific attributes of the Batch block are:

Mode	Attribute	Description
Number	Threshold	The number of objects the Batch block must receive before it passes the objects onto its output path. The default value is 1.
Sum	Attribute Name	An attribute of the input work object whose value the block sums. You can use dot notation to refer to the attribute of a subobject, for example, my-subtable.my-attr .
	Minimum Threshold	The minimum value of the sum of the specified attribute. The block passes the work objects when the sum exceeds the threshold.
Interval	Start Time	A time interval that represents the time at which the block emits its first batch on each of the specified days.

Mode	Attribute	Description
	End Time	A time interval that represents the time at which the block emits its last batch on each of the specified days.
	Period	A time interval that represents a fixed time between batches.
	Days	The days during which the block emits batches according to the Start Time, End Time, and Period.
Trigger	N/A	N/A
Custom	Batch Procedure Name	See Customization Attributes .

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for the Batch block are:

Menu Choice	Description
Choose Trigger Input Path	When Batch Mode is Trigger , this menu choice identifies the input path that carries the trigger object, which triggers the Batch block to execute. You must identify the trigger input path.
Choose Trigger Output Path	When Batch Mode is Trigger , this menu choice identifies the output path that carries the trigger object. If you do not identify the trigger output path, the Batch block deletes the trigger input object.

Menu Choice	Description
Show Trigger Input Path	When the trigger input path is selected, this menu choice displays an indicator arrow next to the trigger input path.
Show Trigger Output Path	When the trigger output path is selected, this menu choice displays an indicator arrow next to the trigger output path.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Customization Attributes

The customization attribute available in Developer mode for the Batch block is:

Attribute	Description
Batch Procedure Name	When Batch Mode is Custom, specifies the procedure name that determines when the Batch block passes the batch. The default value is <code>bpr-batch-number-threshold</code> .

You can customize the threshold procedure to specify a different criteria for passing the batch or container to the output path. For more information, see the *Customizing ReThink User's Guide*.

Branch



The Branch block implements any kind of decision-making operation, for example, routing, sorting, collating, or any other kind of branching.

You can branch work based on probability, type, or attribute value, or you can interactively select the output path. You can also implement branching based on more sophisticated logic by writing rules that set the attribute value upon which the branching is based.

Configuring the Branch Mode

A Branch block supports the following operating modes:

Branch Mode	Description
Proportion	Branches work based on the proportion specified on each output path of the block. This is the default mode.
Dynamic Proportion	Branches work based on the list of proportions specified on each output path. The block uses each subsequent pair of proportions each time the work object loops through the branch.
Type	Branches work based on the path type of each output path of the block and the type of work object that flows into the block.
Prompt	Branches work to the output path that the user selects interactively.
Attribute Value	Branches work based on an attribute value of the work object, a threshold or range, and a mathematical operation that compares the current value to the threshold or range. In Attribute Value mode, you can use rules to set the attribute value based on testing multiple attribute values in a work object.
Custom	Allows you to specify a custom procedure for determining how to branch work objects.

Branching Based on Proportion

You branch work objects based on proportion by specifying a relative proportion on each output path of the Branch block. The work objects flow more often onto the output paths with the relatively higher proportion and less often onto the output paths with the relatively lower proportion.

The value for Branch Proportion can be zero or any positive number, and the sum of the values for all the output paths need not total 1.

When the values you enter add to 1, these numbers represent percentages. For example, you might enter 0.25 on one path and 0.75 on another, in which case ReThink passes objects approximately one-quarter of the time onto one path and approximately three-quarters of the time onto the other path.

When the values add to a number greater than 1, these numbers represent fractions. For example, if you enter 1 on one path, and 1 on the other path, which is the default, ReThink passes work objects half the time on one path and half the time on the other. If you enter 2 on one path and 5 on another path, ReThink passes work objects approximately 2/7 of the time on one path and approximately 5/7 of the time on the other.

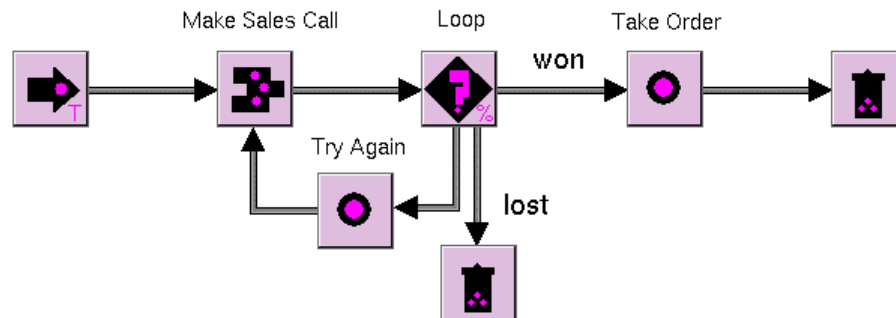
The Branch block computes its probability each time a new work object arrives at the block; the probability does not depend on the previous probability.

To branch work objects based on proportion:

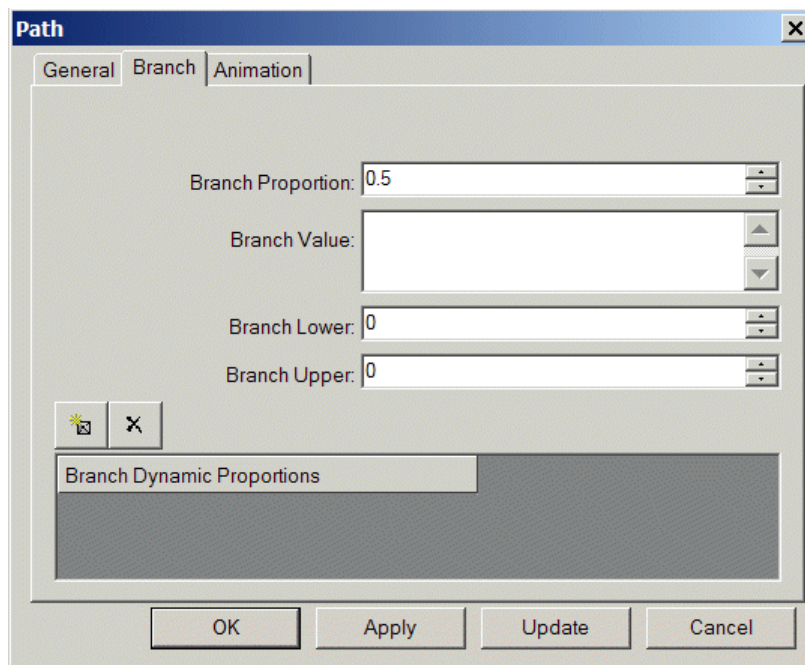
- 1 On the Block tab of the properties dialog, configure the Branch Mode as Proportion, the default.
- 2 Create the desired number of output paths on the Branch block.
- 3 Display the properties dialog for each output path and configure the Type to carry any type of work object.
- 4 Click the Branch tab of the properties dialog for each output path and configure the Branch Proportion to determine the relative proportion of the time that work objects flow onto each path.

This example shows how you use the Branch block to model predictable behavior in a sales process. The leads come in, sales calls are made, and the sales calls pass

to a Branch block. The Branch block uses proportions to model three potential outcomes: the order is won, the order is lost, or a sales call must be made again.



Here is the Branch tab of the properties dialog for the output path that represents the orders that are won:



Branching Based on a Dynamic Proportion

Suppose you have a process in which work objects loop through the Branch block, based on some probability, where the probability of each output path changes each time a work object loops back through the block. You model this type of branching by using dynamic proportion mode.

To use dynamic proportion mode, you configure a list of proportions for each output path. The block uses each subsequent proportion for each output path each time the work object loops through the block.

If a work object loops through the block more than the specified number of proportions, the block uses the last proportion in the list.

To branch work objects based on dynamic proportions:

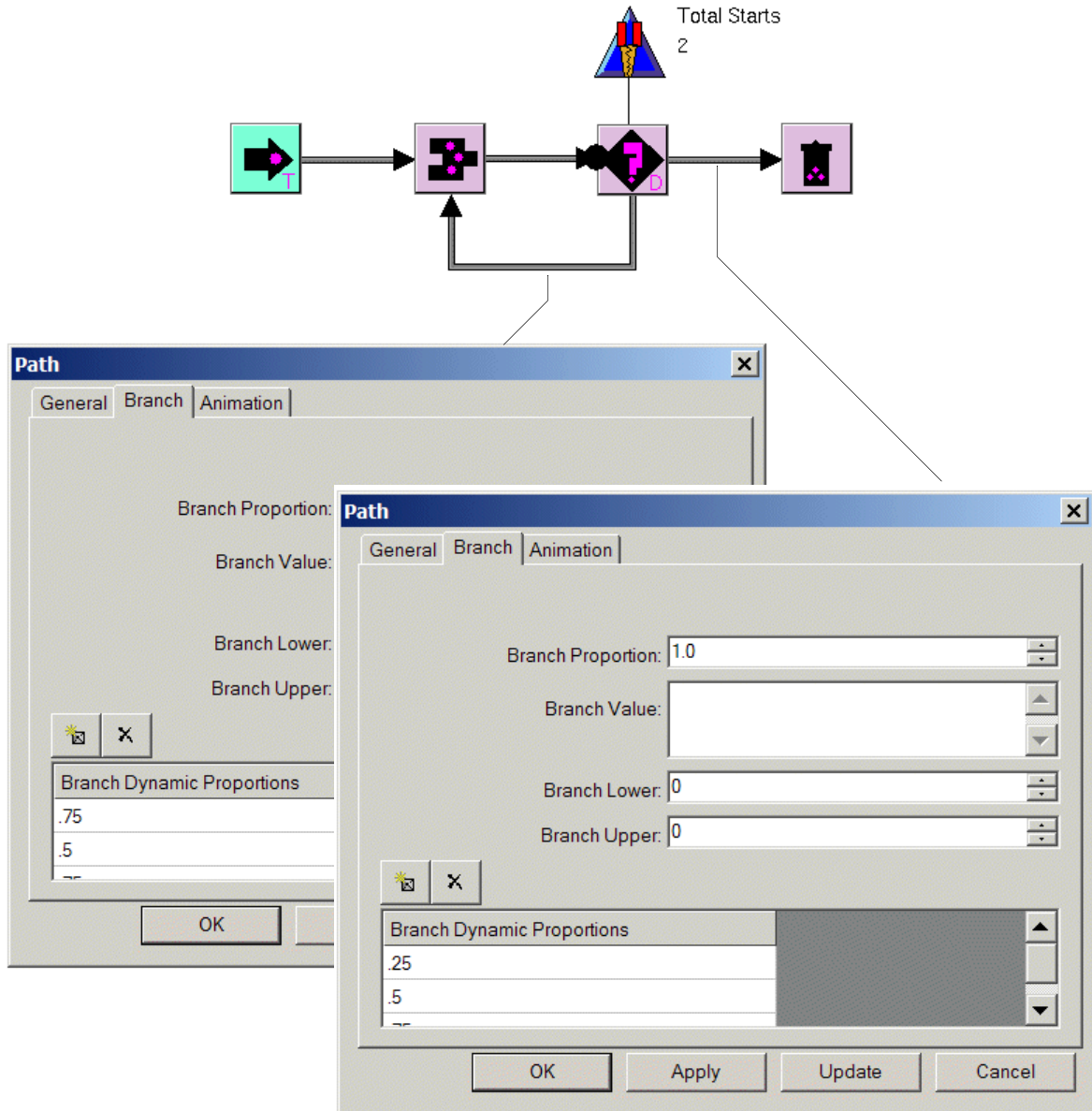
- 1 On the Block tab of the properties dialog, configure the Branch Mode as Dynamic Proportion.
- 2 Create the desired number of output paths on the Branch block and make one of the output paths loop to an upstream path in the model.
- 3 Display the properties dialog for each output path, click the Branch tab, and configure the relative proportions, based on the number of times a work object loops through the model.

You specify values in the same way that you specify values in proportion mode, either as a percentage or as a fraction.

For example, to model a process in which the likelihood of a work object looping back through the Branch block is 75% initially, 50% the next time through, and 25% the third time, the dynamic proportions of each output path would look like this:

Loop Path	Downstream Path
.75	.25
.5	.5
.25	.75

The following example shows a simple model that uses dynamic proportion mode. The Sample probe shows the Total Starts of the Branch block, which indicates that the work object has looped through the model twice.



Branching Based on Type

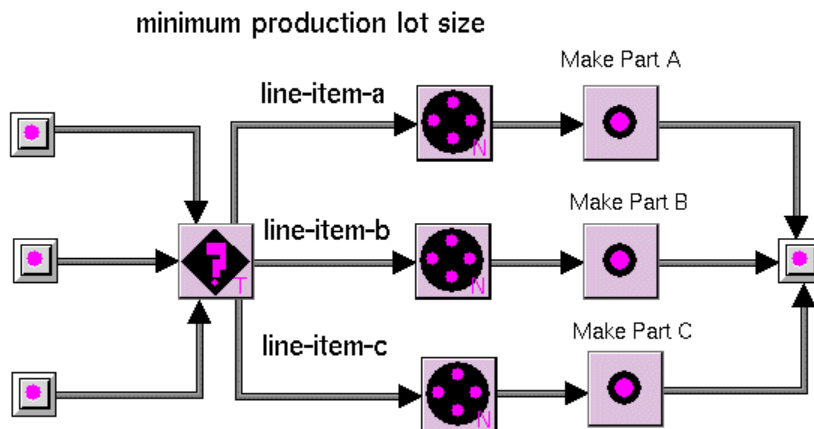
You can branch work objects based on the type of work object by specifying the type of each output path of the Branch block and by specifying type mode. Work objects pass onto the output path whose type most closely matches the object class.

For a general explanation of how ReThink determines the output path type for blocks with multiple output paths, see [Determining the Output Path Based on Its Type](#).

To branch work objects based on type:

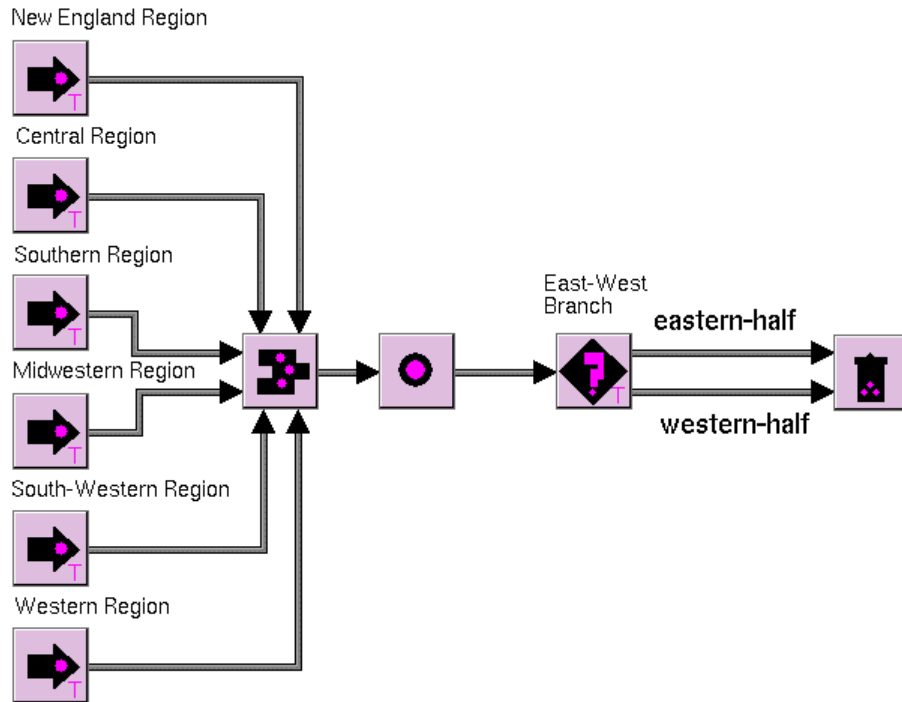
- 1 On the Block tab of the properties dialog, configure the Branch Mode as **Type**.
- 2 Display the properties dialog for each output path and configure the Type to determine on which output path each type of work object passes.

This example shows how you use the Branch block to branch different types of line items before batching them into production lots and storing them in inventory:

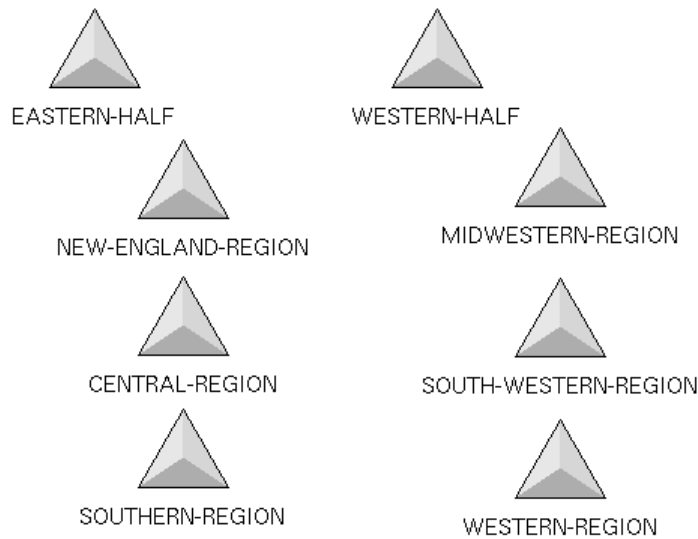


Depending on how you have defined your class hierarchy for the work objects in your model, you can use the Branch block in type mode to branch work according to the superior class of the object, as opposed to the object class itself. For example, the next model generates six different types of sales calls, each from a

different region. The East-West Branch block branches calls according to whether they are in the eastern or western half of the country.



The class hierarchy that supports this type of branching defines two superior classes, **eastern-half** and **western-half**. The three eastern regions are subclasses of the eastern-half class, and the three western regions are subclasses of the western-half class.



Interactively Selecting the Output Path

You can branch work objects by interactively selecting the path on which the work objects should pass. You use this mode for:

- Testing purposes to pass explicitly a work object onto a path that handles exceptional cases.
- Demonstration purposes to control on which output path a work object flows.

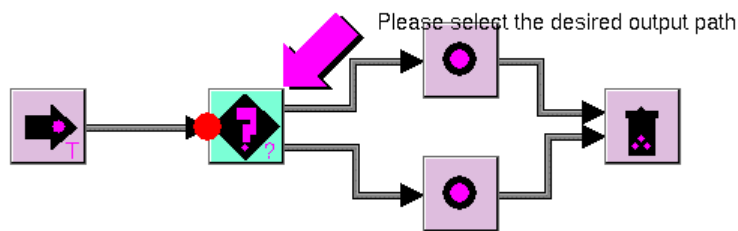
For example, the Branch block might specify a 5% probability that an order is returned. During normal processing, you would only rarely see work objects flow onto this path. Therefore, to test this outcome, you can temporarily override the branch mode and explicitly select the path with a 5% probability.

To select explicitly the output path for branching:

- 1 On the Block tab of the properties dialog, configure the Branch Mode as Prompt.
- 2 Display the properties dialog for each output path and configure the Type so that it can carry any type of input work object.
- 3 When the Branch block receives a work object, it prompts you to select an output path for the work object.
- 4 Choose Select on the desired output path to select it.

The work object passes onto the selected path, and the model continues running.

For example:



Branching Based on Attribute Value

You can branch work objects based on the value of an attribute of the work object. The block tests the current value of the specified attribute against a value specified on the path. You can branch work based on an attribute value by using:

- A branch operation such as equality, greater than, or less than, which the block uses to perform the comparison with an attribute of the block.
- A range, which tests against a lower and upper threshold attribute of the path.

To branch work objects based on an attribute value of the work object:

- 1 On the Block tab of the properties dialog, configure the Branch Mode as Attribute Value.
- 2 Configure the Branch Attribute to name an attribute of the work object whose value the block will test.
- 3 Configure the Operation to determine how the block compares the attribute value of the work object to a value of the path.

When you are comparing symbolic values for equality, specify the = operation, the default.

When you are comparing numeric values, specify one of the following operations: =, !=, <, <=, >, >=, or Range.

For information on branching based on a range of values, see [Branching Based on a Range of Values](#).

- 4 Display the properties dialog for each output path, click the Branch tab, and configure the Branch Value to determine the threshold for comparison.

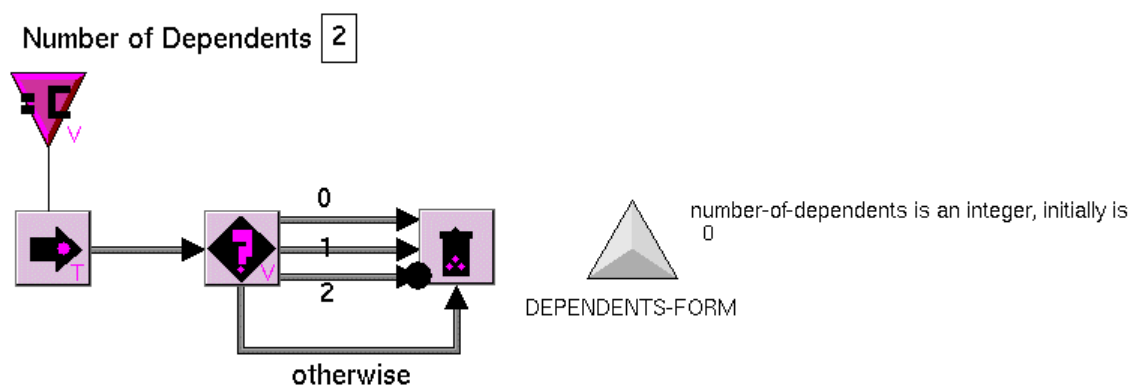
The value can be a number, symbol, string, or otherwise, which passes work objects on its path when no other condition is met.

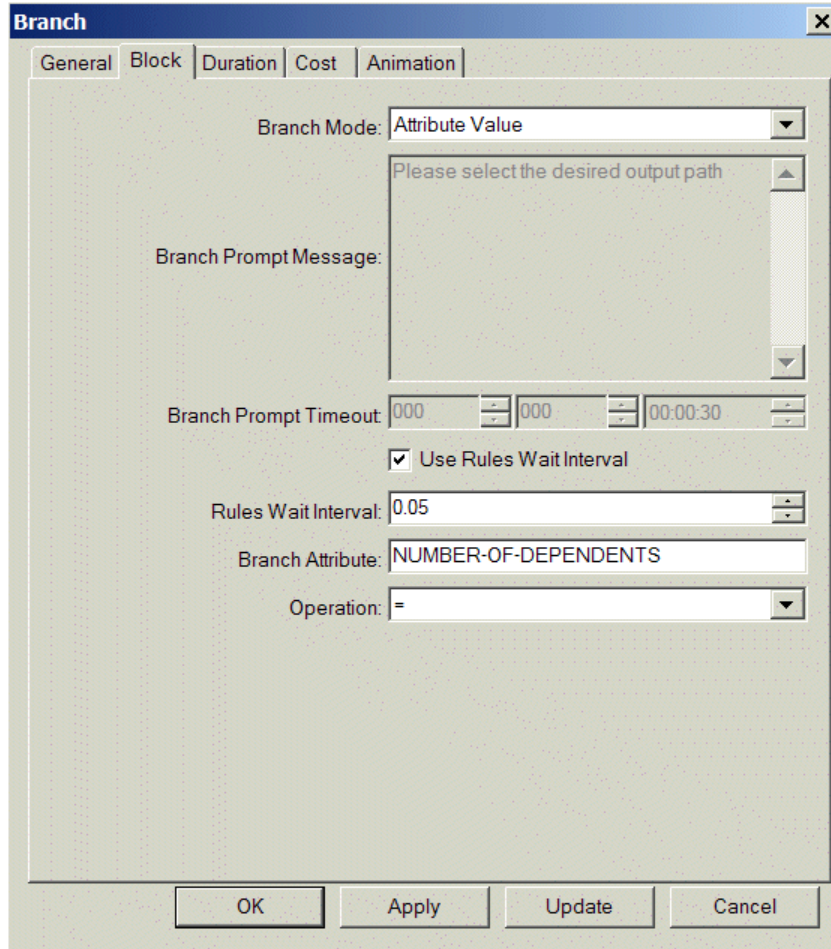
Note Do not specify the same symbol as the Branch Value for more than one output path; otherwise, the output path will be unpredictable.

- 5 On the General tab for each output path, configure the Type so it can carry any type of input work object.

The block compares the current value of the Branch Attribute of the work object to the Branch Value of each output path, using the Operation.

For example, the following model shows how to branch work by testing the number-of-dependents attribute of a dependents-form to see if it is equal to a particular value. The Branch Value of the output paths are 0, 1, 2, and otherwise.





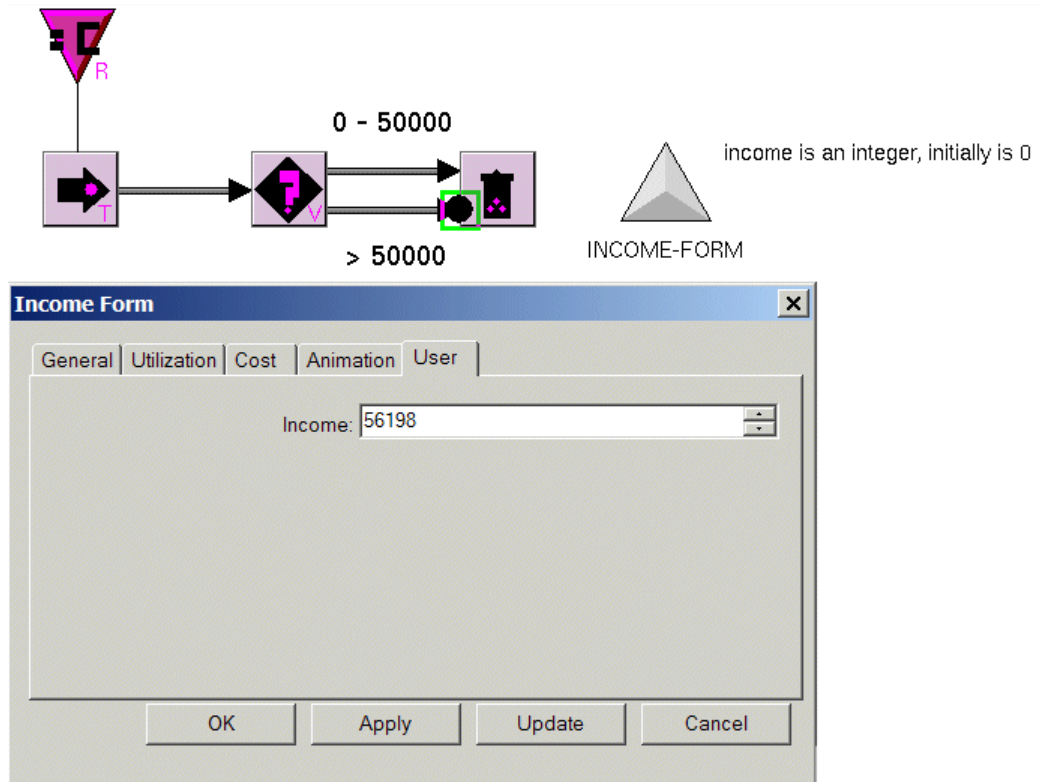
Branching Based on a Range of Values

When the Operation is Range, you have several options. For each output path of the Branch block:

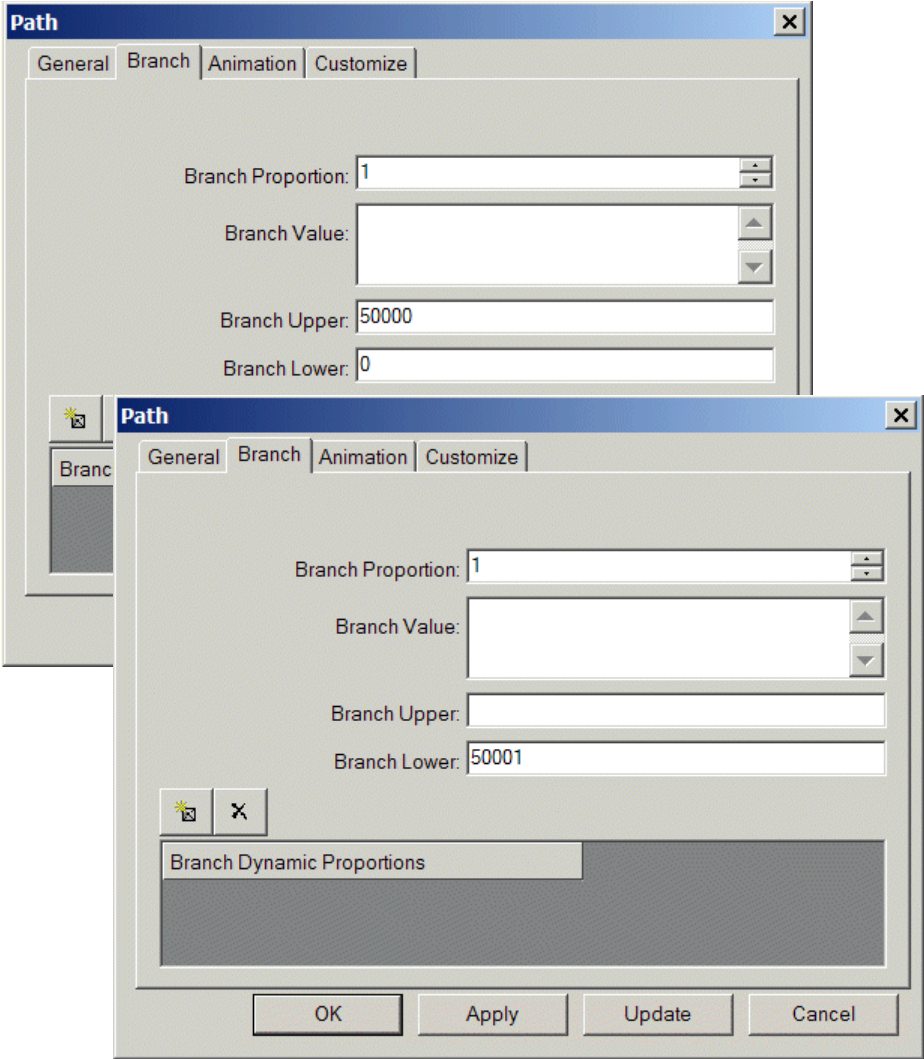
- Specify a distinct range by entering an upper threshold in the Branch Upper attribute of the path and a lower threshold in the Branch Lower attribute. The Branch Upper must be greater than the Branch Lower.
- Specify an open-ended range by entering just an upper threshold for comparison in the Branch Upper attribute of the path.
- Specify an open-ended range by entering just a lower threshold for comparison in the Branch Lower attribute of the path.

Note The Branch Upper and Branch Lower of two output paths should not overlap; otherwise, the output path will be unpredictable.

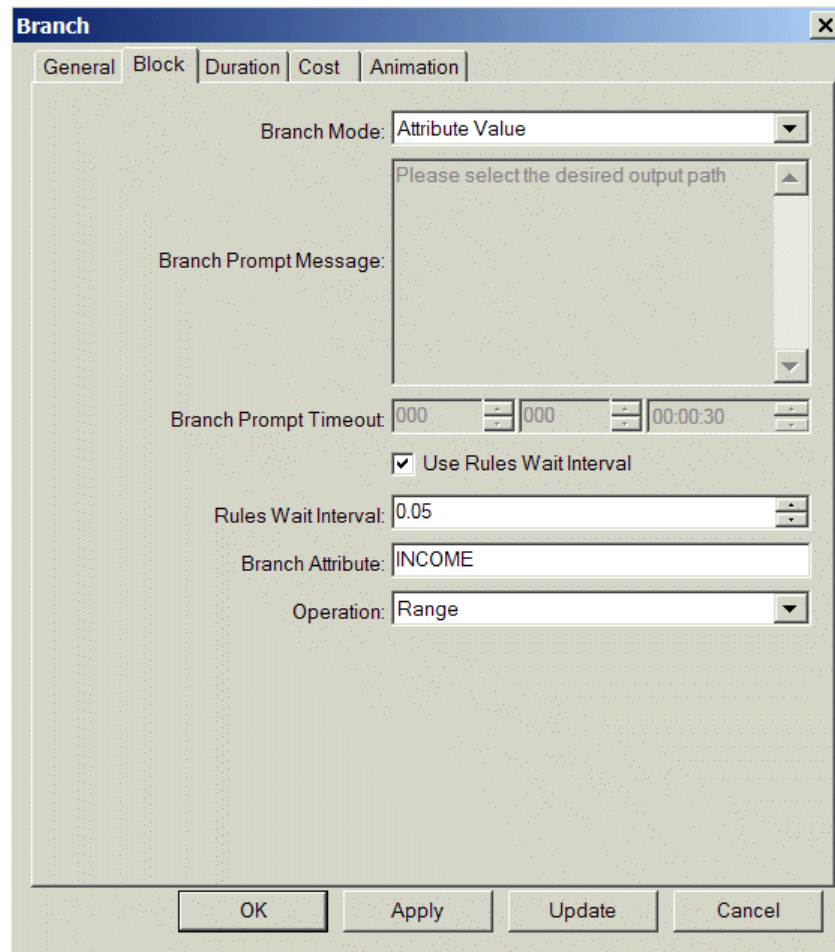
For example, the following model shows how to branch work by testing the income attribute of an income-form, using a range, where the Change feed feeds a random number into the object:



Here are the properties dialogs for each output path:



Here is the Block tab of the properties dialog for the Branch block:



Branching Based on Rules that Set the Attribute Value

In attribute value mode, you can create a rule or a set of rules on the Branch block detail that tests the value of one or more other attributes in the model and sets the attribute value upon which the branching is based. The block uses the attribute value the rule sets in the same way it uses the attribute value in normal attribute value mode.

You can create “if” rules with multiple conditional statements, using “and” or “or” logic. The rule should conclude a value for the attribute of the work object specified in the Branch Attribute of the block. The concluded attribute should be the value of the Branch Value of an output path of the block.

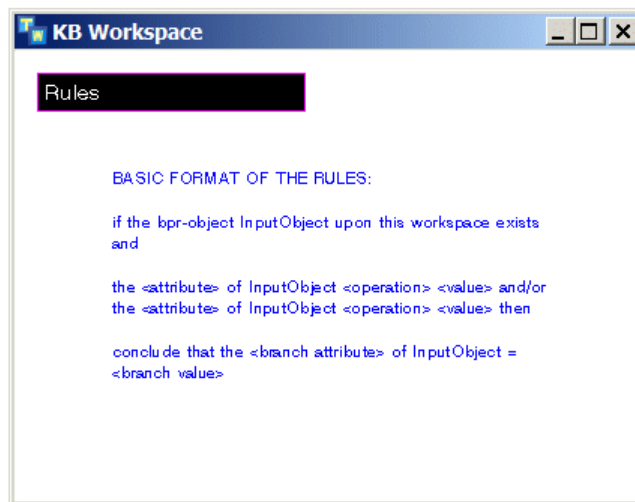
For clarity, we recommend that you create one rule for each possible value of the Branch Value, rather than specifying an initial value for the attribute in the class definition of the work object.

For debugging purposes, we also recommend that you always create an output path whose Branch Value is otherwise so that if no input work object matches the criteria specified in the rule, the block can still process the work object.

To branch work objects based on rules that set the attribute value:

- 1 Follow steps 1 through 5 under [Branching Based on Attribute Value](#).
- 2 Choose Create Rules on the Branch block to display the detail.

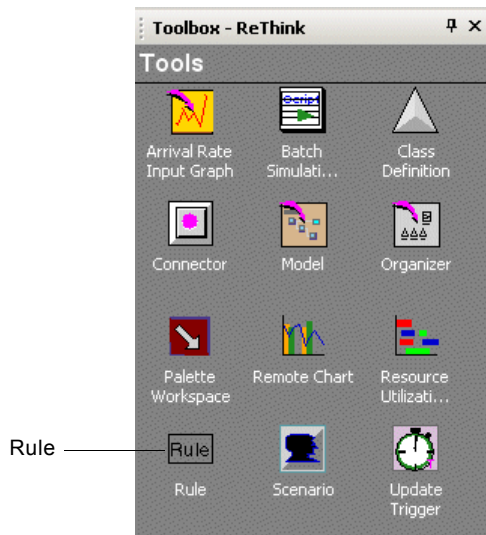
The detail contains a template for creating the rule, which must have this format:



For example, you might conclude the eligibility attribute of a contact-form work object to be eligible by testing the income and number-of-dependents attributes. The default value of the eligibility attribute would be ineligible. To do this, you would substitute the following values for these placeholders:

- Substitute income and number-of-dependents for <attribute>.
- Substitute =, /=, >, >=, <, <= for <operation> and whatever value you want to test for <value>.
- Substitute and/or with the appropriate conditional expression.
- Substitute eligibility for <branch attribute>.
- Substitute the symbol ELIGIBLE for <branch value>.

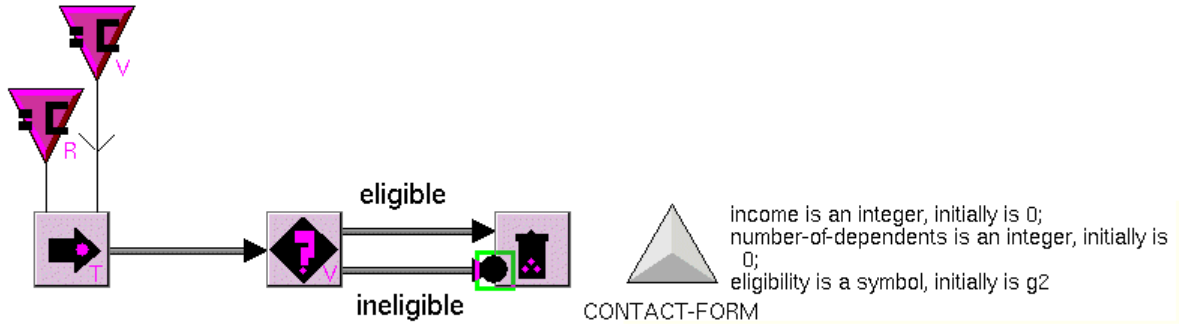
- 3 Display the Tools palette of the ReThink toolbox:



- 4 Select a rule and place it on the Rules detail of the Branch block.
- 5 Choose edit on the rule to display the Text Editor.
The Text Editor guides you through the syntax of the rule by providing prompts below the type-in area, which you can click to enter into the rule.
- 6 Create the rule by clicking the prompts and typing specific attribute names and values.
- 7 Once you have created the rule, delete the free text that describes the format of the rule.
By default, the block waits the specified Rules Wait Interval of 0.05 seconds before proceeding with execution, which should ensure that the rule is finished executing.
- 8 To proceed with execution immediately without waiting, disable the Use Rules Wait Interval option, as needed.

The following example shows a simple model that branches contact-form work objects based on rules. The model feeds the income and number-of-dependents into the contact-form. The Branch block tests each attribute according to rules and concludes the value of eligibility. When eligibility is eligible, the object flows onto the output path whose Branch Value is eligible, and when eligibility is ineligible, the object flows on the other output path.

Number of Dependents



Contact Form [X]

General Utilization Cost Animation User

Eligibility:

Income:

Number Of Dependents:

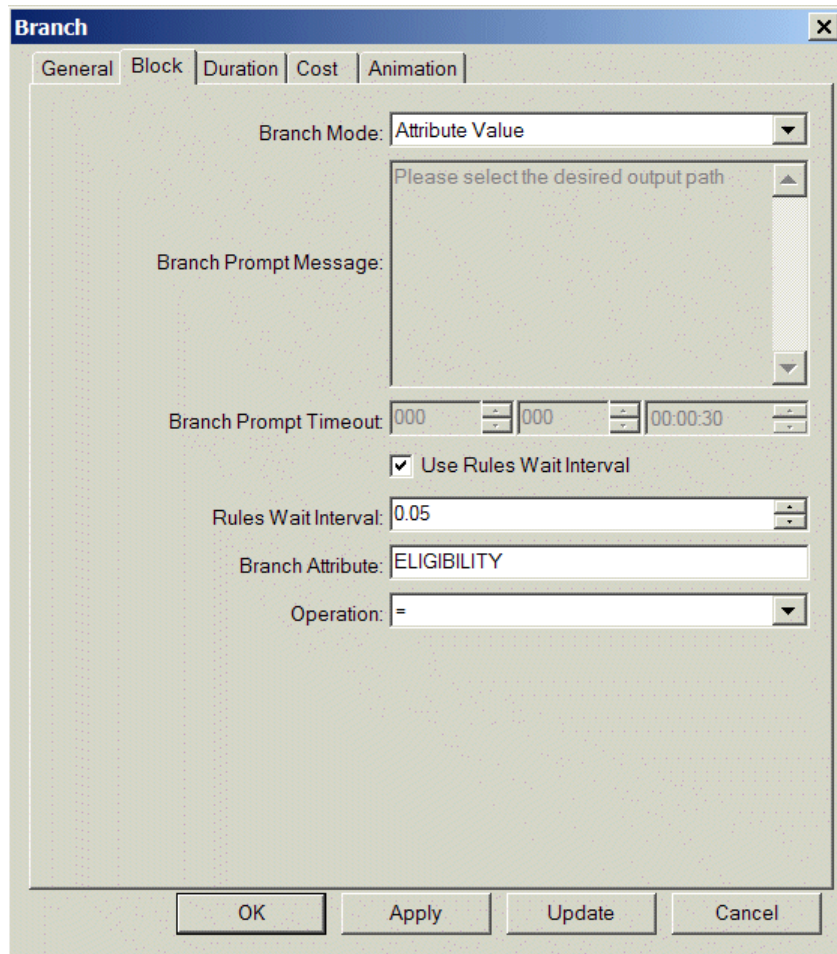
OK Apply Update Cancel

Here are the two rules that conclude values for the eligibility of a contact-form. The first rule concludes a value of **eligible**, while the second rule concludes a value of **ineligible**.

```
if the bpr-object InputObject upon this workspace exists and  
the income of InputObject < 20000 or  
the number-of-dependents of InputObject >= 2 then  
conclude that the eligibility of InputObject = the symbol ELIGIBLE
```

```
if the bpr-object InputObject upon this workspace exists and  
the income of InputObject >= 20000 or  
the number-of-dependents of InputObject < 2 then  
conclude that the eligibility of InputObject = the symbol INELIGIBLE
```

Here is the Block tab of the properties dialog for the Branch block, which branches work based on the attribute named eligibility:

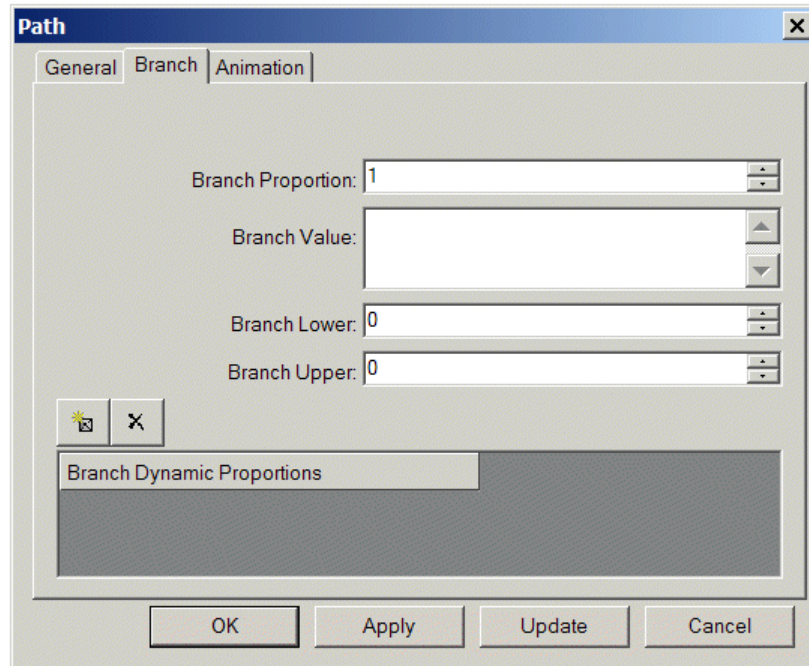


To view the rule on the detail of a Branch block:

→ Choose Show Rules.

Path Attributes that Pertain Only to Branching

The Branch tab of the properties dialog of the output path of a Branch block contains attributes that pertain only to branching:



The attributes of a path that pertain only to branching are:

Attribute	Description
Branch Proportion	<p>In proportion mode, the proportion of the work objects that flow onto this output path, as follows:</p> <ul style="list-style-type: none">• If the output path proportions add to 1, the value of this attribute represents a percentage.• If the output path proportions add to a number greater than 1, the value of this attribute represents a fraction of the total. <p>See Branching Based on Proportion.</p>
Branch Value	<p>In attribute value mode, the value the Branch block uses when it tests the Branch Attribute, using the Branch Attribute Operation. The block behaves as follows:</p> <ul style="list-style-type: none">• If the attribute value meets the criteria, the work object flows onto this output path.• If no attribute value meets the criteria, the work object flows onto the output path whose Branch Value is the symbol otherwise. <p>See Branching Based on Attribute Value.</p>
Branch Lower	<p>In attribute value mode, when the Branch Attribute Operation is Range, this attribute is the lower end of the range that the Branch block uses when it tests the Branch Attribute. If the attribute value meets the criteria, the work object flows onto this output path. See Branching Based on a Range of Values.</p>

Attribute	Description
Branch Upper	In attribute value mode, when the Branch Attribute Operation is Range, this attribute is the upper end of the range that the Branch block uses when it tests the Branch Attribute. If the attribute value meets the criteria, the work object flows onto this output path. See Branching Based on a Range of Values .
Branch Dynamic Proportions	In dynamic proportion mode, the list of proportions the block uses each time a work object loops around the path. The Branch block uses each subsequent proportion in the list each time a work object loops back through the Branch block. See Branching Based on a Dynamic Proportion .

Specific Attributes

Branch: Loop

General | Block | Duration | Cost | Animation

Branch Mode: Proportion

Branch Prompt Message: Please select the desired output path

Branch Prompt Timeout: 000 000 00:00:30

Use Rules Wait Interval

Rules Wait Interval: 0.05

Branch Attribute:

Operation: =

OK Apply Update Cancel

The specific attribute of the Branch block is:

Attribute	Description
Branch Mode	Specifies how the block branches work objects. The options are: Proportion, Dynamic Proportion, Type, Prompt, Attribute Value, and Custom. The default value is Proportion.

The mode-specific attributes of the Branch block are:

Mode	Attribute	Description
Proportion	N/A	N/A
Dynamic Proportion	N/A	N/A
Type	N/A	N/A
Prompt	Branch Prompt Message	The message ReThink displays when it prompts for the output path. The default value is Please select the desired output path.
	Branch Prompt Timeout	The period of time the user has to select the output path on which a work object should pass. The default value is 30 seconds.
Attribute Value	Use Rules Wait Interval	Whether to use the Rules Wait Interval. Disable this option to proceed with execution immediately after the rule executes without necessarily waiting for the rule to complete.
	Rules Wait Interval	The amount of time to wait for rules on the rules workspace to execute before proceeding with execution. The default value is 0.05.
	Branch Attribute	An attribute of a work object that determines the path on which the work object passes. You can use dot notation to refer to the attribute of a subobject, for example, <code>my-subtable.my-attr</code> .
	Operation	The operation the block uses to make the comparison. The default value is <code>=</code> .
Custom	Branch Procedure Name	See Customization Attributes .

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for the Branch block are:

Menu Choice	Description
Create Rules	In attribute value mode, creates a workspace for creating rules that specify logic for concluding the value an attribute of a work object.
Show Rules	In attribute value mode, this menu choice shows the rules that were created on the rules workspace.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Customization Attributes

The customization attributes available in Developer mode for the Branch block are:

Attribute	Description
Branch Procedure Name	When Branch Mode is Custom , specifies the procedure name that determines how the block branches work objects. The default value is <code>bpr-branch-proportion</code> .

You can customize any of the default branch procedure to branch work objects, based on some other criteria. For more information, see the *Customizing ReThink User's Guide*.

BRMS Task



The BRMS Task block invokes Business Rules Management System (BRMS) rules on the work objects it processes. BRMS rules provide a mechanism for easily editing, organizing, analyzing, and executing business rules. You define business rules for a class of business objects in a given category. A business rule consists of one or more conditions and actions, which you define interactively based on the business object class. You invoke rules programmatically by invoking all rules in one or more categories for a set of objects.

You can use BRMS to create individual business rules or to create more complex sets of related business rules in a decision table, which tests conditions and take actions on attributes defined on the same business object class. You can edit these decision tables in G2 or Excel.

BRMS rules and decision tables are described fully in the *Business Rules Management System User's Guide*.

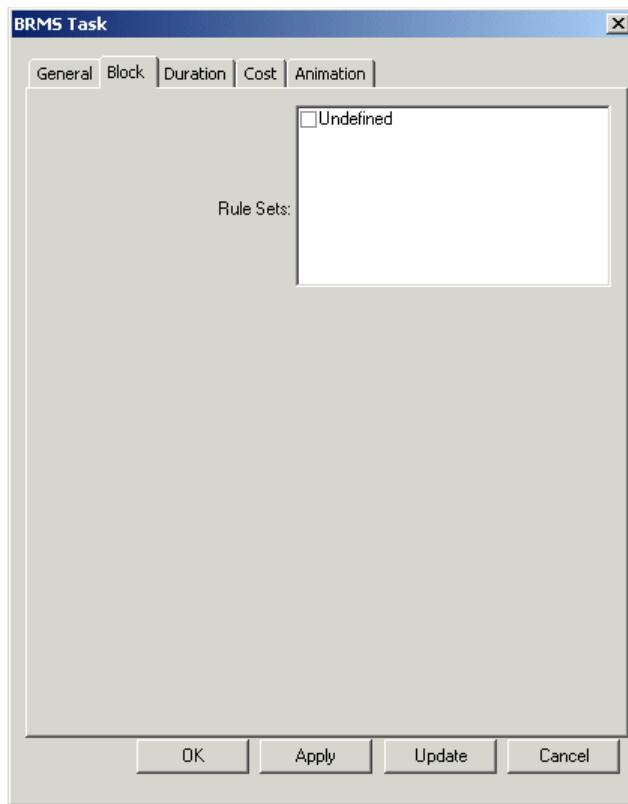
Configuring the BRMS Rules to Invoke

To configure the BRMS rules to invoke:

- 1 Create and configure the BRMS rules you want the BRMS Task block to invoke.
For details, see the *Business Rules Management System User's Guide*.
- 2 Show the properties dialog of the BRMS Task block and click the Block tab.
- 3 Configure the categories of Rule Sets the block should invoke.

When the work object arrives at the block, the specified rules are invoked for associated objects.

Specific Attributes



The specific attributes of the BRMS Task block are:

Attribute	Description
Rule Sets	The rule categories to invoke.

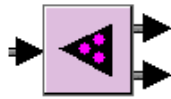
For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The BRMS Task block has no specific menu choices.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Copy



A Copy block creates multiple copies of an object. The Copy block has a single input by default, and it can have any number of outputs. When a Copy block receives its inputs, it outputs as many copies as it has output paths.

The block typically has a single input path. When the block processes the work object, it copies the original work object and passes the original and the copies. However, you can also use the Copy block with multiple input paths, in which case it creates copies of each work object sequentially as it arrives at the block; it does not synchronize the input work objects.

The copied objects have the same attributes and values as the original.

If the work object that the Copy block copies contains an item-list, the copy of the work object contains the same list of items as the original, as of the moment the object is copied.

Creating Copies of a Work Object

To create copies of a work object:

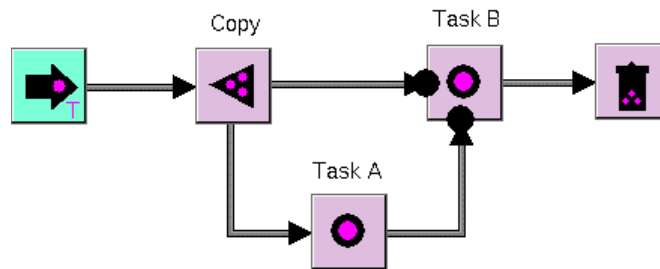
- ➔ Create one output stub for each copy of the work object you want, including the original.

For example, if you want one copy, use the two default output paths on the block, one for the original and one for the copy.

Note Use the default path type for the output paths of a Copy block so that the output work objects are the same type as the input work objects. Do not configure their type to be different from the input path type.

You can use a Copy block with a Task block to synchronize the original work object and its copies, as the following model shows. The Copy block creates a single copy. Task A performs specialized processing on the copy. Task B receives the original on its left input path and the copy on its bottom input path. Task B waits until it receives the copy from Task A before it processes, thereby

synchronizing the original with its copy. Task B then deletes the copy and passes the original on its single output path.



Identifying the Original Output Path

If your model allocates a resource before a Copy block and deallocates it after the Copy block, you need to identify the output path for the original work object of the Copy block. Similarly, if you are associating objects before a Copy block and reconciling them downstream, you also need to identify the output path for the original work object.

If you do not identify the output path for the original work object, you cannot guarantee the work object that caused the resource to be allocated will be the same work object that causes the resource to be deallocated downstream. Similarly, you cannot guarantee that the correct objects will be reconciled.

For more information on...

See...

Allocating and deallocating resources manually

[Allocating the Same Resource for Multiple Sequential Steps.](#)

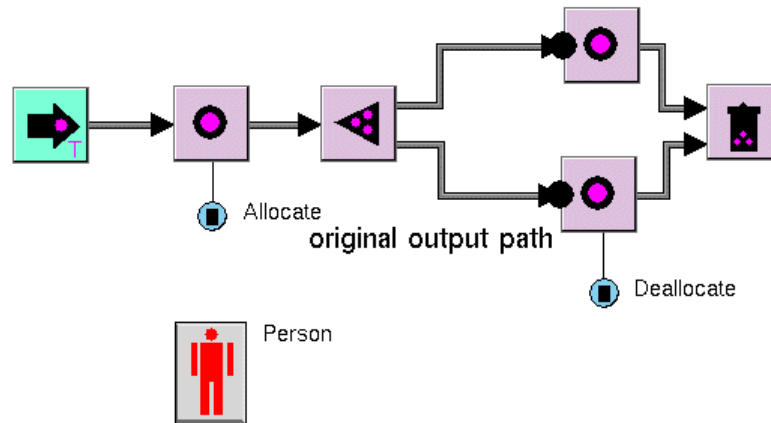
Associating and reconciling objects

- [Associate block.](#)
- [Reconcile block.](#)

To identify the original output path:

- 1 Allocate a resource before the Copy block.
- 2 Deallocate a resource after the Copy block.
- 3 Choose the Choose Original Output Path menu choice on the Copy block to identify the output path for the original work object, which goes to the block that deallocates the resource.
- 4 Choose Select on the output path on which the original work object should be output.

This example shows a model in which you must identify the original output path of the Copy block:



Adding Copies to Associations

If the work object that the Copy block copies is associated with other work objects in the model, you can choose to associate the copy with the other work objects as well. By default, the Copy block does *not* add copies to associations.

To add copies to associations:

- ➔ On the Block tab of the properties dialog, click the Add to Associations option on.

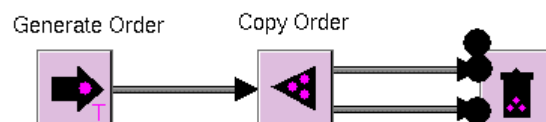
Configuring the Number of Objects to Create

You can generate several copies of new work objects each time the Copy block processes, rather than a single copy. You specify the number of copies to generate for each activity on each output path that does not carry the original work object.

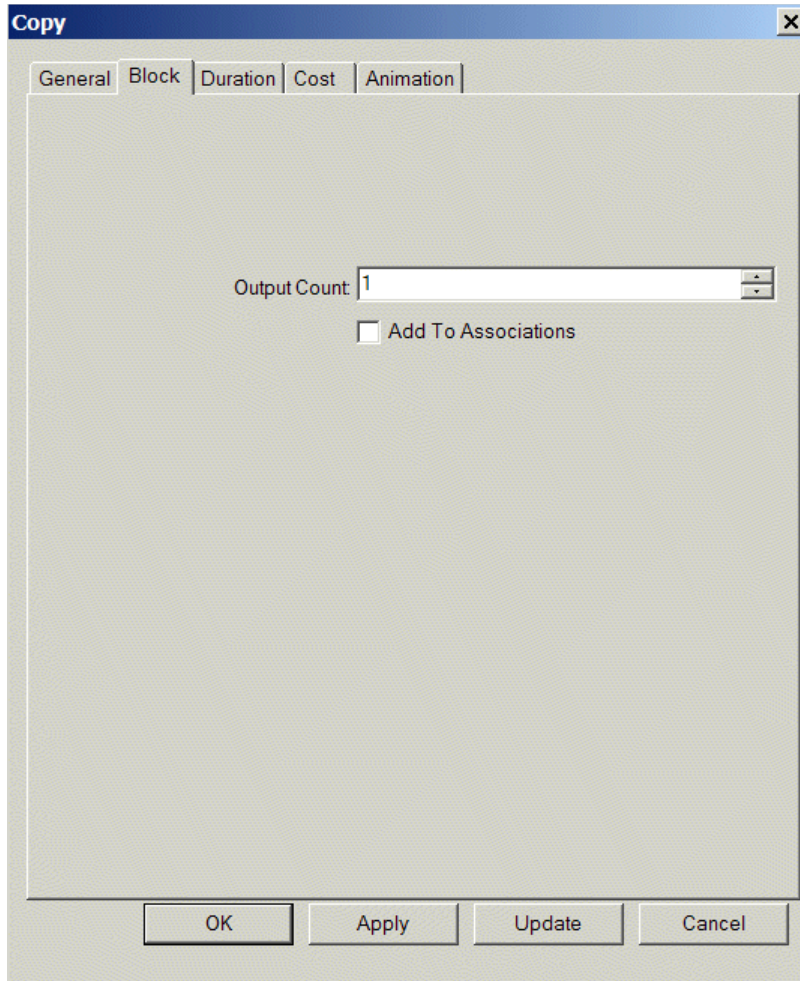
To configure the number of copies to create on each path output path:

- ➔ On the Block tab of the properties dialog, configure the Output Count to be the number of objects to create.

For example, this diagram generates two copies of the input work object, instead of just one. One of the copies has been moved to show both objects.



Specific Attributes



The specific attributes of the Copy block are:

Attribute	Description
Output Count	The number of work object copies to create for each activity and on each output path that does not carry the original work object.
Add to Associations	Whether the copies are associated with the same work objects as the original.

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for the Copy block are:

Menu Choice	Description
Choose Original Output Path	Identifies the output path that carries the original object, which the Copy block copies. Choose Select on the input path that carries the original object to select the path.
Show Original Output Path	Puts an indicator arrow next to the chosen original output path.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Customization Attributes

The customization attributes available in Developer mode for the Copy block are:

Attribute	Description
Copy Item Lists	A truth-value that specifies whether the block should copy the contents of attributes of work objects that contain item-lists. The default value is true .
Copy Item List Items	A truth-value that specifies whether the block should copy the items within item-list attributes of work objects. The default value is true .

For backward compatibility, you can set these attributes to **false**. For more information, see the *Customizing ReThink User's Guide*.

Copy Attributes



The Copy Attributes block copies attribute values from one work object to another. The block copies all of the user-defined attributes that are common to both objects.

The Copy Attributes block takes two input objects, one of which you identify as the source object and the other of which is, by default, the destination object. The block passes the source and destination objects as outputs.

You supply values for the user-defined attributes to be copied by:

- Using a feed.
- Creating an external file of objects with user-defined attributes and values, and use the Source block in file mode to generate objects for the model from the file.

Copying Attributes from One Object to Another

To copy attributes from one object to another, you typically set up a class hierarchy with classes that define the same user-defined attributes.

For example, suppose you were modeling a sales process that generates sales orders, processes the sales orders, generates licenses, then copies all the relevant data from the sales order to the license. The sales order might contain a customer name, company name, order number, and part number. Once the model creates the license, the Copy Attributes block might copy these attributes from the sales order to the license. See the model following the series of steps for an example.

To create a class hierarchy that supports this type of information system, create a superior class that contains the attributes common to both sales orders and licenses. The class definition for sales orders and licenses should both inherit from the same superior class. The sales order and license class definitions can define additional attributes, as needed.

Once you have created the common attributes, you must identify which object is the source. The Copy Attributes block automatically copies all common attributes from the source to the target as soon as both objects arrive at the block. Thus, the Copy Attributes block synchronizes its inputs by waiting to process until it receives all input work objects.

To copy attributes from one object to another:

- 1 Create class definitions for two objects, which share one or more attributes.

Typically, you create a superior class that defines the common attributes, which both classes inherit.

For information on creating work object class definitions, see [Creating a New Class of Work Object](#).

- 2 Configure the path types of the two input paths to the Copy Attributes block.
- 3 Choose the Choose Original Input Path menu choice on the Copy Attributes block, then choose Select on the input path that carries the source object to select the path.

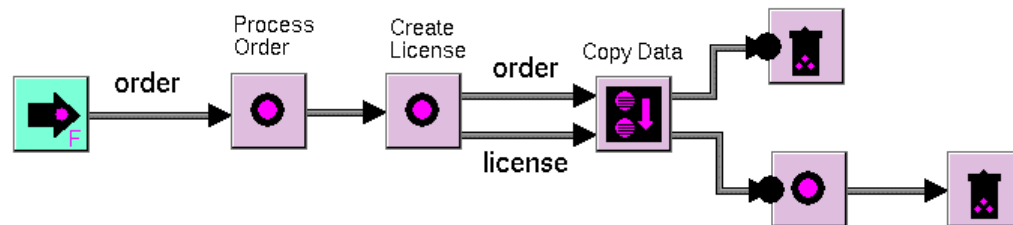
When selected, the path turns magenta, indicating that you have selected it.

- 4 Configure the path types of the two output paths of the Copy Attributes block.

Note If you use the default path type for the output paths of the block, ReThink will process both input work objects on the same output path.

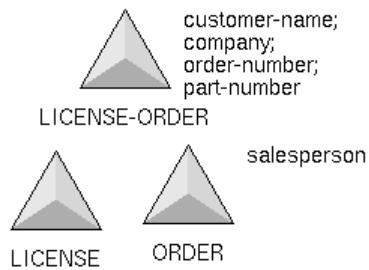
For a general explanation of how ReThink determines the output path type for blocks with multiple output paths, see [Determining the Output Path Based on Its Type](#).

Here is a model that generates and processes orders, creates licenses, copies data from the orders to the licenses, then continues to process the licenses, before deleting the orders and licenses:



The `order` and `license` classes inherit from the `license-order` class, which defines the common attributes. The class-specific attributes appear as attribute displays in the following figure. The `order` class also defines a class-specific attribute named `salesperson`, whose value is given in the source file, along with the common

attributes. Note that it is not necessary to declare data types because ReThink determines the type when it creates the object from the file.



The model generates sales objects from a file, using the Source block's file mode. The external file contains a list of object types, followed by a list of user-defined attribute names and values, separated by carriage returns. This file contains four objects.

```
order, customer-name, "Susan Shore", company, "Shore Sweets", order -  
number, 12345, part-number, "1122-A", salesperson, "ASD"  
order, customer-name, "Linda Longly", company, "Longly Lights", order -  
number, 23456, part-number, "1123-B", salesperson, "FGH"  
order, customer-name, "Allison Albright", company, "Albright  
Associates", order-number, 34567, part-number, "1122 -  
A", salesperson, "JKL"  
order, customer-name, "Vicky Weis", company, "Weis Windows", order -  
number, 45678, part-number, "1122-A", salesperson, "ASD"
```

For information on how to create this external file, using a Source block, see [Generating Work Objects from an External File](#).

To view the attributes of an object:

➔ Run the simulation in step mode and display the properties of a work object.

Here is the User tab of the properties dialog for the first sales order the model produces:

The screenshot shows a dialog box titled "Order" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Utilization", "Cost", "Animation", and "User". The "User" tab is selected. The dialog contains the following fields:

- Company: Shore Sweets
- Customer Name: Susan Shore
- Order Number: 12345
- Part Number: 11227A
- Salesperson: ASD

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Attributes obtained from source file.

Here is the User tab of the properties dialog for the license once the license and the sales order pass through the Copy Attributes block:

The screenshot shows a dialog box titled "License" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Utilization", "Cost", "Animation", and "User". The "User" tab is selected. The dialog contains four input fields, each with a label and a value:

- Company: Shore Sweets
- Customer Name: Susan Shore
- Order Number: 12345
- Part Number: 1122?A

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Common attributes copied from sales order.

Specific Attributes

The Copy Attributes block has no specific attributes.

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for the Copy Attributes block are:

Menu Choice	Description
Choose Original Input Path	Identifies the input path that carries the object from which the block copies the attributes. Choose Select on the input path that carries the source object to select the path.
Show Original Input Path	Puts an indicator arrow next to the chosen original input path.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Insert



The Insert block inserts objects into an item-list attribute of a container. A container is an object of the `bpr-container` class, which is a built-in ReThink class that defines an item-list attribute named `container-list`, any subclass of `bpr-container`, or any user-defined class that defines an item-list attribute.

You use the Remove block to remove the inserted objects from the other object downstream in the process. For example, you might want to add line item objects to an order and remove those line items downstream in a process.

You can insert objects one at a time into the other object, or you can insert a group of objects all at once. When you insert objects one at a time, you create a loop that repeatedly passes the object around the process to the Insert block until some criteria is met. Typically, you use a Branch block to create the loop.

Understanding the Paths of an Insert Block

The Insert block has two input paths and a single output path, as follows:

Path	Description
Container input path	Passes the container object with the item-list attribute.
Object input path	Passes the objects to insert into the item-list attribute of the container.
Output path	Passes the container with the objects inserted.

To configure the Insert block, you must identify which input path passes the container. You do this by specifying the path types for each input path and interactively identifying the container input path. You specify the attribute of the container that defines the item-list. You do not need to identify the output path that passes the objects in the container.

Configuring the Insert Mode

By default, the Insert block inserts a single object into the beginning of the item-list of the container each time the block processes.

You can control whether the block inserts objects one at a time or all at once. When you insert objects one at a time, you can control whether the block inserts

the objects at the beginning or at the end of the list first. You control this behavior by using the Mode attribute of the block as follows:

Mode	Description
First	Inserts objects one at a time into the container list attribute, inserting the object at the beginning of the item-list of the container. This mode is the default.
Last	Inserts objects one at a time into the container list attribute, inserting the object at the end of the item-list of the container.
All	Inserts all the objects into the item-list at one time.

The combination of the Insert and Remove block's Mode specifications determines whether the model uses a FIFO (first in, first out), FILO (first in, last out), LIFO (last in, first out), or LILO (last in, last out) queuing algorithm.

For information on removing objects from a container, see the [Remove block](#).

Inserting a Single Object Into a Container

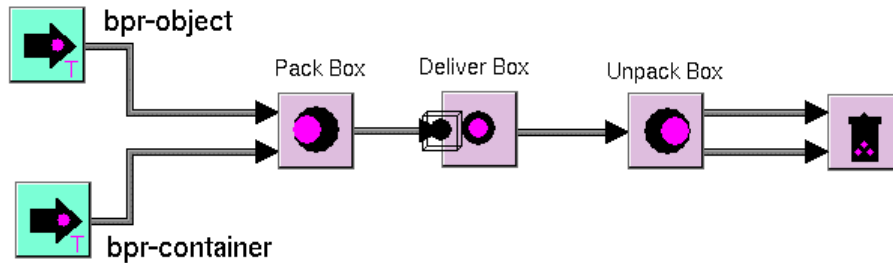
By default, the Insert block inserts a single object into the item-list attribute of the container before passing the container to the downstream block.

To insert an object into a container:

- 1 Display the properties dialog for one input path of the Insert block and configure the Type to be `bpr-container` or a subclass of `bpr-container`.
This built-in ReThink class defines a `container-list` attribute, whose value is an instance of an item-list. The block stores the objects in the `container-list` attribute of this object.
- 2 Choose the Choose Container Input Path menu choice to select the input path that carries the container object, then choose Select on the path that carries the container to select it.
When selected, the path turns magenta indicating that you have selected it.
- 3 On the Blocks tab of the Insert block, configure the Container List Attribute to be the item-list attribute of the container, whose default is the `container-list` attribute of the `bpr-container` class.

The Insert block specifies a Mode of `First`, by default, which inserts one object at a time into the beginning of the container list attribute of the container.

Here is a simple example that packs an object into a container using the Insert block, delivers the box using a Task block, and then unpacks the object from the container using a Remove block. The top output path of the Unpack Box block passes the part that the block removes from the box, and the bottom output path passes the empty box.



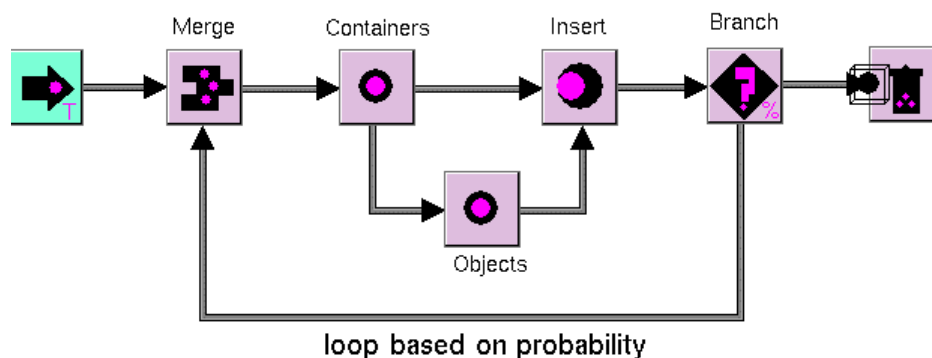
For information on how to remove objects, see the [Remove block](#).

Inserting Objects into the Container By Looping

By default, the Insert block inserts a single object into the container each time the block processes; the block inserts a single object into the container and passes the container downstream. One way of inserting multiple objects into the container is by creating a loop in your diagram that continues to insert objects into the container until a certain condition is met. For example, you might use a Branch block to pass the container around the process based on some probability. Each time the container arrives at the Insert block, a new object is inserted.

To insert objects into the container by looping, specify the Mode attribute of the Insert block to be **First** (the default) or **Last**.

The following example illustrates how to use a Branch block, which loops containers around the process based on probability, in conjunction with the Insert block to insert objects into a container:



The Containers task creates the container object, and the Objects task creates the objects to insert into the container. The Insert block inserts a single object from the

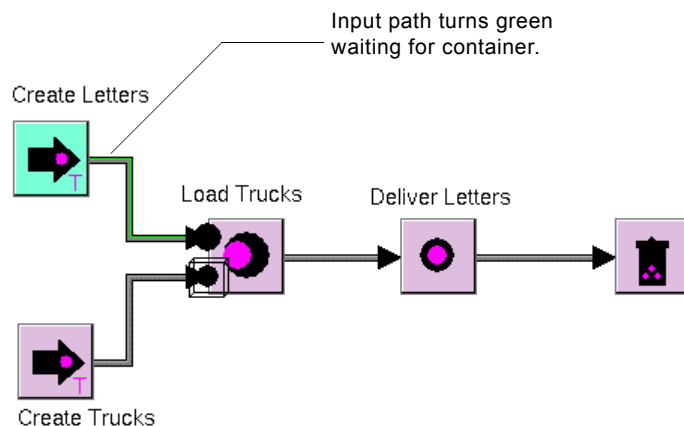
object path into the container from the container path. The resulting container passes to a Branch block, which loops back to a Merge block, based on probabilities. When the container loops around again, the Objects block creates another object, which gets inserted into the container.

Inserting Objects Into the Container All at Once

Sometimes you want to insert a group of objects all at once into a container. This might happen when containers and the objects to insert arrive at the block at different time intervals. For example, when inserting mail into a mail truck, you might insert all the letters that have arrived so far each time a mail truck arrives. To insert objects all at once into a container, you specify the Mode attribute as **All**.

This example shows a mail delivery process in which letters arrive on average once every five minutes, and mail trucks arrive once every hour. Each time a letter arrives at the Load Truck task, it waits on the input path for a truck. The input path turns green indicating that the block is waiting for its other input. When a mail truck arrives at the Load Truck task, the block inserts all the letters that have been waiting on the input path into the truck.

In this model, the container is a truck, the objects to insert into the container are letters, and the Load Truck task is an Insert block:



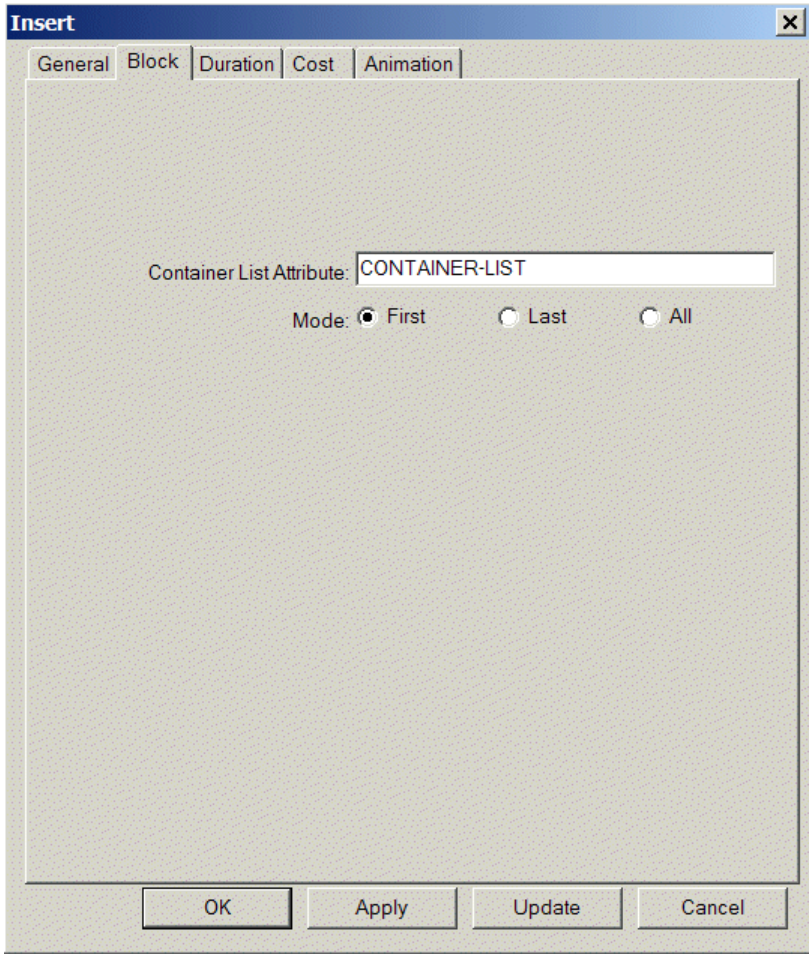
Showing Work Objects in the Container

To show work objects in the container:

- ➔ Run the simulation in step mode and choose Snapshot Container on a container.

ReThink displays a workspace with all the work objects in the container, similar to the workspace you see when you use Snapshot Queue on a path with work backups.

Specific Attributes



The specific attributes of the Insert block are:

Attribute	Description
Container List Attribute	<p>The item-list attribute of the container in which the block inserts objects. The default value is the <code>container-list</code> attribute of a <code>bpr-container</code>.</p> <p>You can use dot notation to refer to the attribute of a subobject, for example, <code>my-subtable.my-attr</code>.</p>
Mode	<p>Specifies how the block inserts objects into the container. The options are:</p> <ul style="list-style-type: none"> • First – Inserts objects one at a time, with each new object first in the list, the default. • Last – Inserts objects one at a time, with each new object last in the list. • All – Inserts a group of items simultaneously into the list.

For information on the common block attributes, see [Common Attributes of Blocks](#).

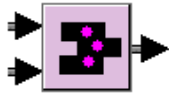
Specific Menu Choices

The specific menu choices for the Insert block are:

Menu Choice	Description
Choose Container Input Path	<p>Specifies the input path that carries the container. Choose Select on the input path that carries the container to select the path.</p>
Show Container Input Path	<p>Puts an indicator arrow next to the chosen container input path.</p>

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Merge



The Merge block merges multiple streams of work objects into a single stream. For example, you use this block to merge two different streams of work objects for similar processing.

The Merge block can have any number of input paths, but it typically has only one output path. As soon as it receives an object on any one of its input paths, it sends the object to its output path, without waiting for its other inputs.

If the Merge block has more than one output path, the block behaves like a Branch block in type mode. For more information, see [Branching Based on Type](#).

Merging Multiple Streams of Work

You use the Merge block to merge together multiple streams of work of the same type or of different types. The Merge block passes its work objects to its output path as soon as the work object arrives. Thus, the Merge block does not synchronize its inputs, like a Task block with multiple input paths does.

For example, if you have multiple Source blocks that emit different types of objects, you might want to merge the objects together into a single stream to perform similar processing of different types of work, using a single Task block.

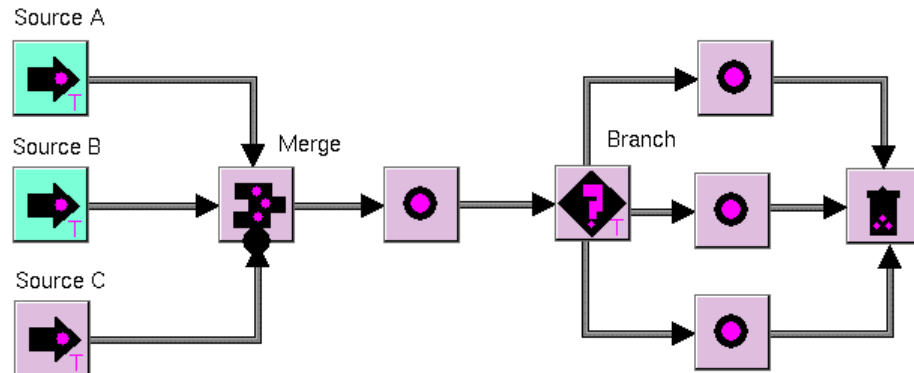
To merge multiple streams of work into a single stream:

➔ Connect as many input stubs to the Merge block as you want to merge.

The Merge block typically has a single output path.

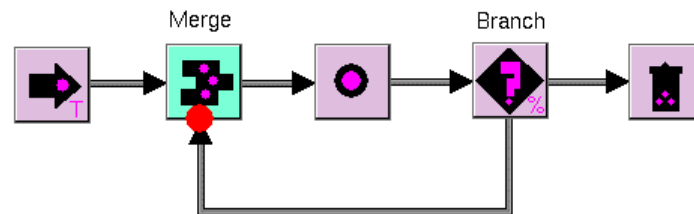
In this example, Source A, B, and C emit different types of work objects, which the Merge block brings together into a single stream. The work objects pass to a task block, which processes them as soon as they arrive. After the task processes the work objects, they flow apart again using a Branch block, which branches the

objects based on their type. The following tasks process each type of work object independently before deleting them.



Merging Work That Loops Around a Process

You often use a Merge block to merge work that originates upstream in a process with work that loops around the process, using a Branch block:



For information on branching, see [Branch block](#).

Specific Attributes and Menu Choices

A Merge block has no specific attributes or menu choices.

For information on the common block attributes, see [Common Attributes of Blocks](#).

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Reconcile



The Reconcile block matches together associated objects. For example, you might use this block to match associated orders and invoices that have gotten out of sequence in a process. When the associated objects arrive, the block outputs the objects together, each on its own output path.

Unlike other blocks with multiple input paths that synchronize their inputs, the Reconcile block does not insert objects waiting to be reconciled into the path queue of the block. This means that the input path does *not* turn green and the wait time of the path does *not* include the time the block spent waiting for the object to be reconciled. Instead, the Reconcile block stores these objects in an internal attribute of the block.

Reconciling Individual Associated Objects

You use the Reconcile block to match individual associated objects. You associate the objects upstream in a process with an Associate block.

By default, the Reconcile block determines the number of associated work objects to reconcile by looking at the number of input paths. Thus, if the block has two input paths, it must wait until two associated work objects arrive at the block before it reconciles them.

To reconcile individual associated objects:

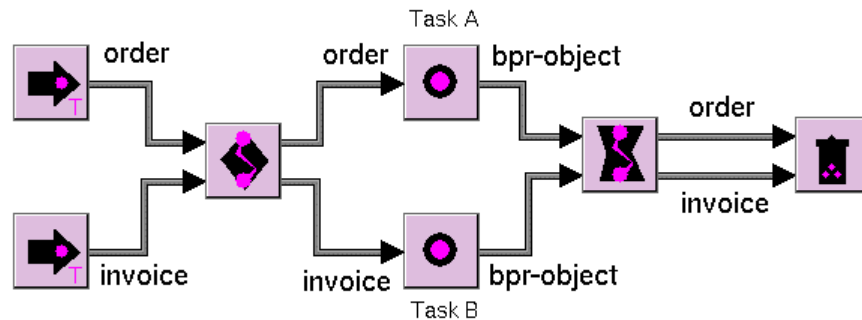
- 1 Associate objects upstream in the process by using an Associate block.
See [Associate block](#).
- 2 On the Block tab of the properties dialog, name the association in the Association Name attribute of the Reconcile block.

This is the value of the Association Name attribute you created upstream in a process with the Associate block.
- 3 Configure the input and output path types of the Reconcile block.

Note If you use the default path type for the output paths of the block, ReThink will pass both input work objects on the same output path.

For a general explanation of how ReThink determines the output path type for blocks with multiple output paths, see [Determining the Output Path Based on Its Type](#).

For example, you might associate an order and an invoice upstream in a process then pass each separately to Task A and Task B, which might have different durations. Once the individual processing is complete, the Reconcile block reconciles the order and invoice, matching associated objects based on the association name.



Reconciling All Objects

By default, the Reconcile block reconciles each individual associated object as it arrives on an input path of the block.

If you have used the Associate block to associate multiple work objects, you can use the Reconcile block to reconcile all objects in the association. When the Reconcile block reconciles all associated objects, it determines the number of associated work objects to reconcile by looking at the named association, instead of the number of input paths. The block must reconcile all associated work objects in the named association. Thus, if the Reconcile block names an association that has four associated work objects, but the block has only three input paths, it must wait until all four associated work objects arrive at the block before it reconciles them.

To reconcile all objects:

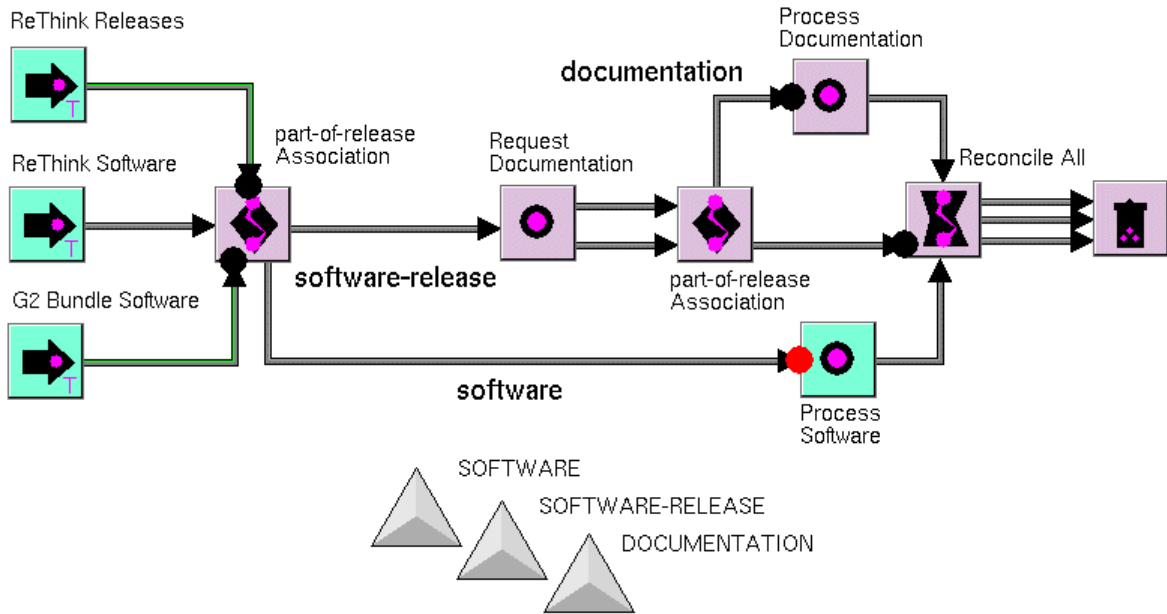
- 1 Use the [Associate block](#) to associate multiple work objects.
You can do this by using the Associate block in Add mode, by enabling the Associate All option, or by using the Associate block in New mode with multiple input paths.
- 2 On the Block tab of the properties dialog, name the association in the Association Name attribute of the Reconcile block.
- 3 Enable the Reconcile All option.
- 4 Configure the input and output path types of the Reconcile block.

The following example models a software release process, which associates and then reconciles a software release, documentation, and two different software components. In the example, the Reconcile block sets the Reconcile All option to be on.

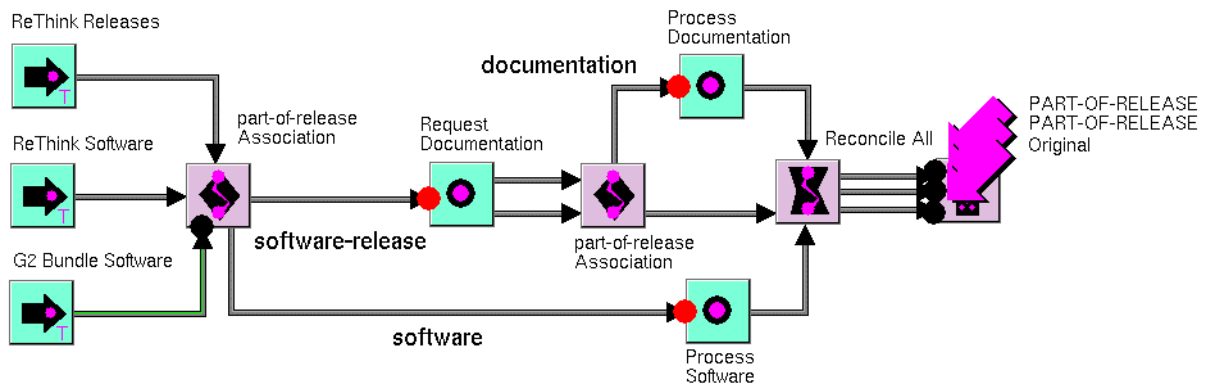
The following figures show two steps in the process:

- 1** The first software release waits on the input path of the Reconcile block for its associated software components and documentation.
- 2** When the associated software components and documentation arrive at the Reconcile block, the block reconciles all the associated work objects in the named association, which includes the software release, the documentation, and both software components. Notice that the software release is still associated with the software and documentation through a **part-of-release** association. In the figure, one of the software components has been moved off the path so both are visible.

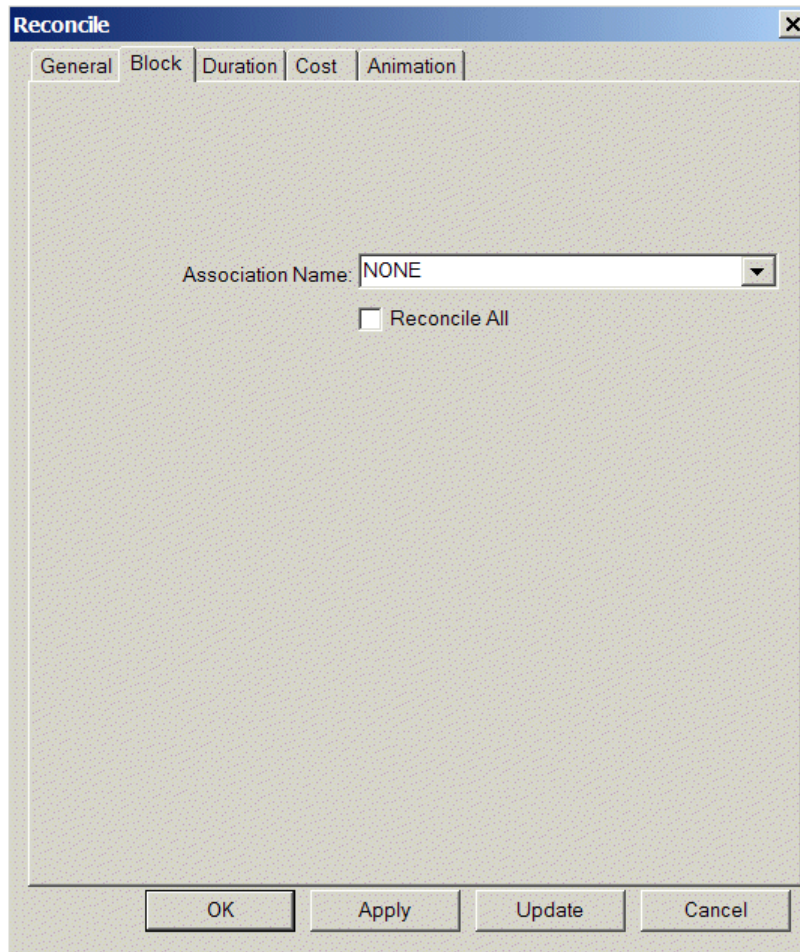
①



②



Specific Attributes



The specific attributes of the Reconcile block are:

Attribute	Description
Association Name	The value that the block uses to determine whether to reconcile two objects.
Reconcile All	When creating an association between multiple work objects, using the Associate block, this option reconciles all associated work objects in the named association. Otherwise, the block reconciles all associated work objects on the input paths.

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The Reconcile block has no specific menu choices.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Customization Attributes

The customization attributes available in Developer mode for the Reconcile block are:

Attribute	Description
Match Procedure Name	Determines the criteria the block uses to reconcile two objects. The default value is <code>bpr-match-by-association</code> , which reconciles objects based on an association name given by the Association Name attribute.

You can customize the match procedure to specify a different criteria for reconciling the two objects. For more information, see the *Customizing ReThink User's Guide*.

Remove



The Remove block removes objects that have been inserted into an item-list attribute of a container. For example, you might add line item objects to an order and remove those line items downstream in a process.

You can remove objects one at a time from the container, or you can remove them all at once. When you remove objects one at a time, you create a loop that continues to pass the container around the process to the Remove block until all the objects have been removed.

You create a container object by using the built-in `bpr-container` class, any subclass of `bpr-container`, or any user-defined class that defines an item-list attribute.

You use the [Insert block](#) or the [Batch block](#) to insert objects into a container.

Understanding the Paths of a Remove Block

The Remove block has one input path and three output paths, as follows:

Path	Description
Input path	Passes the container with the item-list attribute that contains the objects to be removed.
Non-empty container output path	Passes the container when it still has objects in the item-list attribute of the container.
Empty container output path	Passes the container when all the objects have been removed from the item-list attribute of the container.
Object output path	Passes the work objects that the block removes from the item-list attribute of the container.

To configure the Remove block, you must identify the non-empty container output path and the empty container output path. You do this by specifying the path types for each output path and interactively identifying each of the output container paths. You also specify the attribute of the container that defines the item-list. You do not need to identify the path that passes the objects in the container.

Configuring the Remove Mode

By default, the Remove block removes a single object from the beginning of the item-list of the container each time the block processes. When combined with the default behavior of the Insert block, which inserts objects at the beginning of the list, this technique of inserting and removing objects models a LIFO (last-in first-out) queuing algorithm.

You can control whether the block removes objects one at a time or all at once. When you are removing objects one at a time, you can control whether the block removes objects from the beginning or from the end of the list first. You control this behavior by using the Mode attribute of the block, as follows:

Mode	Description
First	Removes objects one at a time from the container list attribute, removing the first object in the item-list first. This is the default.
Last	Removes objects one at a time from the container list attribute, removing the last object in the item-list first.
All	Removes all the objects from the item-list at one time.

The combination of the Insert and Remove block's Mode specifications determines whether the model uses a FIFO (first in, first out), FILO (first in, last out), LIFO (last in, first out), or LILO (last in, last out) queuing algorithm.

For information on inserting objects into a container, see [Insert block](#).

Removing Objects from the Container By Looping

By default, the Remove block removes a single object from the container each time the block processes. By default, the block removes the first object in the item-list from the container. You can also remove objects one at a time by removing the last object first.

To remove objects one at a time from the container, create a loop in your diagram that continues to remove objects from the container until the container is empty.

To remove objects from a container one at a time:

- 1 Insert objects into the item-list attribute of a container by using the Insert block or the Batch block.

For information about how to do this, see the [Insert block](#) or the [Batch block](#).

2 Use one of the output paths of the Remove block to create a loop that connects to an upstream Merge block and configure its path type to be a type of the container.

3 Choose the Choose Nonempty Container Output Path menu choice, then choose Select on the output path you just configured to select the path that carries containers that still contain objects.

This is the path that loops around the process and feeds back into a Merge block.

When selected, the path turns magenta indicating that you have selected it.

4 Configure the path type of a second output path of the Remove block to be a type of container.

5 Choose the Choose Empty Container Output Path menu choice, then choose Select on the output path you just configured to select the path that carries containers that no longer contain any objects.

This is the path that carries the container downstream.

6 Configure the path type of the third output path of the Remove block to be a type of the object that the Remove block removes from the container.

7 On the Block tab of the properties dialog, configure the Container List Attribute attribute of the Remove block to specify the name of the item-list attribute of the container.

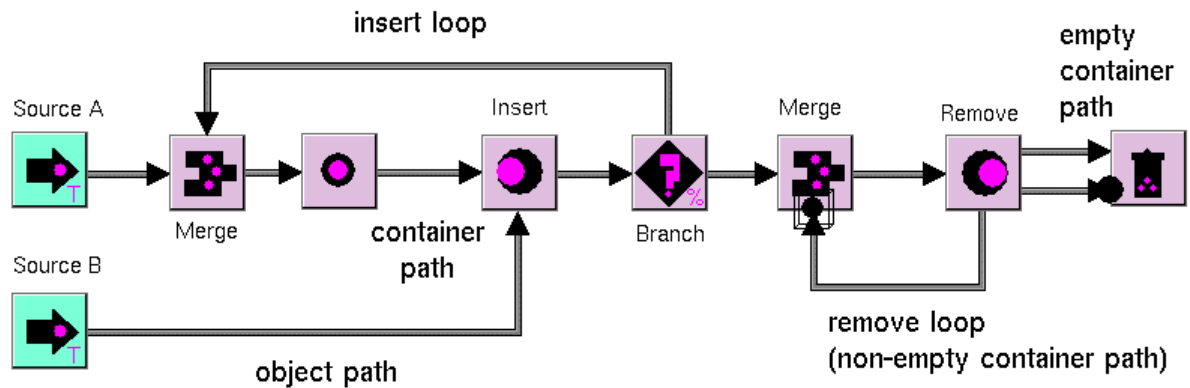
By default, the Container List Attribute refers to the container-list attribute of a bpr-container.

8 Configure the Mode attribute to be First or Last, depending on which objects you want to remove first.

In the following model, Source A produces containers, and Source B produces objects to insert into the container. The Branch block causes the container to loop around the process to insert objects into the container, based on a probability.

The container then passes to the Remove block through a Merge block, which removes objects one at a time from the container. The bottom output path of the Remove block passes the non-empty container, and the right output paths pass

the empty container and the objects that are removed. The labels in the diagram correspond to the preceding steps.



Removing Objects from the Container All at Once

By default, the Remove block removes objects from the container one at a time. You can also remove all the objects at once. When you use this configuration of the Remove block, you do not need to create a loop in your model. Also, the Remove block no longer requires the non-empty container output path.

To remove objects from a container all at once:

- 1 Insert objects into the item-list attribute of a container by using an Insert block or a Batch block.

For information about how to do this, see the [Insert block](#) and the [Batch block](#).

- 2 Delete one of the output paths of the Remove block.

When you remove objects all at once, the Remove block does not have a non-empty container output path.

- 3 Configure the path type of one of the two output paths to name the container.
- 4 Choose the Choose Empty Container Output Path menu choice, then choose Select on the other output path that carries the container to select the output path that carries the empty container.

When selected, the path turns magenta indicating that you have selected it.

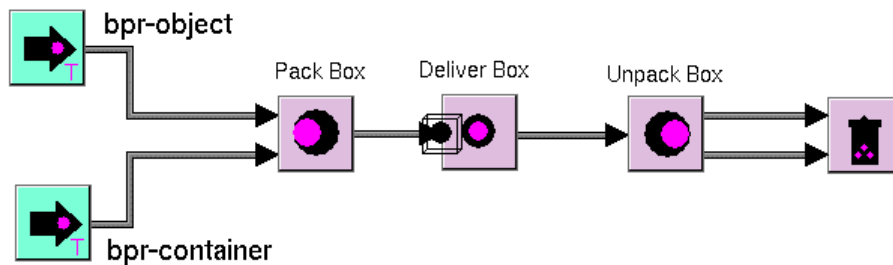
- 5 Configure the path type of the other output path to name the objects that are removed from the container.

- 6 On the Block tab of the properties dialog, configure the Container List Attribute attribute of the Remove block to specify the name of the item-list attribute of the container.

By default, the Container List Attribute refers to the container-list attribute of a bpr-container.

- 7 Configure the Remove block to specify the Mode attribute to be All.

Here is a simple example that packs a part into a box using the Insert block, delivers the box using a Task block, and then unpacks the part from the box using a Remove block. The top output path passes the part that is removed from the box, and the bottom output path passes the empty box. The labels in the diagram correspond to the preceding steps.



Showing Work Objects in the Container

To show work objects in the container:

- ➔ Run the simulation in step mode and choose Snapshot Container on a container.

ReThink displays a workspace with all the work objects in the container, similar to the workspace you see when you use Snapshot Queue on a path with work backups.

Specific Attributes

The image shows a software dialog box titled "Remove" with a close button (X) in the top right corner. The dialog has a tabbed interface with five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Block" tab is currently selected. Inside the dialog, there is a text input field labeled "Container List Attribute:" containing the text "CONTAINER-LIST". Below this field, there are three radio button options for "Mode": "First" (which is selected), "Last", and "All". At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attributes of the Remove block are:

Attribute	Description
Container List Attribute	The item-list attribute of the container from which the block removes objects. The default value is the <code>container-list</code> attribute of a <code>bpr-container</code> . You can use dot notation to refer to the attribute of a subobject, for example, <code>my-subtable.my-attr</code> .
Mode	Specifies how the block removes objects from the container. The default value is <code>First</code> , which removes the first object in the list first. The other options are <code>Last</code> , which removes the last object in the list first, and <code>All</code> , which removes all objects simultaneously from the list.

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for the Remove block are:

Menu Choice	Description
Choose Empty Container Output Path	Specifies the output path that carries empty containers. Choose Select on the output path that carries the empty containers to select the path.
Choose Nonempty Container Output Path	Specifies the output path that carries containers that still have objects in them. Choose Select on the output path that carries containers that still have objects to select the path.
Show Empty Container Output Path	Puts an indicator arrow next to the chosen empty container output path.
Show Nonempty Container Output Path	Puts an indicator arrow next to the chosen non-empty container output path.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

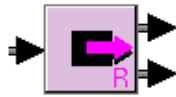
Customization Attributes

The customization attribute available in Developer mode for the Remove block is:

Attribute	Description
Empty Breakpoint	Determines when the container is considered empty and, thus, when it will be passed to the empty output path. The block looks at the contents of the container list when the container first arrives at the block. By default, if the list contains a single object upon arrival, it will pass to the empty container output path. The default is 1.

You can customize when the block passes the container. For more information, see the *Customizing ReThink User's Guide*.

Retrieve



The Retrieve block retrieves objects from a resource pool or from a database. Typically, you add these objects to a pool or database dynamically, using a Store block. For example, you could use this block to retrieve an object from inventory to fill an order, or to model retrieving data from a database.

When you retrieve work objects from a database, ReThink retrieves one object for each record, based on an SQL query. To retrieve work objects from a database, you use the G2 Database Bridge, which provides access to external databases.

For information on how to use the Retrieve block for accessing external databases, see [Retrieving Records from a Database](#).

You specify the output path types to determine on which output paths the input object and the retrieved object flow.

For a general explanation of how ReThink determines the output path type for blocks with multiple output paths, see [Determining the Output Path Based on Its Type](#).

Configuring the Retrieve Mode

A Retrieve block supports the following operating modes:

Retrieve Mode	Description
Random	Retrieves work objects from a pool at random.
Association	Retrieves work objects from a pool, based on the Association Name.
Database	Retrieves work objects from a database.
Attribute Value	Retrieves work objects from a pool, based on an attribute value of the work object, a threshold or range, and a mathematical operation that compares the current value to the threshold or range.
Custom	Allows you to specify a custom procedure for determining how to retrieve work objects from a pool.

Retrieving Objects from a Pool

To retrieve objects from a pool, you must first store the objects in the pool, using the [Store block](#).

You have three options when you retrieve objects from a pool. You can retrieve objects:

- At random.
- Based on an association name.
- Based on an attribute value.

For example, you might want to retrieve an object that is associated with another object as one way of reconciling the associated objects, or you might want to retrieve only those objects with an attribute value within a particular range.

Retrieving Objects from a Pool at Random

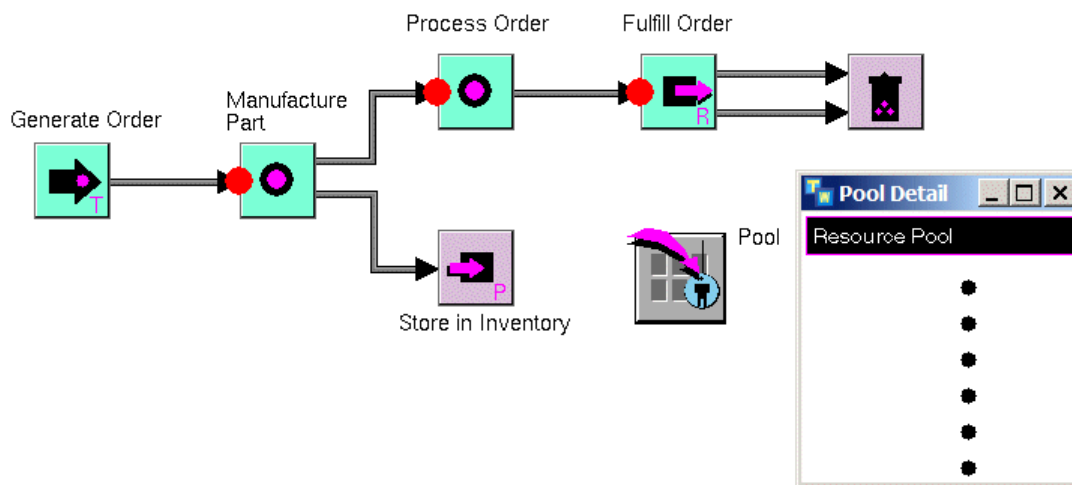
By default, the Retrieve block retrieves objects from a pool at random. You specify the pool from which to retrieve the objects. If the resource from which you are retrieving work objects does not have detail, ReThink displays an error message.

To retrieve objects from a pool at random:

- 1 Store objects in a Resource pool, using the Store block.
For example, you might store the orders in a pool named orders-pool.
See the [Store block](#).
- 2 Configure the output path type of the Retrieve block to name the object to retrieve from the pool.
For example, if you store widgets in the pool, the output path type would be widget.
- 3 Configure the input path type and the other output path type to name an existing object or just use the defaults.
- 4 On the Block tab of the properties dialog, configure the Retrieve Mode of the Retrieve block to be Random.
This mode is the default.
- 5 Choose the Choose Pool menu choice, then choose Select on the resource pool from which the block will retrieve objects to select the pool.
ReThink places an indicator arrow next to the pool to indicate that you have selected it.

Note ReThink adds the Show Pool menu choice to the block, which places an indicator arrow next to the selected pool. ReThink also adds the Show Blocks menu choice to the selected pool, which identifies the blocks that are currently pointing to this pool. These menu choices appear only when the Scenario is active.

This model shows how to store widgets to an inventory and retrieve the widgets from the pool at random to fill an order. The Source block generates orders, which the Manufacture Part block manufactures as widgets and then stores in inventory, using a Store block. The order passes downstream for processing and then retrieves a widget from the pool at random to fill the order. The labels in the diagram correspond to the preceding steps.



Retrieving Associated Objects from a Pool

When you retrieve objects from a pool by association, you specify the pool from which to retrieve the objects and the name of the association. The Retrieve block passes the object and its associated object onto the two output paths of the block.

To retrieve associated objects from a pool:

- 1 Associate objects upstream in a model, using the Associate block.
For example, you might associate an order and an invoice.
See [Associate block](#).
- 2 Store the object with an association name in a resource pool, using the Store block.
For example, you might store the orders in a pool named orders-pool.
See [Store block](#).

- 3 Configure the input path type of the Retrieve block.

The input path type is the class name of the object that is associated with the object in the pool. For example, if you store orders in the pool, the input path type would be `invoice`.

- 4 Configure the output path types of the Retrieve block to reflect the object and its associated object.

For example, the output paths types would be `order` and `invoice`.

- 5 On the Block tab of the properties dialog, configure the Retrieve Mode to be Association.

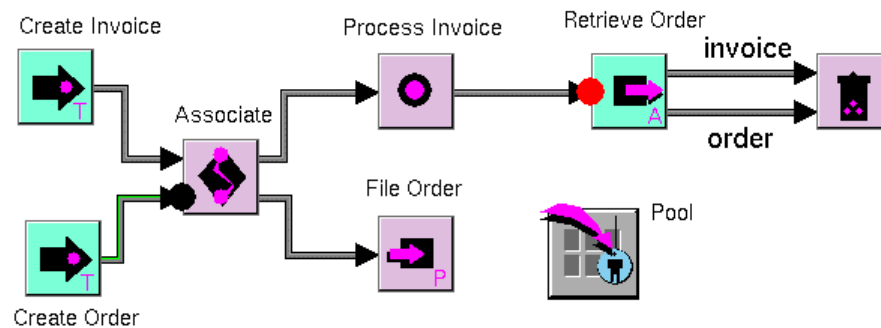
- 6 Configure the Association Name that an Associate block creates as the value of the Association Name attribute of the Retrieve block.

For example, the Association Name might be `order-invoice`.

- 7 Choose the Choose Pool menu choice, then choose Select on the resource pool from which the block will retrieve objects to select the pool.

ReThink places an indicator arrow next to the pool to indicate that you have selected it.

The following figure shows a simple model that associates orders and invoices, stores the orders in a pool, and retrieves the orders from the pool based on the association name. The `order-pool` resource is a pool resource. The labels on the figure correspond to the preceding steps.



Retrieving Objects with a Particular Attribute Value from a Pool

You can retrieve work objects from a pool based on the value of an attribute of the work object. The block tests the current value of the specified attribute of the work objects in the pool against a value specified in the block. You specify the operation that the block uses to perform the comparison in an attribute of the block, such as

equality, greater than, or less than. You can also specify a range operation, which tests against a lower and upper threshold attribute of the block.

To retrieve work objects based on an attribute value of the work object:

- 1** On the Block tab of the properties dialog, configure the Retrieve Mode to be Attribute Value.
- 2** Configure the Retrieve Attribute to name an attribute of the work objects in the pool whose value the block will test.
- 3** Configure the Operation to determine how the block compares the attribute value of the work object to a value of the block.

When you are comparing symbolic values for equality, specify the = operation, the default.

When you are comparing numeric values, specify one of the following operations: =, /=, >, >=, <, <=, or Range.

For information on retrieving work objects based on a range of values, see [Retrieving Based on a Range of Values](#).

- 4** Configure the Attribute Value to determine the threshold for comparison. The value can be a number, symbol, string, true, or false.

For example, here the Block tab of the properties dialog that retrieves objects whose color attribute equals red:

The screenshot shows a dialog box titled "Retrieve: Fulfill Orders for Red Widgets" with a close button (X) in the top right corner. The dialog has several tabs: "General", "Block", "Database", "Duration", "Cost", and "Animation". The "Block" tab is selected. The dialog contains the following fields and options:

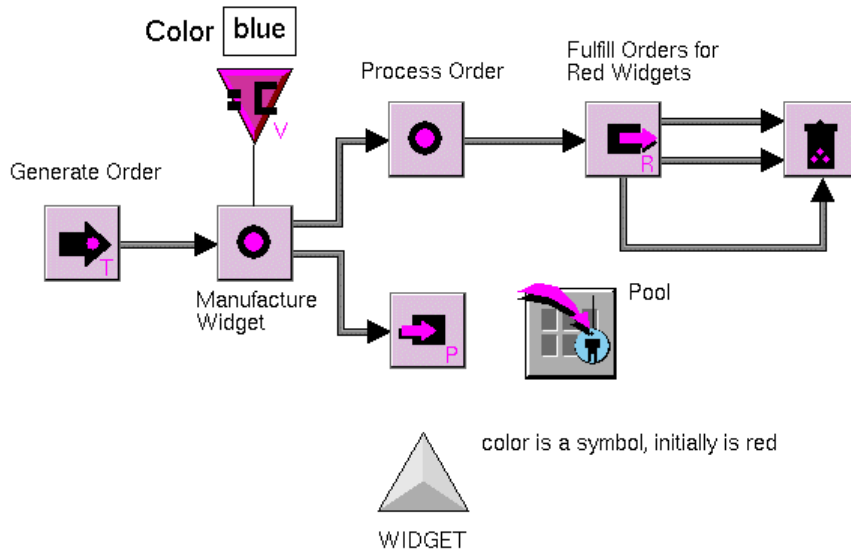
- Retrieve Mode: Attribute Value (dropdown)
- Association Name: NONE (dropdown)
- Retrieve Attribute: COLOR (text field)
- Attribute Value: RED (text field)
- Range Lower: 0 (spin box)
- Range Upper: 0 (spin box)
- Operation: = (dropdown)
- Retrieve All
- Retrieve Copy
- Add To Associations

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

5 Configure the output path types of the Retrieve block.

The block compares the current value of Retrieve Attribute of the work objects in the pool to the Attribute Value, using the Operation.

The following example is a variation on the previous examples of retrieving blocks from a pool, where the Retrieve block retrieves only those widgets whose color is red. A Change feed determines the color of the current widgets in inventory.



Retrieving Based on a Range of Values

When the Operation is Range, you have several options:

- Specify a distinct range by entering an upper threshold in the Range Upper attribute of the block and a lower threshold in the Range Lower attribute. The Range Upper must be greater than the Range Lower.
- Specify an open-ended range by entering just an upper threshold for comparison in the Range Upper attribute of the block.
- Specify an open-ended range by entering just a lower threshold for comparison in the Range Lower attribute of the block.

For example, if the Operation is Range, and you specify the Range Lower as 1 and the Range Upper as 4, the block retrieves work objects from the pool whose specified attribute is greater than or equal to 1 and less than or equal to 4.

Retrieving All Work Object

You can retrieve all work objects from a pool or database, according to the specified retrieve mode:

In this retrieve mode...	The block retrieves...
Random	All work objects in the pool.
Association	All associated work objects in the pool.
Database	All records in the table, and uses the data to create multiple work objects.
Attribute Value	All work objects that meet the criteria in the pool.

Note In database retrieve mode, the default behavior is to retrieve the next work object from the table.

To retrieve all work objects from a pool or database:

➔ On the Block tab of the properties dialog, click the Retrieve All option on.

Retrieving Copies of Work Objects from a Pool

When retrieving work objects from a pool, using random, association, or attribute value mode, you can choose to retrieve a copy of the work object, rather than the work object itself. When retrieving a copy, the block creates a copy of the work object that meets the specified criteria, leaving the original object in the pool.

Note The Retrieve Copy option does not apply to database retrieve mode; database retrieve mode always copies the data from the table and never deletes the record from the database.

To retrieve a copy of a work object from a pool:

- 1 On the Block tab of the properties dialog, configure the Retrieve Mode to be any mode except Database.
- 2 Click the Retrieve Copy option on.

Adding Retrieved Work Objects to Associations

If you are retrieving work objects from a pool by creating a copy, and if the retrieved work object is associated with another work object in the model, you can choose to make the copy be associated with the other work object. By default, the Retrieve block does *not* add copies to associations.

To add retrieved copies to associations:

- 1 On the Block tab of the properties dialog, configure the Retrieve Mode to be any mode except Database.
- 2 Click the Retrieve Copy option on.
- 3 Click the Add to Associations option on.

Determining How the Block Handles Objects Not Found

By default, when the block cannot find an object that meets the association criteria, or when the block cannot find an object in the pool, the block sends the input object onto the specified output path anyway. You can create a specific output path to pass objects for which an associated object or object from the pool is not found.

For example, suppose you are retrieving invoices from a pool based on an association with orders, and the block cannot find an invoice associated with a particular order. You might want to pass the order onto a special output path.

To specify an output path for input objects that the block cannot retrieve:

- 1 Create another output path on the block.
- 2 Configure the path type of the new output path to be the input object type.
Given the example above, the output path type would be invoice.
- 3 Choose the Choose Not Found Output Path menu choice, then choose Select on the path you just created to select the output path that passes the objects whose association cannot be found.

When selected, the path turns to magenta indicating that you have selected it.

Specific Attributes

The image shows a software dialog box titled "Retrieve" with a close button (X) in the top right corner. The dialog has several tabs: "General", "Block", "Database", "Duration", "Cost", and "Animation". The "General" tab is currently selected. Inside the dialog, there are several input fields and checkboxes:

- Retrieve Mode:** A dropdown menu showing "Random".
- Association Name:** A dropdown menu showing "NONE".
- Retrieve Attribute:** An empty text input field.
- Attribute Value:** A text input field containing "0.0".
- Range Lower:** A text input field containing "0" with a spinner control to its right.
- Range Upper:** A text input field containing "0" with a spinner control to its right.
- Operation:** A dropdown menu showing "=".
- Retrieve All:** An unchecked checkbox.
- Retrieve Copy:** An unchecked checkbox.
- Add To Associations:** An unchecked checkbox.

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attributes of the Retrieve block are:

Attribute	Description
Retrieve Mode	Specifies how to retrieve objects. The options are: Random, Association, Database, Attribute Value, and Custom. The default value is Random.
Retrieve All	When selected, retrieves all work objects from the pool or database, according to the Retrieve Mode: <ul style="list-style-type: none">• In random mode, retrieves all work objects from the pool.• In association mode, retrieves all the associated work objects in the pool.• In database mode, retrieves all the records in the table and uses the data to create multiple work objects.• In attribute value mode, retrieves all work objects from the pool that meet the criteria.
Retrieve Copy	When selected, retrieves a copy of the work object, leaving the original work object in the pool. Applies to all retrieve modes except Database.
Add to Associations	When Retrieve Copy is selected, determines whether the block adds the associations of the retrieved work object to the copy.

The mode-specific attributes of the Retrieve block are:

Mode	Attribute	Description
Association	Association Name	The criteria that the block uses to determine which object to retrieve from the pool. Typically, the value of this attribute is an association name that an Associate block creates.
Database	Database Interface Name (Database tab)	The name of the Database Interface object that allows access to an external database. See Retrieving Records from a Database .
	SQL Query (Database tab)	An SQL query that queries the external database named in Database Interface Name. ReThink retrieves one work object for each record that matches the query. The attributes of each work object correspond to the fields in each record. See Creating an SQL Query for Accessing the Data .

Mode	Attribute	Description
Attribute Value	Retrieve Attribute	<p>An attribute of a work object to compare with the Attribute Value to determine if the block should retrieve the work object.</p> <p>You can use dot notation to refer to the attribute of a subobject, for example, <code>my-subtable.my-attr</code>.</p>
	Attribute Value	<p>The value the Retrieve block uses when it tests the Retrieve Attribute, using the Operation. If the attribute value meets the criteria, the block retrieves the work object. If no attribute value meets the criteria, the block does not retrieve the work object.</p>
	Range Lower	<p>When the Operation is Range, this attribute is the lower end of the range that the Retrieve block uses when it tests the Retrieve Attribute. If the attribute value meets the criteria, the block retrieves the work object.</p>
	Range Upper	<p>When the Operation is Range, this attribute is the upper end of the range that the Retrieve block uses when it tests the Retrieve Attribute. If the attribute value meets the criteria, the block retrieves the work object.</p>
	Operation	<p>The operation the block uses to make the comparison. The default value is <code>=</code>.</p>
Custom	Lookup Procedure Name	See Customization Attributes .

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for the Retrieve block are:

Menu Choice	Description
Choose Not Found Output Path	Identifies the output path of the Retrieve block that passes objects for which an associated object or object from the pool cannot be found. Choose Select on the output path you want to select the path.
Show Not Found Output Path	Puts an indicator arrow next to the chosen not found output path.
Choose Pool	In random lookup or association lookup modes, chooses the resource pool from which the Retrieve block retrieves objects. Choose Select on the pool from which to retrieve objects to select the pool.
Show Pool	In random lookup or association lookup modes, displays an indicator arrow next to the pool that the Retrieve block has identified. This menu choice only appears when you have already chosen a pool and when the scenario is active.

Note When you choose a pool for the Retrieve block by using the Choose Pool menu choice, ReThink adds a menu choice to the resource pool's menu called Show Blocks. This menu choice places an indicator arrow next to the block or blocks that are currently pointing to that pool.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

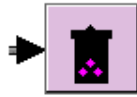
Customization Attributes

The customization attribute available in Developer mode for the Retrieve block is:

Attribute	Description
Lookup Procedure Name	Determines how the block retrieves objects from a pool. The default value of this procedure in the subtable is <code>bpr-random-lookup-from-pool</code> , which retrieves objects from the pool at random.
Copy Item Lists	When <code>true</code> and when Retrieve Copy is <code>true</code> , copies the item-list attributes of the work object.
Copy Item List Items	When <code>true</code> and when Retrieve Copy is <code>true</code> , copies the items contained in the item-list attributes of the work object.

You can customize the procedures that ReThink uses to retrieve blocks from a pool. For backward compatibility, you can set the `copy-item-lists` and `copy-item-list-items` attributes to `false`. For more information, see the *Customizing ReThink User's Guide*.

Sink



The Sink block signals the end of a process and is the counterpart to a Source block. You position this block at the end of a process. A Sink block deletes all the work objects it receives.

Note To probe a Sink block to obtain metrics about work objects, you must configure the probe to trigger in the start phase by configuring the Phase attribute of the probe.

Signalling the End of a Process

You use the Sink block at the end of a process to delete its input work objects. The block permanently deletes the work objects it receives on its input path or paths.

Note If you do not use a Sink block or a block with no output paths at the end of a process in a model, work objects accumulate on the unconnected path. Therefore, we recommend that you always use a Sink block to delete work objects at the end of a process when you do not need to save the data.

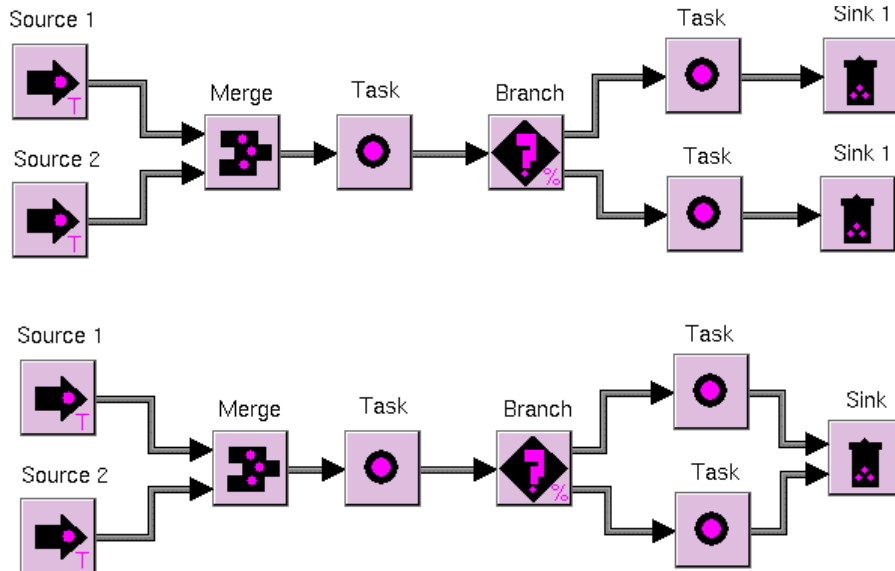
To delete objects at the end of a process:

→ Connect to the Sink block the output stubs from upstream blocks, whose work objects you want to delete.

Tip When you want to delete multiple streams of work, it is sometimes easier to delete the existing input stub on the Sink block first and then connect the output stubs from the upstream block directly to the Sink block.

If you do not want to delete the work objects at the end of a process, you can use a Store block instead to store the objects in a resource pool. For an example, see [Storing Work Objects in a Pool](#).

You can use multiple Sink blocks to delete multiple streams of work at the end of a process, or you can use a single Sink block and connect as many input paths as you need, as these two models show:



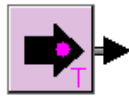
Specific Attributes and Menu Choices

A Sink block has no specific attributes or menu choices.

For information on the common block attributes, see [Common Attributes of Blocks](#).

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Source



The Source block generates work objects as input to a model. In the default mode, the block generates work objects based on the type of its output path and the mean time specified for the block. By default, the Source block uses a random exponential function to compute the arrival time of work objects.

You can generate work objects by specifying:

- Object types on the Source block's output paths.
- Object types, attributes, and values in an external file.
- An SQL query and external database, which generates one object for each record in the database.

When generating work objects from a file, you can determine whether the simulation stops or continues when it reaches the end of the file.

When generating work objects from a database, you must create and configure a Database Interface object to provide access to the external database.

For information on how to use the Source block for accessing external databases, see [Sourcing Records from a Database](#).

You can use the Store block to store object types and attribute values to a file or database, or to store object types, attribute values, and arrival times to a file, and then use the stored data as input to a Source block. In this way, you can rerun the same simulation to perform "what-if" analysis. For example, you might want to experiment with different work flow models or resource constraints, using the same set of input data.

Configuring the Source Mode

A Source block supports the following modes for determining the type of the work objects it generates, and in object file mode and database mode, the attribute values of each work object:

Mode	Description
Type	Generates work objects, based on the Type attribute specified on the output path of the Source block.
Object File	Generates work objects, based on the object types and attribute values specified in an external text file.
Database	Generates work objects, based on records and fields in an external database.
Custom	Uses a custom procedure to determine how the block generates work objects.

Generating Work Objects Based on the Path Type

By default, the Source block uses Type mode to generate work objects, based on the output path type of the Source block.

To generate work objects based on the path type:

- 1 On the the Block tab of the properties dialog, configure the Source Mode to be Type.
You can use the default value for Output Count.
- 2 Create output paths for the Source block for each type of work object you want to create.
- 3 Configure the Type attribute of each output path.

ReThink generates work objects of the type specified on the output path(s), at intervals computed based on the mean time.

If the class definition of the work object does not already exist, ReThink automatically creates one. The work object that ReThink creates is a subclass of `bpr-object`.

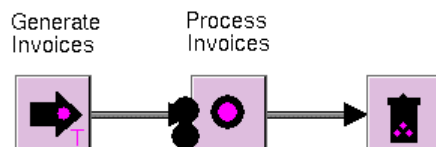
Configuring the Number of Objects to Generate for Each Output Path Type

By default, each time the block executes, it generates one work object on each output path. When Source Mode is Type, you can specify the number of work objects to generate on each output path for each activity of the block.

To specify the number of objects to generate for each output path:

- 1 On the Block tab of the properties dialog, configure Source Mode to be Type.
- 2 Configure the Output Count attribute to be the number of work objects to generate.

For example, this model generates two invoices, one of which has been moved to show both objects:



Generating Work Objects from an External File

You can specify in an external file the types of objects a Source block generates and values for any user-defined attributes the object defines. The Source block generates one work object for every object in the file. You can determine if the block generates work objects continuously or stops when it reaches the end of the file.

Format of Object File

The basic format of the object file looks like this, where each new object is separated by a carriage return:

```
object-type, attribute, value, attribute, value...
object-type, attribute, value, attribute, value...
etc.
```

For example, this file generates three orders:

```
widget, color, red
widget, color, blue
widget, color, yellow
```

You can also specify an output file with attribute values that contain item-lists as follows:

```
object-type, attribute, ((object-type, attribute, value, attribute, value ...),
                        (object-type, attribute, value, attribute, value ...),
                        (...))
object-type, attribute, ((object-type, attribute, value, attribute, value ...),
                        (object-type, attribute, value, attribute, value ...),
                        (...))
etc.
```

For example, this file generates two software products, each with an item-list attribute named `line-items`. Each item in the list is an instance of the `line-item` class and contains two attributes, `product-name` and `product-version`.

```
software-product, line-items, ((line-item, product-name, "G2", product-
version, "8.2 Rev. 0"), (line-item, product-name, "ReThink", product-
version, "5.0 Rev. 0"))
software-product, line-items, ((line-item, product-name, "G2", product-
version, "8.2 Rev. 1"), (line-item, product-name, "GDA", product-version,
"5.0 Rev. 0"))
```

Note The class definition and attribute names must exist before you read object types from a file. If an object file contains attributes that are item-lists, those attributes must also exist. If the class definition does not exist, or if the class definition does not specify all the attributes in the file, ReThink generates an error.

For information on how to create work objects with user-defined attributes, see [Creating a New Class of Work Object](#).

Generating Work Objects Continuously

For example, you might create a file that specifies a single object type that repeats, or you might create a file that specifies numerous different types of object files in a particular order, which stops when it reaches the end of the file.

To generate work objects from a file continuously:

- 1 Create an external file of object types and optional attribute names and values.

You can generate this file by using an external editor or by using the Store block.

For information on how to generate this file by using a Store block, see [Storing Work Objects to a File](#).

- 2 Create class definitions for each object type specified in the external file.

Be sure the class definitions contain class-specific attributes for all user-defined attributes specified in the file.

For information on how to create a class definition with class-specific attributes, see [Creating a New Class of Work Object](#).

- 3 On the Block tab of the properties dialog, configure the Source Mode to be Object File.

- 4 Configure the Object File Name to refer to the external file.

If you have created the external file by using the Store block, the Object File Name attribute is the same as the Object File Name you specify in the Store block.

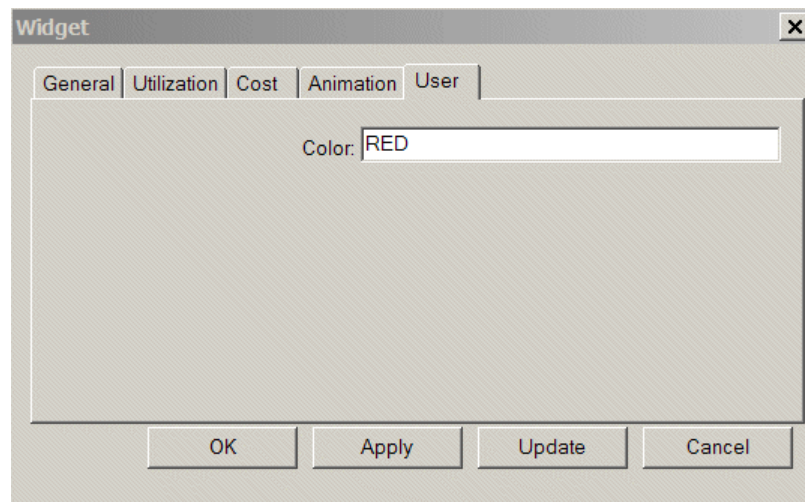
- 5 Configure the output path type of the Source block to be compatible with the work objects in the file.

Note ReThink ignores the objects in the file whose path type is incompatible with the output path type of the Source block.

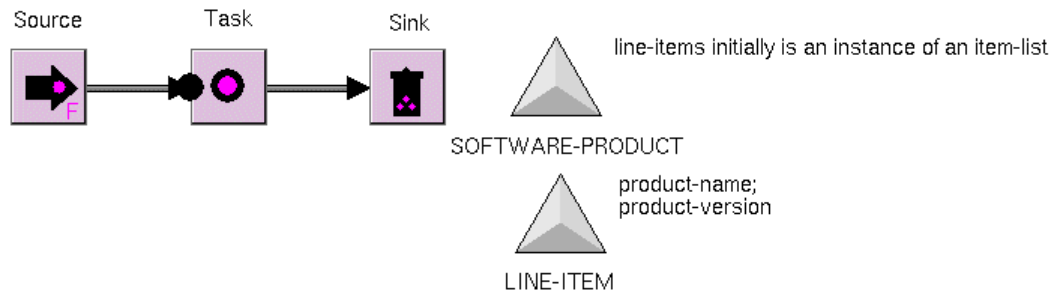
- 6 Configure the duration of the Source block.

The Source block generates one work object for each line in the file. The attributes of the work objects correspond to the attribute names and values specified in the file.

Here is the first example above, which generates widgets with different values for the color attribute:



Here is the second example above, which generates software products with different values for the line-item attribute:



Stopping Generating Work Objects at the End of the File

By default, the Source block generates work objects continuously by looping back to the beginning of the file when it reaches the end. You can cause the Source block to stop when it reaches the end of the file.

To stop generating objects at the end of the file:

- 1 On the Block tab of the properties dialog, configure the Source Mode to be Object File.
- 2 Click the Repeat Object File attribute off.

ReThink generates as many work objects as are specified in the file and then stops.

You can also control when the Source block stops generating work objects by specifying the maximum number of objects to create. For more information, see [Configuring the Maximum Number of Objects](#).

Configuring Duration and Objects from an External File

You can use a combination of a duration file and an object file to create objects, using a Source block. By specifying different values for the following two attributes, you can control which file determines the number of objects the Source block creates:

- The Repeat Duration File attribute on the Duration tab of the properties dialog, which controls whether the duration file repeats. You configure this attribute when the Distribution Mode is Duration File.
- The Repeat Object File attribute on the Block tab of the properties dialog, which controls whether the object file repeats.

The following table summarizes the effects of different combinations of these two attributes:

Repeat File Settings for Object and Duration Files

Repeat File		
Duration	Object	Effect
on	off	Source creates one object for each entry in the object file. If the duration file contains fewer entries than the object file, ReThink cycles to the beginning of the duration file to obtain durations for the remaining entries in the object file.
off	on	Source creates one object for each entry in the duration file. If the object file contains fewer entries than the duration file, ReThink loops to the beginning of the object file to obtain types and attribute values for the remaining entries in the duration file.
on	on	Source creates objects continuously, cycling back to the beginning of each file when it reaches the end. If the files contain a different number of entries, ReThink pairs the current object entry with the current duration entry, regardless of its position in the file.
off	off	Source creates the number of objects in the file that contains the fewest entries. ReThink stops processing when it reaches the end of the shortest file.

You can also specify the maximum number of objects that the source emits to control the number of objects that flow into a model. For more information, see [Configuring the Maximum Number of Objects](#).

For information on configuring duration from a file, see [Specifying Duration from a File](#).

Generating Work Objects

Because a Source block has no input paths, you need to start the block explicitly to generate work objects. You start a source in one of three ways.

Note The Scenario must be active for these menu choices to be available on the Source block.

To generate work objects from a Source block:

→ Choose Start on the Source block to generate work objects continuously according to the specified duration.

or

→ Choose Single Shot on the Source block to generate a single work object.

or

→ Click the Start All button on the toolbar to generate work objects for all Source blocks associated with the scenario.

The Start and Single Shot menu choices start the source and continue running the model. The Start All button also automatically resets the model.

Configuring the Maximum Number of Objects

You can specify the maximum number of work objects that the Source block emits so that the simulation stops when the maximum is reached. This can be useful for testing purposes to limit the number of objects the model processes.

To limit the number of objects a Source block generates:

→ On the Block tab of the properties dialog, configure the Maximum Starts to be the number of objects to generate.

If you are generating object types or durations from an external file, the Maximum Starts attribute takes precedence over the length of the files.

Configuring the Start and End Times

You can specify the start and end times of the first and last work objects, respectively, independent of duration. This feature allows you to designate start and end times for portions of a simulation.

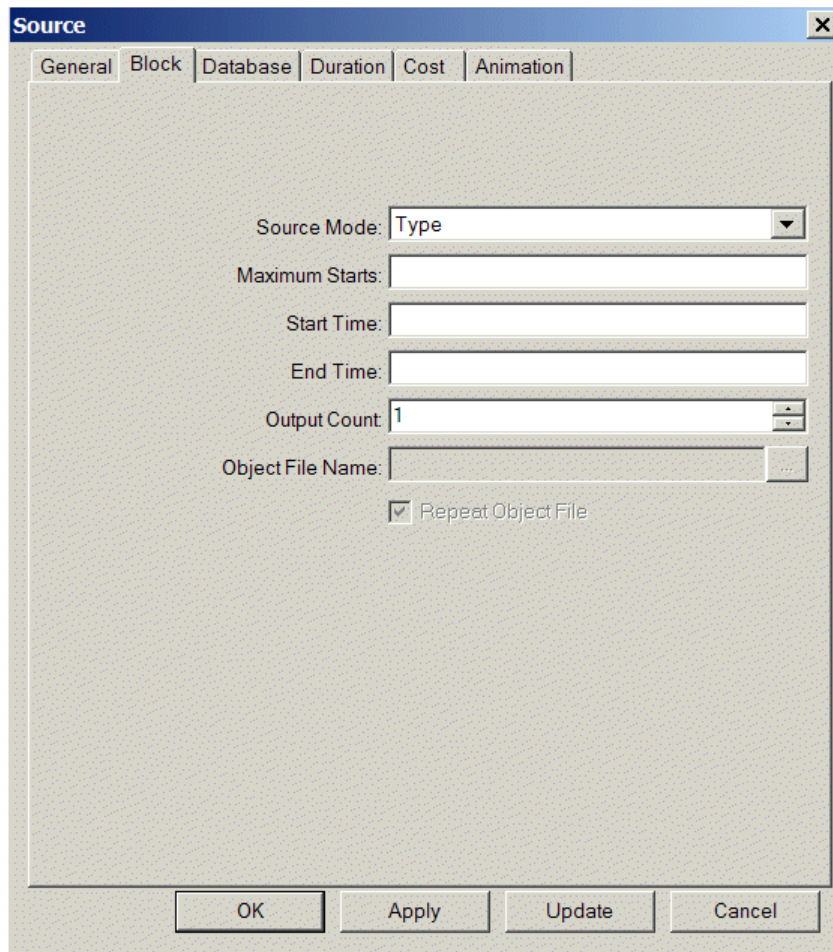
To specify the start and end times of the first and last work objects:

- 1 On the Block tab of the properties dialog, configure the Start Time to be the time at which the first work object is generated, for example, 10 seconds, 8 hours, or 1 week.
- 2 Configure the End Time to be the time at which the last work object is generated.

ReThink converts these durations to seconds in the dialog.

The Source block generates its first work object and its last work object at exactly the start time you specify, regardless of its duration. Thus, if the Source block specifies 45 minutes as the Start Time and 60 minutes as the End Time, and it specifies exactly 10 minutes as its duration, it would generate three work objects at times 45 minutes, 55 minutes, and 60 minutes. When it generates its last work object, the block stops processing.

Specific Attributes



The screenshot shows a dialog box titled "Source" with a close button (X) in the top right corner. The "Block" tab is selected, and the "General" sub-tab is active. The dialog contains the following fields and controls:

- Source Mode: A dropdown menu with "Type" selected.
- Maximum Starts: An empty text input field.
- Start Time: An empty text input field.
- End Time: An empty text input field.
- Output Count: A spin box with the value "1".
- Object File Name: A text input field with a browse button (three dots).
- A checked checkbox labeled "Repeat Object File".

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attributes of the Source block are:

Attribute	Description
Source Mode	Specifies how the block generates work objects. The options are: Type, Object File, Database, and Custom. The default value is Type.
Maximum Starts	The maximum number of work objects the Source block can create.

Attribute	Description
Start Time	The time at which the Source block generates its first work object, specified as a duration. For example, 10 minutes, 1 hour and 30 minutes, 12 days. ReThink converts the duration to seconds.
End Time	The time at which the Source block generates its last work object and stops processing, specified as a duration.

For information on the common block attributes, see [Common Attributes of Blocks](#).

The mode-specific attributes of the Source block are:

Mode	Attribute	Description
Type	Output Count	The number of work objects to create for each activity on each output path.
Object File	Object File Name	The filename of the external file that contains object types, attribute names, and attribute values. The filename must be a complete pathname.
	Repeat Object File	Whether the Source block loops back to the beginning of the object file to continue generating work objects or whether it stops when it reaches the end. The default value is on.
Database	Database Interface Name (Database tab)	The name of the database interface object that allows access to an external database. See Sourcing Records from a Database .

Mode	Attribute	Description
	SQL Query (Database tab)	An SQL query that queries the external database named in Database Interface Name. ReThink generates one work object for each record that matches the query. The attributes of each work object correspond to the fields of each record. See Creating an SQL Query for Accessing the Data .
	Repeat Database (Database tab)	Whether the Source block loops back to the beginning of the database to continue generating work objects or whether it stops when it reaches the end. The default value is on.
Custom	Source Procedure Name	See Customization Attributes .

Specific Menu Choices

The specific menu choices for the Source block are:

Menu Choice	Description
Single Shot	Generates a single work object on the output path.
Start	Generates work objects on the output path continuously. The frequency of work objects depends on the duration of the block.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

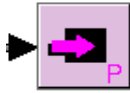
Customization Attributes

The customization attribute available in Developer mode for the Source block is:

Attribute	Description
Source Procedure Name	When Source Mode is custom , specifies the procedure that the block uses to determine how it generates work objects. The default value is bpr-source-type .

You can customize the procedure that ReThink uses to generate work objects. For more information, see the *Customizing ReThink User's Guide*.

Store



The Store block stores objects in a resource pool, file, or database. You use this block to model database operations or inventory fluctuations. For example, you use a Store block to model saving data to a customer database or an order database, or saving line items in a manufacturing inventory. You use the [Retrieve block](#) to retrieve objects from a pool or database.

When you store objects to a file or database, you can store the object types and user-defined attribute values. When you store objects to a file, you can also store the arrival time between objects. You can then use this file or database as input to the Source block to rerun the same simulation, for example, to test a different work flow configuration or a different set of input parameters.

Note When viewing the output files that ReThink generates, use an editor that displays each line of text on its own line.

When you store work objects to a database, ReThink creates or updates one database record for each work object the block processes. To store work objects to a database, you must create and configure a Database Interface object, which provides access to external databases.

For information on how to use the Store block for accessing external databases, see [Storing Work Objects to a Database Table](#).

Configuring the Store Mode

A Store block supports the following modes for determining how it stores work objects:

Mode	Description
Pool	Stores work objects to a resource pool.
File	Stores work objects to a text file.
Database	Stores work objects to an external database.
Custom	Uses a custom procedure to determine how the block stores work objects.

Storing Work Objects in a Pool

One use of resource pools is to store work objects in the pool dynamically during processing. To do this, you use the Store block to store work objects to a pool. You use this technique to model database or inventory operations.

ReThink clears the pool when you reset the model. If your model depends on the fact that objects initially exist in the pool, you must populate the pool with objects before you run the simulation. You can do this by using a G2 procedure or by manually transferring work objects to the pool.

For information on creating procedures, see *G2 Reference Manual*.

To store work objects in a pool, configure the block to refer to the pool.

You have two options when storing objects to a pool:

- You can store work objects in a pool as a way of signalling the end of a process, instead of using a Sink block.
- You can store associated objects in a pool and retrieve the object and its associated object downstream in a process with a Retrieve block.

You can also store resources in a pool dynamically, using the Store block.

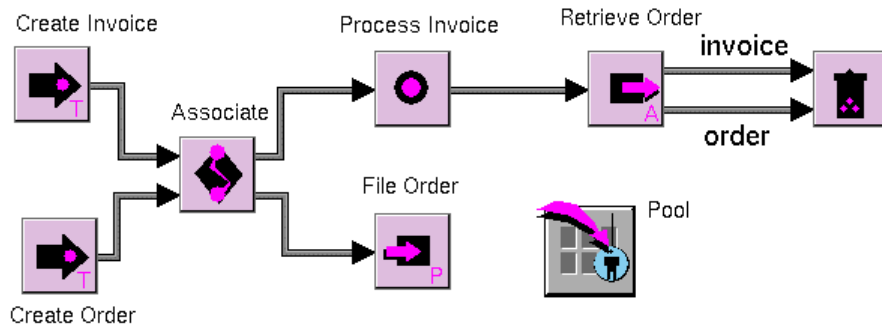
To store work objects to a pool:

- 1 Create a resource pool from the Resources palette of the ReThink toolbox.
You can also create one of the other types of resources; however, if you do, you need to create a detail for the resource.
- 2 On the Block tab of the properties dialog, configure the Store Mode to be Pool.
- 3 Choose the Choose Pool menu choice on the Store block, then choose Select on the pool in which the Store block will store objects to select the pool.
ReThink places an indicator arrow next to the selected pool indicating that you have selected it.

Note ReThink adds the Show Pool menu choice to the block, which places an indicator arrow next to the selected pool. ReThink also adds the Show Blocks menu choice to the selected pool, which identifies the blocks that are currently pointing to this pool.

This example shows a simple model that creates orders and invoices, associates the order and the invoice, stores the orders in a resource pool, then retrieves the

associated orders from the pool. The File Order task stores orders in the PC Database resource pool. The Retrieve Order task retrieves the order from the pool.



Storing Work Objects to a File

You can store objects to a file by using file mode. When you store objects to a file, ReThink stores the object type and any user-defined attributes defined for the object.

The format of the object file looks like this, where each object is separated by a carriage return:

```
object-type,attribute,value,attribute,value...
object-type,attribute,value,attribute,value...
etc.
```

For example:

```
ORDER, TIMESTAMP, 1174
ORDER, TIMESTAMP, 7511
ORDER, TIMESTAMP, 8822
```

You can use this file as input to a Source block to generate work objects in a model. In this way, you can create a reproducible model that uses the same objects and values each time you run the simulation.

For information on how to generate work objects from this file, see [Generating Work Objects from an External File](#).

Note ReThink appends new object data to existing files, rather than overwriting the file. To create a new file, delete the existing file before running the model.

To store objects to a file:

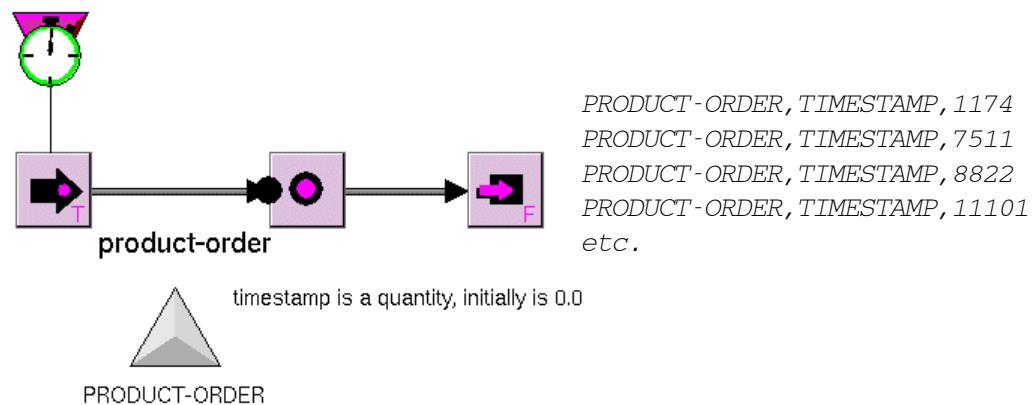
- 1 On the Block tab of the properties dialog, configure the Store Mode to be File.
- 2 Configure the Object File Name to be the file name in which to store the work objects.

You must specify a complete pathname.

Note When you store user-defined attributes to a file, the attributes must have a value; otherwise, ReThink ignores the attribute.

For example, suppose you create a model that generates orders and feeds a timestamp into a user-defined attribute named `timestamp` when the order is created. Thus, this timestamp reflects the creation time of each order.

If you save the objects and their associated timestamps to a file, you could then use the file to generate objects in another model, using a Source block in file mode. This figure shows such a model and sample output file:



Typically, the model updates a user-defined attribute of a work object sometime during the process as opposed to at the beginning of the process. However, this example will also be used to demonstrate how ReThink stores duration times in a file, as the following section describes.

Storing Arrival Times to a File

You can save to a file the difference in creation times between objects. You can then use this file with a Source block to provide arrival times from a file.

For information on how to use this file to specify the duration of a Source block, see [Specifying Duration from a File](#).

To store object arrival times to a file:

- 1 On the Block tab of the properties dialog, configure the Store Mode to be File.
- 2 Configure the Duration File Name to be the file name in which to store the arrival times.

You must specify a complete pathname.

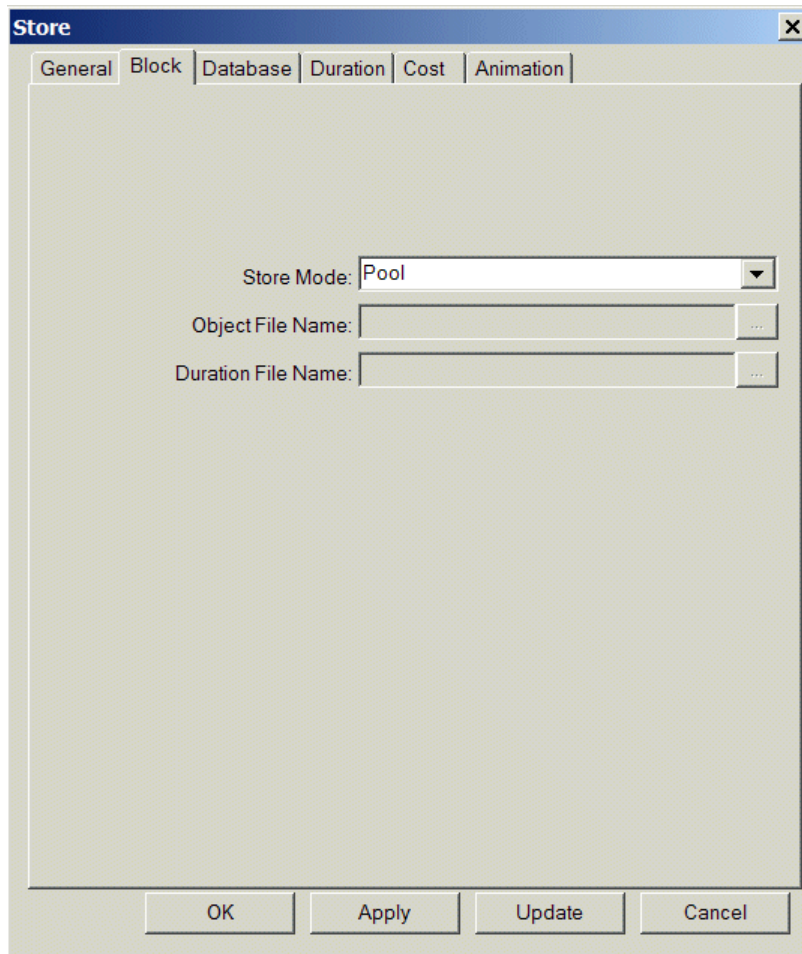
Using the model shown in [Storing Work Objects to a File](#), suppose you were to store arrival times to a file. The duration file would look like this:

1174
6337
1311
2279

Notice that the first time is the same as the first timestamp in the object file created in the previous example, which corresponds to the creation time of the object; the second timestamp is 6337 seconds later; the third timestamp is 1311 seconds after that, and so on.

Thus, ReThink creates arrival times by subtracting each subsequent object's creation time from the previous object's creation time.

Specific Attributes



The specific attribute of the Store block is:

Attribute	Description
Store Mode	Specifies how the block stores objects. The options are: Pool, File, Database, or Custom. The default is Pool.

For information on the common block attributes, see [Common Attributes of Blocks](#).

The mode-specific attributes of the Store block are:

Mode	Attribute	Description
Pool	N/A	N/A
File	Object File Name	The file name that the block uses when it stores object types and user-defined attribute names and values to a file. The filename is a complete pathname.
	Duration File Name	The file name that the block uses when it stores durations to a file. The filename is a complete pathname.
Database	Database Interface Name (Database tab)	The name of the Database Interface object that allows access to an external database. See Storing Work Objects to a Database Table .
	Database Table (Database tab)	The name of a table within the database in which the Store block creates or updates the records. If you do not specify the Database Table, the Store block uses the SQL Statement.
	Database Key (Database tab)	An attribute of the work object, which serves as a key for updating existing records in the database.

Mode	Attribute	Description
	SQL Query (Database tab)	When Database Table is not specified, an SQL query that queries the external database named in Database Interface Name. ReThink stores one work object in each record that matches the query. The attributes of each work object correspond to the fields of each record. See Creating an SQL Query for Accessing the Data .
Custom	Store Procedure Name	See Customization Attributes .

Specific Menu Choices

The specific menu choices for the Store block are:

Menu Choice	Description
Choose Pool	Chooses the resource pool in which the Store block stores objects. Choose Select on the pool in which to store objects to select the pool.
Show Pool	Displays an indicator arrow next to the pool that the Store block has identified. This menu choice only appears when you have already chosen a pool.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

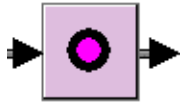
Customization Attributes

The customization attribute available in Developer mode for the Store block is:

Attribute	Description
Store Procedure Name	When Store Mode is Custom , specifies the procedure that determines how the block transfers objects to the detail of the resource pool. The default value is <code>bpr-store-pool</code> .
Database Input Object Name	The name by which you reference the input work object.
Database Quote String	The character that ReThink uses to specify a text string. The default value is <code>'</code> .
Database Quote In Text String	The character that ReThink uses to specify an embedded quote character within a text string. The default value is <code>"</code> .

You can customize the procedure that ReThink uses to store work objects, as well as several attributes related to database mode. For more information, see the *Customizing ReThink User's Guide*.

Task



The Task block represents any activity that processes work objects in a model. This block can have any number of inputs and outputs for processing singular or multiple streams of work.

A Task block represents a core activity or a collection of activities or subprocesses. Typically, you decompose a Task block level by level by adding detail.

If a Task block has multiple inputs, it waits until it has received all of its inputs before it passes objects onto any output paths, thereby synchronizing its inputs.

The number of output paths and their types determines whether the Task block passes an object downstream, creates a new object, or deletes an existing object. The Task block:

- Processes all input work objects for which output paths of a corresponding type exist.
- Creates new work objects on output paths for which no corresponding input path type exists.
- Deletes input work objects for which no corresponding output path type exists.

When configuring the Task block to delete the input work object and create a new work object of a different type, you can configure the block to copy attribute values from the input work object to the output work object.

For a general explanation of how ReThink determines the output path type for blocks with multiple output paths, see [Determining the Output Path Based on Its Type](#).

Processing Work and Sending It Downstream for Further Processing

One basic type of processing involves receiving an object, processing that object, and then sending the same object downstream for further processing. This is the simplest way of using the Task block.

To process an object and send it downstream:

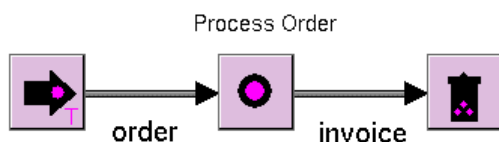
- 1 Configure the type of the input path of the Task block.

Because the default path type is `bpr-object`, you do not need to configure the type of the output path. ReThink automatically passes to the output path the work object that the block receives.

- 2 On the Duration tab of the properties dialog, configure the duration of the task.

The duration of the task represents the amount of processing applied to a particular work object. You determine the amount of time that the Task block contributes to the overall process time based on its duration.

Here is a simple example that processes orders, with the path types labeled:



Notice that the output path type is `bpr-object`, which allows the object to travel on the path, because the `order` class is a subclass of `bpr-object`.

Processing Multiple Streams of Work Synchronously

Another common modeling technique is to process multiple streams of different types of work as part of the same activity. You use this technique when a particular task requires multiple inputs before it can process.

For example, your model might require that an order and an invoice both be signed by the same manager before it is filed and the invoice is sent out.

You typically use a Task block to accomplish this type of processing. The Task block waits to process its inputs until all inputs are available. This type of processing is called synchronized processing.

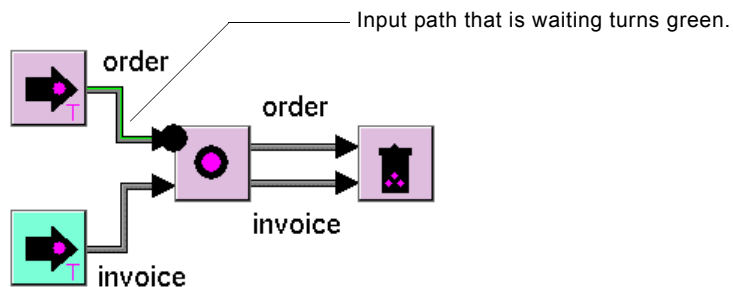
To synchronize the processing of multiple streams of work:

- 1 Create additional input paths for the Task block.
- 2 Configure the path type of all input paths.

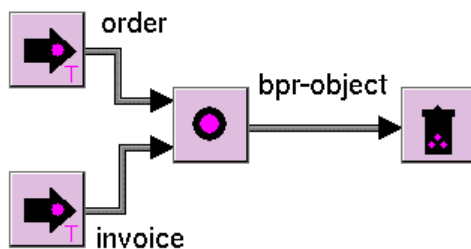
- 3 Depending on how you want to process the objects, you have two options:
- To send the objects downstream on separate output paths, create as many output paths as there are input paths and configure the type of each output path to correspond to the input path types.
 - To send the objects downstream on the single output path of the Task block, use the default path type of `bpr-object`.

When you run the simulation, the block waits to process its inputs until all inputs arrive at the block. The path of the work object that arrives first at the block turns green indicating that the block is waiting for other inputs.

Here is an example of a running model that uses separate sources to generate orders and invoices, which the Task block synchronizes and sends downstream on separate output paths:



Here is a slightly different version of the same example that processes the orders and invoices concurrently but sends them downstream on the same output path:



Processing Multiple Streams of Work Sequentially

When you are processing multiple streams of work, you can also process the work sequentially as opposed to synchronously. You use this technique when a task receives multiple inputs that are completely independent of one another.

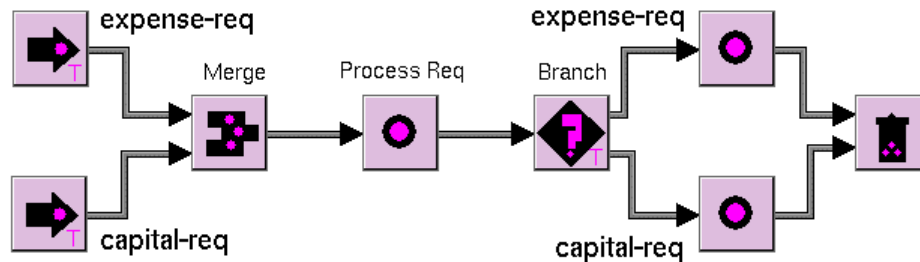
For example, a model of a purchasing department might create two different types of requisitions and process each separately. Rather than being dependent on one another, the requisitions are completely independent.

To use a Task block to process multiple types of work as soon as the work arrives, you must first merge the different streams of work together by using a Merge block. This type of processing is called sequential processing.

To process multiple streams of work sequentially:

- ➔ Insert a Merge block before the Task block, which merges the multiple streams of work together.

Here is an example that merges two different types of requisitions into a single path, which the Task block processes sequentially. The requisitions then pass to a Branch block, which branches them based on their types for separate processing. The input and output path types of the Task block are both `bpr-object`, which allows any type of object to pass.



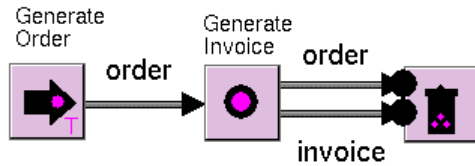
Generating Work in a Process

Typically, you create work objects at the beginning of a process by using a Source block. Often, however, a model requires that you generate work in the middle of a process, as opposed to the beginning. You can use the Task block to generate work objects in the middle of a process by adding output paths to the block.

To generate work in a process:

- 1 Create one or more additional output stubs for the Task block.
For information on how to do this, see [Creating and Deleting Stubs](#).
- 2 Configure the path type of the input and output paths that pass the same type of work object.
- 3 Configure the path type of the output path that generates a new work object.

Here is an example that processes orders, passes the orders downstream for further processing, and generates invoices:



When a Task block has multiple output paths, you typically specify the path types of all output paths to make explicit which objects travel on which paths. However, this step is not, strictly speaking, necessary because ReThink will send the existing object onto the appropriate path, using the default path type.

For more information on specifying path types, see [Determining the Output Path Based on Its Type](#).

Also, the Task block can have multiple input paths as well as multiple output paths. Further, the Task block can generate one or more new work objects.

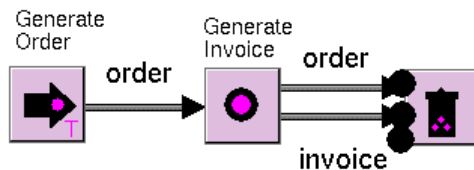
Specifying the Number of Objects to Generate

When using the Task block to generate work objects in the middle of a process, you can generate multiple work objects each time the block processes, rather than a single work object. You specify the number of work objects to generate for each activity and on each output path whose type is different from the input path type.

To specify the number of objects to create for each unique output path type:

- ➔ On the Block tab of the properties dialog, configure the Output Count to be the number of work objects to generate.

For example, this model processes orders on the top output path and generates two invoices on the bottom output path. One of the invoices has been moved to show both objects.



Deleting Work in a Process

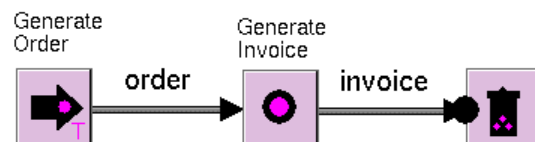
Another common modeling technique is to process one type of work object and generate a different type of work object, thereby deleting the first object. To delete work in a process, specify the path type of the output path to be different from the path type for the input path.

For information on how to copy attributes from the input object to the output object before it gets deleted, see [Copying Attribute Values to the Output Object](#).

To delete work in a process:

- 1 Configure the path type of the input path of a Task block.
- 2 Configure the path type of the output path of the Task block to be a different type.

Here is an example that processes orders and generates invoices, deleting the orders as part of the task:



Modeling the Details of a Task

You can create detail for a Task block to model its detailed tasks. You can place any number of blocks on the detail to create hierarchical views of the model. You can also add Task blocks with detail to a detail to create multiple levels of detail in a model.

When a Task block has detail, ReThink creates a subworkspace for the block with as many connectors as there are input and output paths on the block. The input paths correspond to the connectors on the left side of the detail, and the output paths correspond to the connectors on the right side of the detail. Work objects flow in through the input connectors to the blocks on the detail, then out through the output connectors to the downstream block.

When a Task block has detail, the color of the block's icon changes to salmon, by default.

By default, the Task block with detail computes summary metrics for all the blocks on its detail. Once you create detail for a task, the duration and cost for the top-level task have no effect.

You can enable and disable detail for a Task block. When detail is disabled, the block behaves as if it had no detail, although the detail still exists.

When a Task block has detail and the detail is disabled, the icon looks like this:



Note You cannot probe a Task block with detail when detail is enabled. Probe the individual blocks on the detail instead.

The path type of the right-most output path leading into the connector determines the path type of the object that the superior task passes to the downstream block; the output path type of the superior task has no effect.

For information about how to improve performance in a model that has Task blocks with detail, see [Configuring the Computation Behavior](#).

To model the details of a task:

- 1 Choose Create Detail on the Task block.
- 2 Create, connect, and configure other blocks on the detail.

To enable and disable detail for a task:

➔ Choose Enable Detail and Disable Detail on the Task block.

For additional information on creating hierarchical views in a model and interacting with the detail, see [Creating Hierarchical Views](#).

Copying Attribute Values to the Output Object

You might have a process in which the input work object gets converted into some other type of object, using a Task block. For example, a manufacturing plan might get converted into a build order, or an order might get converted into an invoice. Typically, when you convert an object into a different type, the output object requires at least some of the same attributes as the input object. For example, a manufacturing plan might define the order size, which the build order also requires. When copying attribute values, the attribute names need not be the same.

To copy attribute values to the output object:

- 1 Configure the Task block to delete work in a process.

For details, see [Deleting Work in a Process](#).

- 2 Display the properties dialog and click the Block tab.

By default, the Task block does not copy attributes from the input to the output work object.

- 3 Enable the Copy Attributes option to configure the attributes to copy.

- 4 Configure the attributes to copy, using one of these techniques:
- ➔ Enable the Copy All Attributes option to copy all attributes whose names are the same.

or

- a To copy only certain attributes or to copy attributes whose names are not the same, for each attribute you want to copy, click the Insert Row button.
- b Configure the Source Subtable to be the name of the subtable in which the attribute to copy is defined, if any.

The Source Subtable corresponds to each tab page of the work object on which the attribute to copy appears, for example, **duration-subtable** or **cost-subtable**. If the attribute is on the General tab or is user defined, leave the Source Subtable blank.

- c Configure the Source Attribute to be the name of the attribute of the input work object to copy.

Typically, the source attribute is a user-defined attribute such as **order-size**.

- d Configure the Destination Subtable to be the name of the subtable in which the copied attribute is defined.

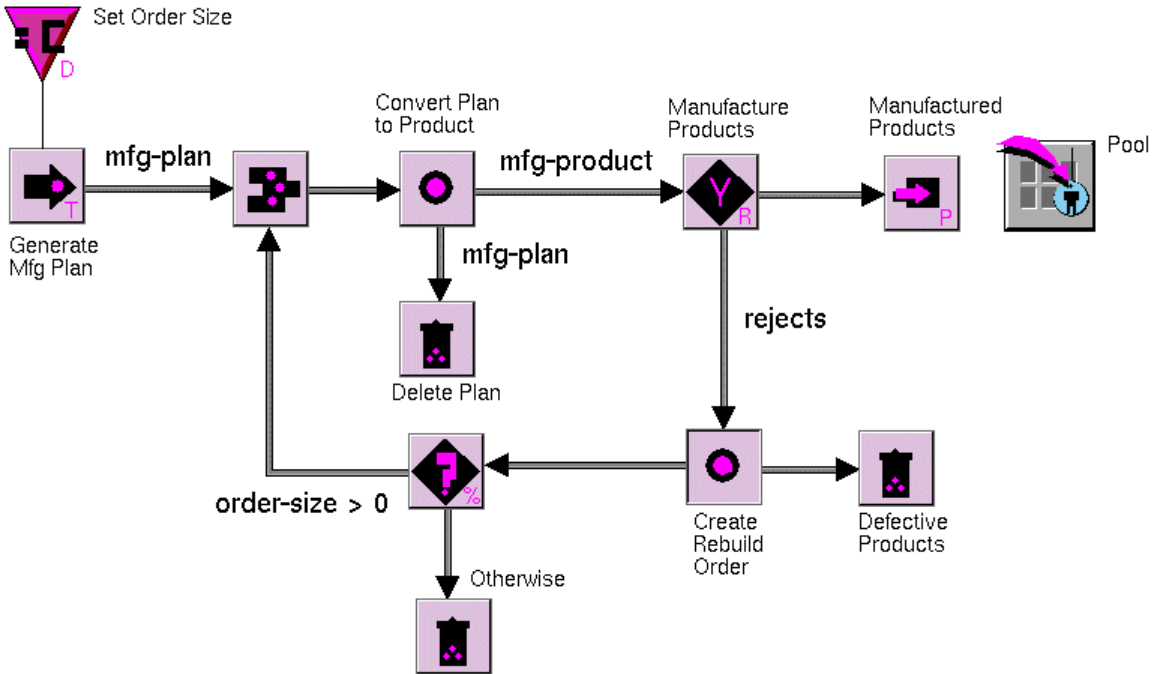
You only need to configure this attribute if you are copying attributes from a subtable.

By default, the block copies the exact value of the source attribute into the destination attribute; however, you can also apply a mathematical operator or function to the source attribute to compute the destination attribute.

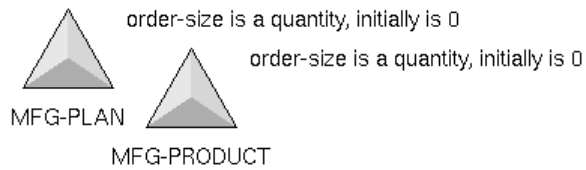
- e Choose the Operator from the available list, or configure the Operator to be FCT and configure the Function to be the name of a G2 function to apply.

Some examples of functions are: **max**, **min**, or **average**. For details, see Chapter 25 “Functions” in the *G2 Reference Manual*.

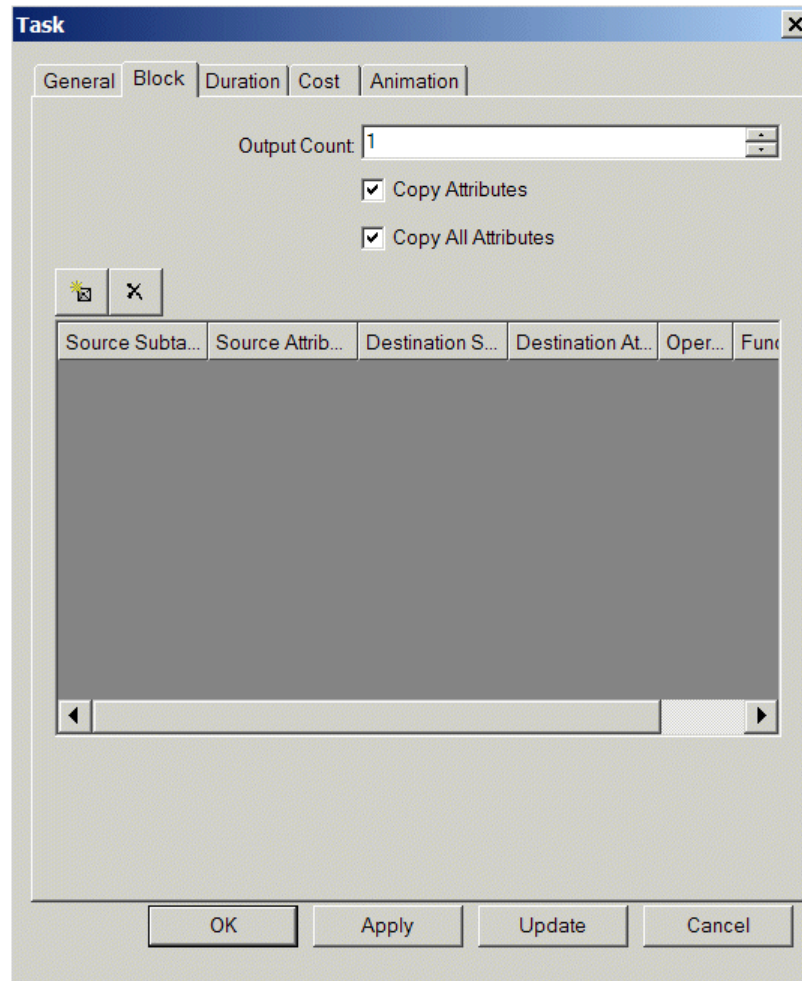
This example shows a manufacturing process that converts mfg-plan objects to mfg-product objects, then manufactures those products. The Convert Plan to Product task copies attributes from the mfg-plan to the mfg-product.



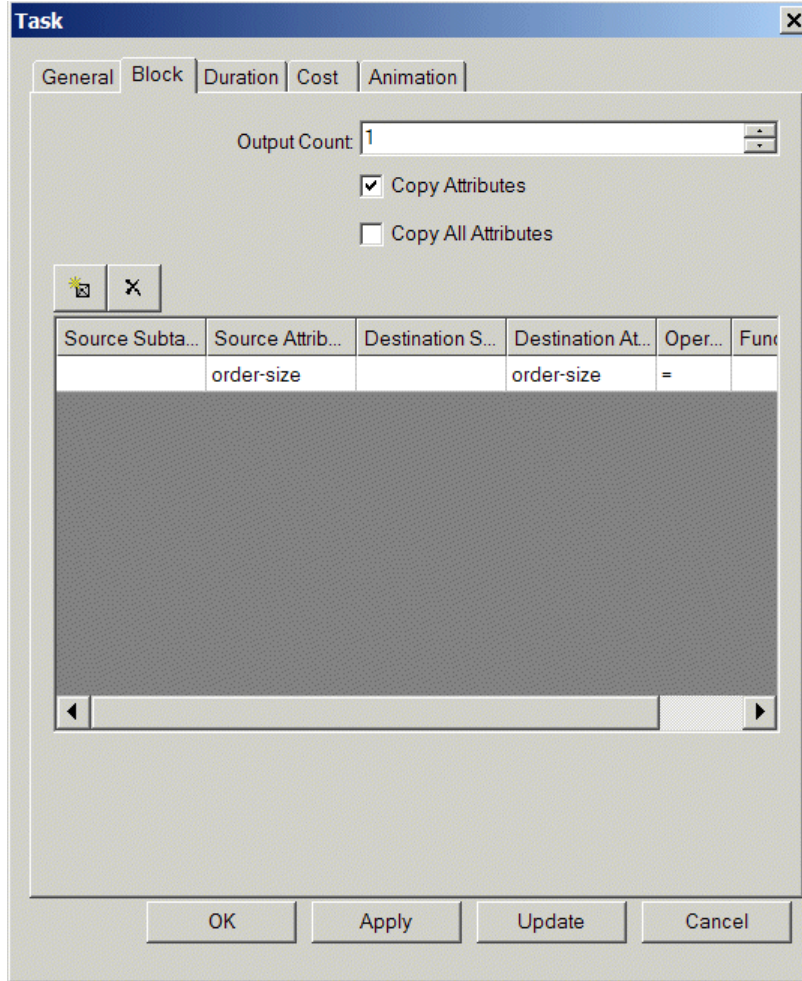
Here are the class definitions for the mfg-plan and mfg-product classes:



Here are two configurations of the Task block that copies attributes:

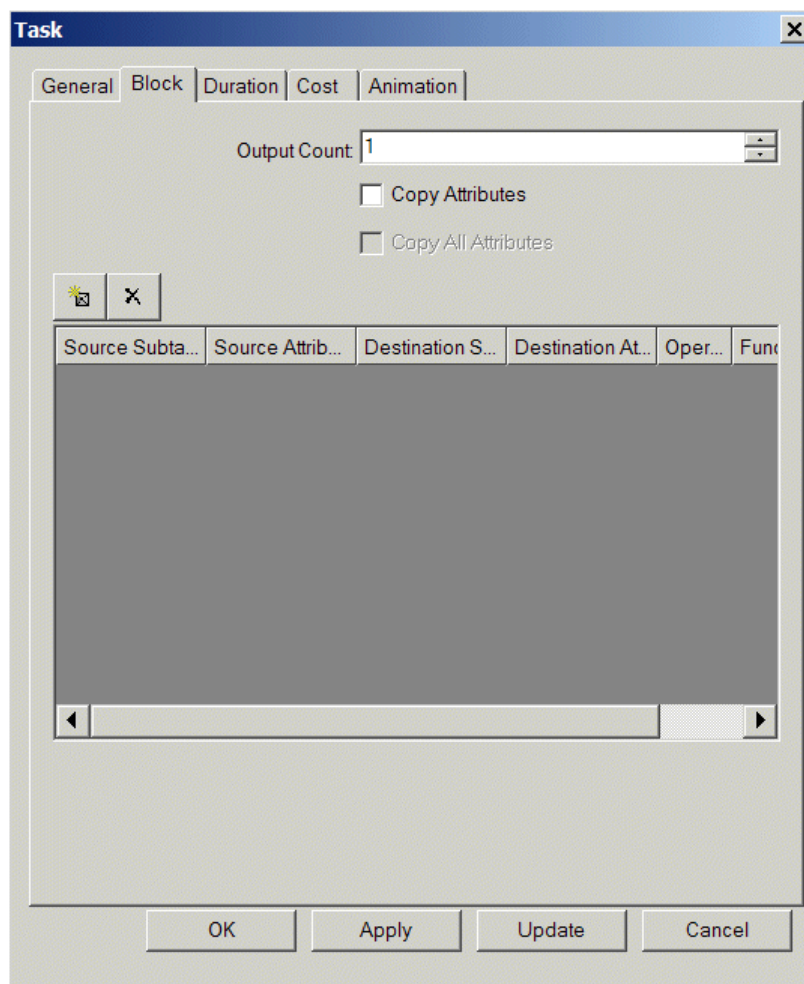


Copies the value of the order-size and time attributes, which are common.



Copies the value of the order-size only.

Specific Attributes



The specific attributes of the Task block are:

Attribute	Description
Output Count	The number of work objects to create for each activity and on each output path whose type is different from the input work object.
Copy Attributes	Enables the copying of attributes from the input to the output work object.

Attribute	Description
Copy All Attributes	Copies all attribute values whose names are common from the input to the output work object.
Source Subtable Source Attribute Destination Subtable Destination Attribute Operation Function	<p>When Copy Attributes is enabled, copies a specific Source Attribute from the input work object to a Target Attribute in the output work object, based on the specified Operator or Function.</p> <p>If the Source Attribute and Destination Attribute are in a subtable of the work object, configure the Source Subtable and Destination Subtable, for example, duration-subtable or cost-subtable.</p>

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for a Task block are:

Menu Choices	Description
Create Detail	Creates a detail subworkspace for the task for adding detail to the model. The default detail has a copy of the superior task with the same input and output paths and the same configuration as the superior task. The input and output paths are connected to connectors on the detail. When a task with a detail processes, the work objects flow through the blocks on the detail of the superior block before passing to the downstream block. See Creating Hierarchical Views .
Show Detail	If you have created detail for the task, this menu choice appears for showing the detail. See Creating Hierarchical Views .

Menu Choices	Description
Enable Detail	If the Task block has detail and the detail is disabled, this menu choice enables the detail so work objects flow to the blocks on the detail.
Disable Detail	If the task block has detail and the detail is enabled, this menu choice disables the detail so work objects flow through the top-level task.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Yield



The Yield block divides the input work object into two output objects, based on an attribute of the input work object and the yield mode. For example, you could use the Yield block to compute the manufacturing yield, based on the order size of the input work object, using a random triangular distribution to compute the yield.

You configure how the block splits the work object by configuring the mode. You can split the work object by configuring percentages, based on random values, a random triangular function, an attribute of the work object, or output path proportions.

The Yield block multiplies the percentages by the value of the specified attribute of the input work object. It then splits the work object into two. It then copies the computed yield into the specified attribute of one output work object. It copies the left-over yield into the attribute of the other work object. You choose which output path carries the rejects.

Configuring the Yield Mode

A Yield block provides the following operating modes:

Yield Mode	Description
Random Yield	You configure the minimum and maximum percentages as values between 0.0 and 1.0.
Random Triangular	You configure the minimum, maximum, and mode percentages as values between 0.0 and 1.0. By configuring a mode that is not equidistant between the two end-points, you can skew the percentages.
Work Object	You configure an attribute of the input work object that determines the yield.
Proportional	You configure the percentages on the output paths of the block, similar to the Branch block.
Custom	See Customization Attributes .

Configuring a Random or Random Triangular Yield

Suppose your model has a manufacturing yield of 70%, plus or minus 3%. You would use the **Random** yield mode and specify the minimum as 0.67 and the maximum as 0.73. The block would choose a random yield between 67% and 73%.

Alternatively, suppose you know that while your manufacturing process has a 70% yield plus or minus 3%, it is more likely to have a higher yield than a lower yield. In this case, you would use a **Random Triangular** yield mode and specify the minimum as 0.67, the mode as 0.71, and the maximum as 0.73. Given these values for a triangular function, the block would choose a random yield that is more often closer to 73% than it is to 67%.

To configure a fixed yield, configure the minimum and maximum to be the same value, for example, .70.

To configure a random or random triangular yield:

- 1 Create a work object class that defines an attribute upon which the yield is to be computed.

For example, you might define a subclass of **bpr-object** called **product** that defines an attribute called **order-size**. The Yield block will multiply the yield percentage by the value of this attribute to determine the yield.

- 2 Create a model that provides a value for the attribute of the work object.

For example, you might use a Change feed to feed random values into the attribute.

- 3 Display the properties dialog of the Yield block and click the Block tab.

- 4 Configure the Yield Mode to be **Random** or **Random Triangular**, depending on how you want the block to compute the yield.

- 5 Configure the Attribute to Split to be the attribute of the work object that will be multiplied by the yield percentage to determine the yield.

In the example, this attribute is **order-size**.

- 6 Configure the Minimum Random Value, Mode Random Value, and Maximum Random Value, depending on the mode you choose.

Each of these values is a number between 0.0 and 1.0. The block multiplies the yield by the value of the Attribute to Split attribute to determine the yield.

- 7 Choose the Choose Reject Path menu choice on the Yield block, then choose Select on the output path that should carry the defective products.

The other output path carries the work object whose yield is computed, based on the mode.

When you run the simulation, the Yield block copies the computed yield to the Attribute to Split attribute of the output work object that represents manufactured

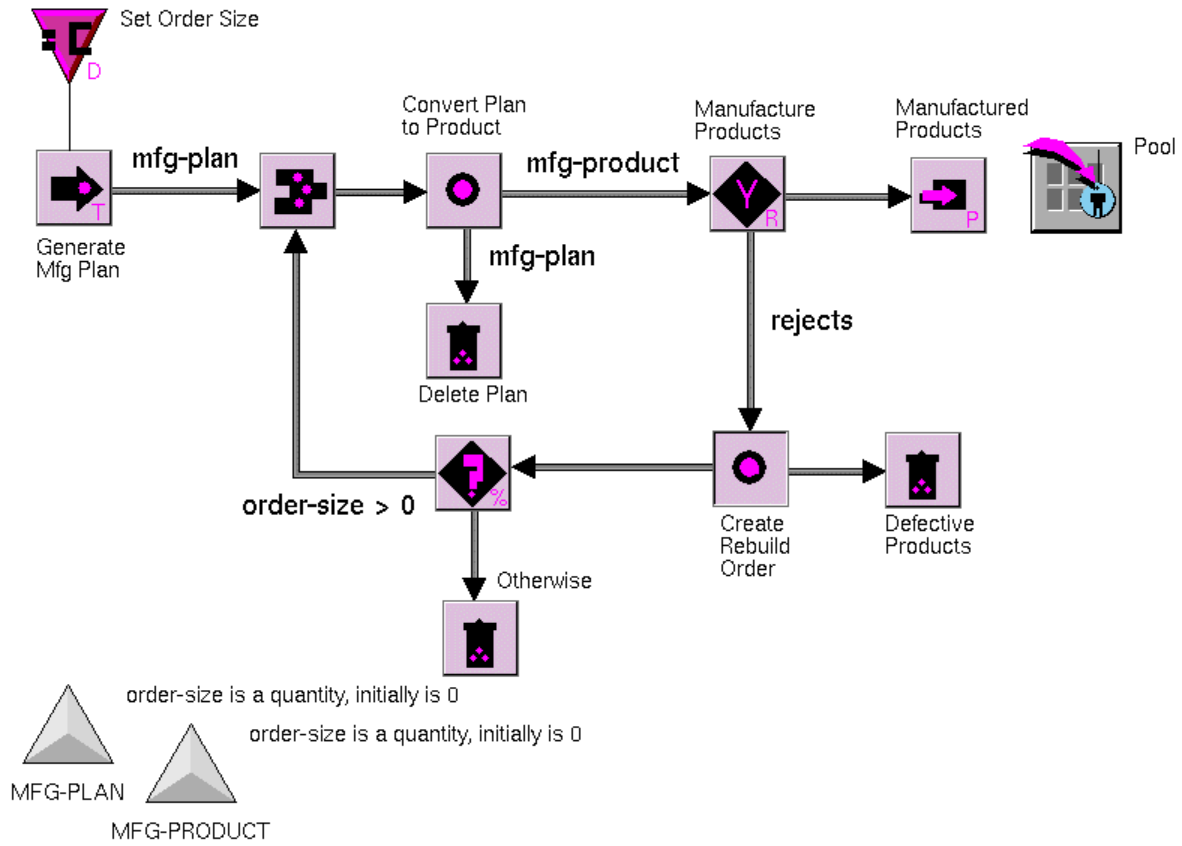
products. It copies the balance of the computed yield to the same attribute of the work object that represents defective products, which travels on the reject path. For example, suppose the yield percentage is 70% and the Attribute to Split is called **order-size**. If the order size of the input work object is 100, then the order size of the manufactured product will be 70, and the order size of the rejected product will be 30.

The following example manufactures products from manufacturing plans. The Change feed sets the **order-size**, using a random triangular function, and the Task block copies the current value of the **order-size** to the product it generates. For details, see the [Task block](#).

The Yield block computes the yield, based on the order size, using a random function. It then copies the computed yield to the **order-size** of the manufactured product and send it to a Store block, which stores it in a pool. It copies the balance of the computed yield to the **order-size** of the rejected product and sends it on the reject path.

The Create Rebuild Order task copies the current order size from the rejected product, deletes the rejected product, and generates a new manufacturing plan, based on the order size. The Branch block then sorts the manufacturing plans, based on order size and sends plans with a positive order size back to the Convert

Plan to Product task. The manufacturing process then begins again to compensate for the defective products.



Here is the Block tab of the Yield block, which uses a random function to compute a 70% yield, plus or minus 3%:

The image shows a software dialog box titled "Yield: Manufacture Products" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Block" tab is currently selected. The dialog contains several input fields and buttons:

- Yield Mode:** A dropdown menu set to "Random".
- Attribute To Split:** A text field containing "ORDER-SIZE".
- Minimum Random Value:** A spin box set to "0.67".
- Mode Random Value:** A spin box set to "0.5".
- Maximum Random Value:** A spin box set to "0.73".
- Work Object Yield Attribute Name:** An empty text field.
- Yield Value:** A spin box set to "0.0".

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Here is the Block tab of the Yield block, which uses a random triangular function to compute a 70% yield, plus or minus 3%, but skewed toward the higher yield rate:

The screenshot shows a dialog box titled "Yield: Manufacture Products" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Block" tab is selected. The settings are as follows:

- Yield Mode: Random Triangular (dropdown menu)
- Attribute To Split: ORDER-SIZE (text input)
- Minimum Random Value: 0.67 (spin box)
- Mode Random Value: 0.71 (spin box)
- Maximum Random Value: 0.73 (spin box)
- Work Object Yield Attribute Name: (empty text input)
- Yield Value: 0.0 (spin box)

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Configuring Yield Based on an Attribute of the Work Object

Suppose you have computed the yield elsewhere in the model and have copied it to an attribute of the input work object. Alternatively, suppose you want to feed the yield into an attribute of the work object, for example, using a Change feed configured to generate random numbers, based on a distribution.

To configure yield based on an attribute of the work object:

- 1 Create a work object class that defines an attribute upon which the yield is to be computed and an attribute that determines the yield.

For example, you might define a subclass of `bpr-object` called `mfg-product` that defines an attribute called `order-size` and an attribute called `yield`. The Yield block will multiply the value of the `yield` attribute by the value of the `order-size` to determine the yield.

- 2 Create a model that provides a value for each of these attributes of the work object.

For example, you might use Change feeds to feed random values into each attribute.

- 3 Display the properties dialog of the Yield block and click the Block tab.
- 4 Configure the Yield Mode to be Work Object.
- 5 Configure the Attribute to Split to be the attribute of the work object that will be multiplied by the yield percentage to determine the yield.

In the example, this attribute is `order-size`.

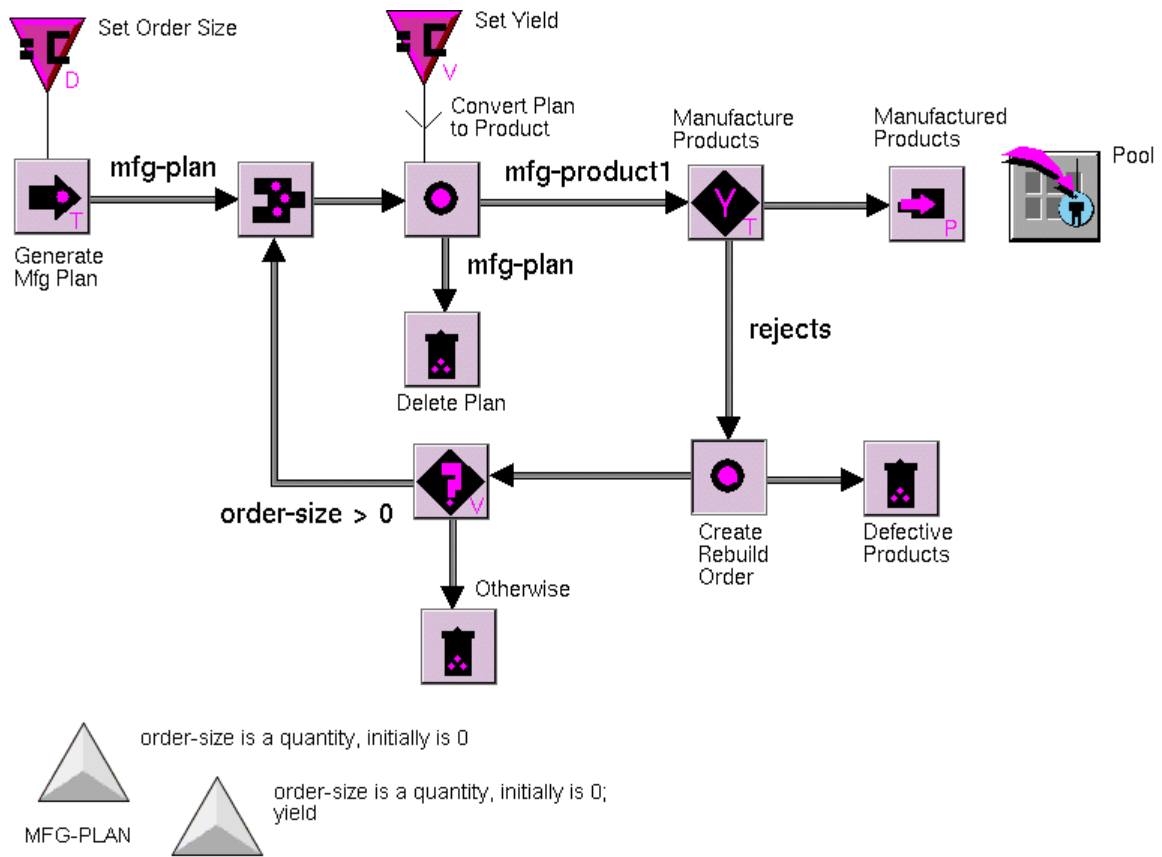
- 6 Configure the Work Object Yield Attribute Name to be the name of the attribute that defines the yield.

In the example, this attribute would be the `yield` attribute.

- 7 Choose the Choose Reject Path menu choice on the Yield block, then choose Select on the output path that should carry the defective products.

For example, suppose the value of the `yield` attribute is 0.70 and the Attribute to Split is called `order-size`. If the order size of the input work object is 100, then the order size of the manufactured product will be 70, and the order size of the rejected product will be 30.

This model is similar to the previous model except that the `mfg-product1` class defines a `yield` attribute and the `Yield` block uses this attribute as the yield percentage. The model uses a `Change` feed to compute a random value for the yield.



Here is the Block tab of the Yield block, which uses the **yield** attribute of the product as the yield percentage and the **order-size** as the attribute to split:

The screenshot shows a dialog box titled "Yield: Manufacture Products" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Block" tab is selected. The settings are as follows:

- Yield Mode: Work Object (dropdown menu)
- Attribute To Split: ORDER-SIZE (text field)
- Minimum Random Value: 0.67 (text field with up/down arrows)
- Mode Random Value: 0.71 (text field with up/down arrows)
- Maximum Random Value: 0.73 (text field with up/down arrows)
- Work Object Yield Attribute Name: YIELD (text field)
- Yield Value: 0.0 (text field)

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Configuring a Proportional Yield

You might know the exact proportion of the input work object that should be converted to manufactured versus defective products. In this case, you can configure the yield percentage as proportions on each output path of the Yield block. You can use this technique to model a manufacturing process that makes different grades of products, for example, low grade, medium grade, and high grade. In this case, you would add extra output paths to the Yield block, and it would split the initial order size proportionally between the output paths; the reject path carries the defective products.

To configure a proportional yield:

- 1 Create a work object class that defines an attribute upon which the yield is to be computed.

For example, you might define a subclass of `bpr-object` called `mfg-product` that defines an attribute called `order-size`. The Yield block will multiply the proportions specified for each output path by the value of the `order-size` to determine the yield for each output path.

- 2 Create a model that provides a value for this attribute of the work object.

For example, you might use Change feeds to feed random values into the attribute.

- 3 Create as many output paths as needed for the Yield block.

For example, to manufacture high, medium, and low grade products, you would add two output paths to the block, which results in three output paths for the manufactured products and one for the defective products.

- 4 Display the properties dialog of the Yield block and click the Block tab.

- 5 Configure the Yield Mode to be Proportional.

- 6 Configure the Attribute to Split to be the attribute of the work object that will be multiplied by the yield percentage to determine the yield.

In the example, this attribute is `order-size`.

- 7 Display the properties dialog for each output path, click the Branch tab, and configure the Branch Proportion.

The Branch Proportion is the proportion of the manufactured and defective product that each output path should carry. The Yield block multiplies the proportion by the value of the Attribute to Split attribute to determine the yield for each output path.

For example, suppose your manufacturing process has a 90% yield that results in equal amounts of low, medium, and high grade products. You could configure the Branch Proportion of the output paths to be 30, 30, and 30 for each manufactured product output path, and 10 for the reject path.

Alternatively, you can configure the Branch Proportion to be a number that represents a percentage, for example, .33, .33, .33, and .10.

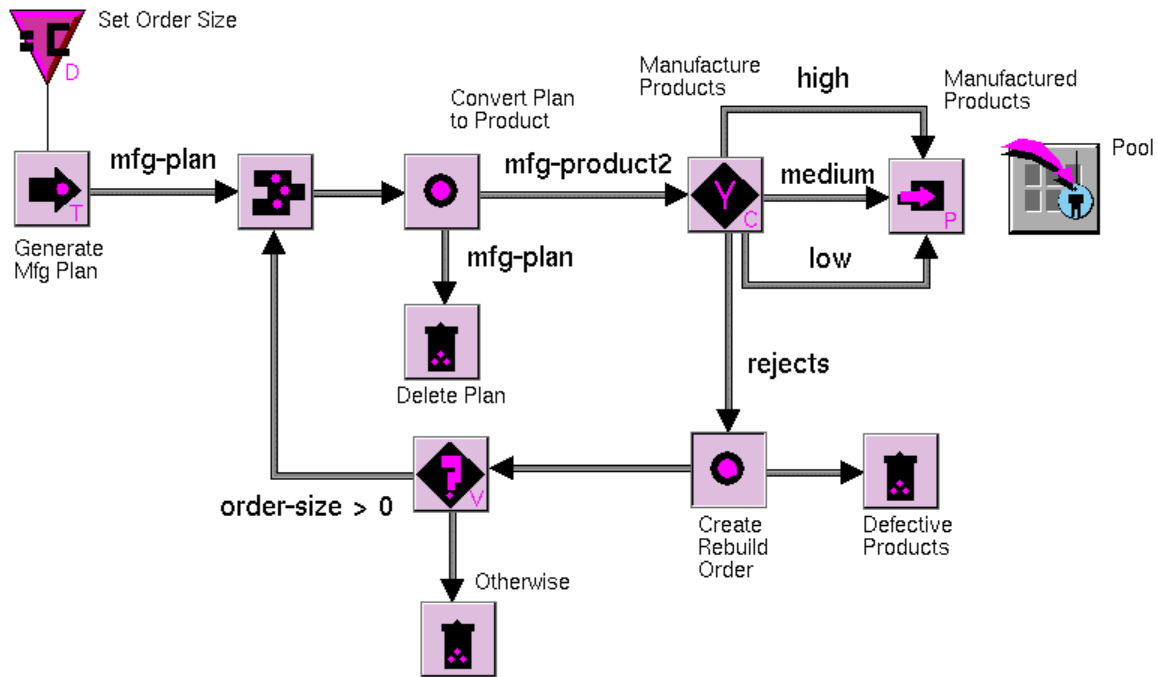
For details on how to configure the Branch Proportion, see [Branching Based on Proportion](#).

- 8 Choose the Choose Reject Path menu choice on the Yield block, then choose Select on the output path that should carry the defective products.

For example, suppose the Branch Proportion values are 30, 30, 30, and 10 for the low, medium, high, and reject paths, respectively, and the Attribute to Split is called `order-size`. If the order size of the input work object is 100, then the order

size for the low, medium, high, and reject path work objects will be 30, 30, 30, and 10, respectively.

This model is similar to the previous models except that it has three output paths for the manufactured products and a reject path, each of which specifies a proportion.



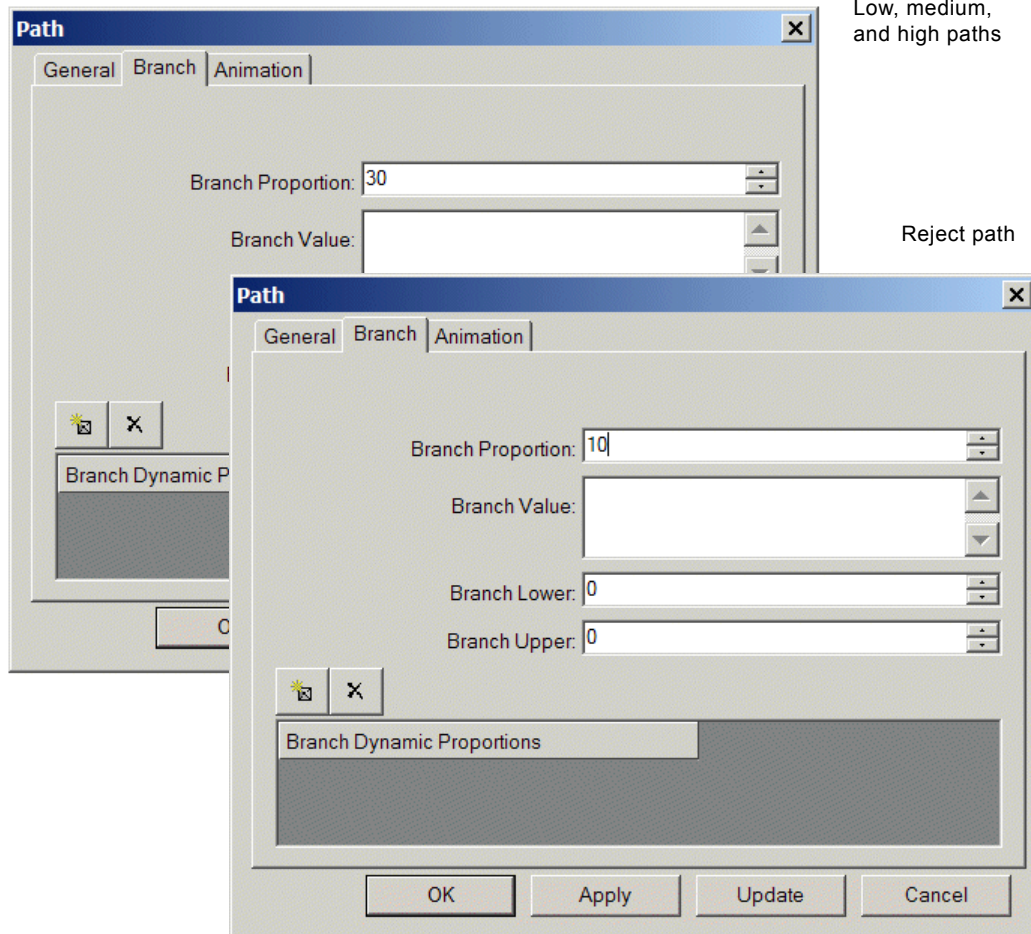
Here is the Block tab of the Yield block, which uses **Proportional** mode to compute the yield percentage and **order-size** as the attribute to split:

The image shows a software dialog box titled "Yield: Manufacture Products" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "Block" tab is selected. The dialog contains the following fields and controls:

- Yield Mode:** A dropdown menu set to "Proportional".
- Attribute To Split:** A text input field containing "ORDER-SIZE".
- Minimum Random Value:** A text input field containing "0.67" with up and down arrow buttons to its right.
- Mode Random Value:** A text input field containing "0.71" with up and down arrow buttons to its right.
- Maximum Random Value:** A text input field containing "0.73" with up and down arrow buttons to its right.
- Work Object Yield Attribute Name:** A text input field containing "YIELD".
- Yield Value:** A text input field containing "0.0".

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Suppose the model has a 90% yield, which is split evenly between low, medium, and high grade products. This figure shows one way to configure the Branch Proportion for the output paths:



Determining the Yield Value

The Yield block computes the current Yield Value for each input work object it receives, which is the yield percentage multiplied by the value of the Attribute to Split attribute of the work object. If the Yield block is configured to use Proportion mode and has more than two output paths, the Yield Value is the sum of the yield values for each output path that carries a manufactured product.

Specific Attributes

The image shows a software dialog box titled "Yield". It has a blue title bar with a close button (X) on the right. Below the title bar are five tabs: "General", "Block", "Duration", "Cost", and "Animation". The "General" tab is currently selected. The main area of the dialog contains several input fields and a dropdown menu:

- Yield Mode:** A dropdown menu with "Random" selected.
- Attribute To Split:** An empty text input field.
- Minimum Random Value:** A text input field containing "0".
- Mode Random Value:** A text input field containing "0.5".
- Maximum Random Value:** A text input field containing "1".
- Work Object Yield Attribute Name:** An empty text input field.
- Yield Value:** A text input field containing "0.0".

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attributes of the Yield block are:

Attribute	Description
Yield Mode	Specifies how the block computes the Yield Value. The options are: Random, Random Triangular, Work Object, proportional, and Custom. The default value is Random.
Attribute to Split	The name of an attribute of the input work object upon which the yield is to be computed. The block multiplies the yield percentage by the value of this attribute, then splits the value between the output work objects. You can use dot notation to refer to the attribute of a subobject, for example, <code>my-subtable.my-attr</code> .
Yield Value	(Metric) The computed yield for the current work object.

The mode-specific attributes of the Yield block are:

Mode	Attribute	Description
Random	Minimum Random Value	The minimum value for the random yield percentage.
	Maximum Random Value	The maximum value for the random yield percentage.
Random Triangular	Minimum Random Value	The minimum value for the random yield percentage.
	Mode Random Value	The mode value for the random yield percentage in a triangular function.
	Maximum Random Value	The minimum value for the random yield percentage.

Mode	Attribute	Description
Work Object	Work Object Yield Attribute Name	The name of an attribute of the input work object that is the yield percentage. You can use dot notation to refer to attributes of subobjects, for example, my-subtable.my-attr.
Proportional	N/A	Specifies that the Branch Proportion of each output path determines the yield percentages.
Custom	Yield Procedure Name	See Customization Attributes .

For information on the common block attributes, see [Common Attributes of Blocks](#).

Specific Menu Choices

The specific menu choices for a Yield block are:

Menu Choices	Description
Choose Reject Path	Identifies the path that carries defective product. Choose Select on the output path that carries the defective product to select the path.
Show Reject Path	Puts an indicator arrow next to the chosen reject output path.

For information on the common menu choices, see [Common Menu Choices for Blocks](#).

Customization Attributes

The customization attribute available in Developer mode for the Yield block is:

Attribute	Description
Yield Procedure Name	When the Mode is Custom , this attribute specifies the procedure that determines how the block computes yield. The default value is bpr-yield-custom-procedure .

You can customize the procedure that ReThink uses to compute yield. For more information, see the *Customizing ReThink User's Guide*.

Instruments Reference

Provides a description and example of each ReThink feed and probe.

Introduction	640
Common Attributes of Instruments	642
Common Menu Choices for Instruments	648
Acknowledge Message Probe	650
Average Probe	652
Copy Attributes Probe	659
Criteria Probe	667
Delete Message Probe	671
Delta Time Probe	673
Interval Sample Probe	678
Message Probe	683
Moving Average Probe	688
N-Dimensional Sample Probe	698
Parameter Probe	702
Sample Probe	708
Statistics Probe	716
Update Trigger Probe	724
Accumulate Feed	725
Attribute Feed	731
Change Feed	738

Copy Attributes Feed 755

Increment Feed 761

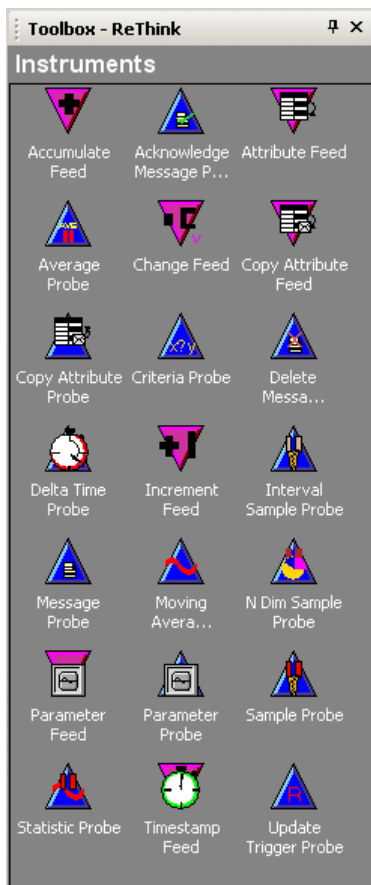
Parameter Feed 766

Timestamp Feed 771



Introduction

ReThink provides two types of instruments: feeds and probes. This chapter describes each instrument in the Instruments palette of the ReThink toolbox:



The chapter begins with sections describing the attributes and menu choices that are common for all instruments. It then describes each instrument including:

- A general description of the instrument.
- Specific uses of the instrument.
- An example.
- Specific attributes and menu choices.

This chapter is organized alphabetically by instrument type, as follows:

Probes	Feeds
Acknowledge Message Probe	Accumulate Feed
Average Probe	Attribute Feed
Copy Attributes Probe	Change Feed
Criteria Probe	Copy Attributes Feed
Delete Message Probe	Increment Feed
Delta Time Probe	Parameter Feed
Interval Sample Probe	Timestamp Feed
Message Probe	
Moving Average Probe	
N-Dimensional Sample Probe	
Parameter Probe	
Sample Probe	
Statistics Probe	
Update Trigger Probe	

For information on how to use instruments in a model, see [Using Instruments](#).

Common Attributes of Instruments

The following sections describe the attributes that are common to all ReThink feeds and probes. You access these attributes by displaying the properties dialog for the instrument. The properties dialog has these tabs:

Tab	Description
General	Parameters that are common to all instruments, such as the Label, Comments, and the class to which the instrument applies. Metrics related to the number of data points.
Instrument	Parameters and metrics that are specific to the particular instrument, for example, the source attribute name for a feed and the computed value for a probe.
Animation	Parameters for specifying animation colors for the instrument.

Following are examples for a Timestamp feed of the tab pages of the properties dialog that are common to all blocks: General and Animation. Following each dialog is a description of the attributes that appear in Modeler mode. Each attribute indicates whether it is a parameter (P) or metric (M).

For information on the attributes that are available in Developer mode, see the *Customizing ReThink User's Guide*.

General Tab for Feeds

Attribute	P/M	Description
Label	P	A text label for the feed, which appears as an attribute display with the instrument.
Comments	P	An area for entering a description of the feed and whatever other information you want to keep.
Apply to Class Name	P	The class to which the instrument feeds values. The default value is <code>bpr-object</code> , which feeds values into any type of work object.
Destination Attribute Name	P	The attribute of the specified class into which the instrument feeds values. Note that you cannot use dot notation to refer to subattributes.

Attribute	P/M	Description
Phase	P	When the feed sets its value. A Phase of Start sets the value before the connected block applies its duration to the simulation clock. A Phase of Stop sets the value after the connected block applies its duration. The default value for feeds is Stop .
Counter	M	The number of data points the instrument has received.
Error	M	A description of any error for the feed. See Debugging Blocks .

General Tab for Probes

The screenshot shows the 'Average Probe' dialog box with the 'General' tab selected. The fields are as follows:

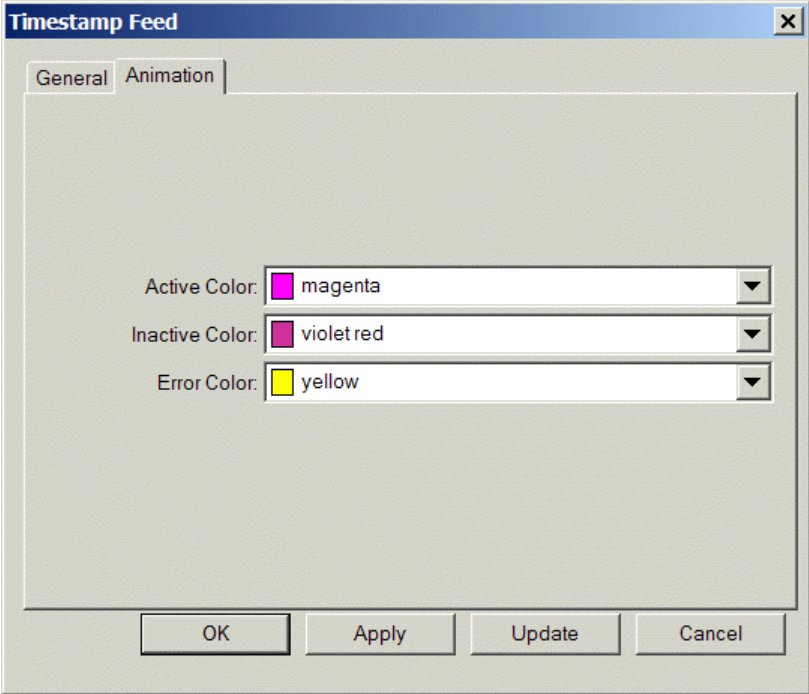
- Label:** An empty text input field.
- Comments:** A multi-line text area.
- Apply To Class Name:** A dropdown menu showing 'BPR-OBJECT'.
- Source Attribute Name:** A dropdown menu showing 'NONE'.
- Phase:** Two radio buttons, 'Start' and 'Stop'. The 'Stop' radio button is selected.
- Counter:** A text input field containing the number '0'.
- Error:** A text input field.

At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Update', and 'Cancel'.

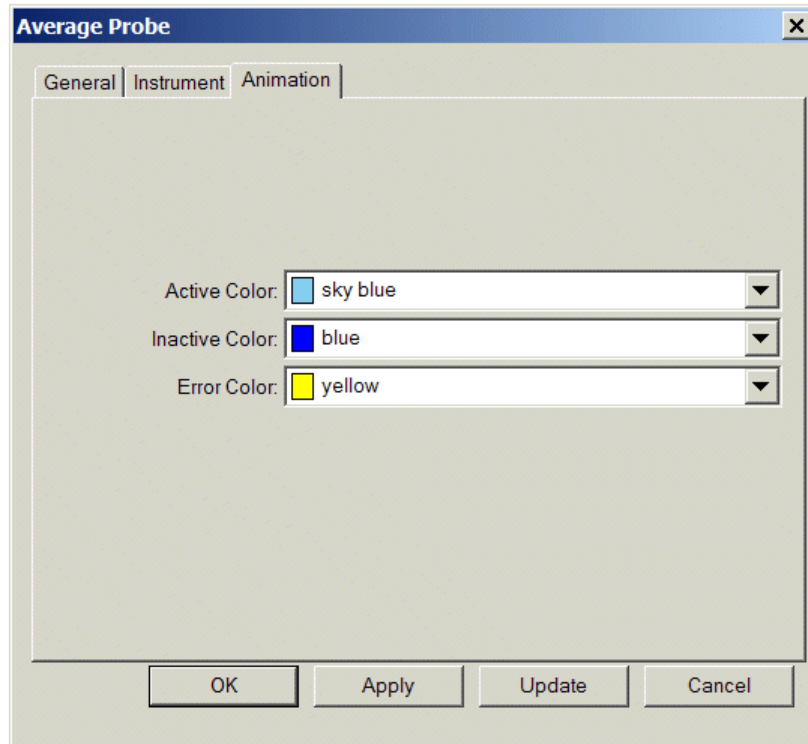
Attribute	P/M	Description
Label	P	A label for the probe, which appears as an attribute display with the instrument.
Comments	P	An area for entering a description of the probe and whatever other information you want to keep.
Apply to Class Name	P	The class from which the instrument probes values. The default value is <code>bpr-object</code> , which probes any type of work object.
Source Attribute Name	P	The attribute of the specified class from which the probe obtains values. Note that you cannot use dot notation to refer to subattributes.
Phase	P	When the probe computes its value. A Phase of Start computes the value before the attached block applies its duration to the simulation clock. A Phase of Stop computes the value after the attached block applies its duration. The default value for probes is Stop .
Counter	M	The number of data points the instrument has received.
Error	M	A description of any error for the probe. See Debugging Blocks .

Animation Tab for Feeds and Probes

Here is the Animation tab for feeds:



Here is the Animation tab for probes:



Attribute	P/M	Description
Active Color	P	The color of the instrument when it is currently feeding or probing values.
Inactive Color	P	The color of the instrument when it is idle.
Error Color	P	The color of the instrument when it is in an error state.

Common Menu Choices for Instruments

All instruments share a common set of menu choices. In addition, all probes share a number of common menu choices.

Common Menu Choices for Feeds and Probes

Feeds and probes both have the following menu choices:

Menu Choices	Description
Delete	Permanently deletes the instrument.
Transfer Clone	Cuts and copies (transfers) the instrument from one workspace to another, or copies (clones) the block to a workspace. ReThink copies all of the instrument's configured attributes.
Order	Lifts the object to the top or drops to the bottom.
Nudge	Moves the object up, down, left, or right by one pixel.
Align or Distribute	Align the left, center, right, top, middle, or bottoms of two or more selected objects. Distributes three or more selected objects vertically or horizontally.
Rotate or Flip	Rotates the object 90 degrees clockwise or counterclockwise or 180 degrees. Flips the object vertically or horizontally.
Properties	Displays the properties dialog for the instrument for configuring parameters and viewing metrics.
Show Scenario	Displays an indicator arrow next to the scenario that controls the instrument. This menu choice is only available when the scenario is active.
Create Connection	Creates a connection stub on the instrument. This menu choice is only available when the connection stub has been deleted.

Common Menu Choices for Probes

All probes have the following menu choices:

Menu Choices	Description
Create Chart	Creates a chart with an associated remote from the probe. The chart plots the probed value. You use the remote to configure the chart. This menu choice is available for most probes. See Creating a Chart Directly from a Probe .
Create Remote	Creates a ReThink remote, which keeps a history of the probed values. ReThink automatically creates a remote when you use the Create Chart menu choice to create a chart from a probe. You use this menu choice when you want to add a probed value to an existing chart. See Plotting Multiple Values on the Same Chart .
Show Remotes	Places an indicator arrow next to the remotes associated with the probe.
Show Chart	Places an indicator arrow next to the chart associated with the probe. This menu choice is available for most probes.

Acknowledge Message Probe



The Acknowledge Message probe acknowledges messages created in the Messages Browser by the Message probe.

Specific Attributes

A screenshot of the 'Acknowledge Message Probe' dialog box. The dialog has a title bar with the text 'Acknowledge Message Probe' and a close button. Below the title bar are three tabs: 'General', 'Instrument', and 'Animation'. The 'General' tab is selected. Inside the dialog, there is a checked checkbox labeled 'Acknowledge All Messages'. Below this, there is a 'Message Type:' label followed by a dropdown menu. Underneath that is a 'Category:' label followed by a text input field. At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Update', and 'Cancel'.

The specific attributes of the Acknowledge Message probe are:

Attribute	P/M	Description
Acknowledge All Messages	P	Whether to acknowledge all messages that arrive on the input path of the associated block.
Message Type	P	The class of message to acknowledge. By default, the probe acknowledges all messages that are instances of the <code>gevm-message</code> class or any subclass.
Category	P	The category of message to acknowledge. By default, the probe acknowledges all message categories.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

An Acknowledge Message probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Average Probe



The Average probe computes an average of all sampled values. The probe keeps track of the minimum and maximum values.

The Average probe is appropriate for computing the average of an attribute of any ReThink object, whose value does not depend on how long it has persisted, such as the average duration of a block.

You can also use the Average probe to compute the average value of a quantitative variable or parameter, then chart the value over time.

Determining the Value of the Probe

The Average probe defines these metrics to determine the value of the probe:

Attribute	Description
Average Value	The average of the sampled values.
Minimum Value	The minimum of the sampled values.
Maximum Value	The maximum of the sampled values.

The Average Value attribute appears as an attribute display of the probe.

Computing the Average of an Attribute Value

You attach an Average probe to a block, probe, or resource to compute the average value of any attribute of a block, activity, work object, path, instrument, or resource.

To compute the average of an attribute value:

- 1 Connect an Average probe to an object in the model whose attribute you want to average.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the class name of the object whose attribute you want to average.

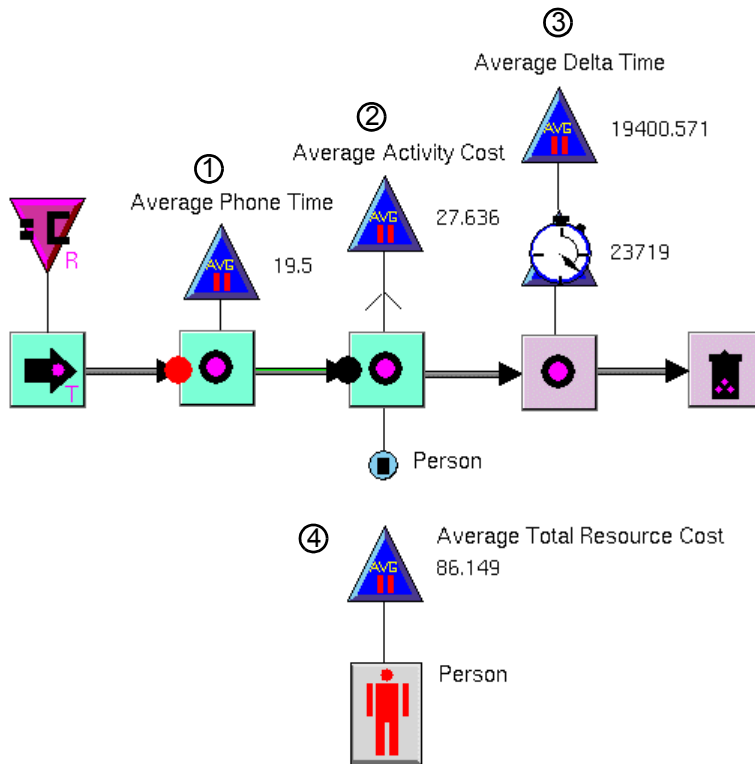
For example, if you are probing the Total Work Time attribute of a work object, specify `bpr-object` or a subclass of `bpr-object`.

- 3 Configure the Source Attribute Name to be the attribute of the class whose values you want to average.

You can choose from the list of attributes or enter your own value. You must specify the attribute name as a symbol. For example, if you are probing the Total Work Time or Total Cost attribute of a work object, choose `total-work-time` or `total-cost`.

- 4 Click the Instrument tab and configure the Precision to be the number of decimal points to round the computed Average Value.
- 5 To provide initial values for the average, configure the Sample Initial Value and Use Initial Value attributes, as needed.

This model shows how you use the Average probe to compute the average of several objects in the model:



- ① The average phone-time of a sales-call.
- ② The average cost of a bpr-activity.
- ③ The average delta-time of a bpr-instrument.
- ④ The average total-cost of a bpr-resource.

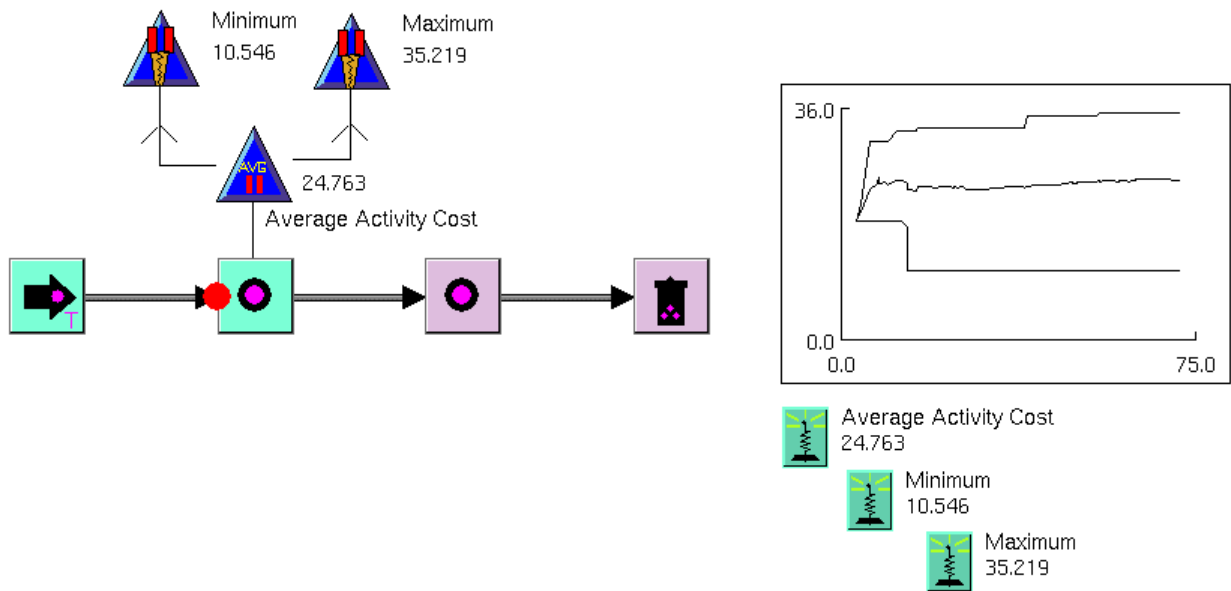
Plotting the Minimum and Maximum Values

The Average probe computes the Minimum Value and Maximum Value of the probed value, which you can plot.

To plot the minimum and maximum values of an Average probe:

- 1** Create and configure an Average probe to compute the average value of an attribute of the model.
For details, see [Computing the Average of an Attribute Value](#).
- 2** Attach two Sample probes to the Average probe whose minimum and maximum values you want to plot.
- 3** On the General tab of the properties dialog for each Sample probe, configure the Apply to Class Name to be `bpr-instrument`.
- 4** Configure the Source Attribute Name of one Sample probe to be `minimum-value`.
- 5** Configure the Source Attribute Name of the other Sample probe to be `maximum-value`.
- 6** Choose Create Chart on the Average probe that computes the average of an attribute of the model.
- 7** Choose Create Remote on the Sample probes that compute the minimum and maximum of the average value.
- 8** To add the new remotes to the chart, for each remote, choose Add Remote on the chart, then choose Select on the remote.
- 9** Display the properties dialog for each remote and, on the Chart tab, configure the Line Color of each plot.

The following figure shows how to plot the minimum and maximum values of an Average probe that computes the average cost of an activity:



Charting the Average of Quantitative Parameters

You can attach an Average probe to a quantitative parameter, then chart the value over time. You can use this feature in conjunction with a Parameter feed, which allows you to feed the value of a quantitative parameter into an attribute of the model, and a Parameter probe, which allows you to probe an attribute of the model that comes from a quantitative parameter.

Tip You can also conclude the current values when an attribute of the model changes by creating a rule. For an example, see [Branching Based on Rules that Set the Attribute Value](#).

To chart the average of a quantitative parameter:

- 1 Create a model that probes the value of a quantitative parameter created from an attribute of the model.

For details, [Parameter Probe](#).

- 2 Attach an Average probe to the quantitative parameter.
- 3 Display the properties dialog and configure the Apply to Class Name attribute of the Average probe to be quantitative-parameter.

Leave the Source Attribute Name blank.

- 4 Choose Create Chart on the Average probe to create a chart and associated remote.

For more information on...

See...

Examples that use a Sample probe [Charting Quantitative Parameters.](#)

Configuring the Parameter probe [Parameter Probe.](#)

Specific Attributes

The specific attributes of the Average probe are:

Attribute	P/M	Description
Use Initial Value	P	Whether to use the Sample Initial Value as the first value.
Sample Initial Value	P	The initial value to use for the Average Value.
Precision	P	The number of decimal places to round the Average Value.

Attribute	P/M	Description
Average Value	M	The average of the sampled values, rounded to the number of decimal places specified by Precision.
Maximum Value	M	The maximum value of the sampled values.
Minimum Value	M	The minimum value of the sampled values.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

An Average probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Copy Attributes Probe



The Copy Attributes probe copies attributes *from* the object to which the probe applies *to* a destination object. You can use the Copy Attributes probe to “roll up” metrics computed on a detail to the higher-level Task block. For example, you might want to “roll up” the Total Cost of the task on the detail to the superior task.

You can configure the Copy Attributes probe to copy the exact value of an attribute, or you can configure it to use an operator or function. For example, you might want to sum the order size of each work object that the task on the detail processes and store the sum in an attribute of the superior task.

Compare the Copy Attributes probe with the Copy Attributes feed, which copies attributes *from* a source object *to* the object to which the feed applies.

Rolling Up Metrics from the Detail to the Superior Task

To “roll up” metrics computed in objects on a detail to the superior task, you attach a Copy Attributes probe to a block or resource on the detail and configure it to copy attributes from an object on the detail to the superior task. You configure the class to which the probe applies, which defines the source object, and the destination class.

For example, you might want to create a subclass of the Task block with additional attributes that the model copies from objects on the detail to the superior task.

To roll up metrics from the detail to the superior task:

- 1 Create a subclass of the Task block with the attributes you want to “roll up” from objects on the detail.

For example, you might create a subclass of the Task block called `bpr-task-1`, which defines the attributes `total-cycles` and `total-products-built`. The model copies the `total-starts` from the task on the detail and copies it into the `total-cycles` attribute of the superior task. It also sums the value of the `order-size` of the work object on the detail and copies it into the `total-products-built` attribute of the superior task.

For information on how to create a subclass, see [Customizing Blocks](#).

- 2 Create a model that uses the subclass of Task block that you created and configure its detail.
- 3 Connect a Copy Attributes probe to an object on the detail, for example, a block or resource.
- 4 On the General tab of the properties dialog, configure the Apply to Class Name to be the class that triggers the probe to copy attributes.

The Apply to Class Name class is the source class from which the Copy Attributes probe copies attributes.

For example, to copy attributes from the Task block, configure the Apply to Class Name to be `bpr-task`. To copy attributes from the work object on the input path of the attached block, configure the Apply to Class Name to be `bpr-object` or a subclass.

- 5 Click the Instrument tab and configure the Destination Class Name to be the class name of the object to which the Copy Attributes probe copies attributes. In the example, the Destination Class Name would be `bpr-task-1`.
- 6 For each attribute to copy, create and configure a row in the List of Operations group, as follows:

- a Click Add Row to create a new row.
- b Configure the Source Subtable to be the name of the subtable in which the attribute to copy is defined, if any.

The Source Subtable corresponds to each tab page on which the attribute to copy appears, for example, `duration-subtable` or `cost-subtable`. If the attribute appears on a tab page other than the Duration or Cost tab, leave the Source Subtable blank.

- c Configure the Source Attribute to be the name of the attribute of the input work object to copy.

The source attribute is a user-defined attribute such as `order-size`.

- d Configure the Destination Subtable to be the name of the subtable in which the copied attribute appears.

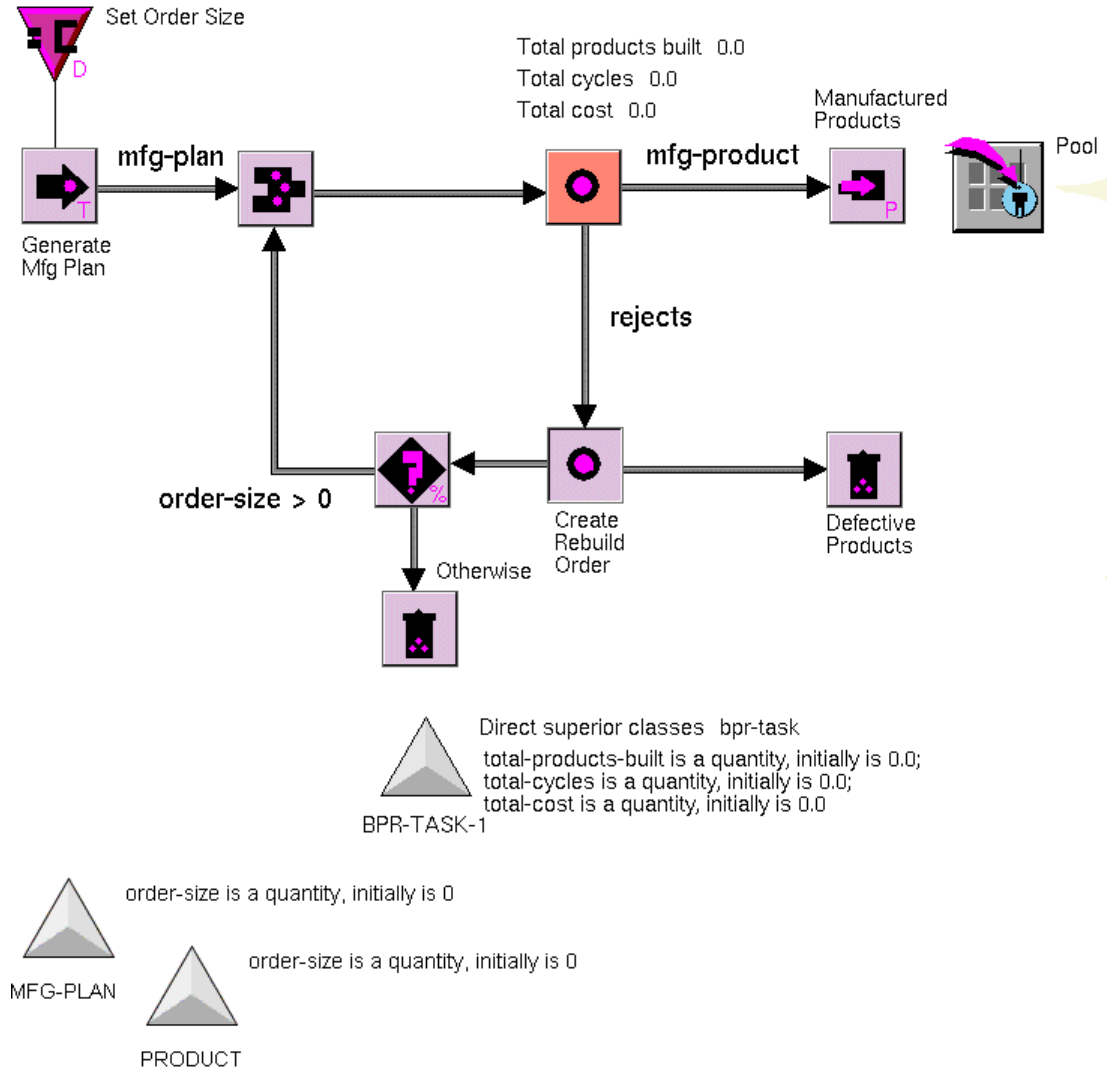
You only need to configure this attribute if you are copying attributes from a subtable.

By default, the probe copies the exact value of the source attribute into the destination attribute; however, you can also apply a mathematical operator or function to the source attribute to compute the destination attribute.

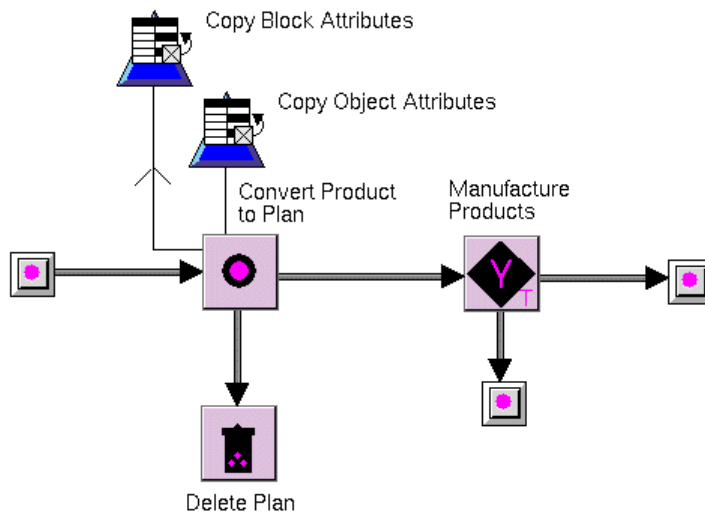
- e Choose the Operator from the available list, or configure the Function to be the name of a G2 function to apply.

Some examples of functions are: **max**, **min**, or **average**. For details, see Chapter 25 “Functions” in the *G2 Reference Manual*.

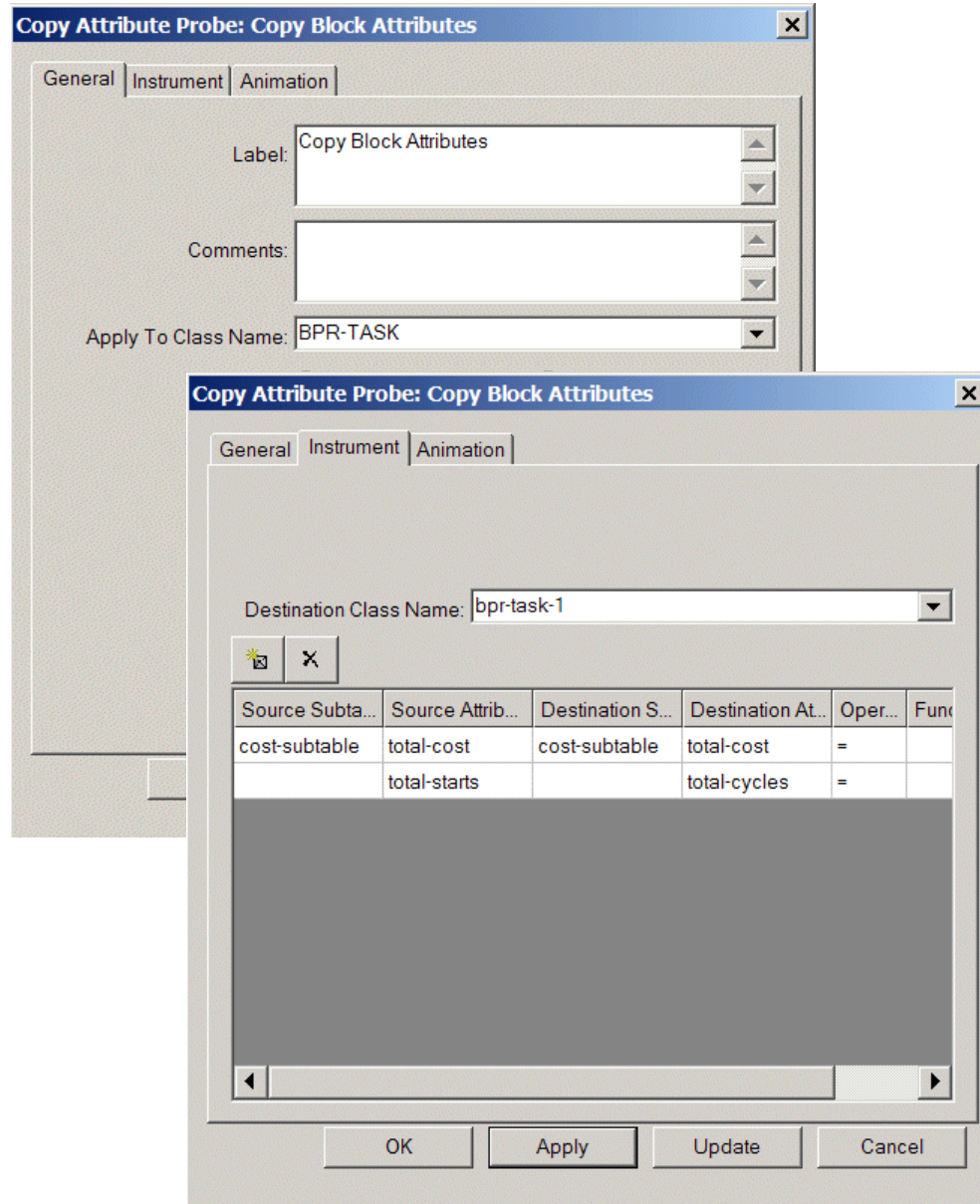
The following example shows how to “roll up” metrics from the detail of a user-defined Task block to the superior task. The **bpr-task-1** class defines three class-specific attributes, **total-products-built**, **total-cycles**, and **total-cost**, which appear as attribute displays above the superior task. The model feeds the **order-size** into each **mfg-plan**, then converts it into a **product**, copying the order size, using the Task block. For details, see the [Task block](#).



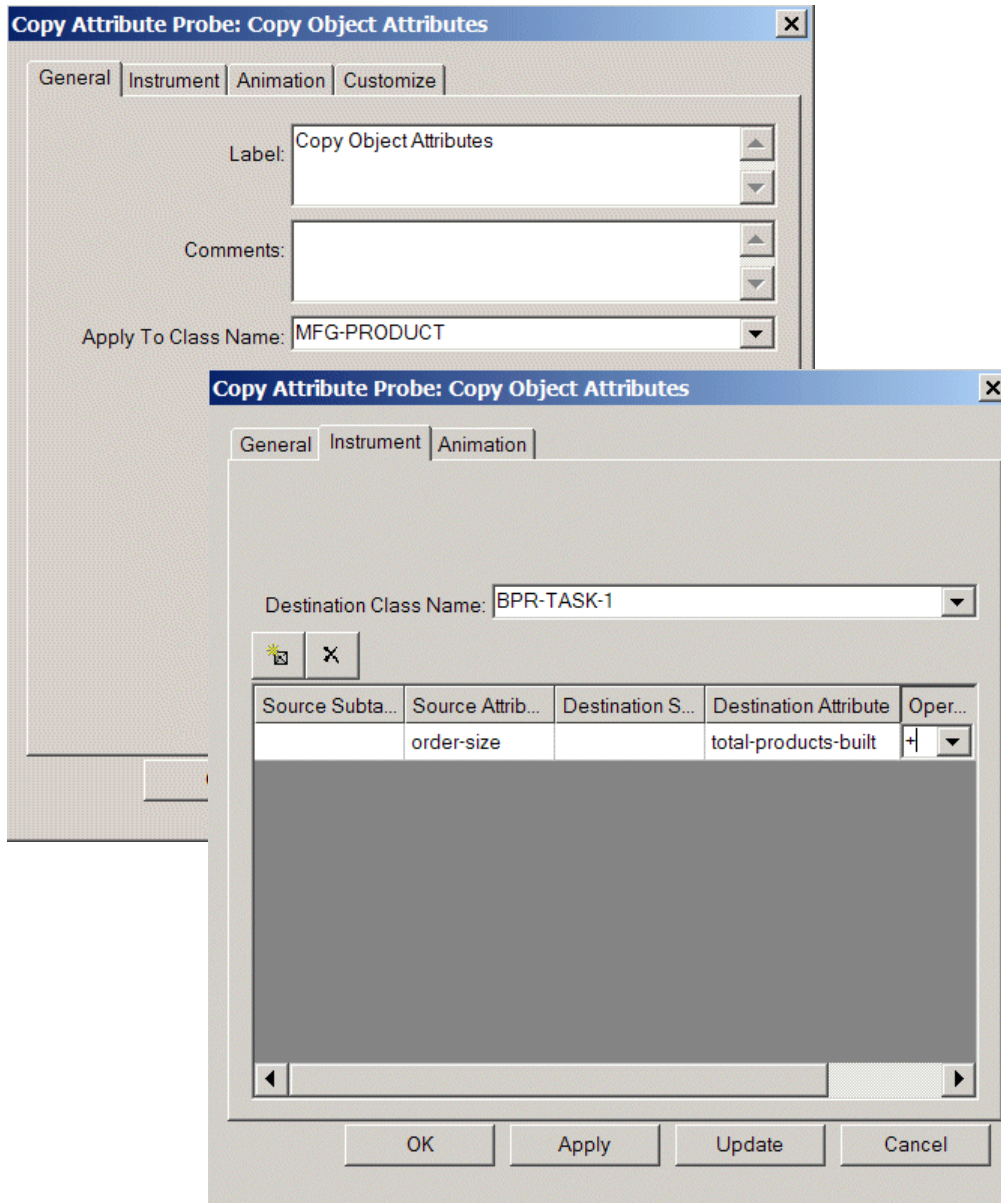
On the detail of the user-defined Task block are two Copy Attribute probes. The Copy Block Attributes probe copies attributes from the Task block on the detail to the superior task, and the Copy Object Attributes probe copies attributes from a work object on the detail to the superior task. The Yield block computes the yield and sends manufactured products out one path and defective products out the other. For details, see the [Yield block](#).



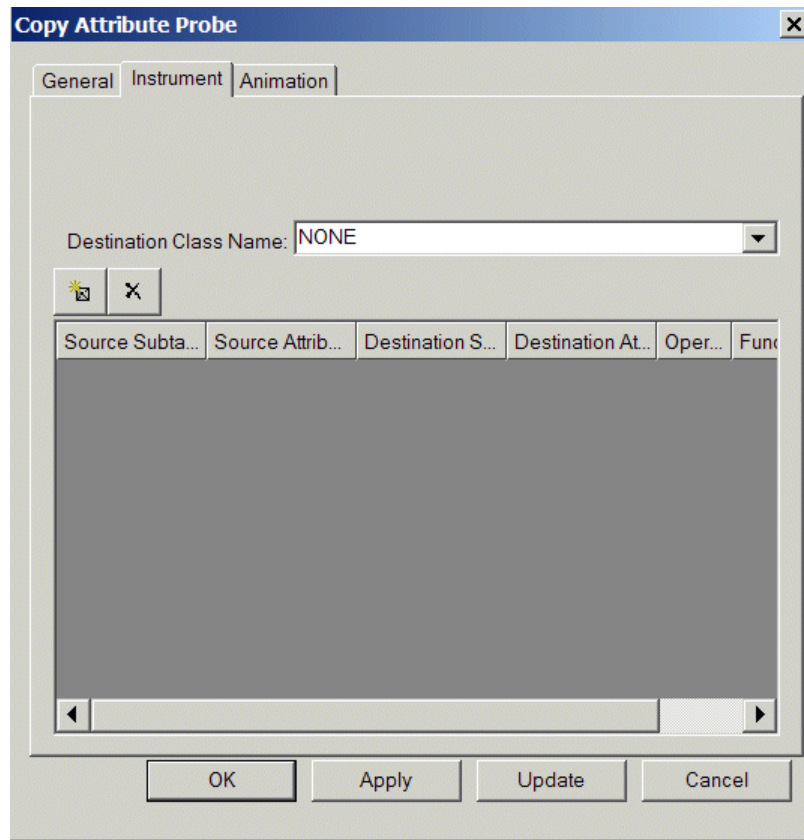
The Copy Block Attributes probe specifies Apply to Class Name as `bpr-task`, which is the source object, and `bpr-task-1` as the Destination Class Name. It copies the `total-cost` on the `cost-subtable` of the source object (`bpr-task`) to the `total-cost` on the `cost-subtable` of the destination object. It also copies the `total-starts` to the `total-cycles` of the destination object. It uses the `=` operator, the default, to copy the exact value for both operations.



The Copy Object Attributes probe specifies Apply to Class Name as mfg-product, which is the source object, and bpr-task-1 as the Destination Class Name. It copies the order-size of the source object (mfg-product) to the total-products-built of the destination object. It uses the + operator to sum the values.



Specific Attributes



The specific attributes of the Copy Attributes probe are:

Attribute	P/M	Description
Destination Class Name	P	The class name of the object whose destination attributes the probe updates. The probe copies attributes from the source object specified in the Apply to Class Name attribute.
Source Subtable Source Attribute Destination Subtable Destination Attribute Operation Function	P	The list of operations to perform when the Copy Attributes probe updates. It copies the Source Attribute of the source object to the Destination Attribute of the destination object, based on the specified Operator or Function. If the Source Attribute and Destination Attribute are in a subtable of the work object, you can configure the Source Subtable Name and Destination Subtable Name, for example, duration-subtable or cost-subtable.

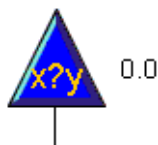
For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Copy Attributes probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Criteria Probe



You use a Criteria probe to compare a sample value in the model against criteria you configure in the probe to determine the percentage of time the sampled value meets the criteria. You configure the value to compare and the operator to use for the comparison. For example, you would use a Criteria probe to determine the percentage of time that the total cost of a work object goes above a certain value.

Determining the Value of the Probe

The Criteria probe defines these metrics to determine the value of the probe:

Attribute	Description
Criteria True Count	The number of times the test was true.
Criteria True Percent	The percentage of times that the test was true.

The Criteria True Percent attribute appears as an attribute display of the probe.

Comparing Sampled Values Against a Criteria

To compare sampled values against a criteria:

- 1 Connect a Criteria probe to a block, instrument, or resource in the model.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the class name of the object whose attribute you want to compare against the criteria.

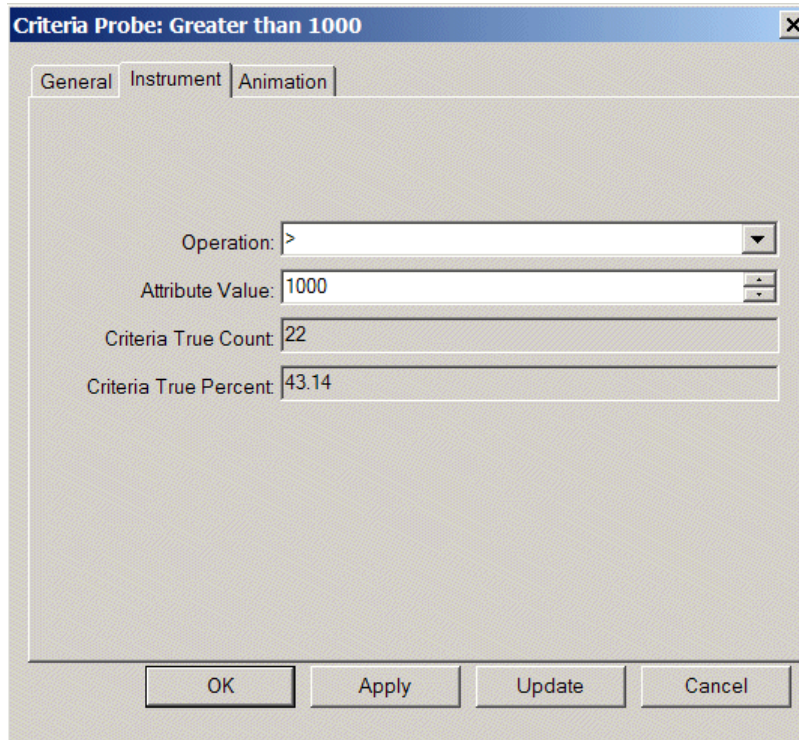
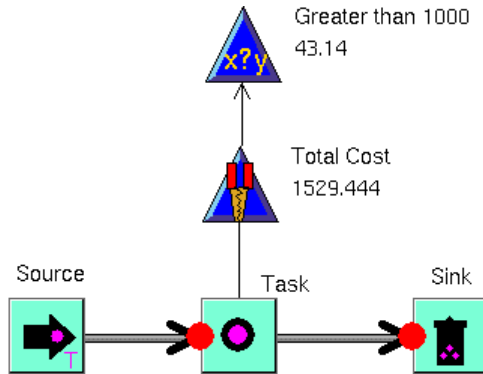
For example, if you are probing the Total Work Time of a work object, specify `bpr-object` or a subclass of `bpr-object`.

- 3 Configure the Source Attribute Name to be the attribute of the class whose value you want to compare against the criteria.

You can choose from the list of attributes to probe or enter your own value. If you enter your own attribute name, you must use a symbol. For example, if you are probing the Total Work Time or Total Cost attribute of a work object, choose `total-work-time` or `total-cost`.

- 4 Click the Instrument tab and configure the Attribute Value to be the value with which to compare the value of the Source Attribute Name.
- 5 Configure the Operation to be the operator to use for comparison.

This example uses a Sample probe to sample the Total Cost of the work object. It then uses a Criteria probe to probe the **sample-value** of the bpr-instrument to test the percentage of time that the total cost exceeds 1000. In this example, the total cost is greater than 1000 approximately 40% of the time.



Specific Attributes

The screenshot shows the 'Criteria Probe' dialog box with the 'General' tab selected. The 'Operation' dropdown is set to '=', 'Attribute Value' is 0, 'Criteria True Count' is 0, and 'Criteria True Percent' is 0.0. The buttons at the bottom are OK, Apply, Update, and Cancel.

The specific attributes of the Criteria probe are:

Attribute	P/M	Description
Operation	P	The operation to use when comparing the Attribute Value against the value of the Source Attribute name of the probe. The default value is =.
Attribute Value	P	The value to use for the comparison.
Criteria True Count	M	The number of times the criteria was true since the start of the simulation.
Criteria True Percent	M	The percentage of time that the criteria was true since the start of the simulation.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Criteria probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Delete Message Probe



The Delete Message probe deletes messages created in the Messages Browser by the Message probe.

Specific Attributes

The screenshot shows a dialog box titled "Delete Message Probe" with a close button (X) in the top right corner. The dialog has three tabs: "General", "Instrument", and "Animation". The "General" tab is selected. Inside the dialog, there is a checked checkbox labeled "Delete All Messages". Below this, there are two input fields: "Message Type:" followed by a dropdown menu, and "Category:" followed by a text input field. At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attributes of the Delete Message probe are:

Attribute	P/M	Description
Delete All Messages	P	Whether to delete all messages that arrive on the input path of the associated block.
Message Type	P	The class of message to delete. By default, the probe deletes all messages that are instances of the <code>gevm-message</code> class or any subclass.
Category	P	The category of message to delete. By default, the probe deletes all message categories.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

An Delete Message probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Delta Time Probe



The Delta Time Probe compares a timestamp of an object with the current time. You use this probe to measure the time change from one event in the model to another, which is called the “delta time.” For example, you can probe the creation time of an object and compare it to the current time at the end of the process to obtain the cycle time of the overall process.

You often use the Delta Time probe in conjunction with a Timestamp feed to compare the current time with a timestamp that you feed into the model.

You can configure the time unit for the Delta Time probe, to report the delta time in the time unit of the model, for example, hours or days.

Determining the Value of the Probe

The Delta Time probe defines an attribute named Delta Time, which is a quantitative parameter that keeps a history of the delta time values. The value of this attribute is the current delta time. The Delta Time attribute appears as an attribute display of the probe.

Computing the Cycle Time

The Delta Time probe is automatically configured to probe the creation time of the work object that flows through the block to which it is attached. Thus, you can easily compute the cycle time of a work object from the time it was created to a specified point in time in the model.

To probe the creation time of a work object:

- 1 Attach a Delta Time probe to a block.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to specify the work object whose creation time you are probing.

By default, the probe computes the delta time in seconds.

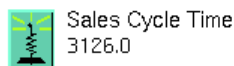
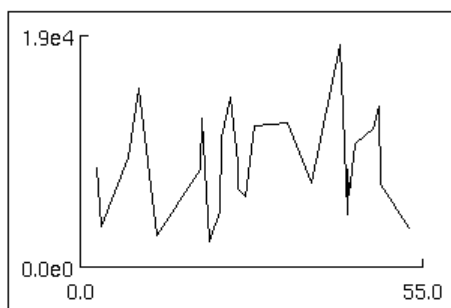
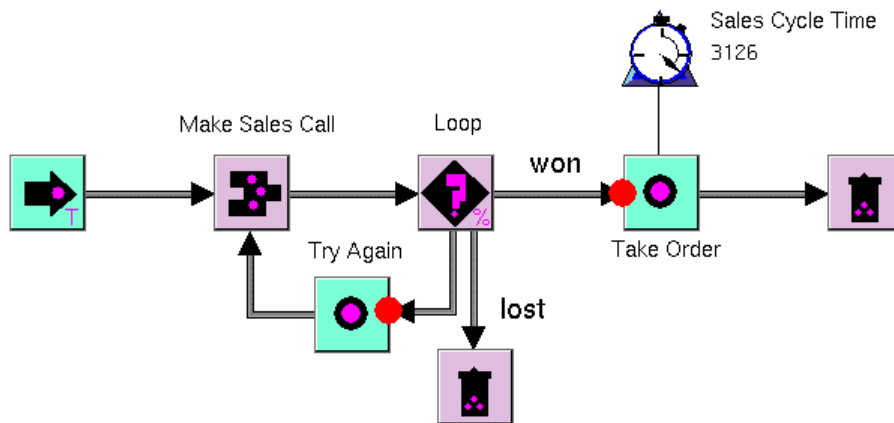
- 3 Click the Instrument tab and configure the Time Unit to use for computing the cycle time, for example, minutes, hours, days, or weeks.

For example, if you configure the duration of blocks in the model in hours, set the Time Unit to 1 hour, the default.

- 4 Configure the Precision to be the number of decimal points to round the Delta Time value.

You use the default value for the Source Attribute Name to probe the creation-time of the work object.

This example probes the creation time of a **sales-call** object to compute the sales cycle time. The model makes a sales call and branches the sales call, based on whether the sale is won or lost or whether another call must be made. The variation in sales cycle time is due to the deviations in the durations of each of the blocks.



For information on how to create charts from probes, see [Charting Performance Metrics](#).

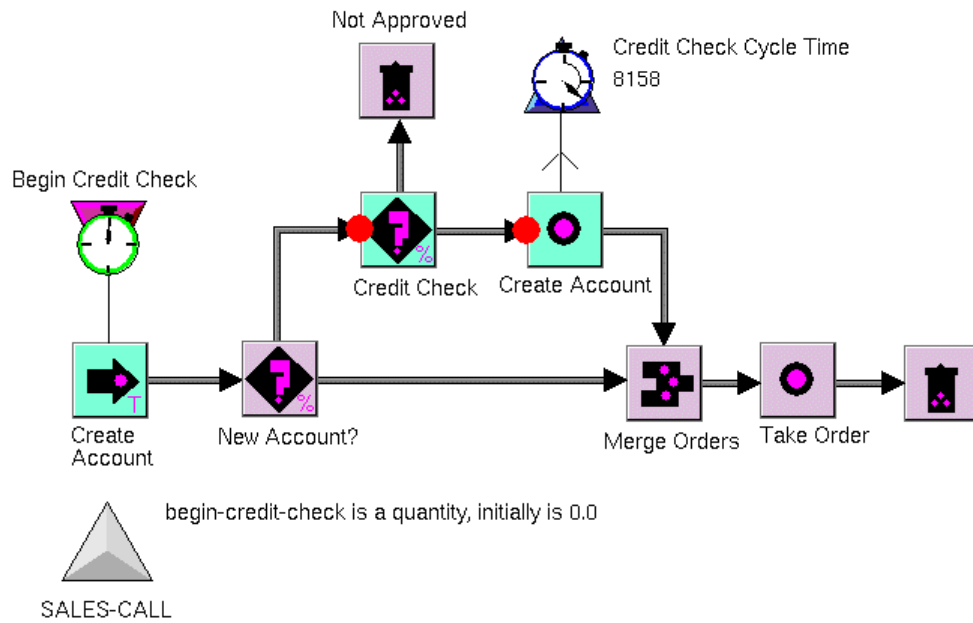
Computing a Partial Cycle Time

Rather than comparing the creation time of an object with the current time, you might want to compare a timestamp that you feed into the model with the current time to compute a partial cycle time. To do this, you use a Timestamp probe in conjunction with a Delta Time probe.

To probe a timestamp that you feed into the model:

- 1 Attach and configure a Timestamp feed to a block in the model where you want to feed a timestamp.
For details, see [Feeding a Timestamp into a Work Object of the Model](#).
- 2 Create a class definition for a work object with a class-specific attribute that holds the timestamp.
For details, see [Creating a New Class of Work Object](#).
- 3 Attach a Delta Time probe to a downstream block where you want to probe the timestamp.
- 4 On the General tab of the properties dialog, configure the Apply to Class Name attribute to be the work object whose timestamp you are probing.
- 5 Configure the Source Attribute Name to refer to the class-specific attribute of the work object that holds the timestamp.
- 6 Click the Instrument tab and configure the Time Unit to use for computing the cycle time.
- 7 Configure the Precision to be the number of decimal points to round the Delta Time value.

Here is a credit check approval process. To monitor the cycle time of the credit approval process, you feed a timestamp into the **sales-call** object at the beginning of the process and probe it at the end of the subtask to determine the credit check cycle time. The Timestamp instrument feeds a timestamp into the **begin-credit-check** attribute of the **sales-call** object, which the Delta Time probe then probes.



Delta Time Probe: Credit Check Cycle Time

General | Instrument | Animation

Label: Credit Check Cycle Time

Comments:

Apply To Class Name: SALES-CALL

Source Attribute Name: BEGIN-CREDIT-CHECK

Phase: Start Stop

Counter: 0

Error:

OK Apply Update Cancel

Specific Attributes

The specific attributes of the Delta Time probe are:

Attribute	P/M	Description
Time Unit	P	The time unit to use for computing the delta time. The default value is 1 second.
Precision	P	The number of decimal places to use for rounding the Delta Time value.
Delta Time	M	The difference in value of the probed value and the current time.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Delta Time probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Interval Sample Probe



You use an Interval Sample probe to average or sum the value of an attribute of the model at regular time intervals, based on simulation time. You use an Update Trigger tool to determine when to sample the model. For example, you would use an Interval Sample probe to chart on a weekly basis the average total cost of a work object at a particular point in the model.

Except for the ability to update based on a time interval, the Interval Sample probe is the same as a Sample probe.

Determining the Value of the Probe

The Interval Sample probe defines an attribute named Sample Value, which is the value of the current sample. The Sample Value attribute appears as an attribute display of the probe.

Sampling the Model at Regular Time Intervals

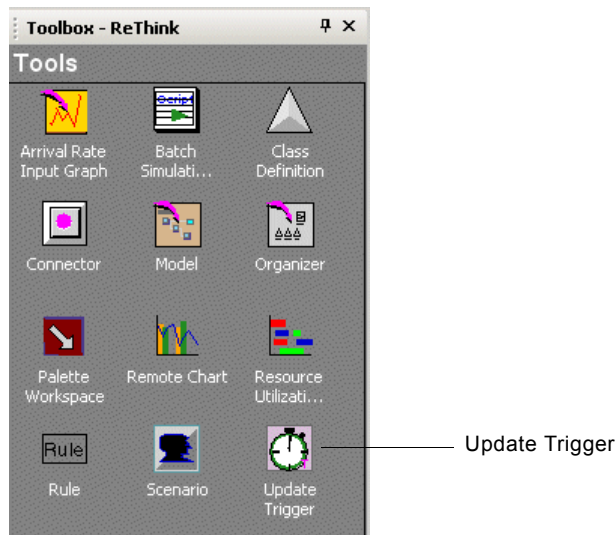
To sample the model at regular time intervals:

- 1 Connect an Interval Sample probe to a block, instrument, or resource in the model.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the class whose attribute value you are probing at regular intervals.

By default, the Interval Sample probe simply samples the values; it does not sum the values.

- 3 To configure the probe to keep a cumulative sum of the sampled values, click the Instruments tab and click the Cumulative Sample option off.

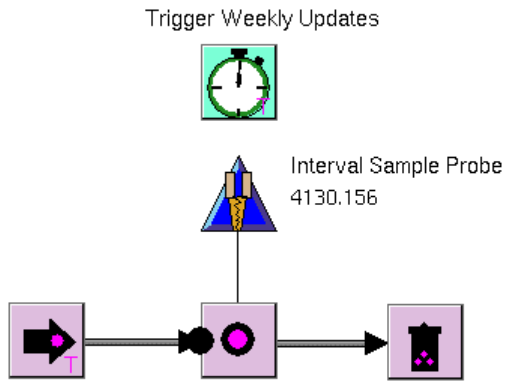
- 4 Display the Tools palette of the ReThink toolbox:



- 5 Create an Update Trigger tool and place it on the model detail or on the detail of an organizer.
- 6 Display the properties dialog for the Update Trigger and configure the Label. By default, the Update Trigger tool triggers updates continuously.
- 7 Click the Block tab and configure the Maximum Starts to be the maximum number of times the trigger should update, as needed.
- You might want to begin triggering updates after a time delay or stop triggering updates after a certain simulation time.
- 8 Configure the Start Time and End Time to be the time at which the trigger should start and finish triggering updates, as needed.
- 9 Click the Duration tab and configure the Period to be the frequency with which to probe the model.
- For example, to probe the model once an hour of simulation time, enter 1 hour.
- 10 Choose the Choose Update Trigger menu choice on the Interval Sample probe, then choose Select on the Update Trigger tool.

The Interval Sample probe samples the model when the Update Trigger tool evaluates, and either averages or sums the sampled values.

This example probes the Total Cost of a block on a weekly basis. The Interval Sample probe samples the block each time the Task block evaluates; however, it does not average the values until the Update Trigger tool evaluates, which happens once a week, based on simulation time.



Interval Sample Probe: Interval Sample Probe

General | Instrument | Animation

Label: Interval Sample Probe

Comments:

Apply To Class Name: BPR-BLOCK

Source Attribute Name: TOTAL-COST

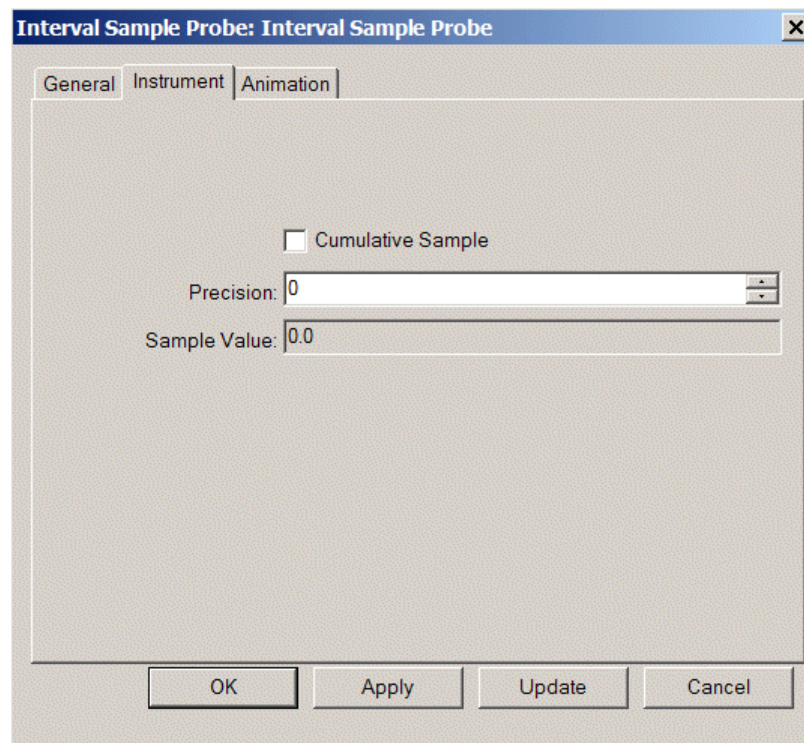
Phase: Start Stop

Counter: 0

Error:

OK Apply Update Cancel

Specific Attributes



The specific attributes of the the Sample probe are:

Attribute	P/M	Description
Cumulative Sample	P	Whether to maintain the sum of the samples in the Sample Value attribute of the probe. The default value is to sample the values without summing them.
Precision	P	The number of decimal places to use for rounding the Sample Value.
Sample Value	M	The current probed value.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

The specific menu choices for the Interval Sample probe are:

Menu Choice	Description
Choose Update Trigger	Identifies the Update Trigger tool that determines when the Interval Sample attribute updates.
Show Update Trigger	Places an indicator arrow next to the chosen Update Trigger tool.
Remove Update Trigger	Removes the association between the Interval Sample probe and the Update Trigger tool.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Message Probe



You use a Message probe to generate a message at a particular location when the simulation is running. By default, ReThink pauses the simulation and displays an indicator arrow with the message text next to the probe. For example, you might want to pause the simulation and display a message when the simulation creates an invalid order.

You can also use the Message probe to send messages to the Messages Browser. You can configure various information about the message, including its type, category, text, and details. You can use the Acknowledge Message probe and Delete Message probe to acknowledge and delete messages in the Messages Browser.

You can also configure the Message probe to write messages to a log file, without pausing the simulation.

You can configure ReThink to send messages to an email account when a message is generated. For details, see [Delivering Messages by Email](#).

Generating Text Messages

When generating messages, you can refer to various attribute values, using the following expressions:

To refer to attributes of the...	Use this expression...
Message probe	[the <attribute> of Probe]
Object that triggers the probe	[the <attribute> of Item]
Object to which the probe is attached	[the <attribute> of ConnectedItem]

For example:

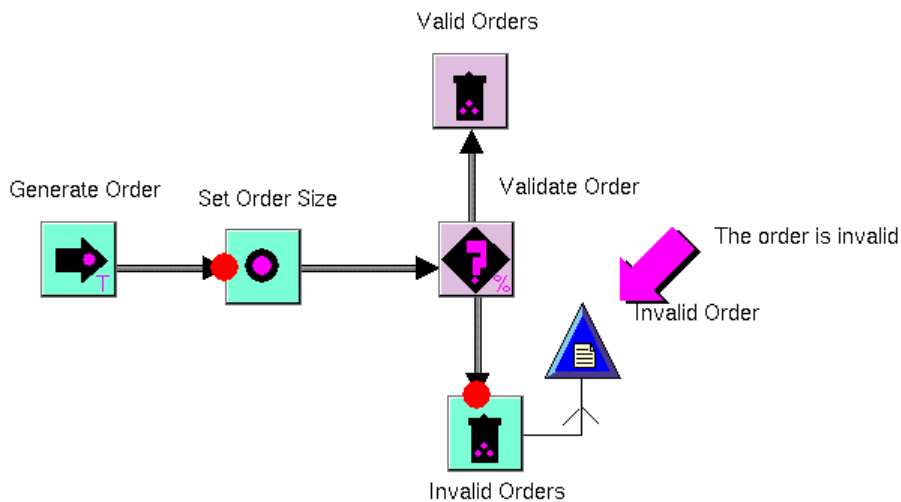
```
"This order is invalid. Probe Counter: [the counter of Probe] ; Object Starts: [the total-starts of Item] ; Block Starts : [the total-starts of ConnectedItem]"
```

To generate a message:

- 1 Connect a Message probe to a block, instrument, or resource in the model.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the name of the class that triggers the message.
- 3 Click the Instrument tab and configure the Message to be a text string to display next to the indicator arrow when the probe triggers.
Refer to attributes of the probe, trigger item, or connected item, as needed.
- 4 Configure the Message Log File to be a text string that names a log file to which the probe writes the messages.
- 5 To cause the instrument to just write messages to the log file, without pausing the simulation, click the Indicate option off.

Tip If you are connecting the Message probe to a Sink or Store block, configure the Phase attribute of the probe to be **Start**; otherwise, the Message probe never triggers.

This example shows the result of running a model that uses a Message probe to flag invalid orders, based on an attribute of the order:

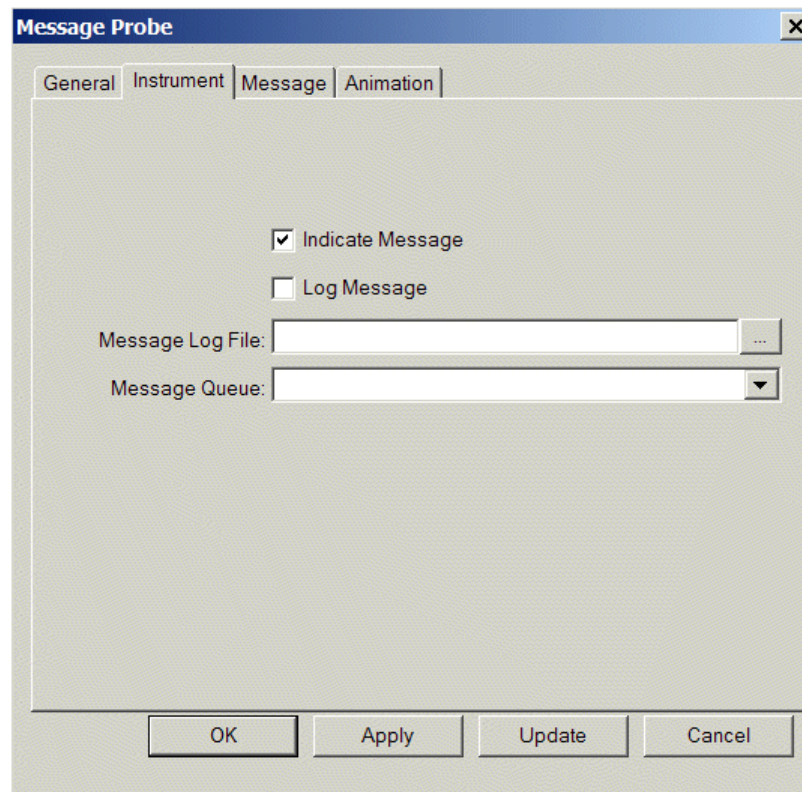


Here is a text of a sample log file:

SATURDAY, 1 Jan 2000 5:54:53 a.m. - Error on Probe Invalid Orders : This order is invalid.

SATURDAY, 1 Jan 2000 8:07:26 a.m. - Error on Probe Invalid Orders : This order is invalid.

Specific Attributes



The specific attributes of the Message probe on the Instrument tab are:

Attribute	P/M	Description
Indicate Message	P	Determines whether to pause the simulation when the Message probe triggers. By default, the Indicate Messages option is on.
Log Message	P	Determines whether to log messages to the specified log file.

Attribute	P/M	Description
Message Log File	P	A text string that names a text file to which the probe writes messages. The log file must be a text file with the extension <i>.txt</i> , <i>.log</i> , or <i>.dat</i> .
Message Queue	P	The message queue to which to send the message. By default, the message is sent to the default message queue, which displays the message in the default Message Browser available by choosing Programs > Message Browser. For details, see Viewing Messages .

The screenshot shows the 'Message Probe' dialog box with the 'Message' tab selected. The dialog has four tabs: 'General', 'Instrument', 'Message', and 'Animation'. The 'Message' tab contains the following fields:

- Message Type:** A dropdown menu.
- Category:** A text input field.
- Priority:** A numeric input field with the value '9' and a spinner control.
- Message:** A large text area with a vertical scrollbar.
- Detail:** A smaller text area with a vertical scrollbar.
- Advice:** A text area with a vertical scrollbar.

At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Update', and 'Cancel'.

The specific attributes of the Message probe on the Message tab are:

Attribute	P/M	Description
Message Type	P	The type of message. To create a text message that appears next to the probe when it triggers, leave the Message Type blank. Choose one of the other message types to create messages that appear in the Message Browser.
Category	P	A text string that is the message category, which appears in the message details.
Priority	P	An integer priority for the message, which appears in the message details. The default value is 9.
Message	P	A text string that is the message to display when the probe triggers. The message can refer to the following expressions: [the <attribute> of Probe] [the <attribute> of Item] [the <attribute> of ConnectedItem]
Detail	P	A text string that provides details about the message, which appears in the message details. The detail can include the same expressions as the message.
Advice	P	A text string that provides advice about the message, which appears in the message details. The advice can include the same expressions as the message.

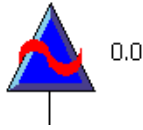
For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Message probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Moving Average Probe



The Moving Average probe computes a time-weighted moving average of all sampled values. This probe is appropriate for computing the average of an attribute of an object, whose value depends on how long it has persisted, such as the number of activities of a block. Thus, a Moving Average probe computes a time-persistent average. It does this by weighting sampled values by how long the values have persisted during the specified time period.

By default, it computes the moving average over the entire simulation. You can also configure the time period over which the probe computes the moving average. The probe computes the moving average starting at the present and moving backwards in time to the time period you specify.

For example, suppose the number of activities of a block has a value of 1 from time 0 to time 3 minutes and a value of 5 from time 3 minutes to time 10 minutes. If the time period over which you are computing the moving average is the entire 10 minutes, the probe computes the moving average as follows:

$$(1 \times 3 \text{ minutes} + 5 \times 7 \text{ minutes}) / 10 \text{ minutes} = 3.8$$

If the time period over which you are computing the moving average is instead 5 minutes, the probe computes the moving average for each 5 minute period, as follows:

$$(1 \times 3 \text{ minutes}) + (5 \times 2 \text{ minutes}) / 5 \text{ minutes} = 2.6$$

$$(5 \times 5 \text{ minutes}) / 5 \text{ minutes} = 5$$

You typically use the Moving Average probe to probe another probe. For example, you can use a Delta Time probe to compute a cycle time, then probe the Delta Time probe to obtain a moving average of this cycle time.

You can also use a Moving Average probe to probe a value directly in the model. For example, you can obtain a moving average of the Current Activities of a block by probing the `current-activities` directly in the Statistics probe.

When you chart a moving average, you often display the probed value and the moving average on the same chart, each in its own color.

Determining the Value of the Probe

The Moving Average probe defines the following two metrics, which are quantitative parameters that keep a history of the probed values:

Attribute	Description
Moving Average	A time-weighted moving average of the probed values.
Moving Standard Deviation	A time-weighted moving standard deviation of the probed values.

The Moving Average appears as an attribute display of the probe.

Computing a Moving Average of a Probed Value

To compute the moving average of a probed value, attach the Moving Average probe directly to another probe.

To compute the moving average of a probed value:

- 1 Connect a Moving Average probe to another probe, such as a Sample or Delta Time probe.
- 2 Display the General tab of the properties dialog for the Moving Average probe.

The default value of Apply to Class Name is `bpr-instrument`, which means the probe is automatically configured to probe another probe.

- 3 Configure Source Attribute Name to be the attribute of the probe for which to compute the moving average.

For example:

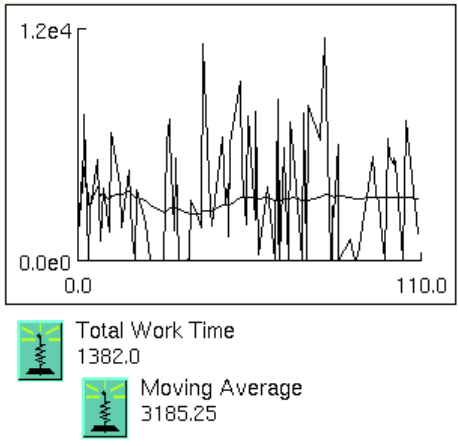
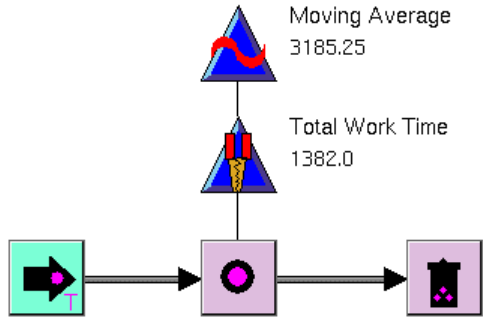
- The `sample-value` attribute of a Sample probe.
- The `delta-time` attribute of a Delta Time probe.

- 4 Click the Instrument tab and configure the Time Period to be the time period over which to compute the moving average, as a duration.

The default value is 1 hour. ReThink converts the duration to seconds.

- 5 Configure the Precision to be the number of decimal places to round the Moving Average and Moving Standard Deviation values.
- 6 To provide initial values for the moving average, configure the Sample Initial Value and Use Initial Value attributes, as needed.

This example computes and plots a moving average of the total work time of the Sample probe:



For information on how to configure a chart to plot two values, see [Plotting Multiple Values on the Same Chart](#).

Here is the General tab of the properties dialog for the Moving Average probe, which shows how it is configured:

The screenshot shows a dialog box titled "Moving Average Probe: Moving Average" with a close button (X) in the top right corner. The dialog has three tabs: "General", "Instrument", and "Animation", with "General" selected. The "General" tab contains the following fields and controls:

- Label:** A text box containing "Moving Average" with up and down arrow buttons on the right.
- Comments:** A larger text box with up and down arrow buttons on the right.
- Apply To Class Name:** A dropdown menu showing "BPR-INSTRUMENT".
- Source Attribute Name:** A dropdown menu showing "SAMPLE-VALUE".
- Phase:** Two radio buttons labeled "Start" and "Stop". The "Stop" radio button is selected.
- Counter:** A text box containing the number "92".
- Error:** A text box with up and down arrow buttons on the right.

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel". The "Apply" button is highlighted with a darker border.

Here is the Instrument tab, which shows its computed values and specific parameters:

The screenshot shows a dialog box titled "Moving Average Probe: Moving Average" with a close button (X) in the top right corner. The dialog has three tabs: "General", "Instrument", and "Animation", with "Instrument" currently selected. The main area contains the following controls:

- An unchecked checkbox labeled "Use Initial Value".
- A text field labeled "Sample Initial Value:" containing the value "0".
- A text field labeled "Time Period:" containing the value "604800".
- A text field labeled "Precision:" containing the value "2".
- A text field labeled "Moving Average:" containing the value "3185.25".
- A text field labeled "Moving Standard Deviation:" containing the value "2979.38".

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Computing a Moving Average Directly

You can attach a Moving Average probe directly to a block to compute the moving average of a value in the model directly. For example, you might want to compute the moving average of the Total Cost of a block, the Total Work Time of a work object, or the Average Utilization of the current resource, without probing these values directly first.

To compute the moving average directly:

- 1 Connect a Moving Average probe directly to a block, instrument, or resource.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the object whose value you are probing.

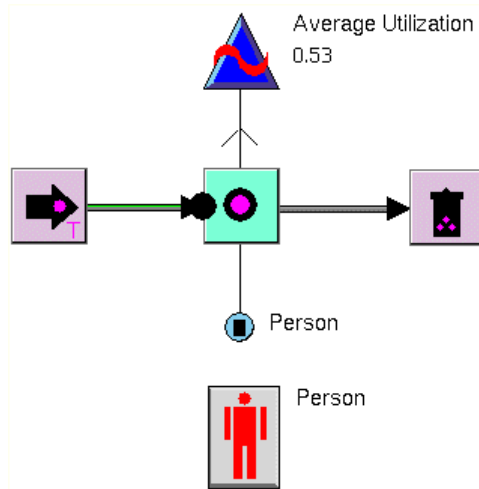
For example, if you are probing the Average Utilization of a resource, the class is `bpr-resource`.

- 3 Configure the Source Attribute Name to be the attribute whose moving average you want to compute.

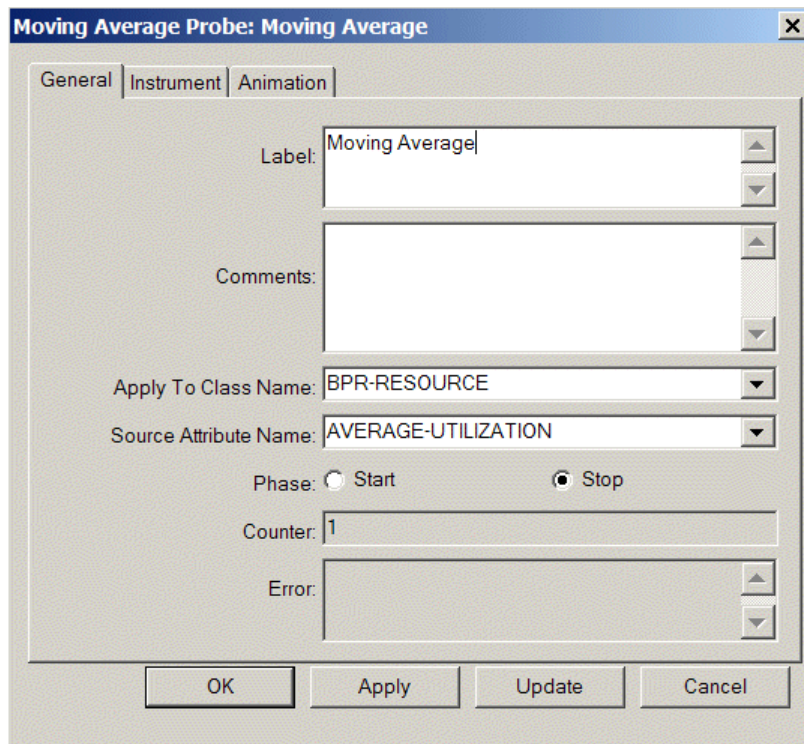
For example, if you are probing the Average Utilization of a resource, the value is `average-utilization`.

- 4 Click the Instrument tab and configure the Time Period to be the time period over which the moving average is computed.
- 5 Configure the Precision to be the number of decimal places to round the Moving Average and Moving Standard Deviation values.

Here is a running model that computes a moving average of the Average Utilization of the current resource allocated to the task:



Here is the General tab of the Moving Average probe:



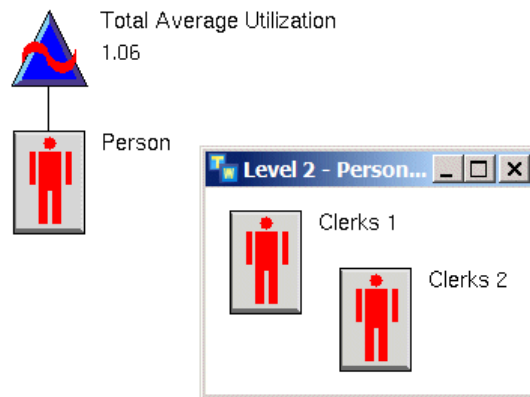
Computing a Moving Average of a Resource Directly

When you probe the attributes of a resource by attaching the probe to a block, the probe computes the moving average of the specified attribute of the *current* resource allocated by the task. You can also probe a resource directly to compute the moving average of the specified attribute of an individual resource or resource pool. When you probe a resource pool directly, you obtain metrics about the *sum* of all the resources in the pool.

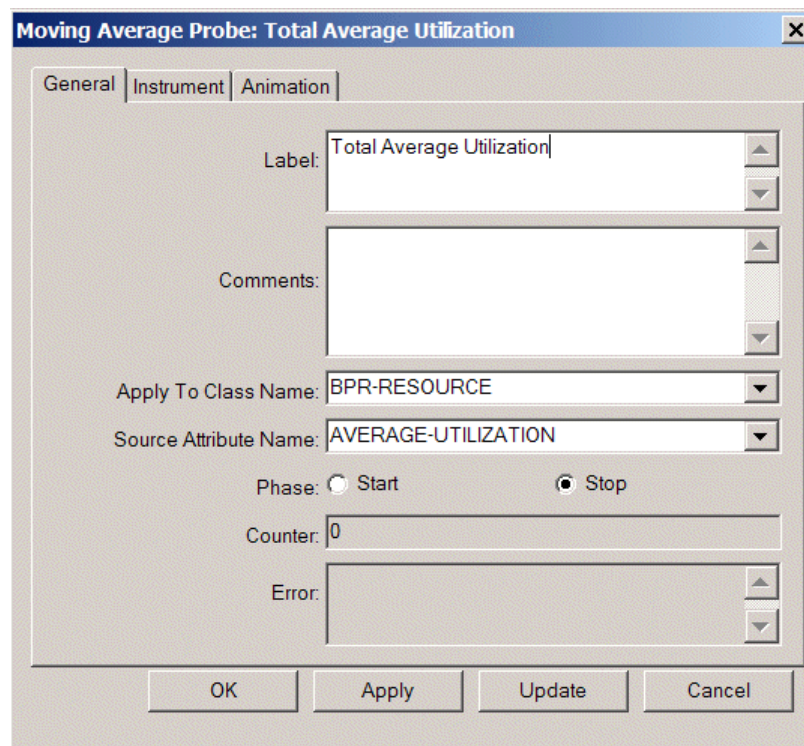
To compute the moving average of a resource directly:

- 1 Connect a Moving Average probe directly to a resource.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name of the probe to be `bpr-resource`.
- 3 Configure the Source Attribute Name Attribute to be the attribute whose moving average you want to compute.
- 4 Click the Instrument tab and configure the Time Period to be the time period over which the moving average is computed.
- 5 Configure the Precision to be the number of decimal places to round the Moving Average and Moving Standard Deviation values.

This figure shows the result of computing a moving average of the sum of the Average Utilization of all resources in the pool:



Here is the General tab for the Moving Average probe:



When you plot the moving average of a resource pool, you might want to divide the moving average by the number of resources in the pool to plot an average of the sum of the moving average values. For information on how to do this, see [Configuring the Colors and Data Points of the Chart](#).

Specific Attributes

The specific attributes of the Moving Average probe are:

Attribute	P/M	Description
Use Initial Value	P	Whether the moving average should include the Sample Initial Value.
Sample Initial Value	P	The initial value to use for the moving average when Use Initial Value is enabled.
Time Period	P	The time interval over which the moving average is computed, for example, 4 weeks . To compute the moving average over the entire simulation, use the default value of 0.
Precision	P	The number of decimal places to round the computed values.

Attribute	P/M	Description
Moving Average	M	A time-weighted moving average of the probed values, based on the Time Period.
Moving Standard Deviation	M	A time-weighted moving standard deviation of the probed values, based on the Time Period.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Moving Average probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

N-Dimensional Sample Probe



You use an N-Dimensional Sample probe to obtain multiple sample values from the model, using a single probe, and optionally keep a history of those values over time. For example, you might use an N-Dimensional Sample probe to collect a history of the Total Cost of a task, the Total Cost of a work object, and the Average Utilization of a resource.

To configure the attributes to sample, you specify the label or unique ID of any object in the model and the attribute to sample. If the object is not unique within the model, you must provide a unique label. If the object does not define a Label, you must go into Developer mode and name the object. For example, to collect the Mean Wait Time of a path, you must name the path.

To view the sampled data, you must export the data to Excel by using an Excel Export tool. For details and an example, see [Exporting Probed Data to a CSV File](#).

The first column of the report displays the simulation time, using an Excel date/time-aware format.

Note The use of an N-Dimensional Sample probe and an Excel Export tool has been superseded by the N-Dimensional Input and Output reports. For details, see [Summary of Input and Output Reports](#) and [Creating Reports in Excel](#).

Collecting N-Dimensional Samples from the Model

To collect n-dimensional samples from the model:

- 1 Connect an N-Dimensional Sample probe to a block, instrument, or resource.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the object that triggers the probe to collect data from the model.

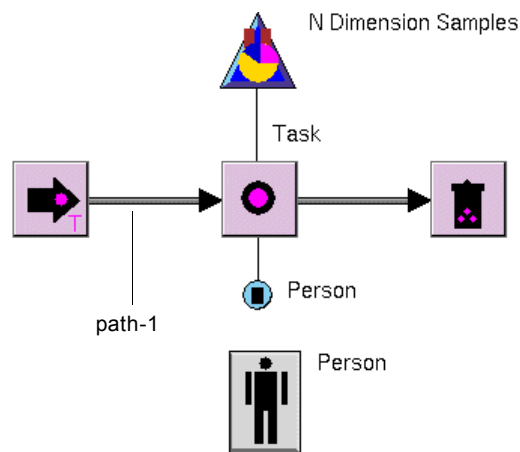
The default value is `bpr-object`, which means it triggers when a work object arrives at a block.

- 3 Click the Instrument tab and configure the attributes to sample and the objects that define those attributes, as follows:
 - a Click Add Row to insert a row above the currently selected row.
 - b Double-click a cell to enter a new value and press Return.

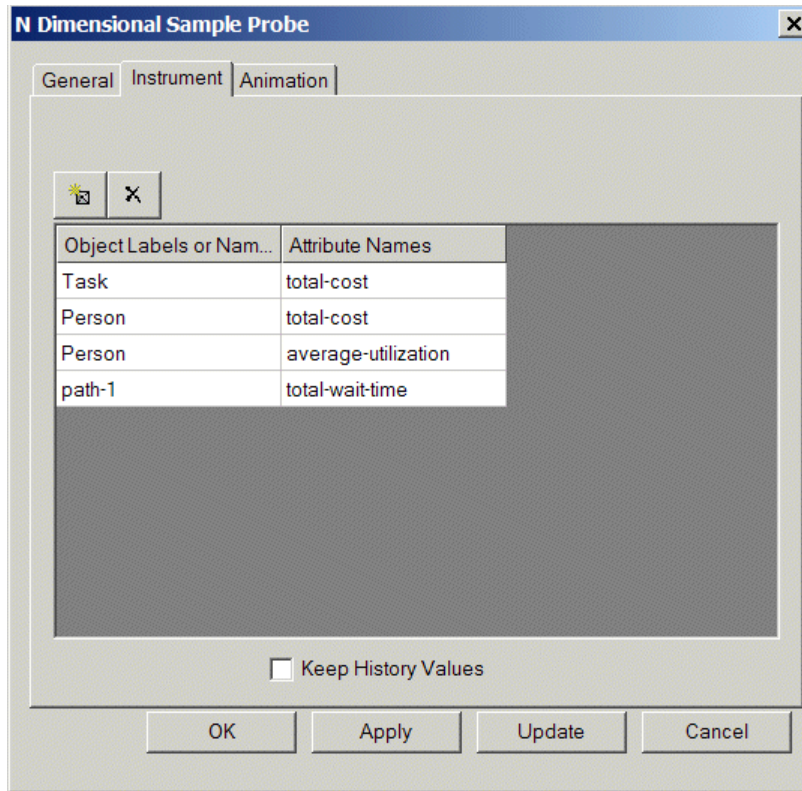
In the left-hand column, enter a unique label of the object whose data you want to collect, and in the right-hand column, enter the name of the attribute whose values you want to collect. Specify the attribute name as a symbol, for example, `total-cost` or `average-utilization`.

- 4 For objects that do not define a Label or whose Label is not unique, you must configure the name of the object, as follows:
 - a Choose Tools > Developer Mode.
 - b Display the properties dialog for the object and click the Customize tab.
 - c Configure the Name to be a unique symbol, for example, `path-1`.
- 5 To keep a history of the sampled data, enable the Keep History Values option.

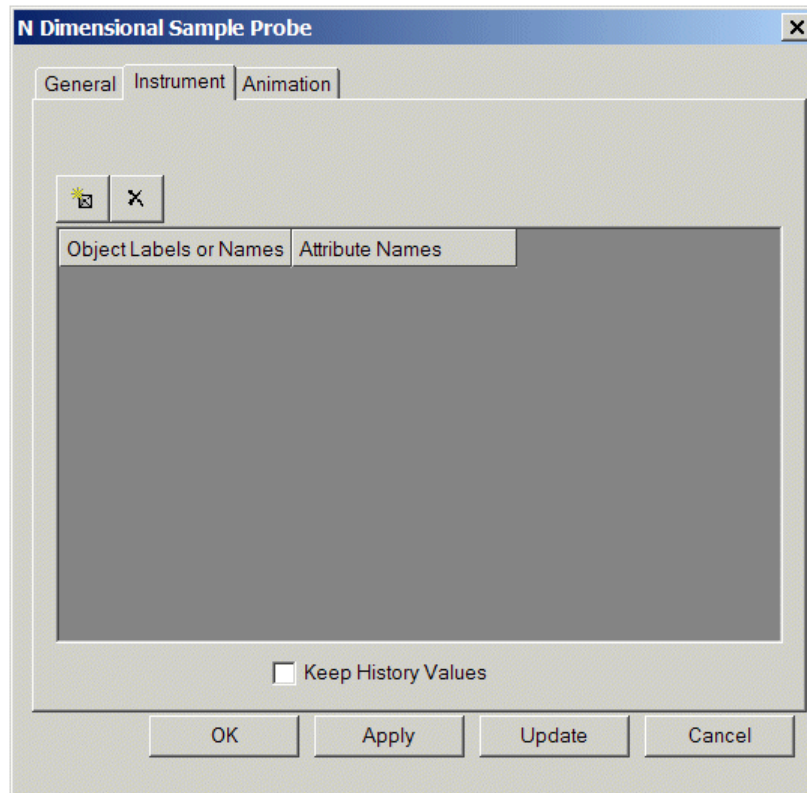
This example shows an N-Dimensional Sample probe that collects the Total Cost of the Task block, the order, and the Person resource, the Average Utilization of the Person resource, and the Total Wait Time of the input path to the Task block named `path-1`:



Here is the Instrument tab for the probe:



Specific Attributes



The specific attribute for the N-Dimensional Sample probe is:

Attribute	P/M	Description
Keep History Values	P	Whether to keep a history of sampled values.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

An N-Dimensional Sample probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#).

Parameter Probe



A Parameter probe sets the value of a parameter to the value of an attribute of the model. You can create the quantitative parameter from the probe.

You can also use the Parameter probe in conjunction with any type of parameter or variable, which you can create in Developer mode.

Used in conjunction with a Sample probe or Average probe, the Parameter probe allows you to plot a history of attribute values or an average of those values, respectively, over time. For details, see:

- [Charting Quantitative Parameters.](#)
- [Charting the Average of Quantitative Parameters.](#)

Used in conjunction with a Parameter feed, the Parameter probe enables you to set attributes of the model that originate from a quantitative parameter. For details, see [Parameter Feed.](#)

Setting the Value of a Parameter

To set the value of a parameter:

- 1 Connect a Parameter probe to an object in the model whose attribute values you want to store in a parameter.

Connect the probe to a block to probe an attribute of the block, its activities, or the work objects, or connect the probe to a resource or to another probe.

- 2 Choose Create Parameter on the Parameter probe to create a uniquely named quantitative parameter.

The Parameter probe sets the Parameter Name to the name of the parameter.

Note If you have already created a parameter, using some other technique, you can also choose the Choose Parameter menu choice on the probe and choose Select on the parameter. For example, you might have already created the parameter by using a Parameter feed. This menu choice gives the parameter a unique name, if it does not have one, and sets the Parameter Name to the named parameter.

- 3 On the General tab of the properties dialog, configure the Apply to Class Name to be the class name of an object in the model whose attribute values you want to store.

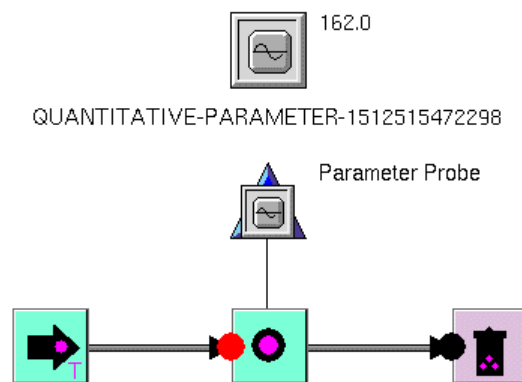
For example, if you are probing the Total Work Time attribute of a work object, specify `bpr-object` or a subclass of `bpr-object`.

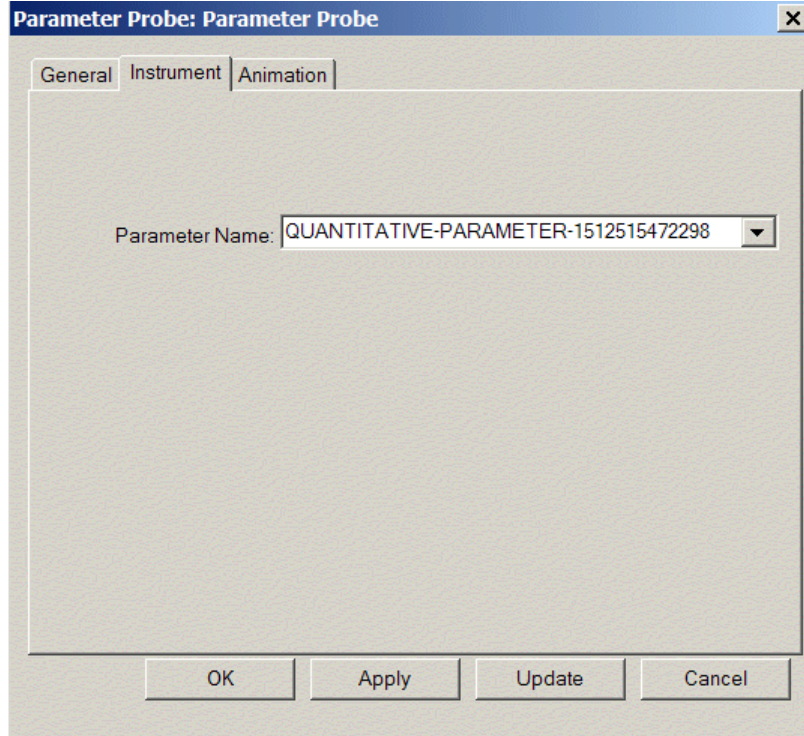
- 4 Configure the Source Attribute Name to be the attribute of the class whose values you want to store in the parameter.

For example, if you are probing the Total Work Time attribute of a work object, specify Source Attribute Name as `total-work-time`.

When you run the simulation, the Parameter probe sets the current value of the parameter to the specified attribute.

For example, in the following model, the Parameter probe obtains the Total Work Time of the work object and stores it in the named quantitative parameter. The Total Work Time of the work object on the output path of the Task block corresponds to the current value of the parameter.





To see the probe associated with the parameter:

→ Choose Show Instruments on the parameter.

Feeding Values into Different Types of Parameters

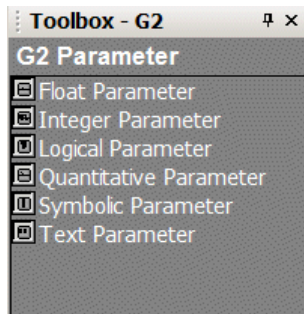
You might want to use a Parameter probe to keep truth values, symbols, or text strings, rather than quantitative values.

The parameter types are:

- logical-parameter, which keeps true or false values.
- quantitative-parameter, which keeps integers or floating point numbers.
- integer-parameter, which keeps integers.
- float-parameter, which keeps floating point numbers.
- symbolic-parameter, which keeps symbols.
- text-parameter, which keeps text strings.

To feed values into different types of parameters:

- 1 Choose View > Toolbox - G2.
- 2 Click the G2 Parameter tab:

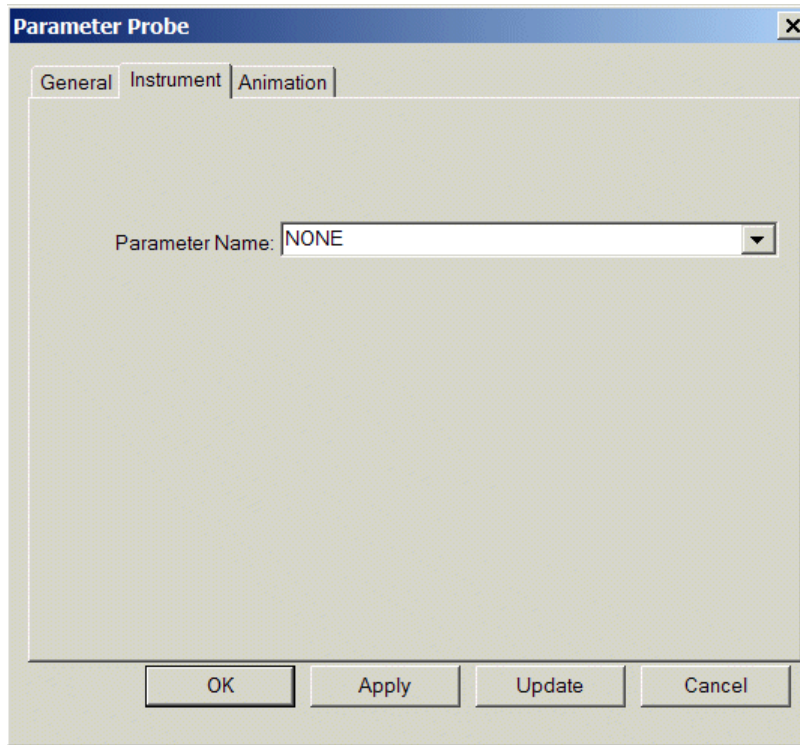


- 3 Create the desired type of parameter.
- 4 Choose the Choose Parameter menu choice on the Parameter probe, then choose Select on the parameter to select it.

The Parameter probe sets the Parameter Name to the name of the parameter.

- 5 Create a model that feeds the appropriate type of value into the parameter.

Specific Attributes



The specific attribute of the Parameter probe is:

Attribute	P/M	Description
Parameter Name	P	The name of the parameter whose value the probe should set. Choosing Create Parameter or Choose Parameter on the probe sets automatically.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

The specific menu choices of the Parameter probe are:

Menu Choice	Description
Create Parameter	Creates a unique named quantitative parameter and sets the Parameter Name attribute of the probe to the parameter name.
Choose Parameter	Associates an existing parameter with the Parameter probe, sets the Parameter Name attribute to the existing parameter, and creates a unique name for the parameter, if needed. Choose Select on a parameter to select it.
Show Parameter	Places an indicator arrow next to the parameter associated with the Parameter probe.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#).

Sample Probe



A Sample probe obtains any attribute value that the model computes. For example, you use the Sample probe to obtain the:

- Total Work Time of a block.
- Total Cost of a work object.
- Average Utilization of a resource.
- Current value of a quantitative parameter or variable, which you can then chart over time.

You can use the Sample probe to obtain individual attribute values, or you can use it to compute the sum of all the individual values.

Note The Sample probe functions in a sample and hold mode, which means it samples the value at the end of block processing. This means that the sampled value might not always match the values shown in the dialog for the sampled value, because these values update at different times.

Determining the Value of the Probe

The Sample probe defines an attribute named Sample Value, which is a quantitative parameter that keeps a history of the probed values. The value of this attribute is the current probed value. The Sample Value appears as an attribute display of the probe.

Probing Attribute Values that the Model Computes

By default, the Sample probe is configured to probe the total-work-time of a work object.

To probe the total work time of an object:

- 1 Connect a Sample probe to a block or instrument.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the class whose attribute values you are probing.

For example, if you are probing the Total Work Time attribute of a work object, specify `bpr-object` or a subclass of `bpr-object`.

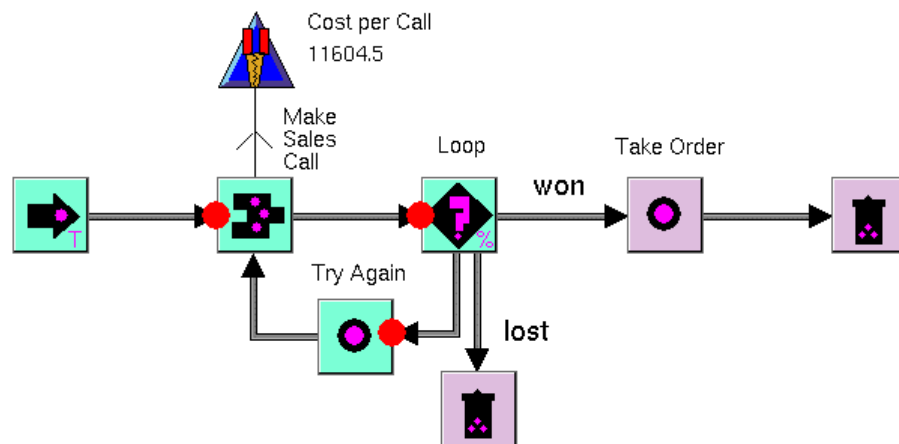
- 3 Configure the Source Attribute Name to be the attribute to sample.

If you are probing the Total Work Time attribute of a work object, you can use the default value for Source Attribute Name, which is `total-work-time`.

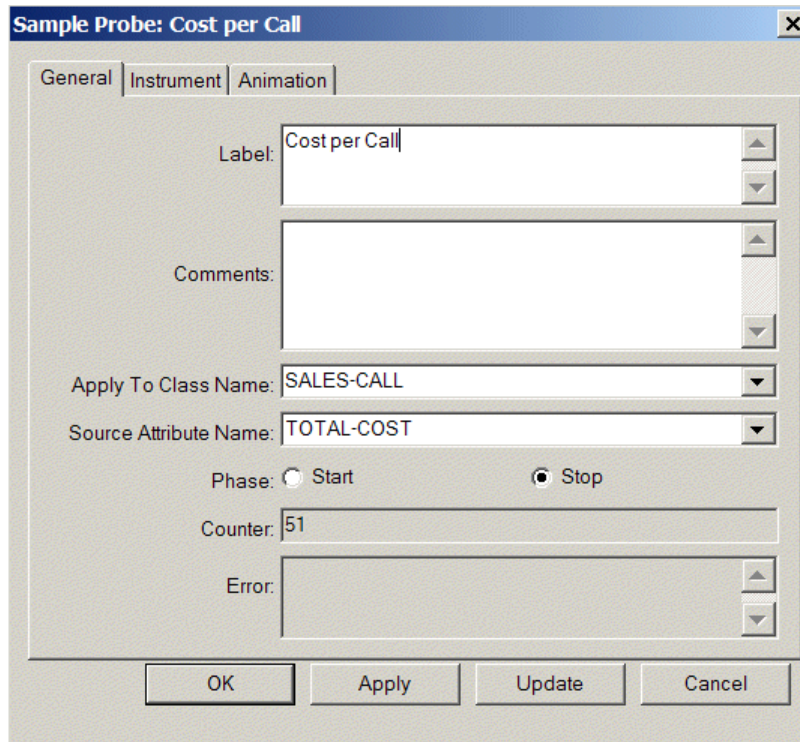
If you are probing some other attribute, such as Total Cost, specify the attribute value as a symbol, for example, `total-cost`.

- 4 To compute the sum of all probed values, click the Cumulative Sample option on; otherwise, use the default.

This example probes the `total-cost` of each sales call in a model of the sales cycle. The Make Sales Call task has a variable cost assigned to it, which represents the cost of the phone call, based on the duration. The cost is 30¢ per minute. By probing the total cost of this task, you can determine the cost of each phone call.



Here is the General tab of the properties dialog for the Sample probe:



Probing Attribute Values of a Resource Directly

When you probe the attributes of a resource by attaching the probe to a block, the probe obtains the sample value from the *current* resource allocated by the task. You can also probe a resource directly to obtain the sample value of an individual resource or resource pool. When you probe a resource pool directly, you obtain metrics about the *sum* of all the resources in the pool.

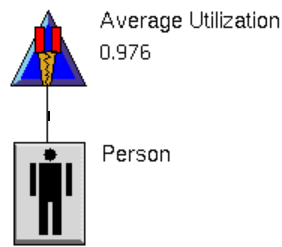
To probe an attribute value of a resource directly:

- 1 Connect a Sample probe directly to a resource or resource pool.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name as `bpr-resource`.
- 3 Configure the Source Attribute Name Attribute to refer to the attribute of the resource you want to probe.

For example, you might want to sample the `average-utilization`.

- 4 To compute the sum of all probed values, click the Instrument tab and click the Cumulative Sample option on; otherwise, use the default.

This example shows how to probe a resource directly, using a Sample probe:



You can perform computations on the sample values, for example, to plot the average of the sum of all the resources in the pool. For an explanation of how to do this, see [Configuring the Colors and Data Points of the Chart](#).

Charting Quantitative Parameters

You can attach a Sample probe to a quantitative parameter, then chart the value over time. You can use this feature in conjunction with a Parameter feed, which allows you to feed the value of a parameter into an attribute of the model, and a Parameter probe, which allows you to probe an attribute of the model that comes from a parameter.

To chart a quantitative parameter:

- 1 Create a model that probes the value of a quantitative parameter from an attribute of the model.

For details, see [Parameter Probe](#).

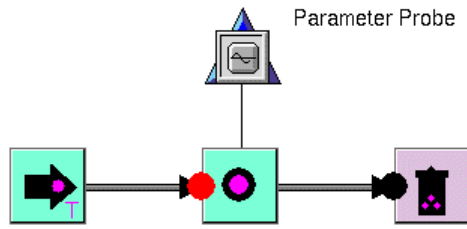
- 2 Attach a Sample probe to the quantitative parameter.
- 3 On the General tab of the properties dialog, configure the Apply to Class Name to be `variable-or-parameter`.

Leave the Source Attribute Name blank.

- 4 To compute the sum of all probed values, click the Instrument tab and click the Cumulative Sample option on; otherwise, use the default.
- 5 Choose Create Chart on the Sample probe to create a chart and associated remote.

The following example shows how to chart over time the value of total cost, which is stored in a quantitative parameter. The model uses a Parameter probe to obtain the total cost of a work object and store it in a quantitative parameter. A Sample probe obtains the current value of the quantitative parameter and plots it on a chart.

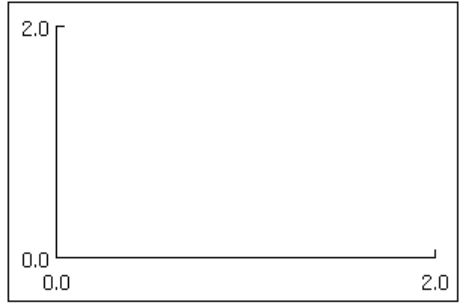
For information on how to configure the Parameter probe, see [Parameter Probe](#).



SAMPLE-VALUE
0.0



QUANTITATIVE-PARAMETER-16125169791



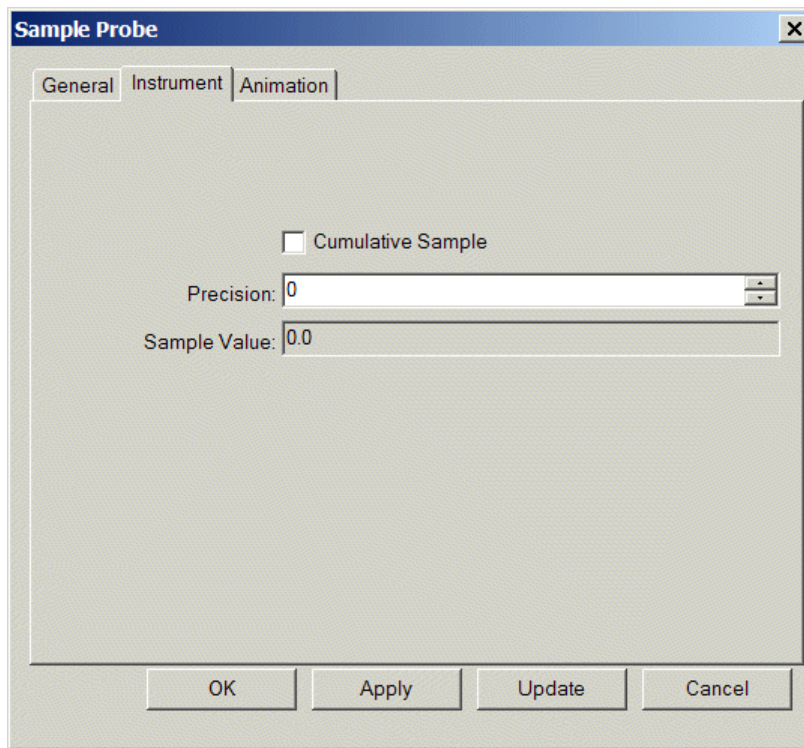
Here is the General tab of the properties dialog for the Sample probe:

The screenshot shows a dialog box titled "Sample Probe" with a close button (X) in the top right corner. The dialog has three tabs: "General", "Instrument", and "Animation". The "General" tab is selected. The "General" tab contains the following fields and controls:

- Label:** A text input field with up and down arrow buttons on the right.
- Comments:** A multi-line text area with up and down arrow buttons on the right.
- Apply To Class Name:** A dropdown menu with "VARIABLE-OR-PARAMETER" selected.
- Source Attribute Name:** A dropdown menu with "NONE" selected.
- Phase:** Two radio buttons labeled "Start" and "Stop". The "Stop" radio button is selected.
- Counter:** A text input field containing the value "0".
- Error:** A text input field with up and down arrow buttons on the right.

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Specific Attributes



The screenshot shows a dialog box titled "Sample Probe" with a close button (X) in the top right corner. It has three tabs: "General", "Instrument", and "Animation", with "Animation" currently selected. The main area contains a checkbox labeled "Cumulative Sample" which is unchecked. Below it is a "Precision" field with a spinner control showing the value "0". Below that is a "Sample Value" field with a text input showing the value "0.0". At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attributes of the Sample probe are:

Attribute	P/M	Description
Cumulative Sample	P	Whether to maintain the sum of the samples in the Sample Value attribute of the probe. The default value is to sample the values without summing them.
Precision	P	The number of decimal places to round the Sample Value.
Sample Value	M	The current sampled value.

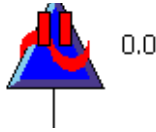
For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Sample probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Statistics Probe



The Statistics probe computes various time-weighted metrics for the sample values. Similar to the Moving Average probe, the Statistics probe is appropriate for computing metrics for the attribute of any object, whose value depends on how long it has persisted, for example, the Current Activities of a block or the Total Cost of a work object or resource. You might also use this probe to compute metrics for user-defined attributes of work objects, such as inventory levels, which you compute as time-weighted values over a given time period.

You provide the time period over which the probe computes metrics. The probe computes metrics for each time period, where the initial value for each new time period is the last value for the previous time period. Unlike the Moving Average probe, you must provide the time period over which to compute the metrics.

For a description of how the probe computes time-weighted metrics, see [Moving Average Probe](#).

You can use the Statistics probe to probe another probe. For example, you can use a Delta Time probe to compute a cycle time, then probe the Delta Time probe to obtain metrics about the cycle time.

You can also use a Statistics probe to probe a value directly in the model. For example, you can obtain metrics the Current Activities of a block by probing the Current Activities directly in the Statistics probe.

Determining the Value of the Probe

The Statistics probe defines the following attributes, which are quantitative parameters that keep a history of values:

Attribute	Description
Sample Value	The current sampled value.
Number of Samples	The number of samples within the time period.
Sum of Incremental Values	The sum of all positive delta values between the last sample value and the new sample value that incremented the value.

Attribute	Description
Sum of Decremental Values	The sum of all negative delta values between the last sample value and the new sample value that decremented the value.
Minimum Value	The minimum of the sampled values within the time period.
Maximum Value	The maximum of the sampled values within the time period.
Average Value	The average of the sampled values within the time period.
Moving Average	The time-weighted moving average of the sampled values, based on the time period.
Moving Standard Deviation	The time-weighted moving standard deviation of the probed values, based on the time period.
Time Weighted Value	The time-weighted value of the sampled value, based on the time period.

The Sample Value appears as an attribute display of the probe.

Computing Statistics for a Probed Value

To statistics metrics for a probed value:

- 1 Connect a Statistics probe to an object in the model for whose attribute you want to compute metrics.

For example, you can attach the Statistics probe to a block, instrument, or resource.

- 2 On the General tab of the properties dialog, configure the Apply to Class name to be the class to which the probe applies.

The default value is `bpr-object`, which means the probe is automatically configured to probe the work object of a block.

- 3 Configure the Source Attribute Name to be the attribute of the probe for which to compute metrics.

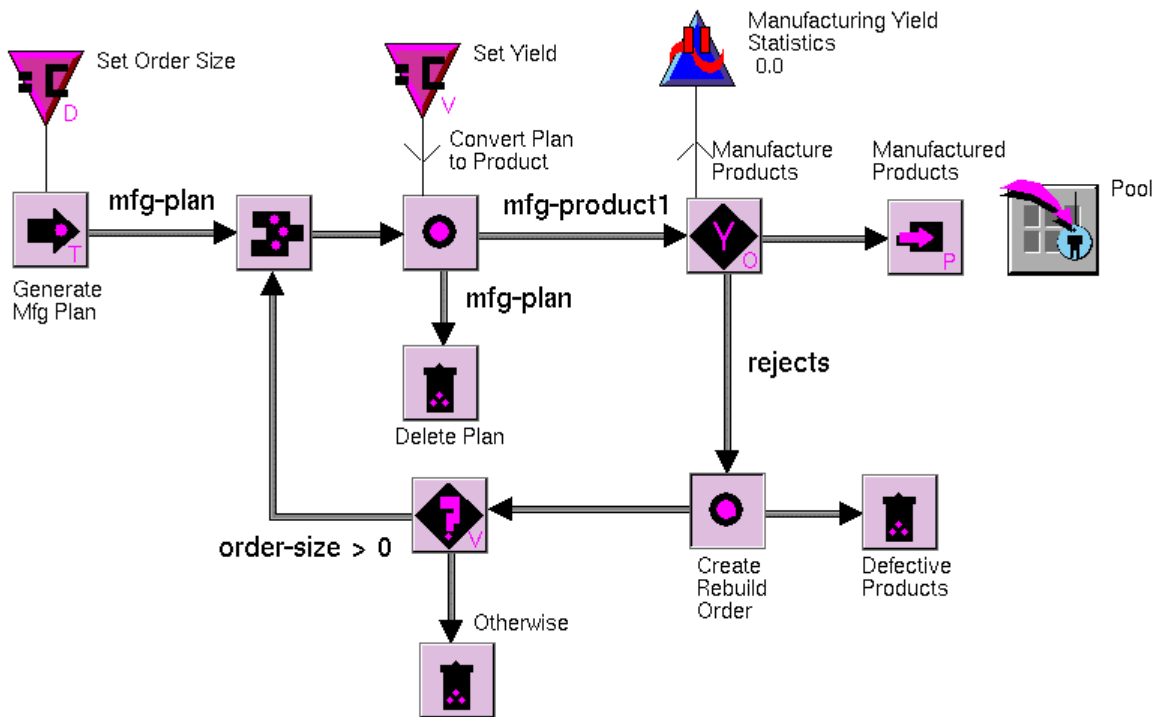
For example, to probe the Sample Value of a Sample probe, the source attribute is `sample-value`, and to probe the Current Activities of a block, the attribute is `current-activities`.

- Click the Instrument tab and configure the Time Period to be the time period over which the time-weighted metrics are computed.

For example, you might want to compute weekly or monthly metrics. The default value is 52 weeks.

- Configure the Precision to be the number of decimal places to round the metrical values.

This example computes metrics for the order size of each manufactured product that the Yield block creates. For details on this example, see the [Yield block](#).



Here is the General tab of the properties dialog for the Statistics probe:

The image shows a software dialog box titled "Statistic Probe: Manufacturing Yield Statistics". It has three tabs: "General", "Instrument", and "Animation". The "General" tab is selected. The dialog contains the following fields and controls:

- Label:** A text box containing "Manufacturing Yield Statistics".
- Comments:** An empty text area.
- Apply To Class Name:** A dropdown menu showing "MFG-PRODUCT".
- Source Attribute Name:** A dropdown menu showing "ORDER-SIZE".
- Phase:** Two radio buttons, "Start" and "Stop". The "Stop" radio button is selected.
- Counter:** A text box containing the number "28".
- Error:** An empty text box.

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

Here is the Instrument tab of the properties dialog for the Statistics probe:

Statistic Probe [X]

General | **Instrument** | Animation

Time Window: 052 000 00:00:00

Precision: 0

Initial Value: 0

Period Start Time: 000.000.00:00:00

Period Initial Value: 0.0

Update Time: 000.001.02:53:27

Sample Value: 8756.0

Number Of Samples: 28

Sum Of Incremental Values: 70714.0

Sum Of Decremental Values: 61958.0

Minimum Value: 0.0

Maximum Value: 14839.0

Average Value: 7816.179

Moving Average: 8703.612

Moving Standard Deviation: 5446.792

Time Weighted Value: 7679.417

OK Apply Update Cancel

Specific Attributes

The specific attributes of the Statistics probe are:

Attribute	P/M	Description
Time Window	P	The time interval over which the metrics are computed, for example, 4 weeks. You must specify a time period. The default value is 52 weeks.
Precision	P	The number of decimal places to round various computed metrics.
Initial Value	P	The initial value for the probed value.

Attribute	P/M	Description
Period Start Time	M	The simulation time at the start of the current time period.
Period Initial Value	M	The initial value of the current time period, which is the last value for the previous time period.
Update Time	M	The time at which the probe last updated its values.
Sample Value	M	The current sampled value, which appears as an attribute display next to the probe.
Number of Samples	M	The number of samples in the current time period.
Sum of Incremental Values	M	The sum of all positive delta values between the last sample value and the new sample value that incremented the value.
Sum of Decremental Values	M	The sum of all negative delta values between the last sample value and the new sample value that decremented the value.
Minimum Value	M	The minimum of the sampled values in the current time period.
Maximum Value	M	The maximum of the sampled values in the current time period.
Average Value	M	The average of the sampled values in the current time period.
Moving Average	M	The time-weighted moving average of the sampled values, based on the Time Period.
Moving Standard Deviation	M	The time-weighted moving standard deviation of the sampled values, based on the Time Period.
Time-Weighted Value	M	The time-weighted sample value, based on the current Time Period.

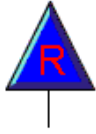
For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Statistics probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Update Trigger Probe



You use an Update Trigger probe to trigger updates, based on model events. The Update Trigger probe updates when the object specified in the Apply to Class Name attribute becomes active. For example, you might want a report to update each time a work object arrives at a block or each time a resource is allocated.

For more information and an example, see [Triggering Updates Based on Model Events](#).

Specific Attributes

An Update Trigger probe has no specific attributes.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

An Update Trigger probe has no specific menu choices.

For information on the menu choices common to all probes, see [Common Menu Choices for Feeds and Probes](#) and [Common Menu Choices for Probes](#).

Accumulate Feed



The Accumulate feed increments a counter by the value specified in the attribute of a work object. For example, suppose you are modeling a sales process that receives local sales calls and regional sales calls, where each type of sales call has an associated mileage attribute. You use an Accumulate feed to increment the total mileage attribute of each sales call object by each specific mileage amount.

The Accumulate feed is very similar to the Count feed, except that you configure the amount by which the feed increments as an attribute of the work object that is accumulating a value. You must also configure the attribute that holds the accumulated value.

By default, the Accumulate feed adds the source attribute to the destination attribute. However, you can also use subtraction, multiplication, division, or exponentiation. For example, if the operation is subtraction, the feed subtracts the source attribute from the destination attribute.

Accumulating Values

To use the Accumulate feed, create a class definition with two class-specific attributes. One attribute specifies the attribute that accumulates the values, and the other attribute specifies the amount by which the counter increments. The Accumulate feed refers to these two attributes as the source and destination, respectively.

To accumulate values into an attribute of an object:

- 1 Create a class definition with two class-specific attributes, one of which is the accumulated value and the other of which is the amount by which to increment the value.

Be sure to configure initial values for each of these attributes.

For details, see [Creating a New Class of Work Object](#).

- 2 Create another class definition that has the same two class-specific attributes but with a different default value for the attribute that increments the feed.

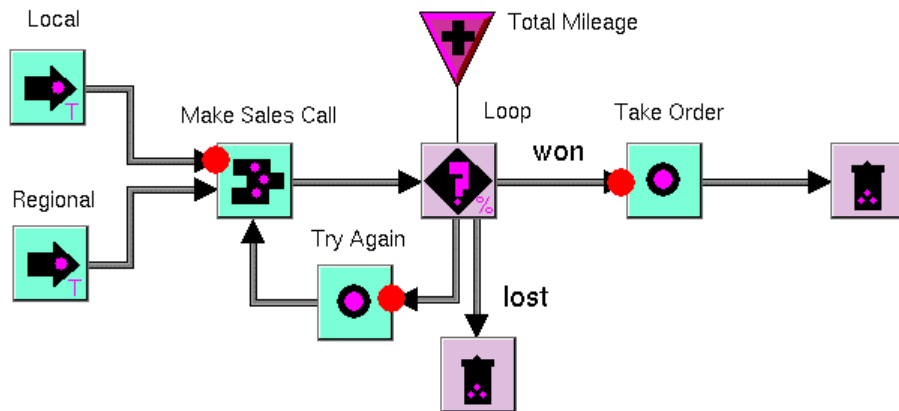
Tip Create two class definitions that inherit from a single superior class, each of which defines different initial values for the attribute that increments the feed.

- 3 Attach an Accumulate feed to a block.

- 4 On the General tab of the properties dialog, configure the Apply to Class Name to be the work object that the model processes.
- 5 Configure the Destination Attribute Name to be the attribute of the work object that stores the accumulated value.
- 6 Click the Instrument tab and configure the Source Attribute Name to be the attribute of the work object that determines how much to increment the accumulated value.
- 7 Configure the Operation to be the mathematical operation that the source attribute should perform on the destination attribute.

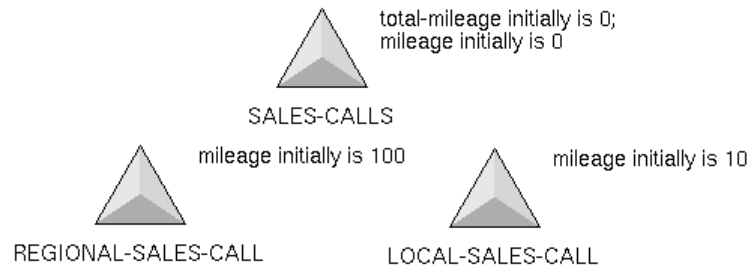
By default, the Accumulate feed adds the source attribute to the destination attribute.

This model of a sales process creates two types of sales calls, a local sales call and a regional sales call, each of which is a type of **sales-call**. The Accumulate feed increments the **total-mileage** attribute of each type of sales call by the **mileage** attribute of each type of sales call, which is different for each type of sales call.

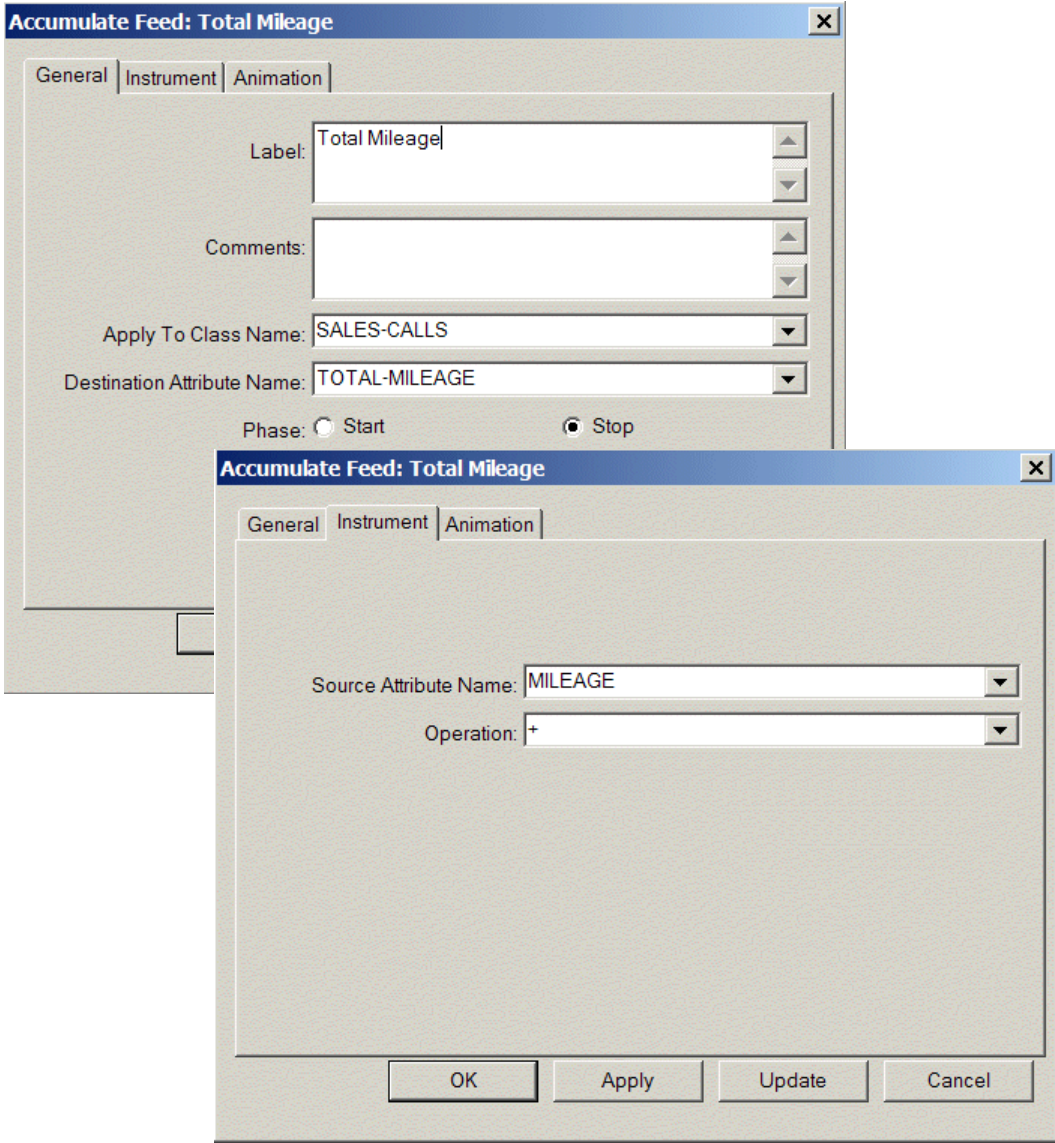


The **sales-calls** class defines a **total-mileage** attribute. The **mileage** attribute specifies the amount by which the Accumulate feed increments the **total-mileage** attribute. Each specific type of **sales-calls** inherits its definition from the **sales-calls** and specifies a different initial value for the **mileage** attribute.

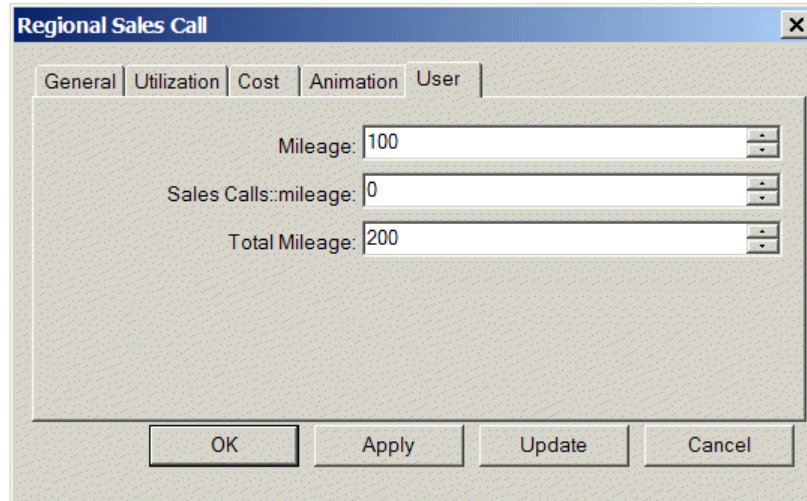
Here are the class definitions for the **sales-calls**, **local-sales-call**, and **regional-sales-call** classes:



Here is General tab and the Instrument tab of the properties dialog for the Accumulate feed. The feed applies to the sales-calls class, which is the superior class for each type of sales call. The mileage and total-mileage attributes are both attributes of the sales-call class.



Here is the User tab of the properties dialog for a regional sales call that has had two sales calls applied to it, as indicated by the value of the Total Mileage attribute:

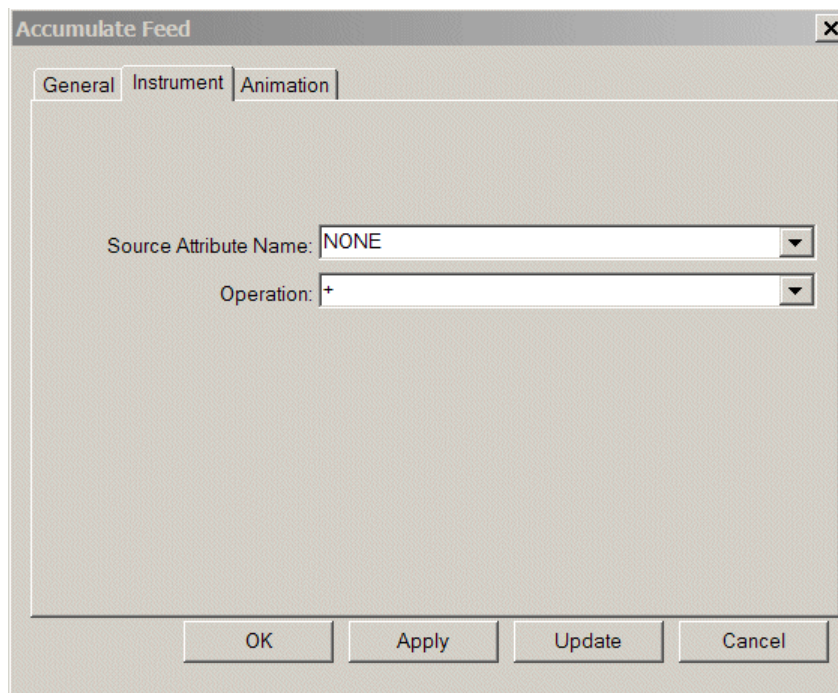


The image shows a dialog box titled "Regional Sales Call" with a close button (X) in the top right corner. The dialog has five tabs: "General", "Utilization", "Cost", "Animation", and "User". The "User" tab is selected. Inside the dialog, there are three input fields with spinners on the right side:

- Mileage: 100
- Sales Calls::mileage: 0
- Total Mileage: 200

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

Specific Attributes



The image shows a dialog box titled "Accumulate Feed" with a close button (X) in the top right corner. The dialog has three tabs: "General", "Instrument", and "Animation". The "Instrument" tab is selected. Inside the dialog, there are two dropdown menus:

- Source Attribute Name: NONE
- Operation: +

At the bottom of the dialog, there are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attributes of the Accumulate feed are:

Attribute	Description
Source Attribute Name	The attribute of the specified class that determines the amount by which the counter increments the Destination Attribute Name attribute of the feed.
Operation	<p>The mathematical operation that the source attribute performs on the destination attribute. The options are: +, -, *, /, and E. The default value is +.</p> <p>For example, if the source attribute is 100, the destination attribute value is currently 500, and the operation is -, the new value of the destination attribute will be 400.</p>

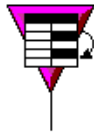
For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

An Accumulate feed has no specific menu choices.

For information on the menu choices common to all feeds, see [Common Menu Choices for Feeds and Probes](#).

Attribute Feed



The Attribute feed copies the value of an attribute of one object to another attribute, either in the same object or in a different object. For example, you might want to copy the work time of an activity of a block into an attribute of the output work object.

An Attribute feed is similar to a Copy Attributes block in that it copies attributes from one object to another, with two differences:

- The Copy Attributes block copies attributes between work objects only, whereas the Attribute feed can copy attributes between any type of object.
- The Copy Attributes block copies all common attributes, whereas the Attribute feed copies only one attribute.

By default, the Attribute feed copies the exact value of the source attribute to the destination attribute. When copying numeric values, you can also use addition, subtraction, multiplication, division, or exponentiation to copy the value. For example, if the operation is subtraction, the feed subtracts the source attribute from the current value of the destination attribute and copies the result into the destination attribute.

Normally, instruments execute either before or after the attached block applies its duration to the simulation, by specifying the Phase attribute as either **Start** or **Stop**, respectively. The Attribute feed provides an additional Phase called **Activity**, which executes *after* the block applies its duration but *before* the block executes its stop method. By setting the Phase to **Activity**, you can copy attribute values from the input work object into the block, after the block applies its duration.

Copying Attribute Values

You can copy attribute values to and from blocks, resources, activities, input work objects, output work objects, input paths, or output paths.

To copy attribute values:

- 1 Connect the Attribute feed to a block in the model.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be source object, that is, the object from which the feed will copy values.
- 3 Configure the Source Attribute Name to be the source attribute, that is, an attribute of the source object whose value the feed will copy.

- 4 Configure the Phase, depending on the source and target objects you specify.

For example:

- To copy attributes from the input work object, input path, or activity, before the block applies its duration to the simulation, configure the Phase as **Start**.
- To copy attributes from the output work object, output path, or activity, after the block has applied its duration to the simulation, configure the Phase as **Stop**.
- To copy attributes from the input work object into the block after the block has applied its duration to the simulation, configure the Phase as **Activity**.

- 5 Click the Instrument tab and configure the Destination Class Name to be the target object, that is, the object to which the feed will copy values.

- 6 Configure the Destination Attribute Name to be the target attribute, that is, the target object whose value the feed will update, using the source attribute.

If the values of the source and destination attributes are numeric, you can configure the operation to perform. By default, the feed does not apply any operator.

- 7 For numeric attributes, configure the Operation to be the operator to apply when copying the source attribute to the destination attribute.

The feed applies the operator to the current value of the destination attribute, as follows:

Operation	Description
+	Destination attribute + source attribute
-	Destination attribute - source attribute
*	Destination attribute x source attribute
/	Destination attribute / source attribute
E	Destination attribute ^{source attribute}

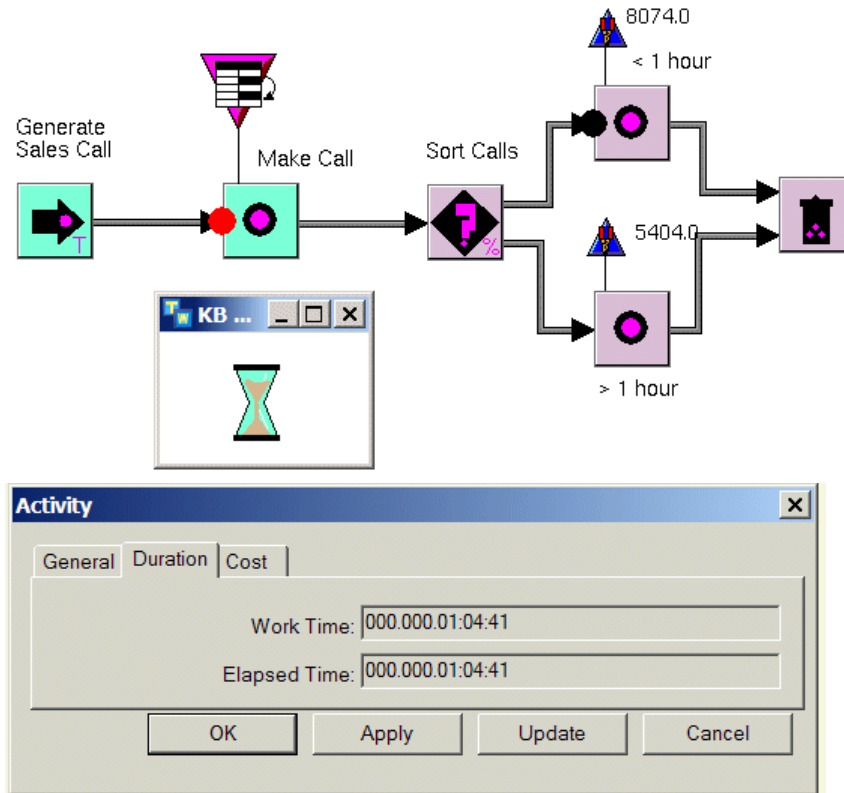
- 8 For numeric values, configure the Precision to be the number of decimal places to round the value of the attribute named by Destination Attribute Name.

This model shows how to copy the work time of the activity of a block into an attribute of the output work object, which requires that the Phase attribute be **Stop**.

The model generates phone calls and uses an Attribute feed to copy the **work-time** of the Make Call activity into the **phone-time** attribute of the phone call. The

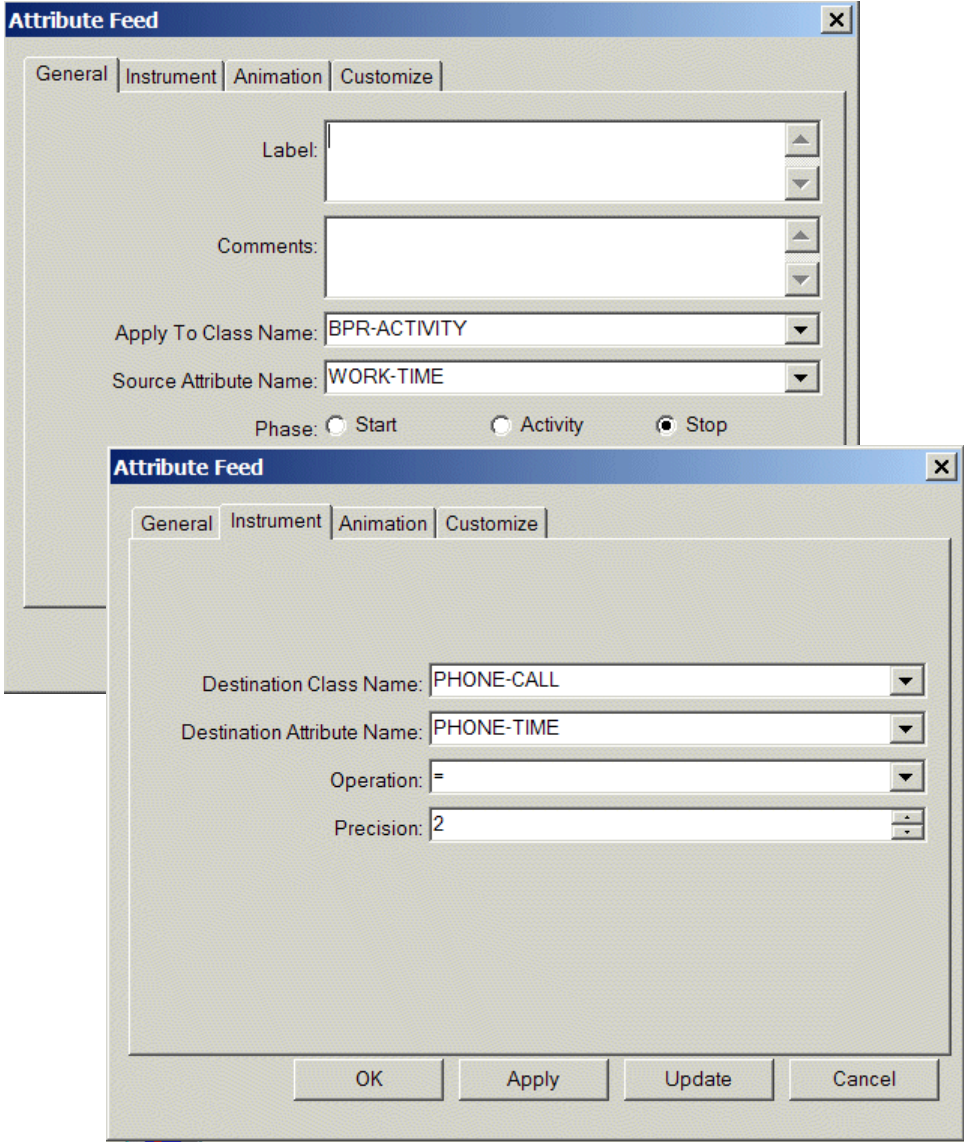
model then branches work, based on the length of each call and counts the number of calls.

The following figure shows the result of choosing Snapshot Activities on the Make Call task and showing its properties dialog. Notice that the Work Time is 2918 seconds.



The Attribute feed configure the Phase as **Stop** so it can copy attributes to the output work object after the Make Call task evaluates. The source attribute is the work-time of the bpr-activity, and the target attribute is the phone-time of the sales-call.

Here are the General and Instrument tabs of the properties dialog for the feed:



Specific Attributes

The screenshot shows the 'Attribute Feed' dialog box with the 'General' tab selected. The dialog has a title bar with a close button (X) and four tabs: 'General', 'Instrument', 'Animation', and 'Customize'. The 'General' tab contains the following fields and controls:

- Label:** A text input field.
- Comments:** A text input field.
- Apply To Class Name:** A dropdown menu with 'BPR-OBJECT' selected.
- Source Attribute Name:** A dropdown menu with 'NONE' selected.
- Phase:** Three radio buttons labeled 'Start', 'Activity', and 'Stop'. The 'Stop' radio button is selected.
- Counter:** A text input field containing the value '0'.
- Error:** A text input field.

At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Update', and 'Cancel'.

The screenshot shows the 'Attribute Feed' dialog box with the 'Instrument' tab selected. The dialog has a title bar with a close button (X) and four tabs: 'General', 'Instrument', 'Animation', and 'Customize'. The 'Instrument' tab contains the following fields and controls:

- Destination Class Name:** A dropdown menu with 'NONE' selected.
- Destination Attribute Name:** A dropdown menu with 'NONE' selected.
- Operation:** A dropdown menu with '=' selected.
- Precision:** A text input field containing the value '0'.

At the bottom of the dialog are four buttons: 'OK', 'Apply', 'Update', and 'Cancel'.

The specific attributes of the Attribute feed are:

Attribute	Description
Phase	(General tab) Determines when the feed copies the value of the source attribute to the destination attribute. A value of Start copies the value before the attached block applies its duration to the simulation. A value of Activity copies the value after the block applies its duration but before the block stops processing. A value of Stop copies the value after the block applies its duration and after it stops processing. Use Activity to copy values from the input work object to the block. The default Phase is Stop .
Destination Class Name	The class name of the object to which the feed copies its value. The feed copies the value of the attribute named by Source Attribute Name to the attribute named by Destination Attribute Name.
Destination Attribute Name	An attribute of the object named in Destination Class Name to which the feed copies a value.
Operation	<p>The mathematical operation to use when copying the value of the source attribute to the destination attribute. The options are: =, +, -, *, /, and E. The default value is =, which copies the exact value.</p> <p>For example, if the source attribute is 100, the destination attribute value is currently 500, and the operation is -, the new value of the destination attribute will be 400.</p>
Precision	The number of decimal places to round the value of the attribute named by Destination Attribute.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

The Attribute feed has no specific menu choices.

For information on the menu choices common to all feeds, see [Common Menu Choices for Feeds and Probes](#).

Change Feed



You use the Change feed to modify an attribute of the model, using a:

- New value
- Random value
- Unique ID
- Mathematical distribution

You typically use a Change feed to modify various attribute values of blocks or resources to experiment with different values. By creating sliders and type-in boxes from the Change feed, you create an easy way of supplying input parameters to the model. However, you can also use the Change feed to generate values for you.

For example, you can use a Change feed to modify:

- The rate at which work flows into the model by feeding values into the Mean attribute of a Source block.
- The number of objects in a batch by feeding values into the Threshold attribute of a Batch block.
- The hourly wage of a resource by feeding values into the Cost per Time Unit attribute of a resource.
- Symbolic values of user-defined attributes of the model, for example, the type of carrier.

Note When ReThink creates a slider or type-in box, it also creates a remote and places it on the object's detail. You do not need to use the remote when you configure sliders and type-ins.

Feeding New Values into Attributes of Blocks

You use a Change feed to modify attributes of blocks to experiment with different parameters. To do this, you configure the Change feed, then create a slider or type-in from the feed. You can use the slider or type-in box to feed values into blocks while the model is running.

You can feed quantitative, textual, or symbolic values into a block.

Feeding Quantitative Values

To feed quantitative values into attributes of blocks:

- 1 Connect a Change feed to the block whose attribute you want to change.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be bpr-block.
- 3 Configure Destination Attribute Name to be the attribute of the block whose value you want to change.
- 4 Click the Instrument tab and configure the Change Mode to be Value, the default.
- 5 Configure the New Value attribute to be the initial value for the Change feed.

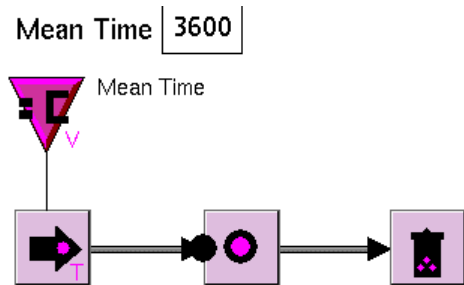
Note When feeding time-based values into the model, you must configure the number of seconds, for example, 3600 for 1 hour, 7200 for 2 hours, and so on.

- 6 To provide an interactive way of feeding new values into the model, choose Create Slider or Create Type In on the Change feed, depending on how you want to enter the new value.

ReThink creates a slider or type-in box just above the Change feed labeled New Value.
- 7 Configure the slider or type-in box, as follows:
 - a Click exactly on the border of the type-in box or exactly on the slider to display the properties dialog.
 - b Configure the Label.
 - c Configure the Value On Activation as a number or use the default, which is none.

If you specify none, the slider or type-in box uses the current value of the Change feed as the default value when you reset the model.
 - d If you create a slider, configure the Minimum Value and Maximum Value attributes of the slider.
 - e To move the slider or type-in box, choose Tools > Developer Mode, move the slider to a different location, then go back into modeler mode.
- 8 To feed a value into the model, adjust the slider or enter a value into the type-in box.

This example shows how you use a type-in box to modify the Mean of a Source block:



Here are the General and Instrument tabs of the properties dialog for the Change feed:

The image displays two overlapping dialog boxes for configuring a 'Change Feed: Mean Time'. The top dialog is in the 'General' tab, and the bottom dialog is in the 'Instrument' tab.

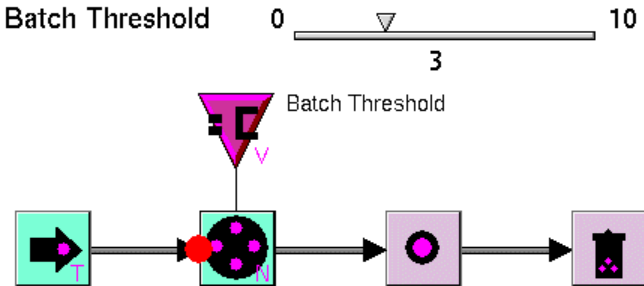
Change Feed: Mean Time (General Tab)

- Label: Mean Time
- Comments:
- Apply To Class Name: BPR-BLOCK
- Destination Attribute Name: MEAN
- Phase: Start Stop

Change Feed: Mean Time (Instrument Tab)

- Change Mode: Value
- New Value: 3600
- Non Quantitative New Value:
- Minimum Value: 0
- Maximum Value: 0
- Initial Id: 1
- Distribution:
 - Distribution Mode: Random Normal
 - Mean: 0
 - Standard Deviation: 0

This example shows how you use a slider to modify the Threshold of a Batch block:



Here are the General and Instrument tabs of the properties dialog for the Change feed:

The image displays two overlapping dialog boxes for configuring a 'Change Feed: Batch Threshold'. The top dialog is on the 'General' tab, and the bottom dialog is on the 'Instrument' tab.

Change Feed: Batch Threshold (General Tab)

- Label: Batch Threshold
- Comments:
- Apply To Class Name: BPR-BLOCK
- Destination Attribute Name: THRESHOLD
- Phase: Start Stop
- OK

Change Feed: Batch Threshold (Instrument Tab)

- Change Mode: Value
- New Value: 1.0
- Non Quantitative New Value:
- Minimum Value: 0
- Maximum Value: 0
- Initial Id: 1
- Distribution
 - Distribution Mode: Random Normal
 - Mean: 0
 - Standard Deviation: 0
- OK Apply Update Cancel

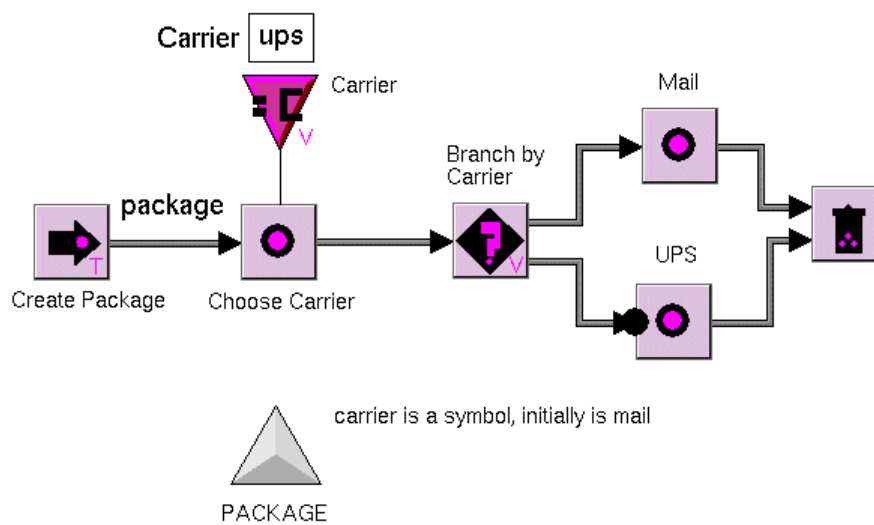
Feeding Symbolic or Textual Values

To provide an interactive way of feeding symbolic or textual values into the model, you use a type-in box.

To feed symbolic or textual values into attributes of blocks:

- ➔ Follow the steps above for feeding quantitative values into attributes of a block, except instead of configuring the New Value attribute, configure the New Non Quantitative Value.

This example shows how you use a type-in box to feed symbolic values into the Carrier attribute of a user-defined work object of type package:



Here are the General and Instrument tabs of the properties dialog for the Change feed:

The image displays two overlapping dialog boxes titled "Change Feed: Carrier".

The top dialog box is on the "General" tab. It contains the following fields and controls:

- Label: Carrier
- Comments: (empty)
- Apply To Class Name: PACKAGE
- Destination Attribute Name: CARRIER
- Phase: Start Stop
- OK button

The bottom dialog box is on the "Instrument" tab. It contains the following fields and controls:

- Change Mode: Value
- New Value: 0.0
- Non Quantitative New Value: UPS
- Minimum Value: 0
- Maximum Value: 0
- Initial Id: 1
- Distribution section:
 - Distribution Mode: Random Normal
 - Mean: 0
 - Standard Deviation: 0
- OK, Apply, Update, and Cancel buttons

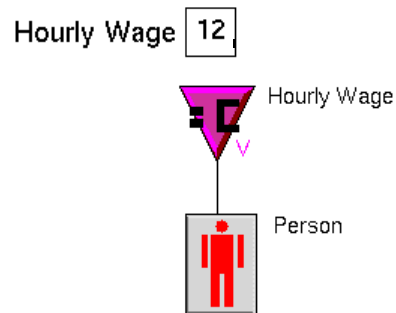
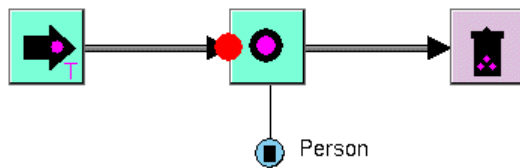
Feeding New Values into Attributes of Resources

You can use a Change feed to modify attributes of resources to experiment with different sets of resource parameters. You follow the same steps as you do when feeding new values into attributes of blocks, except that you connect the feed directly to the resource and you specify `bpr-resource` as the class to which the feed applies. You can use a slider or type-in box to feed values into resources while the model is running.

To feed values into attributes of a resource:

- 1 Connect a Change feed to the resource whose attribute you want to change.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be `bpr-resource`.
- 3 Follow steps 3 through 8 outlined under [Feeding New Values into Attributes of Blocks](#).

This example shows how you use the Change feed to modify the hourly wage of a resource:



Here are the General and Instrument tabs of the properties dialog for the Change feed:

The image displays two overlapping screenshots of the 'Change Feed: Hourly Wage' dialog box. The top screenshot shows the 'General' tab with the following fields:

- Label: Hourly Wage
- Comments:
- Apply To Class Name: BPR-RESOURCE
- Destination Attribute Name: COST-PER-TIME-UNIT
- Phase: Start Stop

The bottom screenshot shows the 'Instrument' tab with the following fields:

- Change Mode: Value
- New Value: 10
- Non Quantitative New Value:
- Minimum Value: 0
- Maximum Value: 0
- Initial Id: 1
- Distribution section:
 - Distribution Mode: Random Normal
 - Mean: 0
 - Standard Deviation: 0

Buttons for 'OK', 'Apply', 'Update', and 'Cancel' are visible at the bottom of the dialog.

Generating Random Numbers and Unique IDs

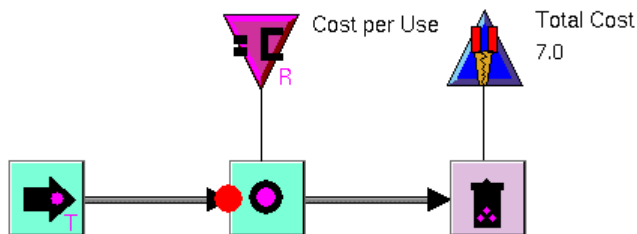
You can use the Change feed to generate random numbers between a maximum and a minimum number, or unique IDs that increment starting from an initial ID.

To generate random numbers:

- 1 On the General tab of the properties dialog, configure the Apply to Class Name and the Destination Attribute Name.
- 2 Click the Instrument tab and configure the Change Mode to be Random.
- 3 Configure the Minimum Value and Maximum Value attributes to be minimum and maximum values for the random number.

ReThink generates random numbers between the minimum and maximum values.

For example, you could use feed random numbers into the cost-per-use of a bpr-block to configure random fixed costs:

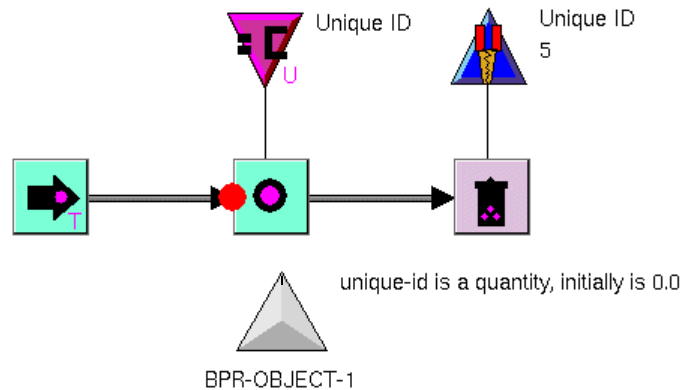


To generate unique IDs:

- 1 On the General tab of the properties dialog, configure the Apply to Class Name and the Destination Attribute Name.
- 2 Click the Instrument tab and configure the Change Mode to be Unique ID.
- 3 Configure the Initial ID to be the initial value for the unique ID.

ReThink generates unique IDs starting with the initial ID and incrementing it by one for each new ID.

For example, you could feed the unique ID into a user-defined attribute of a work object to create a unique ID:



Generating Random Numbers Based on a Distribution

You can use a Change feed to feed random numbers that are generated based on a mathematical distribution. You use the same distributions that you use to configure the Mode Type of a block, including random normal, random exponential, random triangular, and so on.

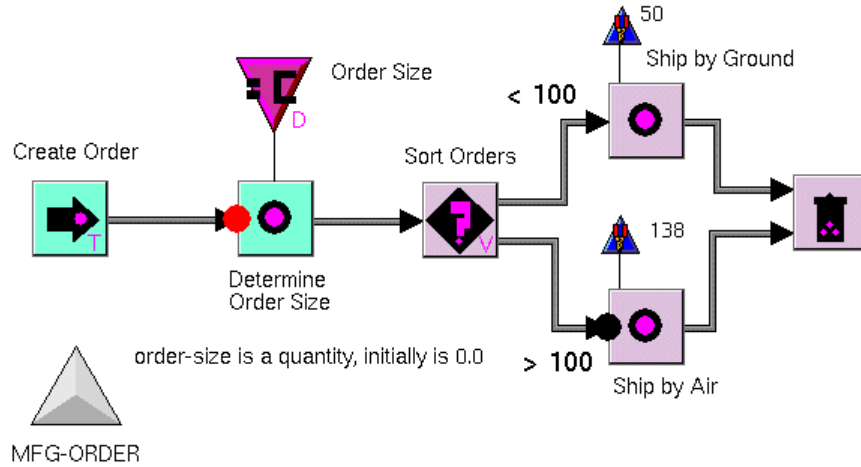
To generate random numbers based on a distribution:

- 1 On the General tab of the properties dialog, configure the Apply to Class Name and the Destination Attribute Name.
- 2 Click the Instrument tab and configure the Change Mode to be Distribution.
- 3 Configure the Distribution Mode to be the random distribution to use, then configure the mode-specific attributes for the distribution you choose.

For details, see [Specifying a Fixed Duration](#) and [Specifying a Random Duration](#).

ReThink generates random numbers, based on the mathematical distribution.

This example shows how you use the Change feed to feed a random number, based on a triangular distribution, into a user-defined attribute named order-size defined for a mfg-order:



Here are the General and Instrument tabs of the properties dialog for the Change feed:

The image displays two overlapping screenshots of the 'Change Feed: Order Size' dialog box. The top screenshot shows the 'General' tab with the following fields:

- Label: Order Size
- Comments:
- Apply To Class Name: MFG-ORDER
- Destination Attribute Name: ORDER-SIZE
- Phase: Start Stop

The bottom screenshot shows the 'Instrument' tab with the following fields:

- Change Mode: Distribution
- New Value: 96.0
- Non Quantitative New Value:
- Minimum Value: 0
- Maximum Value: 0
- Initial Id: 1
- Distribution section:
 - Distribution Mode: Random Normal
 - Mean: 100
 - Standard Deviation: 50

Buttons for 'OK', 'Apply', 'Update', and 'Cancel' are visible at the bottom of the dialog boxes.

Specific Attributes

The screenshot shows a dialog box titled "Change Feed" with a close button (X) in the top right corner. It has three tabs: "General", "Instrument", and "Animation", with "Animation" currently selected. The dialog contains several input fields and dropdown menus:

- Change Mode:** A dropdown menu set to "Value".
- New Value:** A text input field containing "0.0".
- Non Quantitative New Value:** An empty text input field.
- Minimum Value:** A text input field containing "0" with increment and decrement arrows on the right.
- Maximum Value:** A text input field containing "0" with increment and decrement arrows on the right.
- Initial Id:** A text input field containing "1" with increment and decrement arrows on the right.
- Distribution:** A section containing:
 - Distribution Mode:** A dropdown menu set to "Random Normal".
 - Mean:** A text input field containing "0" with increment and decrement arrows on the right.
 - Standard Deviation:** A text input field containing "0" with increment and decrement arrows on the right.

At the bottom of the dialog are four buttons: "OK", "Apply", "Update", and "Cancel".

The specific attribute of the Change feed is:

Attribute	Description
Change Mode	Specifies the type of value the Change feed generates. The options are: Value , Random , Unique ID , Distribution , and Custom .

The mode-specific attributes of the Change feed are:

Mode	Attribute	Description
Value	New Value	A new value for the attribute of the class that the Destination Attribute attribute of the feed specifies. Typically, you set this value with a slider or type-in box, rather than through the dialog. The New Value attribute can be a number, symbol, text string, true, or false.
	New Non Quantitative Value	A new value for the destination attribute when the value is other than a quantity, for example, a symbol.
Random	Minimum Value	The minimum value of the randomly generated number.
	Maximum Value	The maximum value of the randomly generated number.
Unique ID	Initial ID	The initial value for the unique ID.
Distribution	Distribution Mode	The distribution to use when generating the value. For information on the mode-specific attributes for each mode type, see Specifying a Fixed Duration and Specifying a Random Duration .
Custom	Change Procedure Name	See Customization Attributes .

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

The specific menu choices of the Change feed are:

Menu Choice	Description
Create Slider	Creates a slider for setting the New Value or New Non Quantitative Value attribute of the Change feed.
Create Type In	Creates a type-in box for setting the New Value or New Non Quantitative Value attribute of the Change feed.
Show Slider	Places an indicator arrow next to any sliders that have been created from this feed.
Show Type In	Places an indicator arrow next to any type-in boxes that have been created from this feed.

For information on the menu choices common to all feeds, see [Common Menu Choices for Feeds and Probes](#).

Customization Attributes

The customization attribute available in Developer mode for the Change feed is:

Attribute	Description
Change Procedure Name	When Change Mode is Custom, this attribute specifies the procedure name that determines how to compute the value of the feed. The default value is bpr-change-value.

For more information, see the *Customizing ReThink User's Guide*.

Copy Attributes Feed



The Copy Attributes feed copies attributes *from* a source object *to* the object to which the feed applies. You can use the Copy Attributes feed to copy metrics computed in a higher-level Task block to work objects on the detail.

You can configure the Copy Attributes feed to copy the exact value of an attribute, or you can configure it to use an operator or function. For example, you might want to sum the order size of each work object that the task on the detail processes and store the value in an attribute of the superior task.

Compare the Copy Attributes feed with the Copy Attributes probe, which copies attributes *from* the object to which the feed applies *to* a destination object.

Copying Attributes from a Block to a Work Object

To copy attributes from a block to a work object, you attach a Copy Attributes feed to a block on the detail and configure it to copy attributes from the block to an attribute of a work object. You configure the class to which the probe applies, which defines the destination object, and the source class.

For example, you might want to copy attributes from a superior task to a work object on the detail, or you might want to copy an attribute of a block to the work object, such as the Yield Value of a Yield block.

For information about creating a subclass of the Task block with additional attributes that the model copies to objects on the detail, see [Copy Attributes Probe](#).

To copy metrics from a block to a work object:

- 1 Connect a Copy Attributes feed to the block in the model whose attributes you want to copy to a work object.
- 2 On the General tab of the properties dialog, configure the Apply to Class Name to be the class that triggers the feed to copy attributes.

The Apply to Class Name class is the destination class to which the Copy Attributes feed copies attributes.

For example, to copy attributes to a work object, configure the Apply to Class Name to be `bpr-object` or a subclass.

- 3 Click the Instrument tab and configure the Source Class Name to be the class name of the object from which the Copy Attributes feed copies attributes.

For example, if you are copying metrics from a Yield block to the output work object, you would configure the Source Class Name to be `bpr-block` or a subclass.

- 4** For each attribute to copy, create and configure a row in the List of Operations group, as follows:

- a** Click Add Row to create a new row.
- b** Configure the Source Subtable to be the name of the subtable in which the attribute to copy is defined, if any.

The Source Subtable corresponds to each tab page of the object on which the attribute to copy appears, for example, `duration-subtable` or `cost-subtable`. If the attribute is on any tab other than the Duration or Cost tab, leave the Source Subtable blank.

- c** Configure the Source Attribute to be the name of the attribute of the source object to copy.

For example, you might copy the `total-starts` of the superior task to an attribute of a work object on the detail.

- d** Configure the Destination Subtable to be the name of the subtable in which the copied attribute is defined.

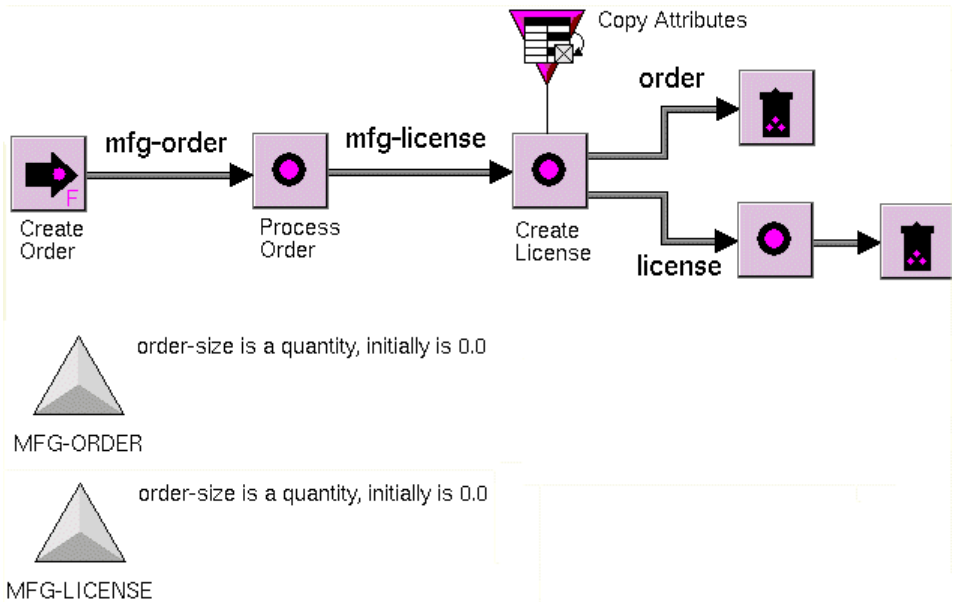
You only need to configure this attribute if you are copying attributes from a subtable.

By default, the feed copies the exact value of the source attribute into the destination attribute; however, you can also apply a mathematical operator or function to the source attribute to compute the destination attribute.

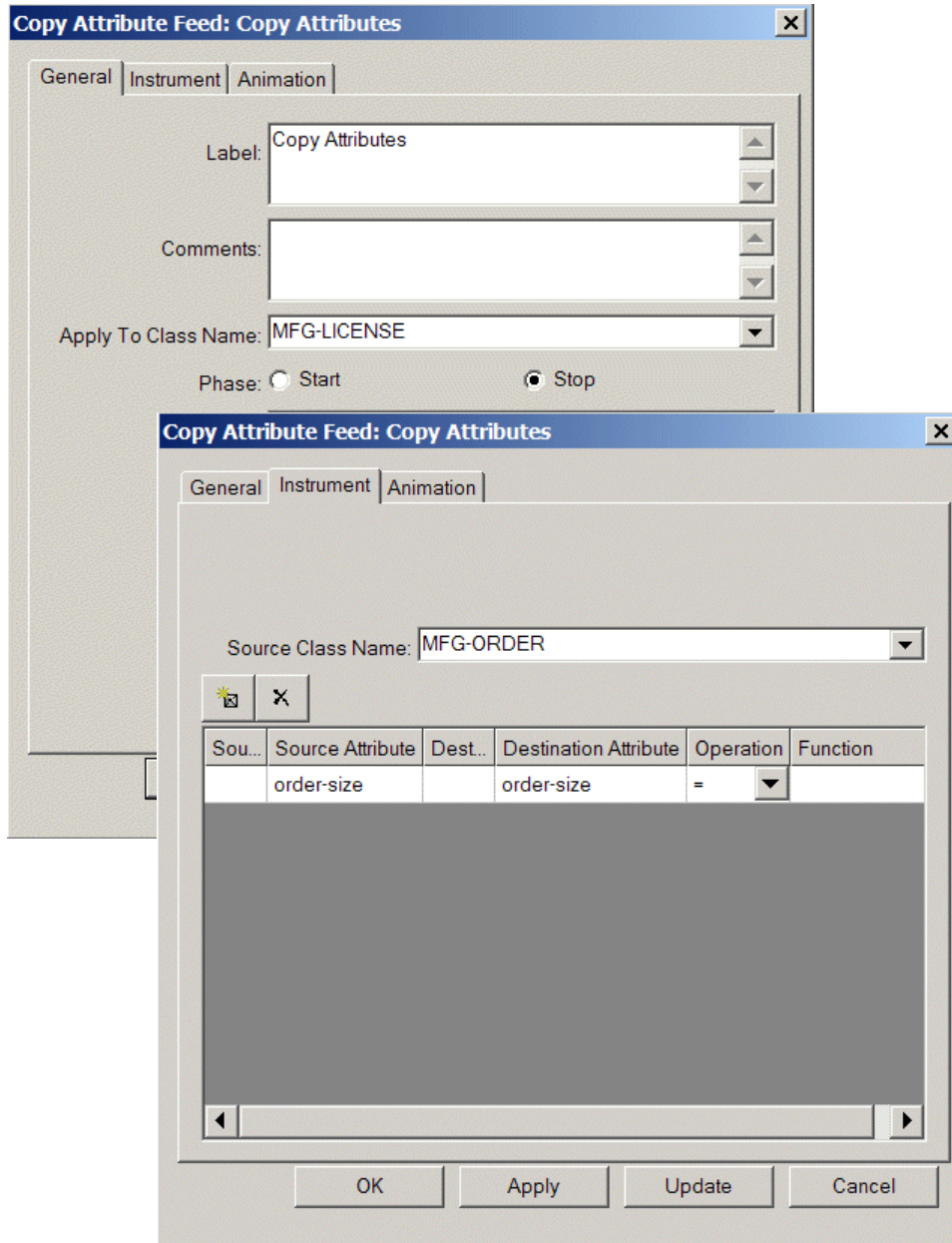
- e** Choose the Operator from the available list, or configure the Function to be the name of a G2 function to apply.

Some examples of functions are: `max`, `min`, or `average`. For details, see Chapter 25 “Functions” in the *G2 Reference Manual*.

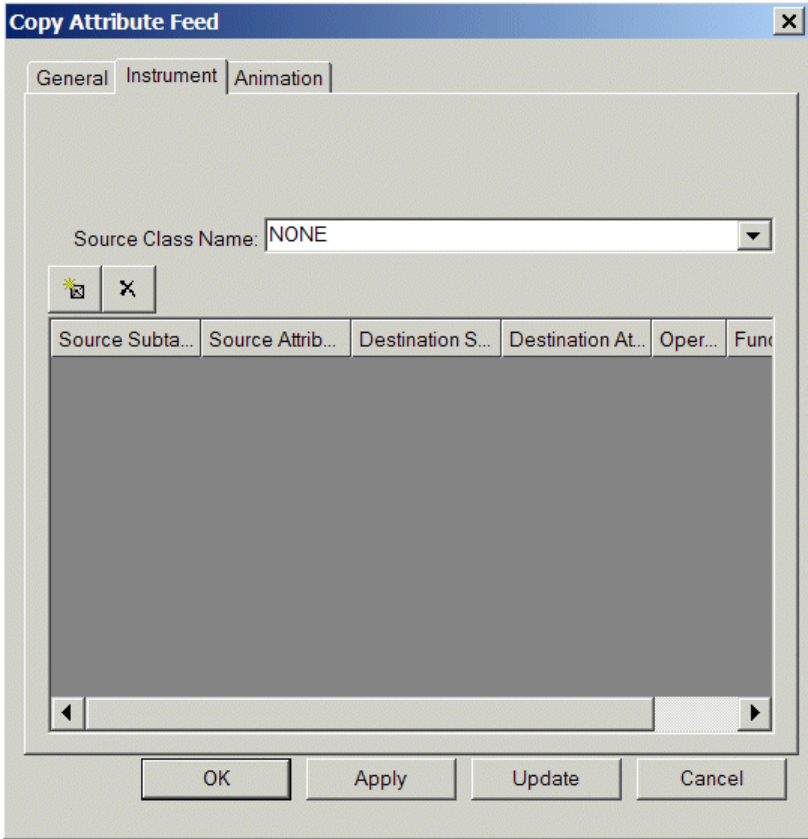
The following example shows how to copy an attribute from one work object to another:



The Copy Attributes feed specifies Apply to Class Name as mfg-license, which is the destination object, and mfg-order as the Source Class Name. It copies the order-size from one object to the other, using the = operator.



Specific Attributes



The specific attributes of the Copy Attributes feed are:

Attribute	P/M	Description
Source Class Name	P	The class name of the object whose source attributes the feed copies. The feed copies attributes to the destination object specified in the Apply to Class Name attribute.
Source Subtable Source Attribute Destination Subtable Destination Attribute Operation Function	P	<p>The list of operations to perform when the Copy Attributes feed updates.</p> <p>It copies the Source Attribute of the source object to the Destination Attribute of the destination object, based on the specified Operator or Function.</p> <p>If the Source Attribute and Destination Attribute are in a subtable of the work object, you can configure the Source Subtable Name and Destination Subtable Name, for example, duration-subtable or cost-subtable.</p>

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Copy Attributes feed has no specific menu choices.

For information on the menu choices common to all feeds, see [Common Menu Choices for Feeds and Probes](#).

Increment Feed



The Increment feed increments a counter by a value. For example, you use this feed to report on the number of times a work object has gone around a loop in the model.

You specify the target attribute in which the count accumulates, the amount by which to increment the counter, and an initial count.

By default, the Increment feed adds the increment value to the destination attribute. However, you can specify a different mathematical operation to perform such as subtraction, multiplication, division, or exponentiation. For example, if the operation is subtraction, the feed subtracts the incremented value from the destination attribute.

Incrementing a Counter

To use the Increment feed, first create a class definition with a class-specific attribute that is the counter. The Increment feed refers to this attribute as the destination attribute of the feed. Next, configure the initial value for the counter and the amount by which to increment the counter. You can also configure an initial value for the counter.

To increment a counter in an attribute of an object:

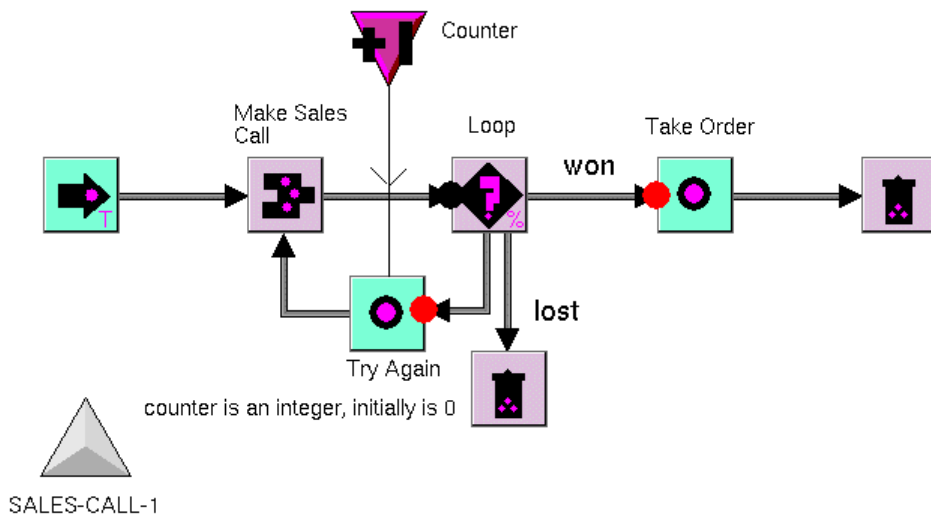
- 1 Create a class definition with a class-specific attribute, which is the counter.
For details, see [Creating a New Class of Work Object](#).

Note If you specify an initial value for the attribute that is the counter, the model uses this default as the initial value for the counter, rather than the value of the Initial Count attribute of the feed.

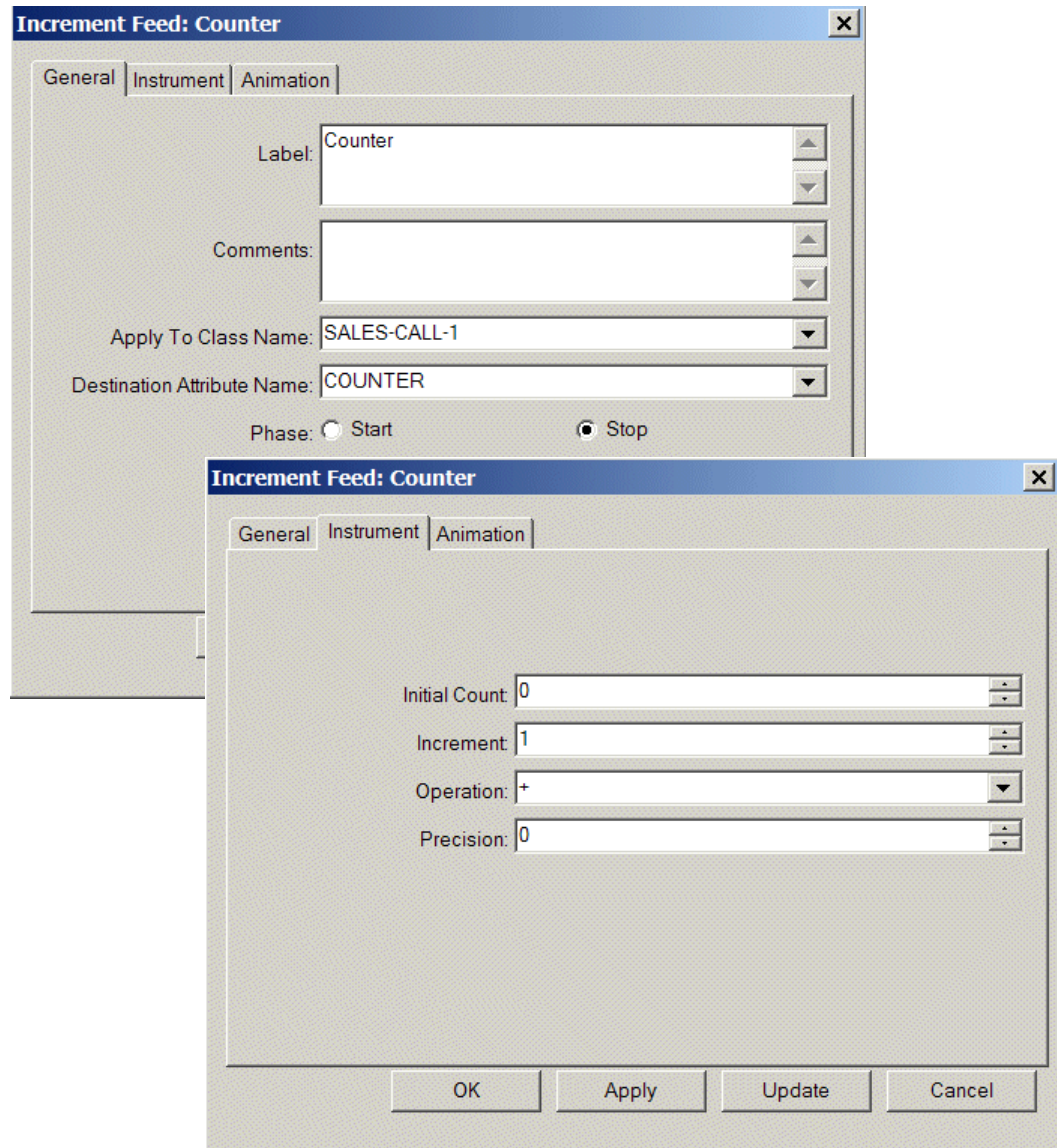
- 2 Connect an Increment feed to a block through which a work object passes multiple times.
- 3 On the General tab of the properties dialog, configure Apply to Class Name to be the work object that the model processes.
- 4 Configure Destination Attribute Name to be the attribute of the work object that is the counter.

- 5 Click the Instrument tab and configure the Initial Count to be the starting number for the counter.
The default value is 0.
- 6 Configure the Increment to be the amount by which to increment the count.
The default value is 1.
- 7 Configure the Operation attribute to be the mathematical operation that the value of the Increment attribute should perform on the destination attribute.
By default, the Increment feed adds the Increment value to the destination attribute.
- 8 Configure the Precision attribute to be the number of decimal places to round the incremented value.

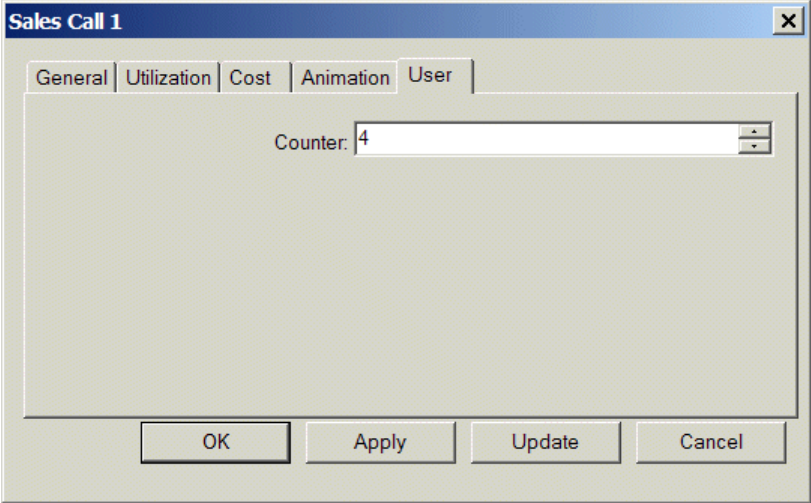
In this model of a sales process, the Branch block passes some percentage of the sales calls back through the process to make another sales call. The Increment feed is attached to the Try Again task to determine how many return sales calls each work object requires.



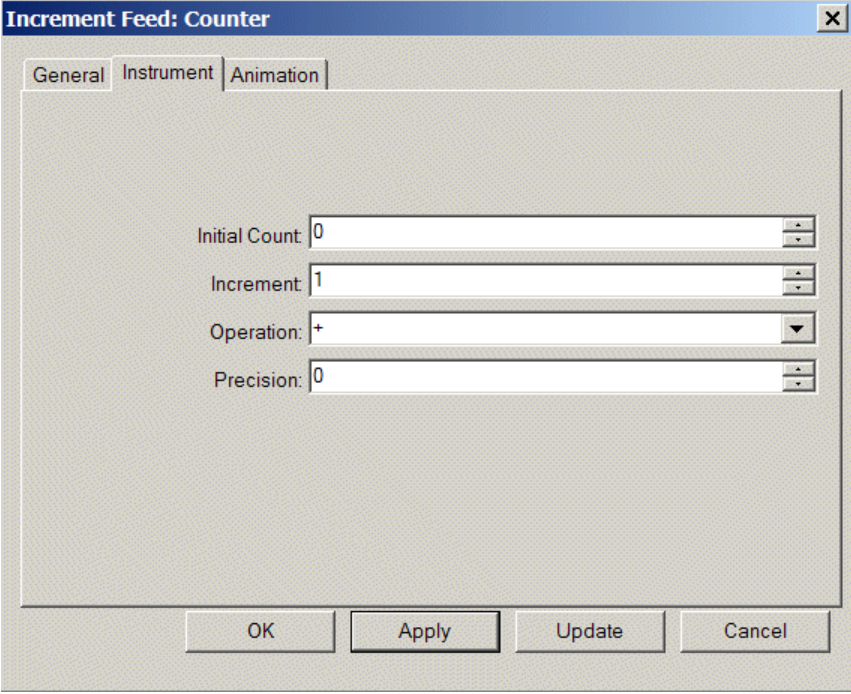
If you probe the sales call just before it is absorbed by the Sink block, you can determine the number of times the sales call went around the loop. Here is the properties dialog for the Increment feed, which applies to the `sales-call-1` class and uses the default values for Initial Count and Increment. The feed specifies the `counter` attribute of the `sales-call-1` as the Destination Attribute Name.



Here is the User tab of the properties dialog for a sales call object that had four return sales calls applied to it, as indicated by the value of the Counter attribute:



Specific Attributes



The specific attributes of the Increment feed are:

Attribute	Description
Initial Count	The starting number for the counter.
Increment	The amount by which to increment the attribute named by the Destination Attribute Name.
Operation	<p>The mathematical operation that the Increment attribute performs on the destination attribute. The options are: +, -, *, -, and E.</p> <p>For example, if Increment is 1, the destination attribute is currently 5, and the operation is -, the new value of the destination attribute will be 4.</p>
Precision	The number of decimal places to round the incremented value.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

The Increment feed has no specific menu choices.

For information on the menu choices common to all feeds, see [Common Menu Choices for Feeds and Probes](#).

Parameter Feed



A parameter feed gets the value of a parameter and sets it as the current value of an attribute of the model. You create the parameter from the feed.

You can also use the Parameter feed in conjunction with any type of parameter or variable, which you can create in Developer mode.

Used in conjunction with a Parameter probe, the Parameter feed enables you to get the value of a parameter that comes from an attribute of the model. For details, see [Parameter Probe](#).

Getting the Value of a Parameter

To get the value of a parameter:

- 1 Connect a Parameter feed to an object in the model whose attribute values you want to set.

Connect the feed to a block to set an attribute of the block, its activities, or the work objects, or connect the feed to a resource or another probe.

- 2 Choose Create Parameter on the Parameter feed to create a uniquely named quantitative parameter.

The Parameter feed sets the Parameter Name on the Instrument tab to the name of the parameter.

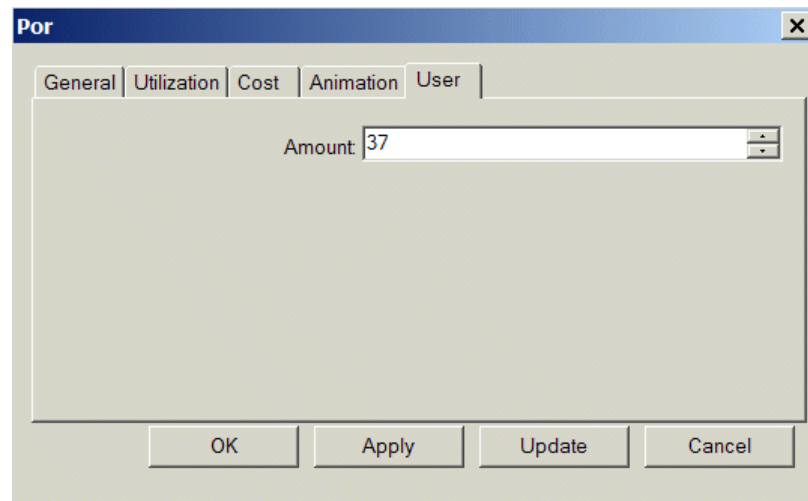
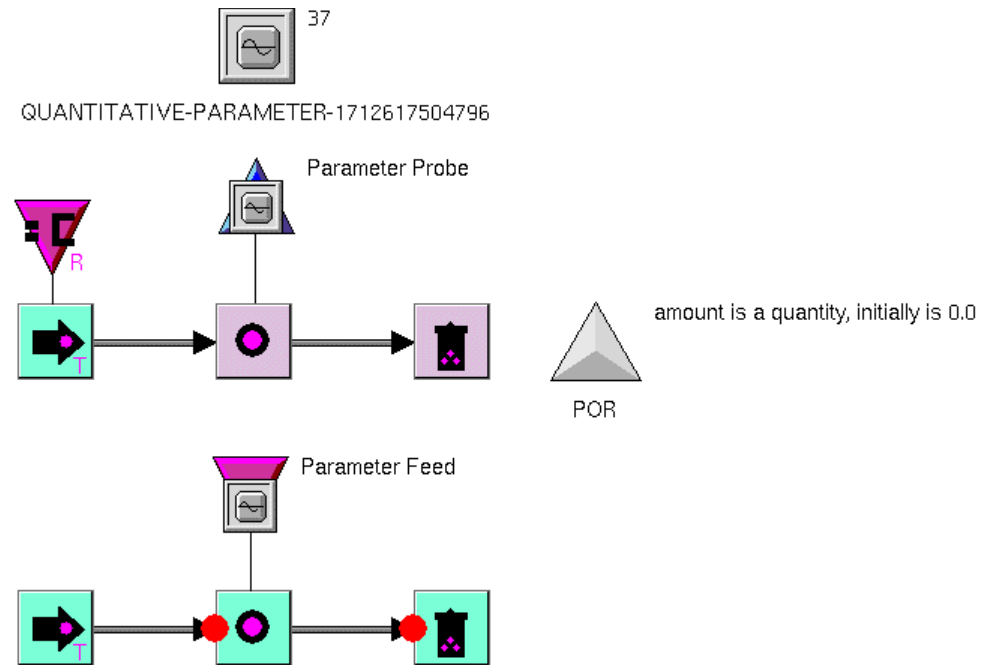
Note If you have already created a parameter, choose the Choose Parameter menu choice on the Parameter feed to choose the existing parameter, or configure the Parameter Name to be the name of the parameter.

- 3 On the General tab of the properties dialog, configure Apply to Class Name to be the object of the model whose attribute values you want to set.
- 4 Configure the Destination Attribute Name to be the attribute of the class whose values you want to set.

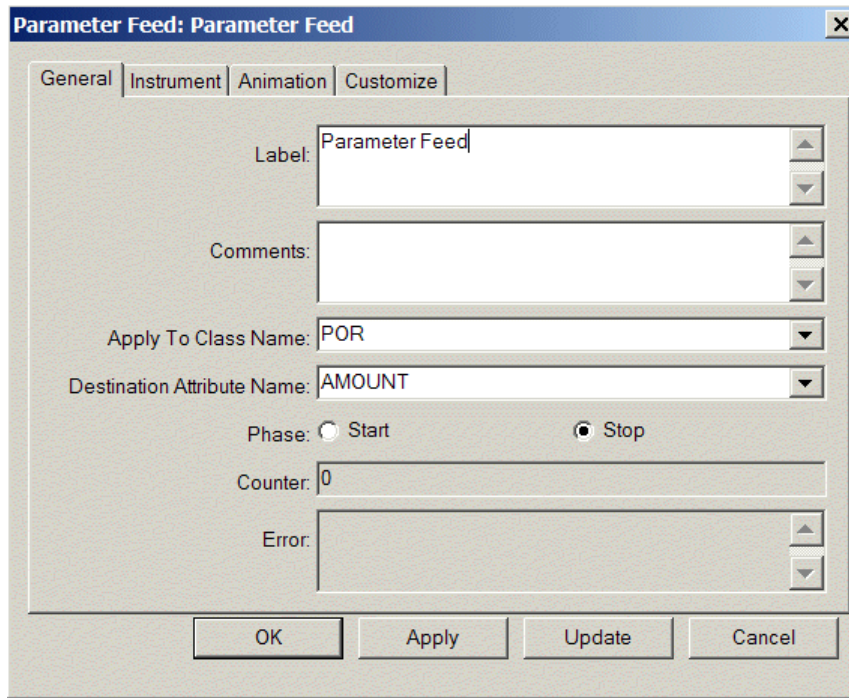
When you run the simulation, the Parameter feed gets the current value of the parameter and sets it as the current value of the specified attribute of the model.

For example, in the first model below, a Change feed feeds a random number into the **amount** attribute of a POR work object. A Parameter probe gets the current value of the **amount** attribute and sets it as the current value of the quantitative

parameter. In the second model, a Parameter feed gets the current value of the quantitative parameter and sets it as the current value of the amount attribute of the POR. The User tab of the properties dialog for the POR on the output path of the Task block in the second model corresponds to the current value of the quantitative parameter.



Here is the General tab of the properties dialog for the Parameter feed:



To see the feed associated with the parameter:

→ Choose Show Instruments on the parameter.

Getting Values from Different Types of Parameters

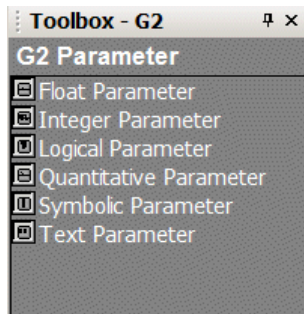
You might want to use a Parameter feed to keep integers, symbols, text strings, or truth values rather than quantitative values.

The parameter types are:

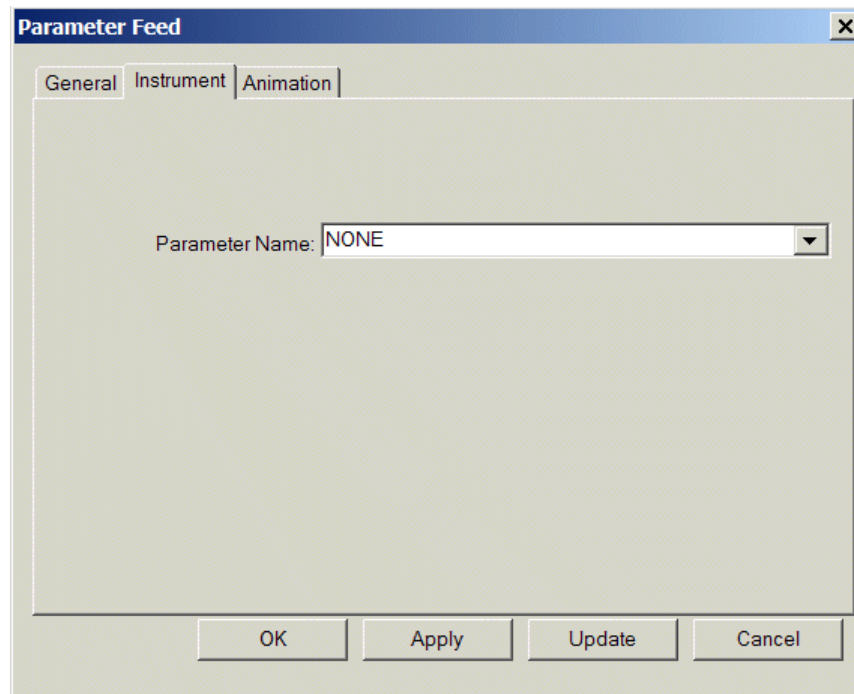
- Logical-parameter, which keeps true or false values.
- Quantitative-parameter, which keeps integers or floating point numbers.
- Integer-parameter, which keeps integers.
- Float-parameter, which keeps floating point numbers.
- Symbolic-parameter, which keeps symbols.
- Text-parameter, which keeps text strings.

To get values from different types of parameters:

- 1 Choose View > Toolbox - G2.
- 2 Click the G2 Parameter tab:



- 3 Create the desired type of parameter.
- 4 Choose the Choose Parameter menu choice on the Parameter probe to choose the existing parameter, then choose Select on the parameter to select it.
- 5 Create a model that sets the parameter values into attributes of the appropriate type.

Specific Attributes

The specific attribute of the Parameter feed is:

Attribute	Description
Parameter Name	The name of the parameter whose current value the feed should get. Choosing Create Parameter or Choose Parameter on the probe sets this attribute automatically.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

The specific menu choices of the Parameter feed are:

Menu Choice	Description
Create Parameter	Creates a unique named quantitative parameter and sets the Parameter Name attribute of the feed to the parameter name.
Choose Parameter	Associates an existing parameter with the Parameter feed, sets the Parameter Name attribute to the existing parameter, and generates a unique name, if needed. Choose Select on a parameter to select it.
Show Parameter	Places an indicator arrow next to the parameter associated with the Parameter feed.

For information on the menu choices common to all feeds, see [Common Menu Choices for Feeds and Probes](#).

Timestamp Feed



The Timestamp feed supplies the current time to an attribute of an object at any point in the model. You use the Timestamp feed in conjunction with a Delta Time probe to compute a partial cycle time for a model. You use a Delta Time probe to compare the timestamp to the current time, which computes the cycle time for that portion of the model.

Typically, you feed a timestamp into a work object of the model and probe the Delta Time of the work object at a point downstream in the model.

Feeding a Timestamp into a Work Object of the Model

To feed a timestamp into a work object, first define a work object class with a timestamp attribute, then configure the Timestamp feed.

To feed a timestamp into a work object of the model:

- 1 Create a class definition for a work object with a class-specific attribute that will hold the timestamp.

For example, you might create a class-specific attribute named `load-time` to indicate the time at which a box is loaded.

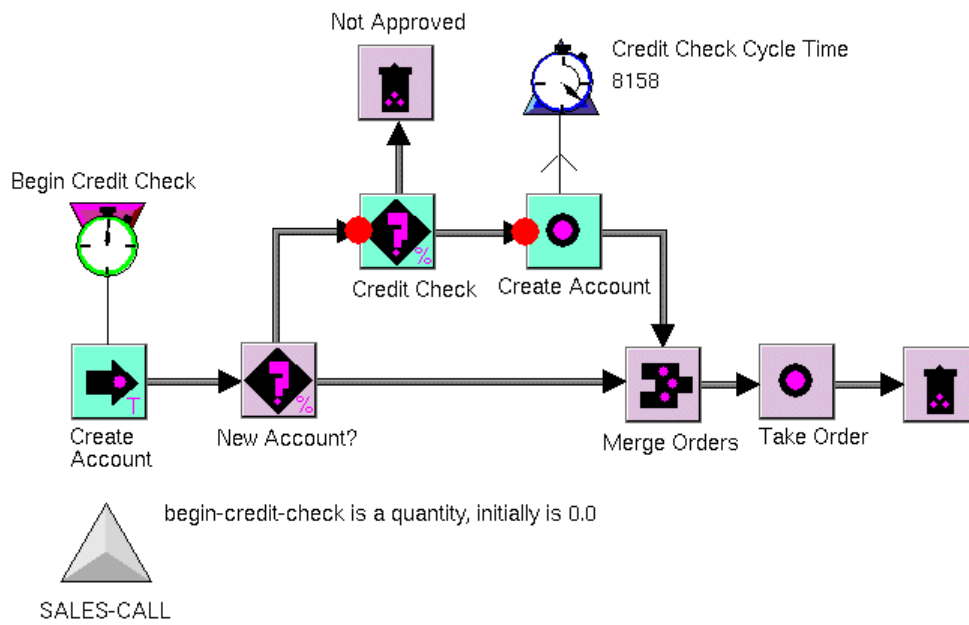
For details, see [Creating a New Class of Work Object](#).

- 2 Connect a Timestamp feed to a block in the model.

Typically, you feed a timestamp into a work object somewhere in the middle of the model, for example, at the beginning of a subtask.

- 3 On the General dialog of the properties dialog, configure Apply to Class Name to be the name of the work object that defines the timestamp attribute.
- 4 Configure the Destination Attribute Name to be the timestamp attribute of the work object.

For example, here is the Take Order detail, which computes a partial cycle time:



Timestamp Feed: Begin Credit Check

General | Animation | Customize

Label: Begin Credit Check

Comments:

Apply To Class Name: SALES-CALL

Destination Attribute Name: BEGIN-CREDIT-CHECK

Phase: Start Stop

Counter: 0

Error:

OK Apply Update Cancel

For a complete explanation of this example, see [Computing a Partial Cycle Time](#).

Specific Attributes

The Timestamp feed has no specific attributes.

For information on the common attributes, see [Common Attributes of Instruments](#).

Specific Menu Choices

A Timestamp feed has no specific menu choices.

For information on the menu choices common to all feeds, see [Common Menu Choices for Feeds and Probes](#).

A

activity: The amount of work associated with processing the inputs of a single block. Each time work objects flow to the input paths of a block, the block creates an activity object. Each activity adds value to the work object; activities have a duration and they can have an associated cost. *See also* concurrent activities.

allocate: To assign a resource to an activity. When the activity is complete, the resource is typically deallocated and can be allocated to another activity. You can allocate a resource for multiple sequential activities. You can allocate particular resources to an activity based on various criteria, such as cost, utilization, or priority. You can also constrain the availability of resources, using temporal constraints.

annotation: A tool that contains detail for adding documentation to a model.

association: A relationship between two or more work objects in a model, which you create by using an Associate block. Because the work objects are associated, they can flow apart in a process. You can create new associations between work objects, or you can add work objects to existing associations. You can reconcile associated work objects downstream in the process based on the association.

attribute displays: Text displayed next to an icon that shows the value of a particular attribute of the object. For example, all blocks show the Label attribute as an attribute display, whose default value is an empty string.

B

batch: A group of work objects that a Batch block processes together. By default, the Batch block waits until the specified number of work objects arrive at the block before passing them together to the downstream block. The block can also pass the batch downstream based on the value of an attribute of an input work object, a particular work object arriving at the block, or a specified time interval.

block: A graphical object that represents an operation within your model. Blocks operate on work objects. Blocks can have a duration and a cost. Blocks can also require resources. ReThink represents each block execution as an activity.

C

chart: A display of historical values that a probe in the model collects. *See also* probe.

concurrent activities: Activity objects that a block processes simultaneously. A block can process multiple activities concurrently, depending on constraints on the model. The number of concurrent activities depends on the arrival rate of work objects to the block, the duration of the task, the number of resources that are available, the maximum number of activities that the block specifies, and whether the block synchronizes its inputs.

connector: An object that connects the blocks on a higher-level workspace to the blocks on the detail of a task. Work objects flow from the top-level block, through the connectors, to the blocks on the detail, and back up through the connector to the top-level blocks.

constraints: *See* [temporal constraints](#).

container: A subclass of work object that defines an attribute into which the Insert block and the Batch block insert work objects, and from which the Remove block removes work objects. A container is an instance of the `bpr-container` class.

cost: The sum of all fixed and variable costs associated with tasks and resources that are applied to different objects in the model. For example, process costs measure costs as they apply to work objects, task costs measure costs as they apply to individual blocks, and resource costs measure costs as they apply to resources or resource pools that are allocated to a task.

creation time: The simulation time at which a ReThink object was created. The creation time of some ReThink objects, like blocks, is typically the start time of the simulation, while the creation time of other ReThink objects, like work objects, can be any time during the simulation when the work object is created.

cycle time: The amount of simulation time from one operation in a model to another. Cycle time is one way of measuring the performance of a process or subprocess. Cycle time is also called delta time.

D

deallocate: To finish using a resource for an activity. When a Resource Manager deallocates a resource, the manager can allocate it to some other activity.

detail: The subworkspace of a ReThink object. For example, a Task block can have detail to define its subprocesses, and a Model has detail on which you place blocks, instruments, and resources. Task block details allow work objects to flow from the top-level task, through the blocks on the detail, and then back up to the top-level blocks. The detail can be multiple levels deep.

duration: The amount of simulation time applied to a particular activity. ReThink computes duration differently for different types of objects. By default, the

duration of most blocks is computed based on a random normal function and represents the amount of simulation time the block has been processing work objects. The duration of a resource is the amount of simulation time the resource has been allocated to activities in the model. The duration of a work object is the amount of simulation time the blocks in the model have spent processing the work object.

E

elapsed time: The amount of simulation time that represents the entire duration of the activity. The total elapsed time associated with a block is the amount of time that has elapsed since the simulation began. *Contrast with* work time.

F

feed: A type of instrument that you use to assign values to objects in the model while the simulation is running. You can feed values into attributes of blocks, paths, work objects, resources, and instruments.

free text: Text that you can place anywhere on a workspace to label the model.

H

hierarchical view: A way of viewing a model such that only the relevant information is visible. You create hierarchical views in a model by adding detail to Task blocks. You can create nested levels of detail in a hierarchical view, as needed.

I

image definition: An object that defines a bitmap image, which ReThink can use as the background for the icon of a block, instrument, resource, or work object, or as the background of a workspace.

indicator: A large magenta arrow that a scenario displays next to an object when the user chooses certain menu choices, such as Show Scenario on any object or Choose Resource on a Resource Manager. The Scenario controls whether clicking the indicator removes it from the workspace, the default, or whether the indicator disappears automatically.

instrument: A type of ReThink object that either supplies values to the model or obtains values from the model. Feeds and probes are the two classes of instruments within ReThink. From ReThink instruments, you create user interface objects to observe and control your model; you create charts from probes and sliders and type-in boxes from feeds.

J

jump mode: The normal discrete event simulation mode. Events occur while a simulation is running. After each event, ReThink advances the simulation clock to the starting time of the next event. Work objects flow through the model continuously. *Contrast with* step mode and synch mode.

junction: A type of connection that exists on the end of a stub when you disconnect two blocks. You can drag a stub into a junction to connect two blocks. When a path contains a junction and you move a connected block, ReThink only reconfigures the path between the junction and the block you move; it does not reconfigure the path between the junction and the block connected to the path on the other side of the junction.

M

metrics: Attributes that the model computes, based on parameters that you configure.

mode: The discrete simulation mode in which a simulation runs, which can be continuous (jump mode), step-by-step (step mode), synchronized (synch mode), or online. You control the mode from the Scenario Control Panel. *See also* user mode.

model: A representation of some part of a business process that you create by using ReThink blocks, instruments, and resources. You create ReThink models on the detail of a Model tool.

module: A set of related information in an application. Typically, you store a single module in a single *.kb* file that has the same name as the module.

module hierarchy: The set of dependent modules that form an application. You can show the module hierarchy, and you can create, delete, rename, and merge modules.

O

organizer: A type of tool with detail on which you can place various types of objects, for example, resources and work object class definitions.

P

parameters: Inputs to a model that determine how the model behaves. For example, the mean time between orders, the hourly wage of a clerk, and the number of boxes in a truckload are all parameters of the model. The model computes metrics, based on parameters that you configure.

path: A connection between two blocks. Paths carry work objects from block to block within a model. Paths have a direction of flow, typically from left to right. Blocks can have input paths and output paths. You determine the type of work object that a block processes by specifying the path type. Paths automatically reconfigure when you move one of the connected blocks. *See also* stub.

path queue: The pending work objects for a particular task. When a block has constraints or when a block synchronizes its inputs, input paths can develop a backlog, called a work backup. Paths keep track of queuing metrics, such as wait time, making it easy to identify and diagnose bottlenecks in a process.

path type: The type of work object that a block receives on its input path or passes on its output path. The path type can be `bpr-object`, `bpr-container`, or any subclass.

phase: An attribute of an instrument that determines whether the instrument evaluates before or after the attached block applies its duration to the simulation. You use the start phase to feed and probe input work objects and paths, and you use the stop phase to feed and probe output work objects and paths.

pool: *See* [resource pool](#).

probe: A type of instrument that obtains values of objects while a simulation is running. You can obtain performance information about blocks, paths, activities, work objects, resources, and instruments by using probes. You create charts directly from probes to plot the history of probed values.

R

readout table: A display that shows the current value of an attribute of an object, which updates at a periodic interval. You use readout tables to display attributes of paths and other objects.

remote: An intermediate object that ReThink creates when you create a chart from a probe. You configure the remote to specify how the chart looks and behaves. Remotes contain a history of probed values.

report: A table of output metrics or input parameters for blocks, paths, probes, resources, and work objects. You can configure the update interval for output reports, and you can apply input report data to the model. You can create reports in the client, in an Excel spreadsheet, in a `.csv` file, or in a database.

resource: An object that a block requires to process its activities. Resources constrain the model, based on availability. You can also apply costs to a model through resources. You use Resource Managers to allocate and deallocate resources for particular tasks.

resource manager: Determines which resources a block uses to perform a particular task and how the block allocates and deallocates them. You create resource managers directly from resources and attach them to blocks that require the particular resource. By default, the resource manager chooses resources from a pool at random and schedules the blocks that are waiting for the resource when

the resource is unavailable. The resource manager also determines the utilization of the resources, which is the amount of resources that are required.

resource pool: A collection of resources available for a particular task. To create a pool for any resource, create detail for the resource and add resources.

S

scenario: The control center of a model. You control the simulation by creating an associated scenario, which keeps track of the simulation time of the model. The scenario advances the simulation clock for each discrete event and either continues running the model or pauses, depending on the mode in which you are running the model. You can use one scenario to control multiple models, or you can use different scenarios to control the same model.

sequential processing: A way of processing in which a block processes each work object in sequence as it arrives at the block. For example, a Merge block performs sequential processing of its inputs. *Contrast with* synchronized processing.

simulation time: The current time at which the simulation is running. Scenarios control when the simulation clock advances, and the duration of activities determines how much it advances for each event.

stand-alone model: A model that runs independently of other models in a knowledge base. You create a stand-alone model by placing a scenario on the detail of a model with the blocks, resources, and instruments that make up the model.

status: The current run status of a simulation, which can be running, stopped, or paused. You control the status from the Simulation menu.

step mode: The mode used for diagnosing and debugging the model. ReThink pauses after each event so that you can walk through the simulation one step at a time. When you continue running, the simulation clock immediately advances to the starting time of the next event. *Contrast with* jump mode and synch mode.

stub: A connection coming into or out of a block that is not yet connected to another block. You can connect stubs to other stubs, directly to a block, or to a junction. *See also* path.

surrogate: A different manifestation of an existing resource, usually placed in a different pool, which the model uses in another location. You use surrogates to share an individual resource between more than one task. For example, in a model of a delivery process, a truck loader might also act as a truck driver.

symbol: A string of alpha-numeric characters without spaces. Use hyphens in place of spaces in any symbol. All names and attributes must be specified as symbols, for example, `model-top-level-workspace`, `creation-time`, and `begin-credit-check`.

synch mode: The mode used to help visualize the delays in the process. ReThink scales the time of the simulation to real time. For example, you can use this mode to run the simulation at one hour simulation time per second of real time. Most of the time when you are building models, however, you focus on diagnosing the work flows and let the clock keep track of the simulation time. *Contrast with* jump mode and step mode.

synchronized processing: A way of processing in which a block waits to process its inputs until all of the inputs have arrived at the block. These blocks synchronize their inputs: the Task block, the Associate and Reconcile blocks, the Insert and Remove blocks, and the Copy Attributes block. *Contrast with* sequential processing.

T

task: A distinct operation within a process that adds value and applies costs to work objects. ReThink represents tasks dynamically as activities.

temporal constraints: Objects that allow you to constrain the date and time availability of resources. To configure the availability of resources, you connect a temporal scheduler to any resource and configure the constraints on the detail of the scheduler. You can configure the availability of resources by the date, month, week, and day.

top-level workspace: A named workspace that you create on which you create models. Each top-level workspace is assigned to a module.

total cost: For a block, the sum of the cost of all of individual activities, which includes the fixed and variable costs assigned to the block and the fixed and variable costs assigned to any resources associated with the block. For a work object, the sum of the individual activity costs applied to the work object.

total elapsed time: The amount of time that has elapsed since the simulation began.

total work time: The sum of all the work time of each activity currently scheduled by the block, as of the current simulation time.

U

user modes: Determines the privileges for different categories of users, such as moving, editing, and deleting objects. *See also* browser mode, modeler mode, and developer mode.

utilization: The amount of a resource allocated to an activity. You specify utilization in the Resource Manager that allocates resources to a task. You specify the amount of the resource that is available for a task in the resource itself. Resources and work objects compute various utilization metrics to analyze the efficiency of the process.

W

“what-if” analysis: A technique of business process reengineering in which you experiment with different parameters, resource constraints, costs, and work flows, to come up with a more efficient process.

wire: A connection between an instrument and another object, or between a Resource Manager and a block. ReThink automatically reconfigures wires when you move one of the connected objects.

work backups: Work objects that ReThink places in the path queue. Work backs up in a process when a block is too busy processing its current inputs to process the work in the queue. This happens when resource constraints exist, when the maximum number of activities for a block is specified, or when the block synchronizes its inputs.

work objects: The objects in a model on which blocks operate. Work objects represent the inputs and outputs of a process, for example, orders and invoices. Work objects compute summary duration and cost metrics that indicate the overall performance of the process. The type of work objects that the model processes depends on the path type, which is **bpr-object**, by default.

work time: The amount of simulation time that an object has actually been active. ReThink computes work time differently depending on the type of object. The work time of a work object is the amount of time the object has been actively operated on by blocks in the process. The work time of a resource is the amount of time the resource has been allocated. The work time of a block is the amount of time the block has been actively processing work objects. *Contrast with* elapsed time.

workspace: An area for creating a model, interfacing with an end user, and storing definitions. *See also* top-level workspace and detail.

Numerics

- 180 menu choice
 - controlling layout, using
 - Layout menu
- 90 Clockwise menu choice
 - controlling layout, using
 - Layout menu
- 90 Counterclockwise menu choice
 - controlling layout, using
 - Layout menu

A

- About ReThink menu choice
- Access Tables menu choice
- Access, Microsoft
- Accumulate feed
 - accumulating values, using
 - reference
- Acknowledge All Messages attribute
- Acknowledge Message probe
- Acknowledge Messages Upon Selection attribute
- action buttons, updating charts, using
- Activate menu choice
 - activating scenarios, using
 - Simulation menu
- activating scenarios
- Active Color attribute
 - of blocks
 - of instruments
 - of resources
- activities
 - See also* blocks
 - attributes of
 - constraining
 - using Maximum Activities
 - using resources
 - cost of
 - current
 - computing
 - showing snapshot of

- when allocating multiple and partial resources
- duration of
 - configuring custom
 - configuring fixed
 - configuring from a file
 - configuring from a report
 - configuring from a report lookup
 - configuring from an attribute
 - configuring random
 - customizing
 - showing allocated resources of
 - understanding for blocks
- Activity phase
- Add Remote menu choice
- Add to Associations attribute
 - of Copy block
 - of Retrieve block
- Address field
- adjusting
 - micro position of objects
 - order of objects
- Administrator mode
 - configuring user preferences for
 - description of
 - Tools menu
- Align or Distribute menu choice
 - controlling layout, using
 - distributing objects, using
 - Layout menu
 - of blocks
 - of instruments
- Allocate Resource attribute
- Alpha attribute
 - random beta distribution
 - random gamma distribution
- animation
 - configuring
 - for blocks
 - for instruments
 - for paths
 - for resources
 - for scenarios

- for surrogates
- Animation Color attribute
- Animation Repeat Counter attribute
- Animation Speed attribute
- Animation tab
 - blocks
 - configuring
 - reference
 - feeds
 - configuring
 - reference
 - paths
 - configuring
 - reference
 - probes
 - configuring
 - reference
 - resources
- annotating models
 - introduction to
 - using Annotation tool
 - using free text
 - using readout tables
- Annotation tool
- applications
 - interacting with objects in
 - navigating
- Apply menu choice
 - Excel toolbar
 - Reports menu
- Apply to Class Name attribute
 - configuring for feeds
 - updating system-defined attributes
 - updating user-defined attributes
 - configuring for probes
 - of feeds
 - of probes
- applying input report data
 - from databases
 - from Excel
- Arrival Rate Input Graph
 - configuring
 - creating
 - editing the shape of
- arrows, indicator
 - configuring behavior of
 - setting and clearing
- Associate All attribute
 - of Associate block
 - using
- Associate block
 - configuring specific features of
 - reference
- Association Name attribute
 - allocating associated resources, using
 - of Associate block
 - of Reconcile block
 - of Retrieve block
- associations
 - adding to existing
 - creating new
 - showing existing
- Attribute Change Event Report
 - description of
 - using
- attribute displays, using
- Attribute feed
 - copying attribute values, using
 - reference
- Attribute Lookup Report
 - configuring durations, using
 - description of
- Attribute Name attribute, of Batch block
- Attribute to Split attribute
- Attribute Value attribute
 - of Criteria probe
 - of Retrieve block
- attributes
 - See also* scenarios and resources
 - computing statistics for
 - configuring
 - block animation
 - block cost
 - block duration
 - block, general
 - block, specific
 - for feeds
 - for probes
 - for reports
 - using Batch Simulation object
 - copying values
 - using Attribute feed
 - using Copy Attributes block
 - using Copy Attributes feed
 - using Copy Attributes probe
 - using Task block
- Auto Refresh Clients attribute
- Average in Process attribute
 - computing for blocks
 - of blocks
- Average probe

- charting averages of quantitative parameters, using
- computing an average, using
- determining value of
- plotting minimum and maximum values of
- reference
- Average Utilization attribute
 - computing for individual resources
 - when allocating multiple resources
 - when allocating partial resources
 - computing for resource pools
 - when allocating multiple resources
 - when allocating partial resources
- of resources
- of work objects
- showing
 - for individual resources
 - for resource pools
 - for work objects
- Average Value attribute
 - current value of Average probe
 - of Average probe
 - of Statistics probe

B

- Back menu choice
 - Go menu
- Background Color attribute
 - of charts
- background images, loading
- Basic Activities palette
 - creating blocks, using
 - reference
 - ReThink toolbox
- Batch block
 - batching
 - at time intervals
 - based on trigger object
 - by summing attribute of work objects
 - in a group
 - into containers, using
 - choosing the batch mode
 - configuring
 - path identity of
 - specific features of
 - reference
- Batch Mode attribute
 - Interval

- Number
 - of Batch block
- Sum
- Trigger
- Batch Procedure Name customization attribute
- Batch Simulation History attribute
- Batch Simulation objects
 - creating
 - keeping histories across multiple simulations, using
 - keywords for
 - report parameters
 - simulations
 - using
- Beep Enabled attribute
- Best Practice URL, of models
- Beta attribute
 - random beta distribution
 - random gamma distribution
- beta distribution
- Block Input Report
- Block Label attribute
- Block Summary Report
- blocks
 - See also* activities and individual block listings
 - activities
 - configuring custom duration
 - configuring duration from a file
 - configuring duration from a report
 - configuring duration from a report lookup
 - configuring duration from an attribute
 - configuring fixed duration of
 - configuring random duration of
 - customizing duration of
 - determining current
 - showing snapshot of current
 - attributes
 - Animation tab
 - common
 - Cost tab
 - Duration tab
 - General tab
 - class names of
 - configuring
 - animation attributes
 - cost attributes
 - duration attributes
 - general
 - general attributes

- hierarchical views
- modes
- path identity
- path types, general
- path types, specific blocks
- specific
- specific attributes
- type of work to process
- connecting
 - by inserting between connected blocks
 - introduction to
 - redisplaying paths when
 - using loops
 - using stubs
- constraining
 - using Maximum Activities
 - using resources
- costs
 - computing, based on resource costs
 - fixed
 - total cost
 - using
 - variable
- creating
- custom
- customizing
- debugging
- disconnecting
- duration
 - computing duration for multiple work units
 - file
 - fixed
 - random
 - report
 - report lookup
 - updating metrics
 - using
 - work object attribute
- errors
- menu choices of
- online
- paths
 - connecting
 - creating loops
 - deleting
- probing performance of
- replacing
- stubs
- verifying metrics of
- Blocks Waiting attribute
- borderless free text
- borders, adjusting workspace
- bpr-acknowledge-message-probe class
- bpr-activity class
- bpr-associate class
- bpr-average-probe class
- bpr-batch class
- bpr-block class
 - feeding values into
 - filtering report data, using
 - probing
 - updating attributes of
- bpr-branch class
- bpr-container class
 - batching objects into
 - configuring as path type
 - configuring path type, using
 - creating subclasses of
 - filtering report data, using
 - inserting objects into
 - removing objects from
- bpr-copy class
- bpr-copy-attribute-probe class
- bpr-copy-attributes class
- bpr-criteria-probe class
- bpr-delete-message-probe class
- bpr-delta-time-probe class
- bpr-insert class
- bpr-instrument class
 - filtering report data, using
 - probing
- bpr-interval-sample-probe class
- bpr-merge class
- bpr-message-probe class
- bpr-module-settings
- bpr-moving-average-probe class
- bpr-n-dim-sample-probe class
- bpr-object class
 - configuring as path type
 - configuring path type, using
 - creating subclasses of
 - filtering report data, using
 - probing
 - vs. bpr-resource
- bpr-parameter-probe class
- bpr-path class
 - filtering report data, using
 - probing
 - updating attributes of
- bpr-pool class
- bpr-probe class

- bpr-reconcile class
- bpr-remove class
- bpr-resource class
 - feeding values into
 - filtering report data, using
 - populating pools dynamically, using
 - probing
 - configuring the probe
 - using Moving Average probe
 - using Sample probe
 - updating attributes of, using Change feed
 - vs. bpr-object
- bpr-retrieve class
- bpr-sample-probe class
- bpr-sink class
- bpr-source class
- bpr-statistic-probe class
- bpr-store class
- bpr-surrogate class
- bpr-task class
- bpr-update-trigger-probe class
- bpr-yield class
- Branch Attribute attribute
- Branch block
 - branching based on
 - a range of values
 - attribute value
 - dynamic proportion
 - interactively selecting output path
 - output path type
 - proportion
 - rules
 - choosing the branch mode
 - configuring
 - general
 - specific features of
 - path attributes of
 - reference
 - testing every outcome, using
- Branch Dynamic Proportions attribute
- Branch Lower attribute
- Branch Mode attribute
 - Attribute Value
 - Dynamic Proportion
 - of Branch block
 - Prompt
 - Proportion
 - Type
- Branch Procedure Name customization
 - attribute
- Branch Prompt Message attribute

- Branch Prompt Timeout attribute
- Branch Proportion attribute
 - configuring
 - for Branch block
 - for Yield block
 - of paths
- Branch tab, of path dialog
- Branch Upper attribute
- Branch Value attribute
- breakpoints
 - setting and clearing
 - verifying model metrics, using
- Bridge Host attribute
- Bridge Port attribute
- Bridge, ODBC
- Bring to Front menu choice
 - controlling layout, using
 - Layout menu
- BRMS Task block
- Business Objects menu choice
- Business Processes menu choice
 - Business Processes menu
 - System Models menu
- Business Rules menu choice
 - Project menu

C

- Category attribute
 - Acknowledge Message probe
 - Delete Message probe
- Change feed
 - changing attributes
 - of blocks, using
 - of resources, using
 - generating
 - random numbers, based on a
 - distribution, using
 - random numbers, using
 - unique IDs, using
 - reference
 - updating system-defined attributes, using
- Change Mode attribute
 - Distribution
 - of Change feed
 - Random
 - Unique ID
 - Value
- Change Procedure Name customization
 - attribute

- charts
 - charting performance statistics, using
 - configuring
 - axes of
 - colors of
 - maximum points of
 - minimum and maximum values of
 - creating from probes
 - offsetting values of
 - plotting multiple values on
 - remote
 - creating from reports
 - creating from Tools palette
 - updating
 - automatically
 - manually
 - using a rule
 - using an action button
- Charts menu choice
- Check Script button
- Choose Connector menu choice
 - associating connectors
 - on model details, using
 - on task details, using
- Choose Container Input Path menu choice
- Choose Detail menu choice
- Choose Empty Container Output Path menu choice
- Choose Manager attribute
 - Priority
 - Random
- Choose Nonempty Container Output Path menu choice
- Choose Not Found Output Path menu choice
- Choose Original Input Path menu choice
- Choose Original Output Path menu choice
- Choose Parameter menu choice
 - of Parameter feed
 - of Parameter probe
- Choose Pool menu choice
 - of Retrieve block
 - of Store block
- Choose Reject Path menu choice
- Choose Resource attribute
 - Highest and Lowest Priority
 - Lowest Cost
 - Lowest Utilization
- Choose Resource menu choice
- Choose Root Workspace menu choice
- Choose Trigger Input Path menu choice
- Choose Trigger Output Path menu choice
- Choose Update Trigger menu choice
 - of Interval Sample probe
 - of reports
 - Update Trigger probe
 - Update Trigger tool
- Class Definition
 - creating subclasses
 - automatically
 - manually
 - of query objects
 - of resources
 - of work objects
- Class Name attribute, work object class definitions
- Class Specific Attributes attribute
 - of query objects
 - of work object class definitions
- Clear Break menu choice
 - of blocks
- Clear Breaks menu choice
 - clearing breakpoints, using Simulation menu
- Clear Indicators menu choice
 - clearing indicators, using Simulation menu
 - using
- client
 - connecting
 - directly to server
 - from Start menu
 - to a specific server
 - disconnecting
- Clone menu choice
 - copying objects, using
- Edit menu
 - of blocks
 - of instruments
- Close menu choice
 - exiting client, using File menu
- collecting n-dimension samples
- colors
 - configuring
 - for blocks
 - for instruments
 - for paths
 - for resources
 - for workspaces
 - editing for objects
- Colors menu choice
- Edit menu

- Comments attribute
 - of blocks
 - of feeds
 - of probes
- comments, adding to scripts
- comparing sampled values against a criteria
- computation behavior of scenarios
- Compute All Blocks attribute
- computer class
- configuring
 - Batch Simulation object
 - blocks
 - animation of
 - concurrent activities of
 - costs of
 - duration of
 - general
 - general attributes of
 - hierarchical views of
 - modes of
 - path types of
 - specific
 - specific attributes of
 - charts
 - database access
 - database interface objects
 - feeds
 - interface pools
 - model environment
 - online blocks
 - paths
 - animation of
 - branch attributes of
 - identity of
 - output types of specific blocks
 - types
 - probes
 - reports
 - attributes to appear in
 - filter criteria of
 - history for
 - scope of
 - time unit of
 - update interval for
 - resource managers
 - allocation and deallocation of
 - association name of
 - criteria for choosing manager
 - criteria for choosing resources
 - utilization of
 - resources
 - animation of
 - constraints for
 - costs of
 - efficiency factor of
 - priority of
 - utilization of
- scenarios
 - animation of
 - constraints for
 - costs of
 - efficiency factor of
 - priority of
 - utilization of
- scenarios
- Connect menu choice
- Database Interface object
- Excel toolbar
- Connect String attribute
- connection posts
 - See* connectors
- connections
 - See* paths and connectors
- connectors
 - associating
 - on model details
 - on Task block details
 - on Task block details
 - showing connected paths of
- Constraints
 - ReThink toolbox
 - Temporal Scheduler
- constraints
 - configuring
 - date availability of
 - hourly availability of
 - monthly availability of
 - temporal scheduler detail of
 - weekly availability of
 - constraining resources
 - using
 - using normal business hours
 - introduction to
 - temporal scheduler
 - default
 - displaying detail of
- Constraints palette
 - constraining resources, using
 - ReThink toolbar
- Container List Attribute attribute
 - of Batch block
 - of Insert block
 - of Remove block
- container-list attribute
 - batching objects into
 - inserting objects into
 - removing objects from
- containers
 - batching objects into

- configuring path types, using
 - creating subclasses of
 - inserting single objects into
 - showing work objects in
 - for Batch block
 - for Insert block
 - for Remove block
- Continue menu choice
 - continuing the simulation, using
 - Simulation menu
- Copy All Attributes attribute
- Copy Attributes attribute
- Copy Attributes block
 - configuring path identity of
 - reference
- Copy Attributes feed
 - copying metrics, using
 - reference
- Copy Attributes probe
 - reference
 - rolling up metrics from details, using
- Copy block
 - adding copies to associations, using
 - configuring
 - number of objects to create, using
 - path identity of
 - specific features of
 - reference
- Copy Item List Items customization attribute
 - of Copy block
 - of Retrieve block
- Copy Item Lists customization attribute
 - of Copy block
 - of Retrieve block
- copying
 - attributes
 - using Attribute feed
 - using Copy Attributes block
 - using Copy Attributes feed
 - using Copy Attributes probe
 - models
 - work objects
- Cost attribute, of activities
- Cost Per Time Unit attribute
 - configuring variable costs
 - for blocks, using
 - for resources, using
 - of blocks
- Cost Per Use attribute
 - configuring fixed costs
 - for blocks, using
 - for resources, using
 - of blocks
- costs
 - of activities
 - of blocks
 - reference
 - using
 - of resources
 - of work objects
- Counter attribute
 - of feeds
 - of probes
- Create Chart menu choice
 - creating charts from probes, using
 - of probes
- Create Connection menu choice
 - creating stubs for instruments, using
 - instruments
- Create Detail menu choice
 - creating detail for tasks, using
 - creating detail, using
 - creating resource pools, using
 - of Task block
 - Organizer tool
- Create Input Graph menu choice
- Create Input menu choice
 - creating new stubs, using
 - of blocks
- Create Manager menu choice
- Create Output menu choice
 - creating new stubs, using
 - of blocks
- Create Parameter menu choice
 - of Parameter feed
 - of Parameter probe
- Create Remote menu choice
 - creating remotes from probes, using
 - of probes
 - remote charts
- Create Rules menu choice
- Create Slider menu choice
 - creating sliders from feeds, using
 - of Change feed
- Create Surrogate menu choice
- Create Type In menu choice
 - creating type-in boxes from feeds, using
 - of Change feed
- creating
 - activities
 - annotations
 - Batch Simulation object

- blocks
- charts
- class definitions
 - for query objects
 - for work objects
 - of resources
- connectors
- database interface objects
- details for Task blocks
- displays
- feeds
- instruments
- interface pools
- JMail interface objects
- models
- online blocks
- organizers
- pools
 - for resources
 - generic
- probes
- projects
- remotes
 - from charts
 - from probes
- reports
 - general
 - in databases
- resource managers
- resource pools
- resources
- scenarios
- sliders
- stubs
- surrogates
- temporal constraints
- type-in boxes
- update trigger
 - probes
 - tools
- user interface objects from feeds
- work objects
 - class definitions for
 - during processing
- Creation Time attribute
 - computing for blocks
 - of blocks
 - of resources
 - of work objects
- Criteria probe
 - comparing sampled values against criteria, using
 - determining value of
 - reference
- Criteria True Count attribute
 - current value of Criteria probe
 - of Criteria probe
- Criteria True Percent attribute
 - current value of Criteria probe
 - of Criteria probe
- .csv files, Excel
- Cumulative Sample attribute
 - of Interval Sample probe
 - of Sample Value probe
- Current Activities attribute
 - determining current block activities, using
 - of blocks
 - of resources
 - of work objects
- Current Utilization attribute
 - computing for individual resources
 - when allocating multiple resources
 - when allocating partial resources
 - computing for resource pools
 - when allocating multiple resources
 - when allocating partial resources
 - displaying
 - for individual resources
 - for resource pools
 - for work objects
 - of resources
 - of work objects
- Current Waiting attribute
 - analyzing wait time of paths, using
 - of paths
- custom blocks
- customer support services
- customizing
 - blocks
 - description of
 - how to
 - instruments
 - Resource Managers
 - resources
 - scenarios
 - surrogates
 - work objects
- cycle time
 - computing
 - for work objects
 - partial

using Delta Time probe

D

- database access
 - configuring
 - connect string
 - database interface objects
 - host and port
 - ODBC data source
 - ReThink for
 - connecting to the database
 - creating
 - record work objects for
 - SQL queries for
 - creating the database
 - generating work objects, using
 - introduction to
 - online mode
 - retrieving objects, using
 - starting ODBC Bridge
 - storing work objects, using
 - updating database records, using
 - using reports
- Database Commit block
- Database Input Object Name customization attribute
- Database Interface Name attribute
 - configuring for reports
 - creating work objects from a database, using
 - of Retrieve block
 - of Source block
 - of Store block
 - retrieving work objects from a database, using
 - storing work objects to a database, using
- database interface objects
- Database Key attribute
 - of Store block
 - updating work objects in a database, using
- Database Query block
- Database Quote In Text String customization attribute
- Database Quote String customization attribute
- Database Reporting Enabled parameter
- Database Rollback block
- Database SQL DML block
- Database Stored Procedure block
- Database tab
 - reports
 - Retrieve block
 - Source block
 - Store block
- Database Table attribute
 - of Store block
 - storing work objects to a database, using
- Database Table Name attribute
- Database Update Object block
- Date and Time as Durations attribute
- Date Constraint
- Date-Time Time as Duration attribute
- Days attributes
- DB Function Query block
- db-*qo-record* class
- deactivating scenarios
- Deallocate Resource attribute
- debugging
 - blocks
 - branching work onto explicit path
 - verifying model metrics
 - viewing errors
- Default User Mode attribute
- Default Web Location attribute
- definitions
 - See* class definitions
- Delay block
- Delete All Messages attribute
- Delete Background Image menu choice
 - deleting background images, using
 - Workspace menu
- Delete CSV Report File menu choice
- Delete menu choice
 - deleting objects, using
 - deleting workspaces, using
- Edit menu
 - of blocks
 - of instruments
 - of paths
- Delete Message probe
- deleting
 - objects
 - stubs
 - workspaces
- deleting messages
- Delta Time attribute
 - current value of Delta Time probe
 - of Delta Time probe
- Delta Time probe
 - computing
 - cycle time, using

- partial cycle time, using
 - determining value of
 - reference
- demo models, viewing
- Destination Attribute Name attribute
 - configuring for feeds
 - updating system-defined attributes, using
 - updating user-defined attributes, using
- of Attribute feed
- of feeds
- Destination Class Name attribute
 - of Attribute feed
 - of Copy Attributes probe
- Detail Color attribute
- details
 - associating connectors on
 - for models
 - for Task blocks
 - computing metrics for Task blocks with creating
 - creating hierarchical views, using
 - definition of
 - deleting
 - for Model and Organizer tools
 - for resource pools
 - for Task blocks
 - disabling
 - displaying for objects
 - enabling
 - replacing
 - for models
 - for organizers
 - rolling up metrics from
 - showing
 - connected paths on
 - for container objects
 - for pools
 - for Task blocks
 - superior object of
 - Task block
- Developer mode
 - configuring user preferences for
 - description of
- dialogs
 - See* menu choice listings for launching specific dialogs
- Direct Superior Classes attribute
 - for query objects
 - for resource subclasses
 - for work object subclasses
- Disable Detail menu choice
 - disabling Task block details, using
 - of Task block
- Disconnect menu choice
 - Excel toolbar
 - of blocks
- disconnecting
 - from the client
 - using menu
- Displays
 - Annotation tool
 - Readout Table
 - ReThink toolbox
- Displays palette
- displays, attribute
- distributed workflow applications
- distributing
 - objects
- Distribution Mode attribute
 - Arrival Rate Input Graph
 - Custom
 - Duration File
 - Fixed Distribution
 - description of
 - specifying for blocks
 - of blocks
 - of Change feed
 - Random Beta
 - Random Erlang
 - Random Exponential
 - Random Gamma
 - Random Lognormal
 - Random Normal
 - Random Triangular
 - Random Uniform
 - Random Weibull
 - Report Indexed Lookup
 - Report Lookup
 - Work Object Duration
- Documentation menu choice
- Down menu choice
- duration
 - of activities
 - of blocks
 - of resources
 - of work objects
- Duration attribute
- Duration File Name attribute
- Duration tab
 - activities

blocks

E

- Edit menu
- Efficiency Factory attribute
- Elapsed Time attribute
 - computing Total Work Time, using
 - of activities
- email
 - configuring
 - address
 - format
 - to send
 - delivering messages by
 - examples of sending
 - sending
 - starting JMail Bridge
 - startup parameters for sending
- Empty Breakpoint customization attribute
- Empty Color attribute
- Enable Animation attribute
- Enable Charting attribute
- Enable Detail menu choice
 - enabling Task block details, using
 - of Task block
- Enable Macros button
- Enable Metrics Toolbar Update attribute
- Enable Status Bar Message Browser attribute
- Enable Tracking attribute
- End Time attribute
 - of Batch block
 - of Source block
 - of Update Trigger tool
- Erlang distribution
- Error attribute
 - of blocks
 - of feeds
 - of paths
 - of probes
 - viewing block errors, using
- Error Color attribute
 - of blocks
 - of instruments
 - of paths
 - of resources
- errors
 - debugging blocks
 - handling in online mode
 - viewing

- Event and Alarm Metrics menu choice
- Events queue
- Excel
 - connecting to the server from
 - manually
 - controlling simulation from
 - creating reports in
 - enabling macros in
- Excel CSV File Reporting Enabled attribute
- Excel *.csv* files
- Excel File Name attribute
- Excel Report Enabled attribute
 - importing data, using
 - writing data, using
- Exit menu choice
 - exiting the server, using
- exponential distribution
- Export Excel tool
- Export Tools button, ReThink toolbox
- Export Tools palette
- Extended Menus attribute

F

- F4 key
- feeds
 - Animation tab
 - configuring
 - attribute to update
 - class to which feed applies
 - introduction to
 - creating
 - sliders from
 - type-in boxes from
 - user interface objects from
 - feeding values into models, using
- General tab
- introduction to
- menu choices of
- updating
 - system-defined attributes of the
 - model, using
 - user-defined attributes of the model,
 - using
- FIFO queuing algorithm
- File menu
- files
 - .csv*
 - g2.ok*
 - generating work objects from
 - .kb*

rethink-40-online-examples.kb
rethink-online.kb
ReThink-Summary-Reports.xls
StartServer.bat

storing

arrival times in

work objects in

twng.exe

FIFO queuing algorithm

Filter menu choice, Excel

filtering report data

general

in Excel

Fixed Distribution

Flip Horizontally menu choice

controlling layout, using

Layout menu

Flip Vertically menu choice

controlling layout, using

Layout menu

Foreground Color attribute

formatting Excel reports

Forward menu choice

Go menu

free text

G

G2 Help Topics menu choice

G2 JMail Bridge menu choice, Start menu

G2 JMSLink

G2 toolbox

g2.ok file

gamma distribution

General tab

activities

blocks

feeds

paths

probes

generating messages

Get menu choice

Workspace menu

GIF files, loading as background images

Go menu

Go To menu choice

manage dialog

project hierarchy

Search dialog

Go to Superior menu choice

View menu

H

Help menu

Hide menu choice

View menu

hierarchical views, modeling the details of a task, using

histories

charting

keeping for reports

Home menu choice

Go menu

Home Process Map attribute

Hourly Constraint

HTTP menu choice

I

IDs, unique

Import Data from Database menu choice

Import Data from File menu choice

Inactive Color attribute

of blocks

of instruments

of resources

Include All Details attribute

Include Tasks with Detail attribute

Increment attribute

Increment feed

incrementing counters, using reference

Indexed Lookup Report

configuring duration, using description of

Indicate Busiest Path menu choice

Indicate Items attribute

configuring

Indicate Longest Wait Time Path menu choice

Indicate Message attribute

Indicate Mode attribute

Indicate Most Used Path menu choice

Initial Count attribute

Initial ID attribute

Initial Value attribute

Average probe

Moving Average probe

Statistic probe

Initialize Application menu choice

deleting messages, using

Project menu

Input directory

- input reports
 - applying data to the model
 - from databases
 - from Excel
 - general
- input stubs
- Insert block
 - choosing insert mode
 - configuring
 - path identity of
 - specific features of
 - inserting
 - objects into a container by looping
 - single objects into containers, using
 - reference
 - understanding the paths of
- instruments
 - See also* feeds and probes
 - attributes, common
 - class names of
 - configuring
 - animation attributes
 - feeds
 - probes
 - connecting to objects
 - creating
 - creating instruments, using ReThink
 - toolbox
 - customizing
 - feeding values into models, using
 - menu choices of
 - probing performance of models, using
 - replacing
- Instruments palette
 - reference
 - ReThink toolbar
- Interface Name attribute
 - Database Interface object
- Interface Pools menu choice
 - Project menu
- Interfaces menu choice
 - configuring network interfaces, using
 - Project menu
 - SMTP
 - SQL
- Interval Sample probe
 - determining value of
 - reference
 - sampling models, using
- inventory fluctuations

J

- Java Mail (JMail)
 - configuring
 - in configuration file
 - in user preferences
 - sending email, using online blocks
- Java Message Service (JMS)
- JMS menu choice
- JMS Sink
- JMS Source
- JMS Task
- JPEG files
 - loading as background images
 - saving workspaces to
- Jump Mode menu choice
 - description of
 - running simulations, using
- Simulation menu

K

- .kb* files
 - See also* files
 - description of
 - opening
 - saving
- Keep History attribute
 - N Dimensional Sample probe
 - of reports
- knowledge bases (KB)
 - See .kb* files

L

- Label attribute
 - hiding
 - of feeds
 - of probes
- layering
- Layout menu
- Layout toolbar
 - View menu
- Left menu choice
- LIFO queuing algorithm
- LIFO queuing algorithm
- Line Color attribute
- Load Background Image menu choice
 - loading background images, using
 - Workspace menu
- loading projects

- Log Message attribute
- lognormal distribution
- Lookup Attribute Name attribute
- Lookup Label Attribute Name attribute
- Lookup Subtable customization attribute
- loops, creating in diagrams

M

- machine class
- Make Temporal Connector menu choice
- Manage dialog
 - displaying object properties and details using
- Manage menu choice
- Manager Priority attribute
- managing
 - objects
 - using Manage dialog
 - using Project menu
- Match Procedure Name customization attribute
- Max attribute
 - random beta distribution
 - random triangular distribution
 - random uniform distribution
- Maximum Activities attribute
 - limiting concurrent activities, using of blocks
- Maximum Random Value attribute
- Maximum Starts attribute
 - debugging blocks, using of Source block
 - of Update Trigger tool
- Maximum Utilization attribute
 - computing for individual resources
 - when allocating multiple resources
 - when allocating partial resources
 - computing for resource pools
 - when allocating multiple resources
 - when allocating partial resources
 - configuring resource availability, using displaying
 - for individual resources
 - for resource pools
 - of resources
- Maximum Value attribute
 - configuring slider values, using
 - generating random numbers, using
 - maximum value of Average probe

- of Average probe
- of Change feed
- of Statistics probe
- Maximum Values attribute, of remotes
- Mean attribute
 - configuring fixed distribution, using fixed distribution
 - random erlang distribution
 - random exponential distribution
 - random lognormal distribution
 - random normal distribution
- Mean Wait Time attribute
 - analyzing wait time of paths, using of paths
- menus
 - Edit
 - File
 - Go
 - Help
 - Layout
 - Model
 - Project
 - ReThink
 - Simulation
 - Tools
 - Workspace
- Merge block
 - configuring
 - merging work
 - multiple streams of using a loop
 - processing multiple streams of work asynchronously, using reference
- Message attribute
- Message Board menu choice
 - View menu
 - viewing messages, using
- Message Browser menu choice
 - View menu
- message browsers
 - configuring
 - for modeler mode
 - for operator mode
 - showing by default in operator mode
- Message Browsers menu choice
- Message Log File attribute
- Message probe
 - generating text messages, using reference
- Message Type attribute

- Acknowledge Message probe
- Delete Message probe
- messages
 - acknowledging
 - deleting
 - delivering by email
 - generating
 - viewing
- Messages queue
- metrics
 - disabling updating of toolbar
 - updating
 - in properties dialogs
 - in reports
- Microsoft
 - Access
 - Excel
- Min attribute
 - random beta distribution
 - random triangular distribution
 - random uniform distribution
- Minimum Random Value attribute
- Minimum Threshold attribute
- Minimum Value attribute
 - configuring slider values, using
 - generating random numbers, using
 - minimum value of Average probe
 - of Average probe
 - of Change feed
 - of Statistics probe
- Mobile Email
 - address
 - Notification
- Mode attribute
 - All
 - Insert block
 - Remove block
 - Associate block
 - Add
 - New
 - First and Last
 - Insert block
 - Remove block
 - of Associate block
 - of Insert block
 - of Remove block
 - random triangular distribution
- Mode Random Value attribute
- Model button, Manage dialog
- model environment, configuring
- Model menu
- Model tools
 - See* models
- Model Version column
- Modeler Browser attribute
- Modeler mode
 - configuring
 - user preferences for
 - description of
- models
 - See Also* simulations
 - annotating
 - associating connectors on detail of
 - comparing different versions of
 - controlling
 - using different scenarios
 - using single scenario
 - copying
 - creating
 - definition of
 - demo
 - feeding values into
 - hierarchical
 - large, working with
 - organizing
 - introduction to
 - organizing, using Organizer tools
 - performing what-if analysis on
 - probing performance of
 - replacing default detail of
 - working with
- modes
 - configuring
 - distribution
 - for blocks
- Monthly Constraint
- Moving Average attribute
 - current value of Moving Average probe
 - of Moving Average probe
 - of Statistics probe
- Moving Average probe
 - computing moving averages
 - directly, using
 - of probed values, using
 - of resources directly, using
 - determining current value of
 - reference
- Moving Standard Deviation attribute
 - moving standard deviation of Moving
 - Average probe
 - of Moving Average probe
 - of Statistics probe

My User Preferences menu choice
configuring user preferences, using
Project menu

N

navigating applications
Navigator
 menu choice
Navigator menu choice
 View menu
N-Dimensional Input Report
 creating
 description of
N-Dimensional Output Report
 creating
 description of
N-Dimensional Sample probe
 collecting samples, using
 reference
New Instance menu choice, project hierarchy
New menu choice
 creating
 projects, using
 top-level workspaces, using
 File menu
 Workspace menu
New Point menu choice
New Value attribute
 of Change feed
 updating system-defined attributes, using
normal distribution
Normal menu choice
Not Available Time attribute
 computing for resources, using constraints
 of resources
Nudge menu choice
 controlling layout of objects, using
 Layout menu
 of blocks
 of instruments
Number of Samples attribute

O

Object File Name attribute
 of Source block
 of Store block
Object Input Report
Object Models menu choice

Object Summary Report
object tracking
objects
 adjusting the order of
 aligning
 copying
 deleting
 displaying properties for
 distributing
 editing colors
 flipping
 interacting with
 in Modeler mode
 managing
 nudging
 resizing
 rotating
 selecting
 all
 individual
 transferring
ODBC Bridge
Start menu
 using for database access
Online Activities palette
 ReThink toolbar
 using
online mode
 configuring delays
 how it works
 interacting with databases
 introduction to
 IO Interface pools
 modeling distributed workflow
 applications
 online blocks
 using
Online Mode menu choice
 description of
 Simulation menu
 using
Open menu choice
 File menu
 opening projects, using
Operation attribute
 of Accumulate feed
 of Attribute feed
 of Branch block
 of Criteria probe
 of Increment feed
 of Retrieve block

Operator Browser attribute
Operator mode
 configuring user preferences for
 description of
 user mode

Oracle

Order menu choice
 Layout menu
 of blocks
 of instruments

organizers
 creating
 definition of
 placing resources in
 replacing default detail of

otherwise symbol

Output Count attribute
 of Copy block
 of Source block
 of Task block

Output directory

output reports
 generating report data
 in databases
 in Excel
 in the client

output stubs

P

Palette Workspace

Parameter feed
 getting parameter values, using
 getting values from different types of
 parameters, using
 reference

Parameter Name attribute
 of Parameter feed
 of Parameter probe

Parameter probe
 feeding values into parameters, using
 reference
 setting parameter values, using

parameters
 See Also attributes
 getting values of
 setting values of

Path Input Report

path queues

Path Summary Report

path types
 configuring
 for blocks
 introduction to
 using containers
 using default
 using user-defined objects

paths

See also stubs

attributes
 Animation tab
 common
 for branching
 General tab

class name of

configuring
 animation attributes
 for particular blocks
 identify of
 output types of specific blocks
 type
 using containers

determining, based on type

disabling redrawing for

duration of

green

 analyzing wait time of
 due to path synchronization
 due to resource constraints
 when limiting concurrent activities

menu choices of

redisplaying for connected blocks

type

 configuring
 default

work backups on

 due to resource constraints
 showing using Snapshot Queue

Pause menu choice

 Batch Simulation object
 pausing the simulation, using
 Simulation menu

performance, enhancing

 by configuring
 animation
 computation behavior
 simulation speed

Period attribute

 of Batch block

 of Update Trigger tool

Period Initial Value attribute

- Period Start Time attribute
- person class
- Phase attribute
 - configuring
 - for feeds
 - for probes
 - of activities
 - of Attribute feed
 - of feeds
 - of probes
- pools
 - See also* resources
 - allocating particular resources from
 - creating
 - for resources
 - generic
 - introduction to
 - deleting details
 - determining when to use
 - populating dynamically
 - showing details
 - vs. other resources
- popup menus
 - interacting with objects, using
- popup menus, displaying
- Precision attribute
 - of Attribute feed
 - of Average probe
 - of Delta Time probe
 - of Increment feed
 - of Interval Sample probe
 - of Moving Average probe
 - of Sample Value probe
 - of Statistics probe
- Print menu choice
 - File menu
 - printing workspaces, using
- priority
 - allocating the same resource to different
 - blocks, based on
 - choosing resources, based on
- Probe Input Report
- Probe Summary Report
- probes
 - Animation tab
 - charting performance statistics, using
 - configuring
 - attribute to probe
 - class to which probe applies
 - introduction to
 - examples of probing

- average utilization of current resource
- average utilization of resource in pool
- average utilization of top-level
 - resource
- General tab
- introduction to
- menu choices of
 - common
 - probes
- probing
 - blocks
 - models
 - resources
 - work objects
- showing current value of
- Procedure Name attribute
- Project
 - menu
 - managing objects, using
 - using
 - using submenus
- Project menu
- projects
 - creating
 - opening
 - saving
 - working with
- properties dialogs
 - shortcuts for displaying
- properties dialogs, displaying
- Properties menu choice
 - blocks
 - Edit menu
 - for items on workspaces
 - instruments
 - paths
 - popup menu for objects

Q

- quantitative values, feeding
- quantitative-parameter class, charting average
 - value
- query objects
- Queues menu choice

R

- Random Beta distribution
- Random Erlang distribution

- Random Exponential distribution
- Random Gamma distribution
- Random Lognormal distribution
- Random Normal distribution
- random numbers
 - generating
 - based on a distribution
 - using Change feed
- Random Triangular distribution
- Random Uniform distribution
- Random Weibull distribution
- Range Lower attribute
- Range Upper attribute
- readout tables
- Reconcile All attribute
- Reconcile block
 - configuring specific features of reconciling
 - all associated objects, using
 - individual associated objects, using
 - reference
- records, database
- Redraw Path attribute
 - disabling path redrawing, using
 - of paths
- Refresh menu choice
 - Go menu
- remote charts
 - creating
 - from reports
 - from Tools palette
- Remote Process Sink block
- Remote Process Source block
- Remote Process Task block
- remotes
 - creating
 - from charts
 - from feeds
 - from probes
 - interpreting value when configuring charts
 - offsetting values of
 - plotting multiple values on same chart, using
 - scaling values of
 - statistics of
- Remove block
 - configuring
 - general
 - path identity of
 - remove mode
 - specific features of
 - reference
 - removing objects
 - all at once
 - by looping
 - understanding the paths of
- Remove Update Trigger menu choice
- Repeat Database attribute
 - determining when to stop creating work objects, using
 - of Source block
- Repeat Duration File attribute
- Repeat Object File attribute
- Report Title attribute
- reports
 - accessing databases, using
 - Batch Simulation object script keywords for
 - configuring
 - attributes to appear in
 - filter criteria of
 - for database access
 - history for
 - scope
 - time units of
 - update interval for
 - creating
 - in databases
 - in Excel
 - in the client
 - specialized
 - templates
 - filtering
 - data
 - in Excel
 - generating output data
 - in .*csv* files
 - in databases
 - in Excel
 - in the client
 - importing input data
 - from .*csv* files
 - from databases
 - from the client
 - introduction to
 - keeping a history of data values in
 - refreshing data
 - summary of input and output
 - updating
 - based on model events
 - Excel

- introduction to
 - manually
 - multiple
 - regularly
 - triggering
- Reports menu choice
- Reports palette
 - creating reports, using ReThink toolbar
- Reset menu choice
 - resetting the simulation, using Simulation menu
- resizing objects
- Resource Input Report
- Resource Managers
 - See also* resources
 - allocating resources
 - for multiple sequential steps, using to tasks, using
 - associating with different resources
 - configuring utilization of
 - creating
 - multiple identical, for different tasks
 - multiple, for a single task
 - customizing
 - identifying associated resource of
- resource pools
 - See* pools
- Resource Priority attribute
- Resource Summary Report
- Resource Utilization chart
- resources
 - See also* pools and Resource Managers
 - allocating
 - associated
 - based on priority
 - for multiple sequential steps
 - from a pool
 - lowest cost resource
 - multiple resources from a pool
 - multiple resources to the same task
 - partial resources from a pool
 - particular resources from a pool
 - resource with lowest utilization
 - same resources to multiple tasks
 - surrogate resources to different tasks
 - the same resource to different blocks
 - based on priority
 - to tasks
 - with constraints
 - availability of
 - configuring, using constraints
 - configuring, using maximum utilization
 - class names of
 - comparing with work objects
 - configuring
 - animation of
 - costs of
 - priority of
 - constraining
 - availability of, using temporal constraints
 - model, using
 - using normal business hours
 - costs of
 - activities
 - assigning fixed and variable
 - introduction to
 - total cost
 - creating
 - general
 - generic pools
 - pools
 - creating pools
 - for any resource
 - generic
 - creating resources
 - customizing
 - deallocating explicitly
 - disabling
 - displaying attributes with
 - duration of
 - efficiency factors for
 - examples of
 - allocating multiple resources from a pool
 - allocating partial resources from a pool
 - showing attributes
 - feeding attribute values into
 - metrics
 - General tab
 - Utilization tab
 - pools
 - creating
 - determining when to use
 - populating dynamically
 - probing
 - average utilization of current resource, example

- average utilization of resource in pool,
 - example
- average utilization of top-level
 - resource, example
 - general
 - moving average of
 - performance of
 - sample values of
- red, when allocated
- replacing
- showing currently allocated
- surrogates
- timing
- utilization of
 - charting
 - computing
 - individual
 - when allocating multiple resources
 - from a pool
 - when allocating partial resources
- work backups due to
- Resources palette
 - creating pools, using
 - creating resources, using
 - ReThink toolbar
- Restore Last Pane Settings attribute
- Resume menu choice, Batch Simulation object
- ReThink
 - connecting client
 - from Start menu
 - to specific server
 - demo models
 - exiting
 - menus
 - running
 - running in secure G2 environment
 - starting server
 - using Start menu
 - with your application loaded
- ReThink Help Topics menu choice
- ReThink toolbox
 - Basic Activities
 - Constraints
 - Displays
 - Export tools
 - Instruments
 - Online Activities
 - Reports
 - Resources
 - generic pool
 - pools
 - resource
- Tools
 - Arrival Rate Input Graph
 - Batch Simulation Object
 - Class Definition
 - Connector
 - Model
 - Organizer
 - Palette Workspace
 - Scenario
 - Update Trigger tool
- using
 - rethink-40-online-examples.kb* file
 - rethink-online.kb* file
 - ReThink-Summary-Reports.xls* file
- Retrieve All attribute
- Retrieve Attribute attribute
- Retrieve block
 - adding copies to associations
 - choosing the retrieve mode
 - configuring
 - path identity of
 - configuring specific features of
 - determining how the block handles objects
 - not found
 - reference
 - retrieving objects
 - all
 - at random
 - based on a range of values
 - by association
 - copies of
 - from a database
 - from a pool
 - with particular attribute values
- Retrieve Copy attribute
- Retrieve Mode attribute
 - Association
 - Attribute Value
 - of Retrieve block
 - Random
- Right menu choice
- Rotate or Flip menu choice
 - controlling layout, using
- Layout menu
 - of blocks
 - of instruments
- Row ID column
- Rule Sets attribute
 - of BRMS Task block
- rules

- branching work objects based on updating charts, using
- Rules Wait Interval attribute
- running
 - ReThink
 - simulations
 - using Batch Simulation object
 - using scenarios

S

- Sample
- Sample Initial Value attribute
 - Average probe of Moving Average probe
- Sample probe
 - charting quantitative parameters, using
 - determining the value of probing
 - attribute values of the model, using
 - resources directly, using
 - reference
- Sample Value attribute
 - current value of Interval Sample probe
 - current value of Sample probe of Interval Sample probe of Sample probe of Statistics probe
- sampling attribute values
 - at regular time intervals of objects
- Save A menu choice
 - saving projects, using
- Save as JPEG menu choice
 - File menu
 - saving workspaces, using
- Save As menu choice
 - File menu
- Save menu choice
 - File menu
 - saving projects, using
- Scale attribute, random weibull distribution
- Scaling Divisor attribute
- Scaling Offset attribute
- scaling workspaces
- scenarios
 - activating and deactivating
 - Batch Simulation object script keywords configuring
 - computation behavior

- duration
- indicator arrow behavior
- mode
- object tracking
- online mode
- simulation speed
- start time
- version
- creating
- customizing
- definition of
- introduction to
- simulation time of
- Script attribute, for Batch Simulation object
- Search Criteria attribute
- Search menu choice
 - searching for objects, using
 - Tools menu
- Seconds per Tick attribute
- secure G2, running in
- Select All menu choice
 - Edit menu
 - selecting objects, using
- Selected Color attribute
- Send to Back menu choice
 - controlling layout, using
 - Layout menu
- sending email
- server
 - connecting to
 - default
 - from Excel, initially
 - from Excel, manually
 - specific
 - disconnecting from
 - from Excel
 - shutting down
 - using menus
 - starting
 - on specific port
 - using Start menu
 - with your application loaded
- Server Information menu choice
- Set All Available menu choice
 - Date Constraint
 - Hourly Constraint
 - Monthly Constraint
 - Weekly Constraint
- Set All Not Available menu choice
 - Date Constraint
 - Hourly Constraint

- Monthly Constraint
- Weekly Constraint
- Set Break menu choice
 - debugging blocks, using
 - of blocks
- Set Default User Mode attribute
- Shape attribute, random weibul distribution
- Show Associations menu choice
- Show Blocks menu choice
 - showing blocks pointing to pool
 - Retrieve block
 - Store block
- Show Chart menu choice
 - output reports
 - probes
- Show Constraint menu choice
- Show Container Input Path menu choice
- Show Detail menu choice
 - Model tool
 - of Task block
 - Organizer tool
 - showing details
 - for pools
 - for Task blocks
 - summary of common tasks
 - View menu
 - workspaces
- Show Empty Container Output path menu choice
- Show Flow History menu choice
- Show Instruments menu choice
 - Parameter feed
 - Parameter probe
- Show Items to Update menu choice
- Show Logbook attribute
- Show Nonempty Container Output Path menu choice
- Show Not Found Output Path menu choice
- Show Original Input Path menu choice
- Show Original menu choice
- Show Original Output Path menu choice
- Show Parameter menu choice
 - of Parameter feed
 - of Parameter probe
- Show Pool menu choice
 - of Retrieve block
 - of Store block
 - showing chosen pool
 - for Retrieve block
 - for Store block
- Show Reject Path menu choice
- Show Remotes menu choice
- Show Report menu choice
- Show Resource menu choice
- Show Resources menu choice
- Show Rules menu choice
- Show Scenario menu choice
 - of blocks
 - of instruments
- Show Slider menu choice
- Show Source menu choice
- Show Trigger Input Path menu choice
- Show Trigger Output Path menu choice
- Show Type In menu choice
- Show Update Trigger menu choice
- Show URL menu choice
 - blocks
 - models
- Show Users menu choice
- Shrink Wrap menu choice
 - Layout menu
 - shrink wrapping workspaces, using
- Shut Down G2 menu choice
 - shutting down server
 - using menus
- Simulation
 - menu
 - toolbar
 - simulation clock
 - advancing
 - based on real time
 - based on wall clock
 - continuously
 - for each discrete event
- Simulation menu
 - activating and deactivating scenarios
 - clearing indicators
 - configuring simulation mode
 - starting and stopping simulations
- Simulation Speed attribute
- simulation time
 - scaling to real time
 - scenarios
- Simulation toolbar
 - toggling
 - View menu
- Simulation Version attribute
- Simulation Version column
- simulations
 - See Also* scenarios
 - configuring attributes
 - using Attribute Change Event Reports

- controlling
 - from Excel
 - introduction to
- running
 - in jump mode
 - in step mode
 - in synch mode
 - using Batch Simulation object
- starting and stopping
- Single Shot menu choice
 - debugging blocks, using
 - of Source block
- Sink block
 - configuring
 - reference
- sliders, creating from feeds
- SMTP menu choice
- Snapshot Activities menu choice
 - of blocks
 - showing current activities, using
- Snapshot Container menu choice
 - showing work objects in containers, using
 - for Batch block
 - for Insert block
 - for Remove block
 - when using container path types
- Snapshot Queue menu choice
 - example of showing work backups, using
 - due to limiting concurrent activities
 - due to resource constraints
 - of paths
 - showing work backups, using
- Source Attribute Name attribute
 - configuring for probes
 - of Accumulate feed
 - of probes
- Source block
 - configuring
 - duration and objects, using files
 - general
 - maximum number of objects
 - number of objects to generate
 - objects from external file
 - source mode
 - specific features of
 - start and end times
 - generating work objects
 - based on path type
 - by evaluating the block
 - from a database
 - from an external file
 - reference
- Source Class Name attribute
- Source Mode attribute
 - Database
 - Object File
 - of Source block
 - of Store block
 - Type
- Source Procedure Name customization
 - attribute
- SQL menu choice
 - creating database interface objects, using
- SQL queries
- SQL Query
- SQL Query attribute
 - generating objects from a database, using
 - of Retrieve block
 - of Source block
 - retrieving objects from a database, using
 - Store block
- SQL2000
- Standard Deviation attribute
 - random lognormal distribution
 - random normal distribution
- Standard toolbar
 - View menu
- Start All menu choice
 - Simulation menu
 - starting the simulation, using
- Start menu choice
 - Batch Simulation object
 - of Source block
- Start Time attribute
 - of Batch block
 - Source block
- Start Time attribute, Update Trigger tool
- Start Time tab
- starting simulations
 - using Batch Simulation Object
 - using Scenario
- StartServer.bat* file
- State attribute
- Statistics probe
 - computing statistics, using
 - determining the value of
 - reference
- Status Bar menu choice
 - toggling status bar, using
 - View menu
- Step Mode menu choice
 - debugging blocks, using

- description of
- running simulations, using
- Simulation menu
- Stop menu choice
 - Batch Simulation object
 - Go menu
- stopping simulations
 - using Batch Simulation Object
 - using Scenario
- Store block
 - configuring
 - specific features of
 - store mode
 - reference
 - storing arrival times to a file, using
 - storing work objects
 - in a resource pool, using
 - to a database, using
 - to an external file, using
 - updating database records, using
- Store Mode attribute
 - Database
 - File
 - storing arrival times
 - storing objects
 - Pool
- Store Procedure Name customization attribute
- stubs
 - See also* paths
 - connecting to blocks
 - creating for blocks
 - input
 - output
 - deleting
- subclasses
 - See* Class Definition
- subworkspaces
 - See* details
- Sum of Decremental Changes attribute
- Sum of Incremental Changes attribute
- summary reports
 - See* reports
- surrogates
 - configuring animation attributes of
 - customizing
 - sharing resources, using
 - showing associated resource of
- symbols, feeding
- Synch Mode menu choice
 - description of
 - running simulations, using

- Simulation menu
- System Performance menu choice
- System Settings menu
- System-Administrator mode
 - configuring user preferences for
 - description of

T

- Tabbed Mdi Mode attribute
- Task block
 - associating connectors on detail of
 - configuring
 - general
 - specific features of
 - configuring the number of objects to generate, using
 - copying attribute values to output objects, using
 - deleting work objects, using
 - generating work objects, using
 - modeling details of
 - creating hierarchical views by reference
 - processing multiple streams of work objects
 - sequentially, using
 - synchronously, using
 - processing work objects and sending downstream
 - reference
- Telnet Command attribute
- Templates menu choice
- temporal constraints
 - See* constraints
- temporal scheduler
 - displaying detail of
 - introduction to
 - using default
- text values, feeding
- Threshold attribute
 - Batch block
 - batching objects in a group, using
- Time per Unit Attribute attribute
 - computing duration, using
 - of blocks
- Time Period attribute
- Time Unit attribute
 - configuring costs
 - for blocks, using

- for resources, using
 - of Arrival Rate Input Graph
 - of blocks
 - of Delta Time probe
 - of reports
- Time Window attribute
- Timeout attribute
- Timestamp feed
 - computing partial cycle times, using
 - feeding timestamps, using
 - reference
- Time-Weighted Value attribute
- timing
 - See Also* duration
 - resources
- toolbars
 - Layout
 - Simulation
 - Standard
 - using
 - Web
- Toolbars menu
- toolbox
 - G2
 - ReThink
- Toolbox - G2 menu choice
 - using
- Toolbox - ReThink menu choice
 - View menu
- Tools
 - Class Definition
 - Connector
 - menu
 - Organizer
 - ReThink toolbox
 - Scenario
 - Update Trigger tool
- Total Cost attribute
 - computing
 - based on resource costs
 - for blocks
 - for work objects
 - of blocks
- Total Elapsed Time attribute
 - computing
 - Average in Process, using
 - for blocks
 - using constraints
 - of blocks
 - of resources
 - of work objects
- relating to activities
- understanding for blocks
- Total Idle Time attribute
 - computing, using constraints
 - of resources
 - of work objects
- Total Insertions attribute
 - analyzing wait time of paths, using
 - of paths
- Total Starts attribute
 - determining current block activities, using
 - of blocks
 - of resources
 - of work objects
- Total Stops attribute
 - determining current block activities, using
 - of blocks
 - of resources
 - of work objects
- Total Wait Time attribute
 - analyzing wait time of paths, using
 - of paths
- Total Work Time attribute
 - computing
 - Average in Process, using
 - for blocks
 - using constraints
 - of blocks
 - of resources
 - of work objects
 - relating to activities
 - understanding for blocks
- transaction processing
- Transfer menu choice
 - cutting and pasting objects, using
 - Edit menu
 - of blocks
 - of instruments
- transferring
 - blocks
 - instruments
- triangular distribution
- truck class
 - twng.exe* file
- Type attribute
 - configuring
 - using a user-defined object
 - using bpr-container
 - using bpr-object
 - using query object
 - using user-defined objects

- configuring for paths
- of paths

Type of Database attribute

type-in boxes

- creating from feeds
- example of

U

uniform distribution

Uninitialize Application menu choice

- Project menu

unique IDs, generating

Up menu choice

Update All Related Items menu choice

Update button, reports

Update button, updating duration statistics

Update Chart menu choice

Update Chart menu choice, for charts

Update Charts attribute

Update Input Graph menu choice

Update Interval attribute

Update menu choice

- Excel toolbar
- of blocks
- Reports menu
- updating duration statistics, using

Update Mode attribute

Update Report menu choice

Update Time attribute

Update Trigger probe

- exporting probed data, using reference
- updating reports, using

Update Trigger tool

- exporting probed data, using
- triggering updates for multiple reports, using
- using with Interval Sample probe

updating

- charts
- reports

 - at regular time intervals
 - Excel
 - manually

URL

- attribute of blocks

Use Initial Value attribute

- Average probe
- of Moving Average probe

Use Rules Wait Interval attribute

user interface objects, creating from feeds

User Interface Theme attribute

User Mode menu choice

- switching user modes, using

Tools menu

user modes

- configuring default
- specifying user preferences for different
- switching

User Name attribute

- Modeler mode

user preferences

- configuring

 - in Modeler mode

- creating and configuring
- specifying for different types of users

User Preferences menu choice

- configuring user preferences, using
- Project menu

User tab of work objects

Users menu choice

utilization

- of individual resources
- of Resource Managers
- of resource pool
- of resources
- of work objects

 - computing
 - displaying
 - example with no constraints
 - example with resource constraints

Utilization attribute

- allocating

 - associated resources, using
 - multiple resources, using
 - partial resources, using

- of Resource Managers

 - allocating multiple resources from a pool, using
 - allocating partial resources, using

UUID attribute

V

Value on Activation attribute

- configuring initial value
- for sliders
- for type-in boxes

variable-or-parameter class, charting values

- variables
 - getting values of
 - setting values of
- viewing messages and errors

W

- Waiting Color attribute

- Web toolbar

 - View menu

- Weekly Constraint

- Weibull distribution

- what-if analysis

 - introduction to

 - performing on models

- Window menu

- wires

 - on instruments

 - on Resource Managers

- work backups

 - analyzing wait times due to

 - due to

 - Maximum Activities

 - path synchronization

 - resource constraints

 - showing interactively

 - showing on input paths

 - due to resource constraints

 - using Snapshot Queue

- Work Object Duration Attribute attribute

- Work Object Yield Attribute Name attribute

- work objects

 - accumulating values in attributes of

 - associating

 - automatically creating class definitions for

 - batching

 - branching

 - class names of

 - comparing with resources

 - computing cycle time for

 - copying

 - attributes of

 - onto output path

 - costs

 - computing based on resource costs

 - using

 - creating

 - class definitions for

 - during processing

 - using a Source block

 - using Task block

 - customizing

 - deleting

 - using Sink block

 - using Task block

 - duration, computing

 - incrementing values in attributes

 - inserting into containers

 - merging multiple streams of

 - path types, configuring using

 - probing

 - performance

 - processing

 - and sending downstream

 - multiple streams sequentially

 - multiple streams synchronously

 - using blocks

 - reconciling associated

 - removing from containers

 - retrieving from pools and databases

 - showing

 - allocated resources of

 - associated

 - in containers, for Batch block

 - in containers, for Insert block

 - in containers, for Remove block

 - statistics of

 - storing to pools, files, and databases

 - updating user-defined attributes of

 - user-defined attributes of

 - utilization

 - computing

 - displaying

 - example with no constraints

 - example with resource constraints

- Work Time attribute

 - of activities

 - relating to Total Work Time

 - using to compute Total Work Time

- Workspace Margin attribute

- Workspace menu

 - Delete Background Image

 - description of

 - Get

 - Load Background Image

 - New

- workspaces

 - adjusting borders for

 - deleting

 - editing

 - colors of

- margins of
- name of
- properties
- hiding
- interacting with
- loading background images
- printing
- saving as JPEG
- scaling
- showing superior object of detail
- shrink wrapping

X

- X attribute
- X Maximum attribute
- X Minimum attribute
 - of Arrival Rate Input Graph
 - of charts
- X Range attribute
- X Scale attribute
- X Shift attribute
- X Size attribute
- X Step attribute
- XMB files, loading as background images

Y

- Y attribute
- Y Maximum attribute
- Y Minimum attribute
 - of Arrival Rate Input Graph
 - of charts
- Y Range attribute
- Y Scale attribute
- Y Shift attribute
- Y Size attribute
- Yield block
 - configuring
 - based on an attribute of a work object
 - mode
 - path identity of
 - proportional yield
 - specific features of
 - using random and random triangular
 - yields
 - determining yield value
 - reference
- Yield Mode attribute
 - of Yield block

- Proportional
- Random
- Random Triangular
- Work Object
- Yield Procedure Name customization attribute
- Yield Value attribute

Z

- Zoom In menu choice
 - scaling workspaces, using
 - View menu
- Zoom menu choice
 - scaling workspaces, using
 - View menu
- Zoom Out menu choice
 - scaling workspaces, using
 - View menu
- Zoom to Fit menu choice
 - scaling workspaces, using
 - View menu