User's Manual

Part C - Advanced Tools

SIMPROCESS

Release 5

CACI

EVER VIGILANT

Organization of the SIMPROCESS Documentation Set

The SIMPROCESS documentation set consists of four manuals:

- *Getting Started With SIMPROCESS*
- *SIMPROCESS User's Manual*
- *SIMPROCESS Metadata Manual*
- *SIMPROCESS OrgModel Manual*

All of the manuals can be opened directly from the **Help/SIMPROCESS Manuals** menu. Also, each manual includes this section which provides links to the chapters in every manual. Press the Control key when clicking any link to open the linked file in a new window. For Windows systems, in order for links between manuals to work properly, Adobe Acrobat or Acrobat Reader must be used to view the manuals.

Getting Started

The *Getting Started With SIMPROCESS* manual is a must for first time SIMPROCESS users. This manual can also be used for evaluation purposes. The chapters are

- Business Process Modeling With SIMPROCESS
- Installation

- Building Your First Model With SIMPROCESS
- Evaluating Alternatives With SIMPROCESS
- Demonstration and Reference Models

User's Manual

The *User's Manual* is divided into four parts with each part being a separate file. Part A is an excellent reference for beginners and casual users. This part contains detailed documentation of the basic and intermediate functions of SIMPROCESS. The chapters are

- Process Modeling and Analysis with SIMPROCESS
- SIMPROCESS Basics
- Statistical Modeling Constructs
- Activity Modeling Constructs
- Resource Modeling Constructs
- Graphical Modeling Constructs
- Activity-Based Costing
- Statistical Output Reports

Part B is a reference intended for advanced users of SIMPROCESS. This part contains detailed documentation of the programming and library management functions in SIMPROCESS Professional Edition. The chapters are

- Reusable Templates and Libraries
- Customizing a Model with Attributes and Expressions
- More Advanced Model Building
- Exporting Results

Part C describes the integrated tools included with SIMPROCESS Professional. The chapters are

- Advanced Data Analysis
- SIMPROCESS Database
- Experiment Manager
- OptQuest for SIMPROCESS
- SIMPROCESS Dashboards
- Model Bundles
- Custom Reports
- Scenarios

The Appendices are

- Importing Version 2.2.1 Models
- Activity Summary Table
- SIMPROCESS File Structure
- Statistical Distributions
- Statistical Tools Glossary
- SIMPROCESS System Attributes and Methods
- External Event Files
- Simulation Results File
- UML Interfaces
- Running Models Without GUI
- SIMPROCESS and External Java Classes

Metadata Manual

The Metadata Manual describes how to build and edit SIMPROCESS metamodels, assign metamodels to a SIMPROCESS model, and enter metadata in a SIMPROCESS model. The chapters are

- SIMPROCESS Metadata
- SIMPROCESS Metamodel Editor
- Assigning Metamodels
- Entering Metadata
- BPEL Metadata

OrgModel Manual

The OrgModel Manual describes how to build and edit SIMPROCESS Organization and Resource Models (OrgModels) and assign OrgModels to a SIMPROCESS model. The chapters are

- SIMPROCESS Organization and Resource Models
- SIMPROCESS OrgModel Editor
- Assigning OrgModels
- Using OrgModels with SIMPROCESS

CHAPTER 1

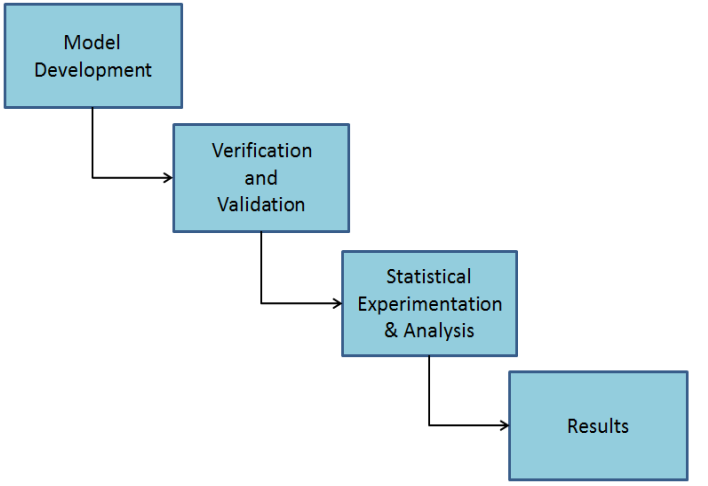
Advanced Data Analysis

SIMPROCESS comes with ModelFit, a data analysis tool that extends the ability to conduct statistical data analysis.

ModelFit provides automated fitting of distributions from sample data. It is used with the SIMPROCESS Auto Fits feature (see "Auto Fits Distributions").

The ModelFit tool:

- Increases the accuracy of experiments
- Decreases the time to find the solution to the problem
- Provides a statistical software tool tailored to the statistical simulation environment
- Performs data analysis
- Models random processes



ModelFit is used to analyze data and construct a data model of the random process that generated the data. An introduction to the statistical background for data analysis and data modeling is provided in the following section, “An Introduction to Data Analysis and Modeling.”

An Introduction to Data Analysis and Modeling

Introduction

Probability distributions are a way of describing the random variations that occur in the real world. Although the variations are called random, there are different degrees of randomness, and the different distributions correspond to how the variations occur. Consequently, different distributions are used for different purposes.

Probability distributions are represented by probability density functions. Probability density functions show how likely a certain value is. The more likely the value, the larger the probability density function at that point. The total probability at all points should sum to 100%, so the area under a probability density function is equal to one.

Cumulative density functions give the probability of selecting a number at or below that value. For example, if the cumulative density function value at 1.7 was equal to .75, then 75% of the time, selecting from that distribution would give a number less than 1.7. The value of a cumulative density function at a point is the area under the corresponding probability density curve to the left of that value. Since the total area under the probability density function curve is equal to one, a move towards more positive values in a cumulative density function approaches one.

Knowledge of all these details is not needed to effectively model a situation. Knowing which distribution best fits the data is most important. The curve fitting capabilities of ModelFit makes it easy to find the correct distribution for data.

In addition, standard distributions can be examined visually for different combinations of input parameters.

Standard Statistical Indicators

There are several statistical indicators that can tell a lot about the distribution of random values. Consult an introductory statistics textbook for detailed discussions of these indicators. This section simply gives their meanings and some of their implications. More definitions can be found in "Statistical Tools Glossary".

The mean is the most important characteristic of a data sample. Always try to get a good value for the mean (from actual data, if possible) if the scenario represents a time to complete some task or the number of arrivals in a certain time period.

The mode is the most likely value in a set of data. Consider the following list of numbers:

2, 3, 3, 3, 9, 10

The mode for this set of data is 3, since it occurs most frequently. The mean for this set of data is 5. If a distribution is symmetric about the mean, and singly-peaked, then the mode and the mean will be equal. In this case, the distribution contains a couple of large values which shift the mean away from the mode.

As discussed later, the mode is an important parameter for specifying a triangular distribution.

The standard deviation is a simple measure of the spread in the data. It is calculated by finding the mean, summing the squares of all the differences between the mean and the data points, taking the square root of this value, and then dividing by the square root of the number of data points. Loosely, it is the average difference from the mean. Why are the square and square root are taken? Because the result would be zero if just the differences were summed!

The standard deviation is a measure of how much spread there is in the data. If the standard deviation is large, there is a wide variation in the data. If it is small, the data are tightly clustered around the mean. The standard deviation is a necessary piece of information for many standard distributions. The variance is the standard deviation squared.

Most of the parameters for continuous distributions can be classified, on the basis of their physical or geometric interpretation, as being one of three basic types: location, scale, or shape parameters. The following discussion is taken from the book *Simulation Modeling and Analysis* (Third Edition) by Law and Kelton (2000).

> A location parameter l specifies an abscissa (x axis) location point of a distribution's range of values; usually, l is the midpoint (e.g., the mean of a normal distribution) or lower endpoint (e.g., location for a Pearson type V distribution) of the distribution's range. As l changes, the associated distribution merely shifts left or right without otherwise changing. Also, if the distribution of the random variable X has a location parameter of 0, then the distribution of the random variable $Y = X + l$ has a location parameter of l.
>
> A scale parameter b determines the scale (or unit) of measurement of the values in the range of the distribution. (The standard deviation is a scale parameter for the normal distribution.) A change in b compresses or expands the associated distribution without altering its basic form. Also, if the random variable X has a scale parameter of 1, the distribution of the random variable $Y = bX$ has a scale parameter of b.
>
> A shape parameter a determines, distinct from location and scale, the basic form or shape of a distribution with the general family of distributions of interest. A change in a generally alters a distribution's properties (e.g., skewness) more fundamentally than a change in location or scale. Some distributions (e.g., exponential and normal) do not have a shape parameter, while others (e.g., beta and Pearson type VI) may have two.

A probability density function is a representation of a statistical distribution. The probability density

function is large for likely values sampled from a distribution and is small for unlikely values. The maximum value for the probability density function is equal to the mode for that distribution. The area under the probability density function curve within a certain range of values gives the probability of sampling from that distribution and getting a number within that range. Of course, the total area under the curve must be equal to one, since a number from the distribution must be less than or equal to one.

A confidence interval is the range of values a certain percentage of the population would be expected to fall into if the sample were drawn from a normal distribution. Plus or minus one standard deviation corresponds to about a 68% confidence interval; plus or minus two standard deviations corresponds to about a 95% confidence interval; and plus or minus three standard deviations corresponds to about a 99.7% confidence interval.

Why Statistical Simulation Experiments?

Mean-Value Analysis

Mean-value analysis is a simple, although often quite useful, approach to modeling. The basic philosophy is to model processes by their average output.

For example, suppose there is a station that can process five parts per hour, and on average four parts arrive every hour. Mean-value analysis shows that the station should be able to handle the expected load.

However, in the real world there might be some hours the station may process only three parts, and other hours it may process seven. Likewise, there may not be a perfectly steady flow of four parts into the station every hour. Sometimes there may be more and sometimes less.

This statistical nature of the real world will lead to a diminished throughput, for there will be times when the station is idle, and other times when it is backed up, disrupting the flow to and from it in the rest of the factory. This is the limitation of mean-value analysis, and the reason that simulation is necessary for accurate predictions.

The Importance of Experimental Data

It is very important to have experimental data as the basis of a model. This may seem like a paradox: If I am modeling something that does not exist, where will I get the experimental data?

This is not as big a problem as it sounds. The individual components of a similar existing system will probably be close enough to ones in the proposed system to be useful in the model. The data do not have to be exhaustive, *any data is much better than no data*.

In fact, a good test of the model building process is to build a model of an existing system and check how closely the model and the system match up. This can often be a key to convincing others of the validity of the simulation modeling approach.

SIMPROCESS Statistical Distributions

The following table contains the statistical distributions available in SIMPROCESS. Eleven of the distributions also have an implementation from Apache Commons Math. These distributions have ACM included in the designation. Other distributions can be created from these by selecting **Distributions...** on the **Define** menu (see "User Defined Distributions"). See "Statistical Distributions" for descriptions of each distribution.

Beta Distribution	**Bet(shape1, shape2, minimum, maximum, stream)** **BetACM(shape1, shape2, minimum, maximum, stream)**
Binomial Distribution	**Bin(trials, probability, stream)** **BinACM(trials, probability, stream)**
Erlang Distribution	**Erl(mean, shape, stream)**
Exponential Distribution	**Exp(mean, stream)** **ExpACM(mean, stream)**
Gamma Distribution	**Gam(mean, shape, stream)** **GamACM(mean, shape, stream)**
Geometric Distribution	**Geo(probability, stream)**
Hyper Exponential Distribution	**Hex(mean1, mean2, probability1, stream)**
Inverse Gaussian Distribution	**InG(location, scale, shape, stream)**
Inverted Weibull Distribution	**InW(location, scale, shape, stream)**
Uniform Integer Distribution	**Int(minimum, maximum, stream)**
Johnson S_B Distribution	**JSB(minimum, maximum, shape1, shape2, stream)**
Johnson S_U Distribution	**JSU(location, scale, shape1, shape2, stream)**
Log-Logistic Distribution	**LLg(location, scale, shape, stream)**
Log-Laplace Distribution	**LLp(location, scale, shape, stream)**
Lognormal Distribution	**Log(mean, standard deviation, stream)** **LogACM(mean, standard deviation, stream)**

Negative Binomial Distribution	**NgB(s, probability, stream)**
Normal Distribution (non-negative)	**Nor(mean, standard deviation, stream)** **NorACM(mean, standard deviation, stream)**
Normal Distribution (unbounded)	**Nrm(mean, standard deviation, stream)**
Pareto Distribution	**Par(location, shape, stream)**
Pearson Type V Distribution	**PT5(location, scale, shape, stream)**
Pearson Type VI Distribution	**PT6(location, scale, shape1, shape2, stream)**
PertBeta Distribution	**Per(minimum, mode, maximum, lambda, stream)**
Poisson Distribution	**Poi(mean, stream)** **PoiACM(mean, stream)**
Random Walk Distribution	**RnW(location, scale, shape, stream)**
Triangular Distribution	**Tri(minimum, mode, maximum)** **TriACM(minimum, mode, maximum)**
Uniform Distribution	**Uni(minimum, maximum, stream)** **UniACM(minimum, maximum, stream)**
Weibull Distribution	**Wei(shape, scale, stream)** **WeiACM(shape, scale, stream)**

CHAPTER 2

SIMPROCESS Database

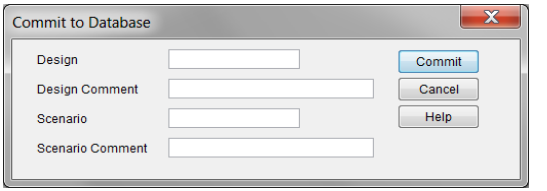
The Windows version of SIMPROCESS includes a Microsoft Access 2000 database (SimProcDB.mdb) designed to hold simulation results from simulation runs. This database includes predefined queries, graphs (forms), and reports, each of which can be modified to tailor the database for output analysis. The database can be updated to later versions of Access as SIMPROCESS will still be able to write to the database. (The Access images in this chapter are from Access 2000 through Access 2007.)

SIMPROCESS can use other non-Access SQL databases by making appropriate entries in the `sProcDB.properties` file provided by the installer. This file is installed on both Windows and non-Windows systems and is pre configured for use with the Access database described above. Additional information is included in this chapter to aid in using a non-Access SQL database patterned after the Access database (SimProcDB.mdb) to hold simulation results.

Note that the discussion in this chapter is only concerned with storing simulation results in SimProcDB.mdb or in a user-created SQL database patterned after SimProcDB.mdb. There are expression statements that allow SIMPROCESS to read from or write to any SQL database. See "Interfacing With A Database" for more information.

Committing Results to the Database

To commit simulation results to the database, select **Commit to Database** from the **Report** menu. The menu item will not be enabled until a simulation run is complete (either terminating normally or user terminated). If the database is not found, an error dialog will appear. If this occurs recheck the values in the `sProcDB.properties` file. Normally, the **Commit To Database** dialog will appear. A design name and scenario name must be entered. The design and scenario comments are optional. What constitutes a design will be discussed in the next section.



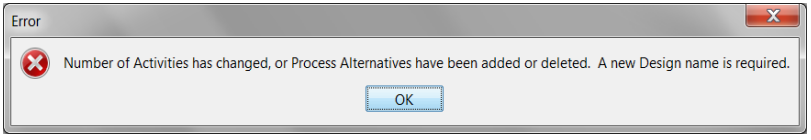
Once the information has been entered, click on **Commit**. At this point SIMPROCESS verifies that the design name and scenario name are allowable. If so, the results are stored in the database. When finished, the status bar shows **Commit Complete**.

System, Design, and Scenario

Three tables in the database control the results placed there: **System**, **Design**, and **Scenario**. The **System** table is the highest level table and contains the model name. Therefore, the database is designed to hold results from more than one model. If a record is deleted from the **System** table, all records related to that model are deleted from the Access database. (In order for this to be true in another SQL database, the appropriate relationships and/or constraints must be established when the database is created.) The **Design** table holds data from different versions of the same model. A model design consists of the following:

- Number of entity types
- Entity type names
- Number of resources
- Resource names
- Resource costs
- Number of Activities, Processes, and Process alternatives
- Names of Activities, Processes, and Process alternatives
- Activity/Process structure
- Start and End date
- Start and End time
- Number of replications
- Output time units.

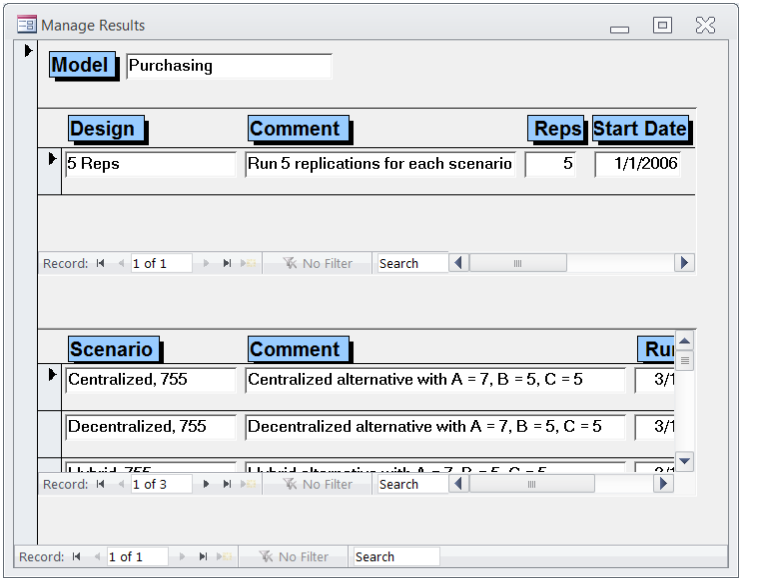
If any of the above change, then a new design name is required. For instance, assume results have been committed to the database with a design name of *Design 1*. After that a Process alternative is added to the model and the model is run again. If the same design name as before is used, SIMPROCESS will prompt for a new design name because the number and names of Process alternatives have changed.



Note that there will be no prompt for a new design name if a different alternative is run that was already in the model. If another alternative is run, the design name or scenario name can be changed. When a design is deleted from the **Design** table, all records associated with that design are removed from the Access database. (In order for this to be true in another SQL database, appropriate relationships and/or constraints must be established when the database is created.) Therefore, to reuse a design name for the same model name (and there has been a design change), first delete the design.

A scenario is used to track various runs of the same system (model) and design. For example, one run might have a certain resource level at 3 while another run has the same resource level at 4. Differentiating those runs in the database is done through the scenario name. The scenario comment field is a good place to document the important settings for that scenario. Again, if a scenario is deleted, all records associated with that scenario will be deleted from the Access database. (As before, in order for this to be true in another SQL database, appropriate relationships and/or constraints must be established when the database is created.)

In the Access database, the **Manage Results** form should be used to delete results for a Model (System), Design, or Scenario. DO NOT modify the tables directly. Using the **Manage Results** form ensures the tables do not become corrupted. This form can be run from the **Forms** tab in Access. Also, when using **Launch Database Application** from the **Report** menu, the **Manage Results** form launches automatically. If the form is blank, then there are no run results in the database.

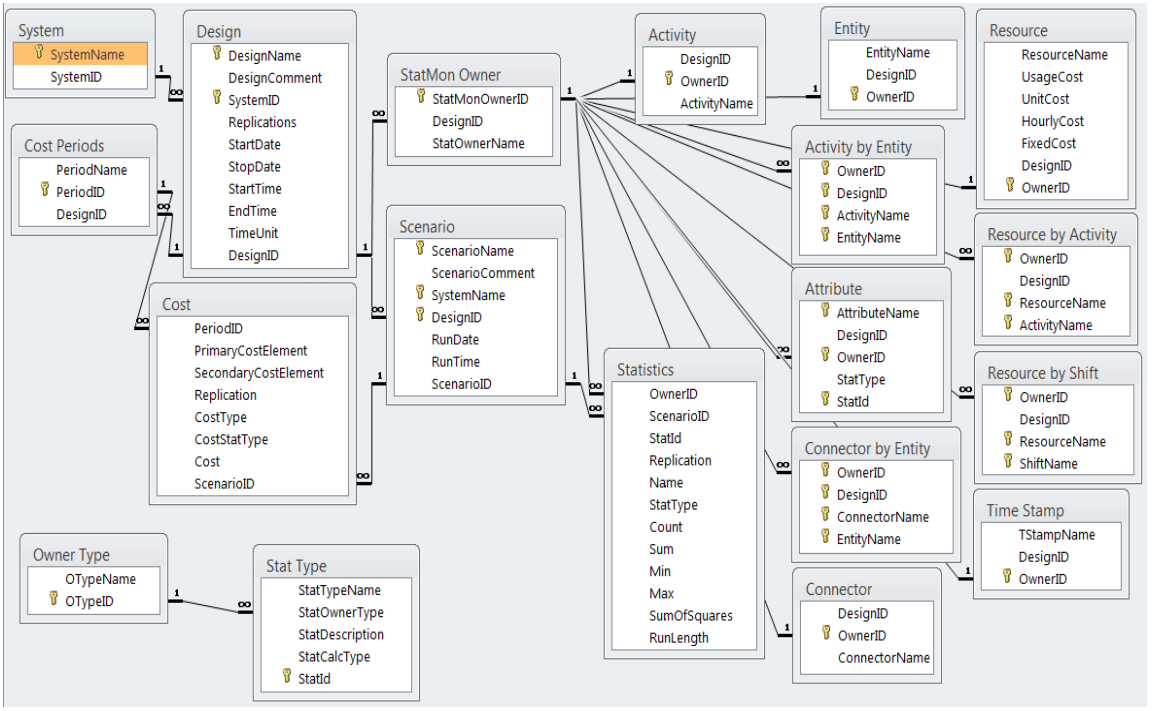


The **Launch Database Application** menu item is only enabled on Windows systems. It will be enabled even if the sProcDB.properties file is configured for a non-Access database.

IMPORTANT: It is good practice to regularly compact the Access database, particularly after deleting records.

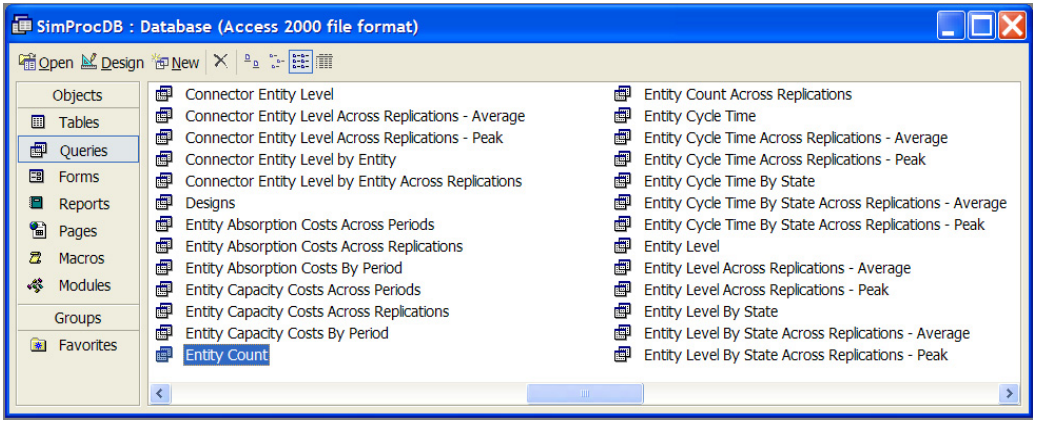
Database Table Relationships

Viewing all the tables and the relationships between them in the Access database is helpful in understanding how the queries were constructed. Because other database tools may or may not support constraints and relationships in the same way as Access, it will be up to the user to create comparable mechanisms there if desired.

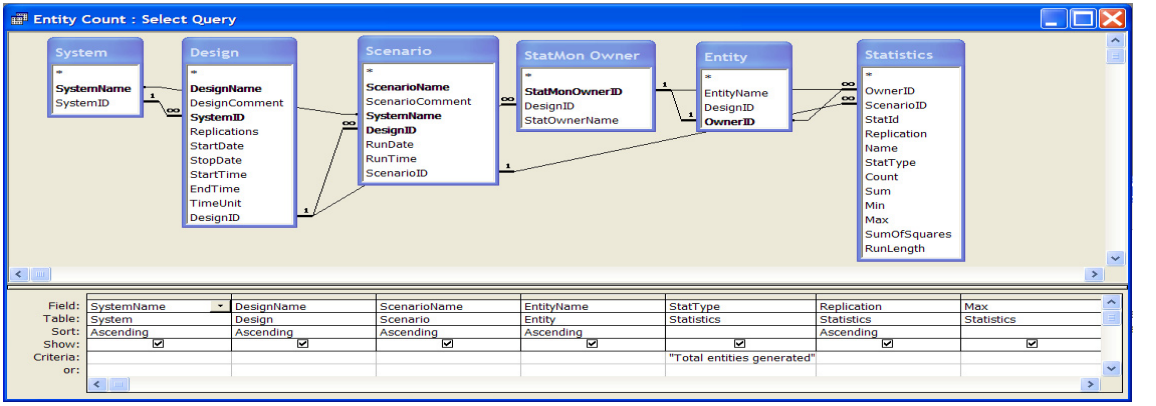


Database Queries

The predefined queries in the SIMPROCESS Access database are designed to offer the same information provided in the Standard Report. (See "Standard Report".) The queries can be used as-is or copied and modified. The queries are not restricted by system, design, or scenario. So when the **Entity Cycle Time** query is run, all entity cycle times without restriction will be displayed. Copying a query and restricting its search is one example of tailoring a query for analysis.

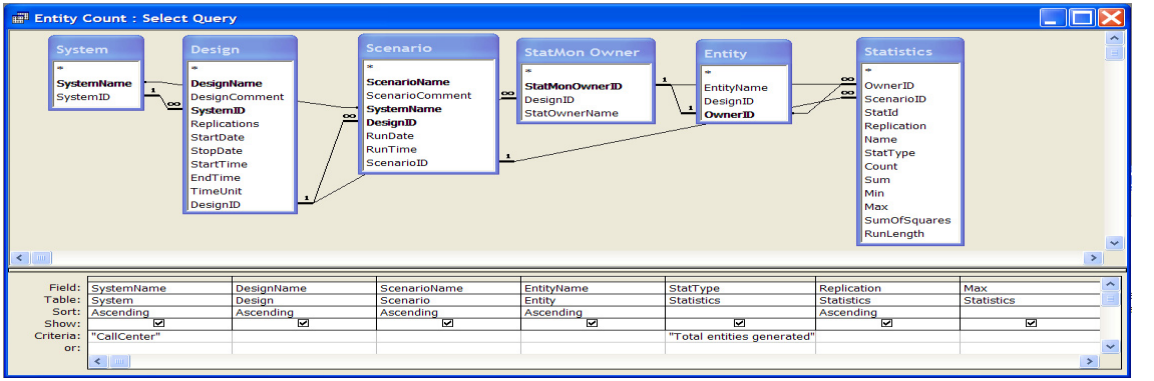


All of the queries were created in Access' Design View rather than SQL. Shown below is the Design View of the **Entity Count** query.

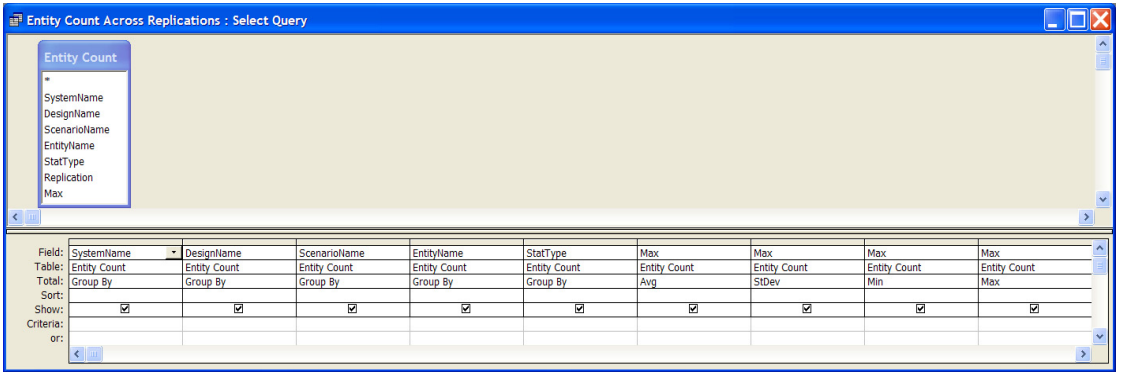


Even though the whole criteria cannot be seen, note that the **StatType** field from the **Statistics** table is restricted by "Total Entities generated" or "Entities remaining in system" or "Total Entities disposed."

These are the statistics types that make up **Entity Count**. Other fields could be restricted as well. For instance, the **SystemName** field from the **System** table could be restricted so only one model's results would be available. In the example below the query to the demonstration model **CallCenter** is restricted by placing "CallCenter" in the **Criteria** row of the query under **SystemName**.



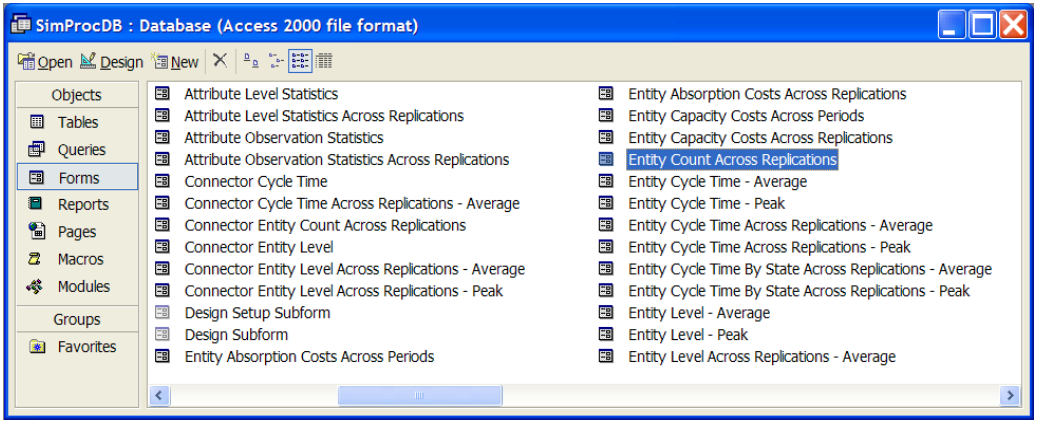
The queries that generate results across replications were developed using the queries that give results by replication. So if the **Entity Count** query is restricted to **CallCenter**, then the **Entity Count Across Replications** query will also be restricted to **CallCenter**. Looking at the Design View for **Entity Count Across Replications**, the only data source for the query is the **Entity Count** query.

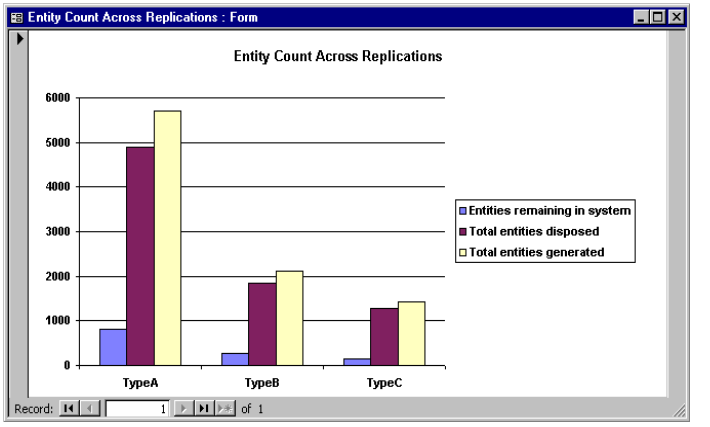


The average, standard deviation, minimum, and maximum were determined by selecting the appropriate statistic in the **Total** row for the field from the **Entity Count** query that was to be summarized across replications. This was accomplished by not including the **Replication** field from the **Entity Count** query.

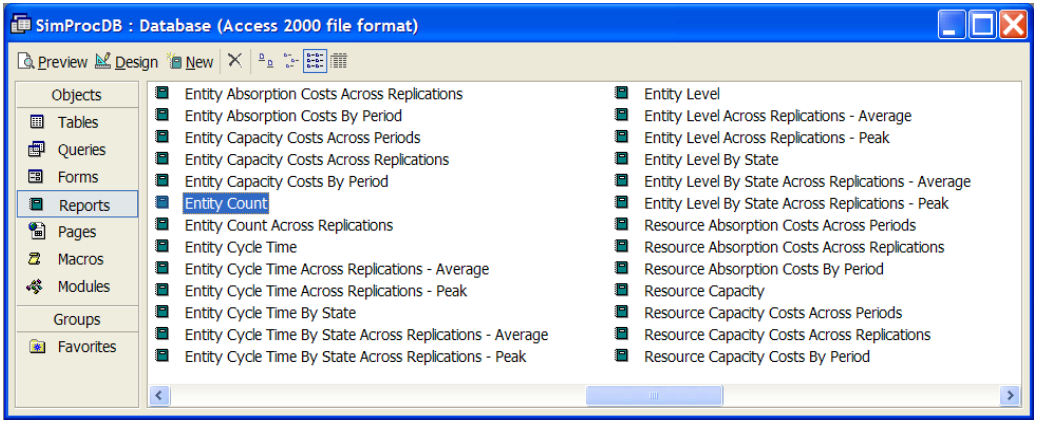
Forms (Graphs) and Reports

The **Forms** tab contains the predefined graphs plus the **Manage Results** form. These graphs are based on the predefined queries. Therefore, the **Entity Count Across Replications** form (or graph) is based on the **Entity Count Across Replications** query. In order to restrict the graph to only certain records, go to the original query (in this case **Entity Count** since **Entity Count Across Replications** is based on it) and restrict it.





The **Reports** tab contains the predefined reports. Like the graphs, the reports are based on the predefined queries. Therefore, the **Entity Count** report is based on the **Entity Count** query. In order to restrict the report to only certain records, go to the original query (in this case **Entity Count**) and restrict it.

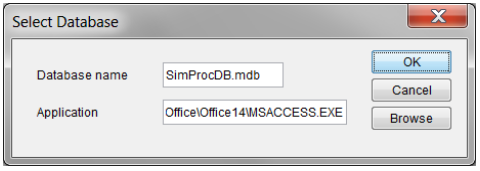


When using a non-Access SQL database, the predefined queries, graphs and reports described in this chapter may serve as a basis for developing SQL statements for retrieving and analyzing database contents. Those using SIMPROCESS on non-Windows systems where the SimProcDB.mdb file is not installed may obtain a copy for reference at the SIMPROCESS web site, or contact SIMPROCESS Technical Support for assistance.

Launch Database Application

This menu item on the **Report** menu (only enabled on Windows systems) brings up a dialog that selects the database and database application to use. SimProcDB.mdb and Microsoft Access are the defaults. (The remaining portion of these instructions assume MS Access is the database application.) If the database field is blank, then Access will start and prompt to open a database. If only a database is in the field (with no path), SIMPROCESS assumes the database is in the `SPUser` directory. If this is not the case, Access will start and prompt for a database. When the database desired is not in `SPUser`, enter the full path, and Access will open with that database.

If SimProcDB.mdb is entered in the database name field, the **Manage Results** form will launch automatically when the database opens. If other actions are desired, simply close the form and continue. Always use the **Manage Results** form to delete records from SimProcDB. If the tables are edited directly, they could become corrupted and run results placed in the database would be lost. The **Manage Results** form can be launched from the **Forms** tab. Copies of SimProcDB.mdb may be used for results. However, the **Manage Results** form will not launch automatically.



SIMPROCESS and Other Databases

Most of this chapter assumes the use of the Microsoft Access database with SIMPROCESS. Those not using Windows will be compelled to use another database, and Windows users may wish to have the flexibility to do so. This section will provide information which those users must know to employ another database for analyzing results.

To store results in any database, SIMPROCESS must have the file `sProcDB.properties` in the `SPUser` directory. The one provided by the installer is configured for use with the Access database. Also included there are two example files which can aid in using another SQL database:

- `mysql.sProcDB.properties`
- `simprocessdb.sql`

The `mysql.sProcDB.properties` file is an example of what needs to be put into the `sProcDB.properties` file in order to use MySQL, a popular open source database, with SIMPROCESS. It contains two required properties and others that are commented.

- `jdbc.drivers` - This property must provide the fully qualified name of a Java class which is a JDBC driver compatible with the database to be used. The one in this example is from the **MySQL Connector/J** driver, available with MySQL at www.mysql.com. The driver must be in the SIMPROCESS classpath at runtime, so the JAR file containing it should be placed into the `jre/lib/ext` directory under the SIMPROCESS installation directory (see "Java Extension Mechanism" for more information). Administrator privileges may be required.
- `jdbc.url` - This property identifies a specific database and is used to connect to it via the driver named above. Details on the possible forms and values of this property will be documented with available Java JDBC drivers. The example in this file shows using a MySQL database named "simprocess" on the local system.
- `jdbc.username` - This property is disabled in the example. If the database configuration requires a user name, remove the "#" to enable the property and give it an appropriate value.
- `jdbc.password` - This property is disabled in the example. If the database configuration requires a user password, remove the "#" to enable the property and give it an appropriate value.

The `simprocessdb.sql` file is an example of Data Definition Language (DDL) for creating a database in MySQL that is suitable for use with SIMPROCESS. The file contains necessary DDL statements to drop and create a database named "simprocess" and then to create each of the tables needed for storing simulation results, some of which require insertion of static data. While the Access database contains some tables and/or columns with spaces in their names, this DDL uses underscore characters instead. Other features, such as foreign keys, referential integrity, cascading deletes, etc., are not supported by all databases, a fact which should be considered in the management of any selected SQL database.

The DDL statements in this example name the database tables and columns using the same case that SIMPROCESS uses in its internal SQL statements, though not all databases will require or respect

case. For instance, testing showed that MySQL server 3.23.52 insisted that table names in SQL statements match the case used in the DDL, while MySQL server 4.0.14-standard allowed the use of all lower case names irrespective of the DDL used.

The included MySQL examples were tested with SIMPROCESS in the following environments:

- SIMPROCESS on Linux, MySQL server 3.23.52 on the same Linux host
- SIMPROCESS on Windows, MySQL server 3.23.52 on a Linux host via LAN
- SIMPROCESS on Windows, MySQL server 4.0.14-standard on a Mac OS X host via LAN

SIMPROCESS users are encouraged to send information to the SIMPROCESS Technical Support team about their own experiences with other non-Access SQL databases. Reports of good (or bad) JDBC drivers for various databases are welcomed. Please send compressed example files similar to these to `simprocess@caci.com`. When sending such reports and/or examples, please be sure and provide as much detail as possible about any special circumstances, platforms involved, etc., and indicate whether it's all right to share any samples with others.

CHAPTER 3

Experiment Manager

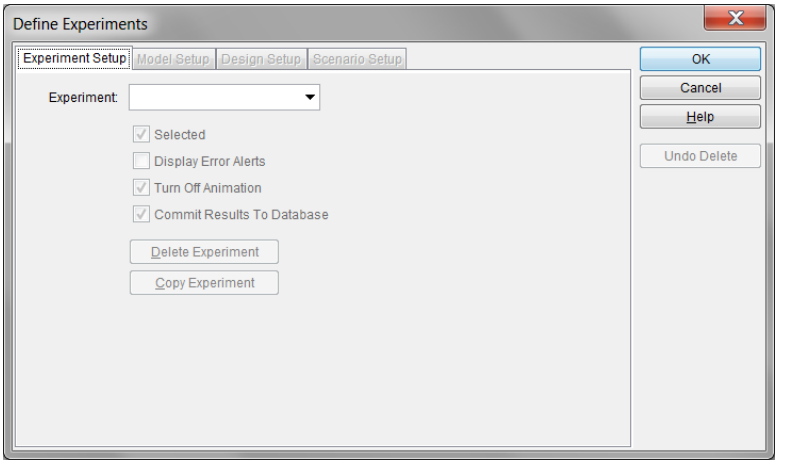
The Experiment Manager sets up model runs that SIMPROCESS will run automatically and, optionally, after each run, place the results in the database. This is accomplished by defining experiments in SIMPROCESS, which are stored in `Experiments.xml` in the SIMPROCESS working directory (see "SIMPROCESS Working Directory"). SIMPROCESS reads the information from `Experiments.xml`, loads the appropriate model, runs the model, then (optionally) commits the results to the database.

The four items on the **Experiment** menu control the operation of the Experiment Manager.

- Define Experiments
- Run All Experiments
- Run Selected Experiments
- Run Specific Experiment

Defining Experiments

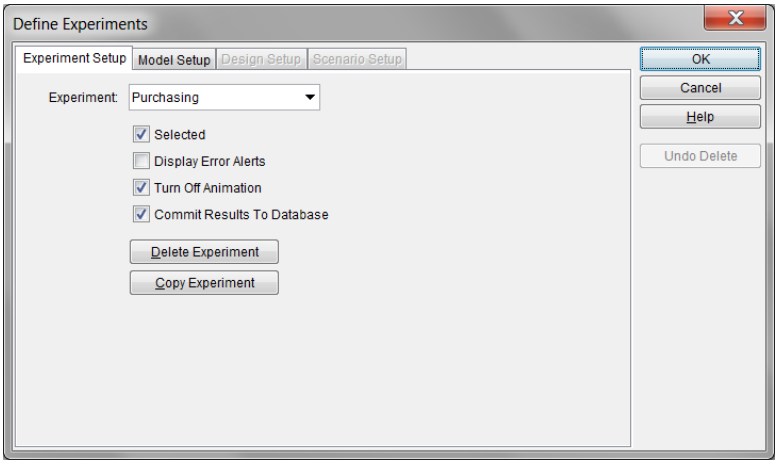
An experiment can have multiple models, a model can have multiple designs, a design can have multiple scenarios, and a scenario can (optionally) have multiple **Model Parameters** (Creating a User Defined Attribute) and **Process Alternatives** (Processes). Selecting **Define Experiments/Using SIMPROCESS...** brings up a dialog experiment information is entered.



If no experiments have been defined, everything is disabled except the **Experiment** combo box. If there are already experiments defined, the first experiment appears in the **Experiment** combo box, and all appropriate items are active. To add to or modify an experiment already defined, select an experiment from the **Experiment** combo box. There may be a slight delay when a predefined experiment is selected. This is because the first model for the experiment must be loaded and then processed.

Entering Experiment Information

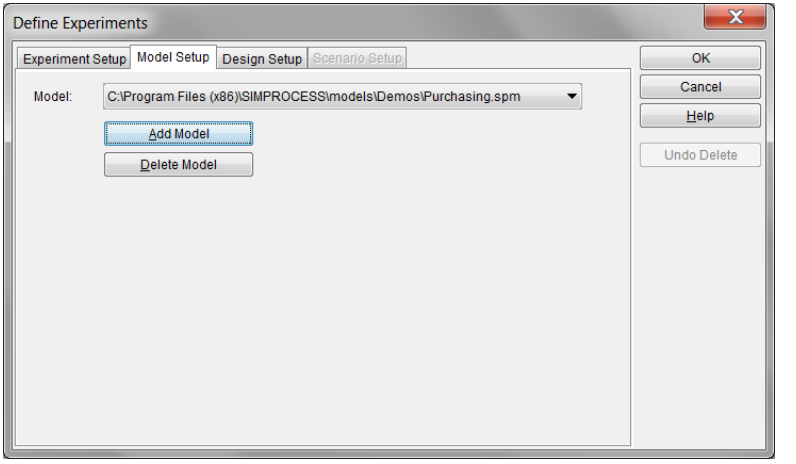
There are five fields that require entries for each experiment: **Experiment**, **Selected**, **Display Error Alerts**, **Turn Off Animation**, and **Commit Results To Database**. This shows an example experiment called **Purchasing**. Type the name in the **Experiment** combo box, then press **Enter**. If not already active, entering an experiment name causes the remaining items on the **Experiment Setup** tab and the **Model Setup** tab to activate.



Once the experiment has been named, then check or uncheck **Selected**. The default is checked. **Selected** is used as an option for running experiments. **Display Error Alerts** defaults to unchecked. If unchecked, then SIMPROCESS will run the experiment without stopping to notify of errors. This means SIMPROCESS will run every combination of model, design, and scenario that it can. Those with errors will be skipped over. If **Display Error Alerts** is checked, when SIMPROCESS encounters an error when running the experiment, an error dialog will display that will stop the experiment until the error has been acknowledged by the user. **Turn Off Animation** defaults to checked. If checked, all models in the experiment will run with animation turned off. **Commit Results To Database** defaults to checked as well. When selected, the results from each run will be placed in the SIMPROCESS database. Multiple experiments can be entered. Although not necessary, it is recommended that model, design, and scenario information be entered before adding another experiment.

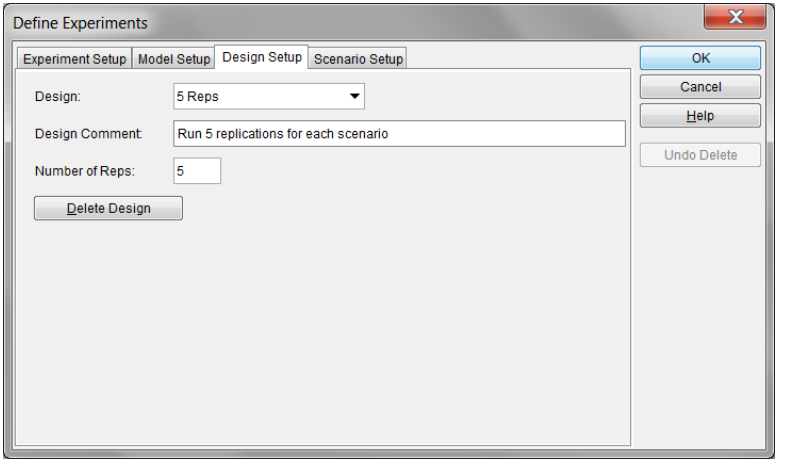
Entering Model Information

The next tab is **Model Setup**. This tab is active when an experiment name is entered or selected on the **Experiment Setup** tab. To select a model, click the **Add Model** button. This brings up a file chooser dialog. The complete path and the name of the model selected is added to the **Model** combo box. All models added are part of the experiment that was selected on the **Experiment Setup** tab. Although multiple models can be entered before entering design and scenario information, it is recommended that all the design and scenario information (along with parameter and process information, if needed) be entered before entering the next model. If the selected model is moved after being added to an experiment, the model and all associated experiment information (design, scenario, etc.) will be deleted from the experiment the next time **Define Experiments/Using SIMPROCESS...** is selected.



Entering Design Information

The **Design Setup** tab is active when a model is selected in the **Model** combo box on the **Model Setup** tab. At least one design name is required for each model. The **Design Comment** and **Number of Reps** fields are optional. If no value (integer only) is entered for **Number of Reps**, the last saved model value will be used for each scenario in this design unless there was a previous design for the same model. One design called "5 Reps" is entered for the model `C:\Program Files\SIMPROCESS\models\Demos\Purchasing.spm`.



Entering Scenario Information

The **Scenario Setup** tab is active when a design is selected in the **Design** combo box on the **Design Setup** tab. At least one scenario must be entered for each design. If creating a new scenario, a name for the scenario must be entered in the **Scenario** combo box. The **Scenario Comment** field is optional. At this point no more information is required to run a scenario.

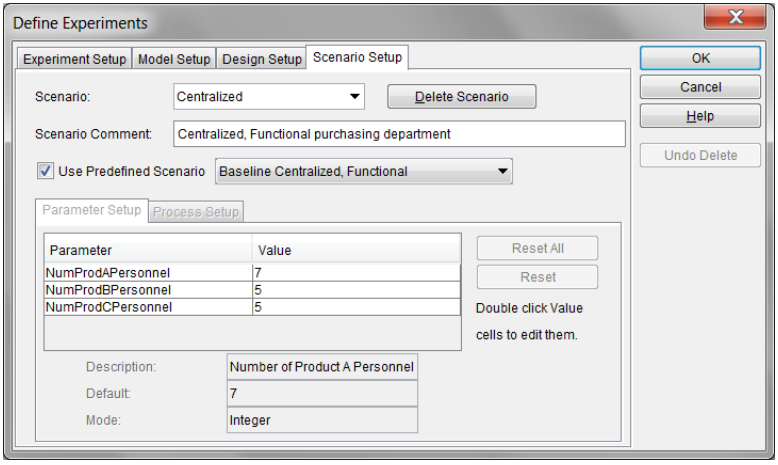
There are two ways to further define a scenario and both are optional.

- Selecting **Use Predefined Scenario**
- Setting **Model Parameter** values and **Process Alternatives**

If either of these are not done, then the last saved value for each **Model Parameter** and **Process Alternative** are used.

Using a Predefined Scenario

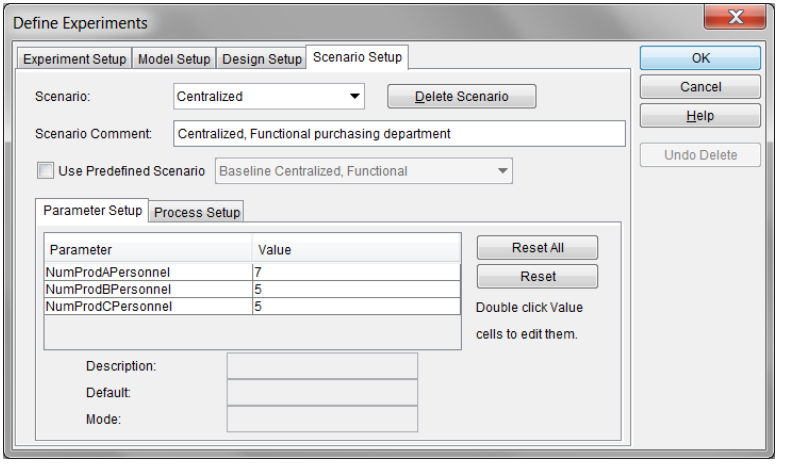
When **Use Predefined Scenario** is selected the **Parameter Setup** and the **Process Setup** tabs are disabled and the list of predefined scenarios is enabled.



The predefined scenarios list contains scenarios defined using **Scenarios...** on the **Simulate** menu (see "Scenarios" on page 121). These scenarios define more than just **Model Parameter** values and **Process Alternatives** and they override the **Number of Reps** setting on the **Design Setup** tab. When the simulation completes the results from the scenario are saved into an external file (`Scenarios.xml`) for later use. Scenario Reports can be created to compare results from different scenarios or within the same scenario (see "Comparing Scenarios" on page 129).

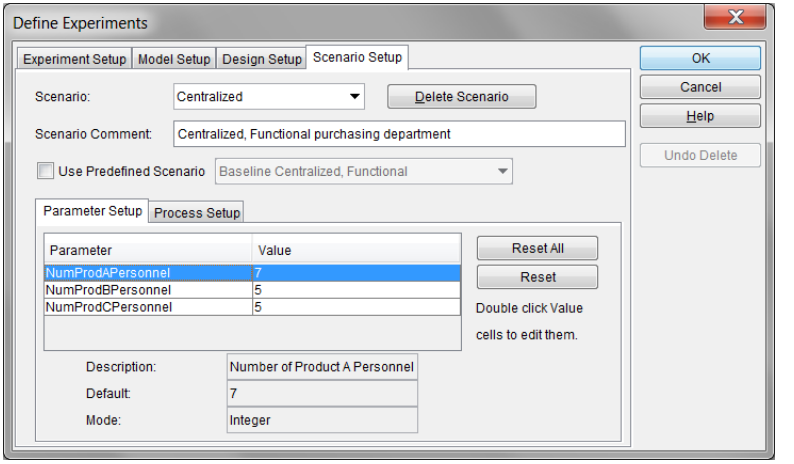
In order to set **Model Parameter** values and **Process Alternatives** directly, leave **Use Predefined Scenarios**

deselected. **Model Parameters** are set on the **Parameter Setup** tab, and **Process Alternatives** are set on the **Process Setup** tab.



Entering Model Parameter Information

The **Parameter Setup** tab is used to enter **Model Parameter** values. The attributes for the selected model that have been designated as **Model Parameters** appear in the table. When a **Model Parameter** is selected, its description (comment from the attribute definition), mode (Integer, Real, or String), and default value are displayed in the text fields below the table. Once selected, the value can be changed by typing into the field on the table. Values entered that do not match the mode will not be accepted. The **Reset** button will reset the selected **Model Parameter** to its default value. The **Reset All** button resets all **Model Parameters** to their default values. In this example there are three Model attributes that can be changed for a scenario. (When changing selected scenarios, there may be a slight lag for the **Model Parameter** values to update. This is due to the processing required to update the values.)

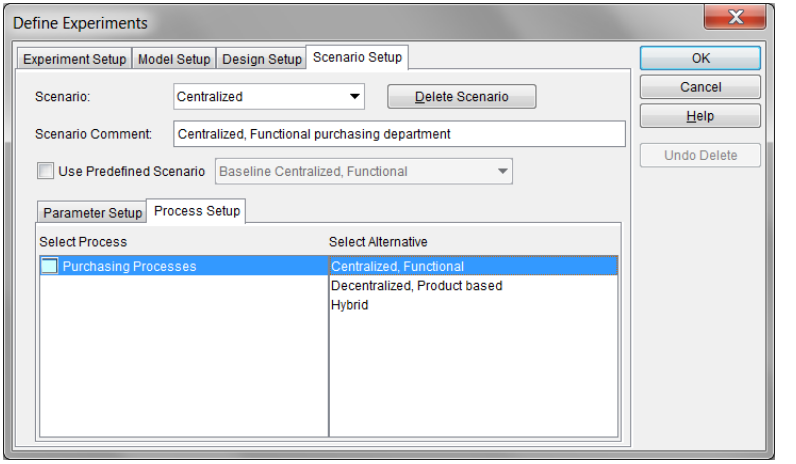


NOTE

If no **Model Parameter** information is entered for a scenario, then the model will run with the last saved values for the **Model Parameters**. If a parameter has been changed for a scenario, and the next scenario uses the same model (that is, a new model is not loaded), then the **Model Parameter** will have the same value as the previous scenario. It will not reset to its default value.

Entering Process Alternative Information

The **Process Setup** tab is used to set the active alternatives for processes that have more than one alternative. The panel on the left displays the processes in the selected model that have more than one alternative. (Since processes are displayed by name, it is recommended that process names not be duplicated within a model.) Once a process is selected, the alternatives for that process will appear in the panel on the right. If an alternative has already been selected, that alternative will be highlighted. If no alternative is highlighted then the model will run with the alternative that was active during the last save of the model. Select the alternative to be active for the current scenario. If a Master Process (Master Processes) has multiple alternatives, the copies of the Master Process will not be displayed. The selected alternative for the Master Process will be applied to all copies.



In this example the **Centralized, Functional** alternative for the **Purchasing** process will be active for the current scenario.

NOTE

If no process/alternative information is entered for a scenario, then the model will run with the last saved active alternative for each process. If an alternative has been changed for a scenario, and the next scenario uses the same model (that is, a new model is not loaded), then the active alternative for a process will be the same as the previous scenario unless a different alternative is specified in the Experiment Manager.

Undo Delete Button

The **Undo Delete** button is active whenever an experiment, model, design, or scenario has been deleted. The button remains active until used or until the scope of the deleted item has been lost. An experiment can be restored any time before another delete occurs. A model can be restored as long as the experiment to which it belongs remains the selected experiment. A design can be restored as long as the experiment and model to which it belongs remain selected. Finally, a scenario can be restored as long as the experiment, model, and design to which it belongs remain selected.

Running Experiments

Once the experiments have been defined, run the experiments. This section discusses starting experiments, the operation of experiments, how to interact with experiments, and how to generate standard reports from experiments.

Starting Experiments

There are three options for running experiments.

- Run All Experiments
- Run Selected Experiments
- Run Specific Experiment

Run All Experiments

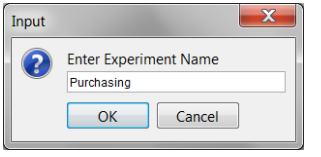
The menu item **Run All Experiments** causes SIMPROCESS to attempt to run every experiment defined. This option ignores the **Selected** field.

Run Selected Experiments

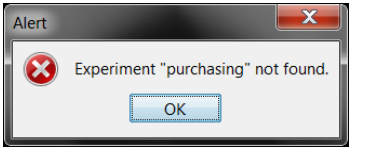
When this menu item is selected, SIMPROCESS attempts to run the experiments in which the **Selected** field is checked.

Run Specific Experiment

This menu item specifies one experiment to run. When selected, enter the experiment name in the dialog.



If no match occurs, an error message appears stating the experiment could not be found in the database. Case does matter when selecting a specific experiment. The case and spelling must match.


Alert
Experiment "purchasing" not found.
OK

Experiment Operation

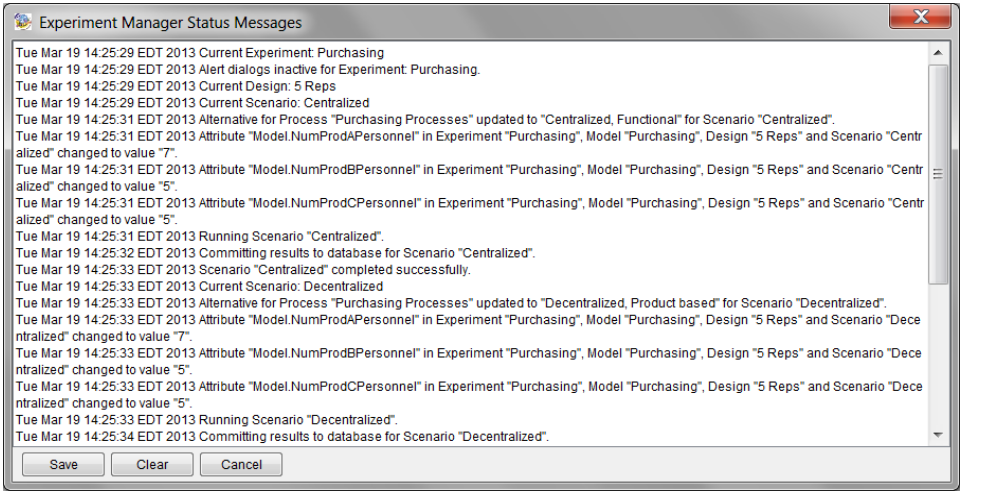
The Experiment Manager operates as follows when running experiments:

Once an experiment begins, SIMPROCESS begins a sequence of steps. Each step occurs if the previous completes successfully. The steps to running experiments are:

1. Load Experiments.xml.
2. Prompt to save current work if necessary.
3. Enter a loop with sub loops.
    Find Experiment.
        Find Model.
            Find Design.
                Find Scenario.
                Setup Predefined Scenario or Find Parameters and Find Process Alternatives.
                Load Model.
                Verify Design.
                Run Model.
                If selected commit results to database.
                If Predefined Scenario save results to Scenarios.xml.
                End loop.
            End loop.
        End loop.
    End loop.
4. Display status message on success of experiments.

Experiment Trace

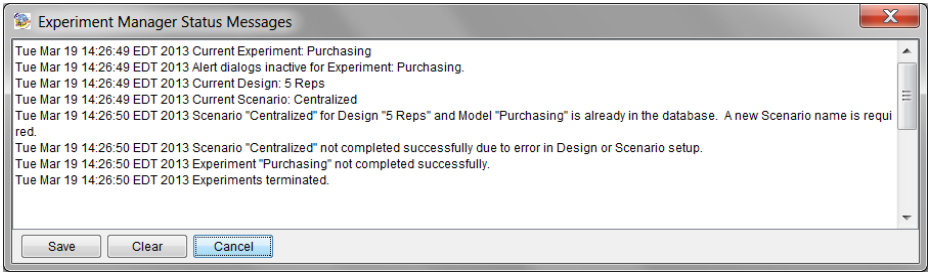
During experiment execution, the Experiment Manager displays a trace showing the steps that have been completed and the current step. The trace includes the date, time, and a message.



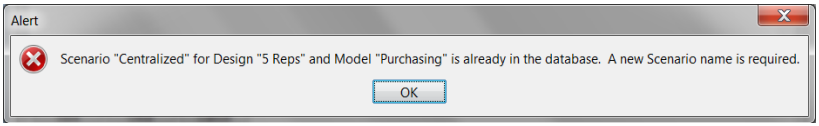
It is important to examine the Experiment Manager trace, since it will give the complete status of the experiments. The above example shows the experiment, alert dialog status (see below), model, design, and scenario. Also, it shows the **Process Alternatives** that were active and the values of the **Model Parameters** for the scenario.

Experiment Errors

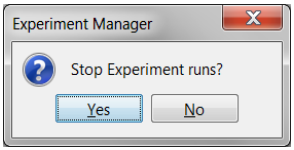
As explained in the instructions for defining experiments, when the **Display Error Alerts** field is not selected, interactive error messages are suppressed. However, error messages will appear in the Experiment Manager trace. The Experiment Manager trace shows which experiments completed successfully and which encountered errors. Especially important are the messages that show errors with the **Model Parameters**.



When the **Display Error Alerts** field is selected, error dialogs will appear that pause the Experiment until the error is acknowledged.



Also, after the error displays, if there are more experiments to be run, a dialog appears with the option to stop all experiment runs.



If experiments are to be run overnight, it is recommended that the **Display Error Alerts** field not be selected for every experiment that is to be run. That way SIMPROCESS will run every combination of Experiment, Model, Design, and Scenario that it can. Otherwise, if an error occurs, processing stops until the error is acknowledged.

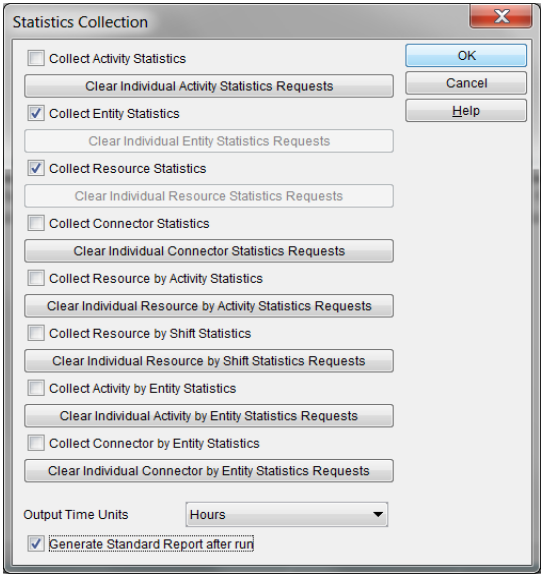
Interacting With Experiments

A model run by the Experiment Manager can be interacted with in the same manner as a model that was user initiated. This means the animation can be turned on or off, the animation settings can be changed, the model can be navigated, etc.

Experiments can be stopped early by stopping a model run. An experiment cannot be stopped while loading experiment information, loading a model, or committing results to the database. If the **Display Error Alerts** field is selected, there will be an option to stop completely. Otherwise every run must be stopped to stop all experiment, model, design, and scenario combinations.

Generating Standard Reports From Experiments

If selected, the Experiment Manager automatically commits the run results to the database. If an ASCII version of the Standard Report is needed, go to **Define Global Statistics Collection** on the **Report** menu. Select **Generate Standard Report after run**. This will create a file in the model's directory that will contain the report for every replication, the sum of the replications, and the average of the replications.


Statistics Collection
OK
Cancel
Help
Collect Activity Statistics
Clear Individual Activity Statistics Requests
Collect Entity Statistics
Clear Individual Entity Statistics Requests
Collect Resource Statistics
Clear Individual Resource Statistics Requests
Collect Connector Statistics
Clear Individual Connector Statistics Requests
Collect Resource by Activity Statistics
Clear Individual Resource by Activity Statistics Requests
Collect Resource by Shift Statistics
Clear Individual Resource by Shift Statistics Requests
Collect Activity by Entity Statistics
Clear Individual Activity by Entity Statistics Requests
Collect Connector by Entity Statistics
Clear Individual Connector by Entity Statistics Requests
Output Time Units
Hours
Generate Standard Report after run

CHAPTER 4

OptQuest for SIMPROCESS

In today's highly competitive global environment, people are faced with many difficult decisions, such as allocating financial resources, building facilities, managing inventories, determining product mix strategies, and more. Modeling a decision problem in SIMPROCESS discovers what performance measures can be expected using a certain strategy.

A strategy can be defined as a certain set of values for the model parameters. Other words for strategy include scenario and solution. Note that "solution" refers to the model parameter values (inputs to the simulation), *not* the resulting values of the performance measures.

Suppose the goal was to find the best strategy (without using OptQuest). Model parameter values for a strategy being considered would have to be entered into SIMPROCESS and the results analyzed. Then, this process would have to be repeated for every strategy under consideration. (Experiment Manager automates running models with various values for model parameters. See "Experiment Manager" on page 28.) Often, it would not be clear how to adjust the model parameters from one simulation to the next. This type of search is tedious and, in problems with thousands or millions of potential alternatives, impractical.

OptQuest enhances SIMPROCESS by automating the search for an optimal strategy.

Note that OptQuest for SIMPROCESS is licensed separately from SIMPROCESS.

Overview of OptQuest for SIMPROCESS

Since OptQuest is an optimization tool, it is attempting to minimize or maximize the value of a performance measure based on limits (constraints, upper bounds, and lower bounds). OptQuest automatically runs SIMPROCESS models varying the values for the model parameters searching for optimum results within the specified limits.

When the optimization runs:

OptQuest feeds a potential solution into SIMPROCESS model by setting the decision variable (model parameter) values.

SIMPROCESS runs one simulation (which may include multiple replications) to evaluate the solution.

OptQuest takes the resulting response values from SIMPROCESS.

OptQuest analyzes the results of the simulation and uses its intelligent search procedures to generate a new potential solution, which it then sends to SIMPROCESS.

OptQuest repeats this process. Its ultimate goal is to find the solution that optimizes (maximizes or minimizes) the value of the model's objective.

Elements of an OptQuest Optimization

Optimizations consist of an objective (minimize or maximize), decision variables, and constraints (constraints are optional).

Objective: An expression that represents the model's objective, such as minimizing queues or maximizing profits.

Decision Variables: Variables that can be meaningfully manipulated to affect the performance of a simulated system. The model parameters in SIMPROCESS are called decision variables in OptQuest.

Constraints: Relationships among decision variables and output variables. For example, a constraint might ensure that the total amount of money allocated among various investments cannot exceed a specified amount, or the processing time of a system cannot be greater than a certain value.

There can be bounds (upper and lower) on the decision variables and performance measure constraints (statistics).

Using OptQuest: An Overview

Follow these steps to use OptQuest.

Create a simulation model, or open a simulation model.

Prepare the model for optimization. This includes creating and assigning model parameters, establishing a performance measure to optimize, making sure the model is complete, verifying and validating the model, and making sure the model runs with no errors.

Set up the optimization, or select previous optimization settings (**Tools/OptQuest** menu). The following steps apply to creating a new optimization. Previously defined optimization settings can be modified or deleted.

a. Define the objective.

b. Select the decision variables.

c. Define constraints.

d. Set process alternatives.

e. Select optimization options.

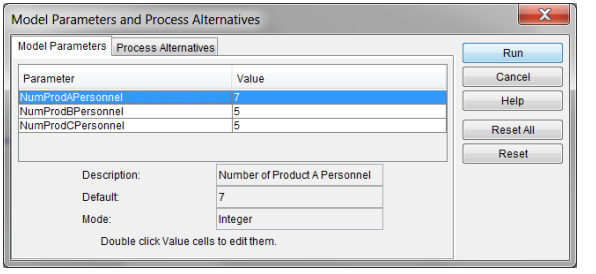
Run the optimization. When **Start** is selected, the model is saved, and the OptQuest interface appears. From this interface the user can monitor the optimization, stop the optimization, and create reports once complete. Also, if selected, OptQuest will prompt for more iterations when approaching the maximum iteration.

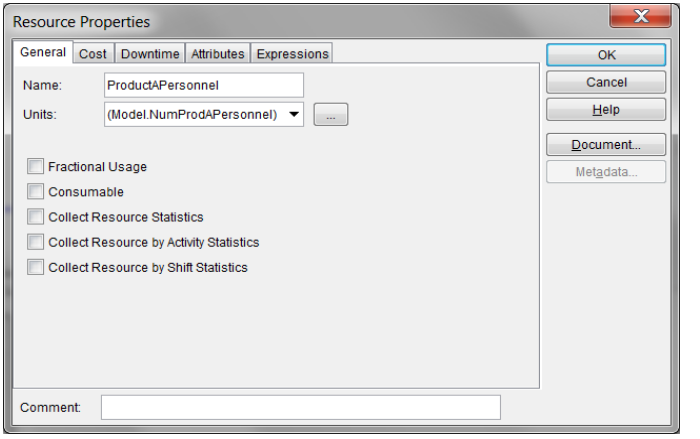
Interpret the results.

Optimization Setup

Preparing for Optimization

In order to use OptQuest with a model, the model must have model parameters of type Integer and/or Real. Model parameters are user controlled values that typically set some of the initial conditions of a simulation. For instance, if Truck is a resource in the model, a model parameter could be used to set the number of the resource Truck available. In the Inventory demonstration model (see *SIMPROCESS Getting Started Manual*) reorder levels are controlled by model parameters. When a model is run that has model parameters, a dialog appears that allows values of the model parameters to be changed before the run begins. The image below shows the model parameter dialog from the Purchasing demonstration model. There are three model parameters: `NumProdAPersonnel`, `NumProdBPersonnel`, and `NumProdCPersonnel`. All three of these are Model Attributes and their **Mode** is Integer. These model parameters set the number of units of the resources ProductAPersonnel, ProductBPersonnel, and ProductCPersonnel.





Since OptQuest uses the Integer and Real model parameters as decision variables, when OptQuest runs the model as part of the optimization, it sets the values of the model parameters for each run.

Another requirement for optimization is an objective. Therefore there must be a performance measure in the model to maximize or minimize. This performance measure could be one of the standard statistics collected by SIMPROCESS or a specialized statistic, which could consist of a linear or non-linear combination of various standard statistics. Again, referring to the Purchasing demonstration model, the objective is to minimize the variable cost of all the resources. Since variable cost is collected automatically, this statistic does not have to be created. However, in the Inventory demonstration model, the objective is to minimize inventory and minimize the cycle time of an order. Since only one value can be optimized, these performance measures must be combined into one. In the Inventory demonstration model there is a linear combination of the two performance measures. This combination is in the **End Simulation** expression of the **Model Expressions** (**Define/Model Expressions**).

```
Model.OrderTime := GetEntityStatistic("Order", "tokendelay", "Avg",
Replication);
Model.FinishedProduct := GetResourceStatistic("FinProduct",
"resrcidle", "Avg", Replication);


Model.InventoryPlusOrderTime := Model.FinishedProduct + (5.0 *
Model.OrderTime);
```

When combining performance measures the Expression language will be needed to retrieve statistics collected during the simulation. (See "Accessing Statistics During Simulation".) The example above shows values being assigned to three Model Attributes: `OrderTime`, `FinishedProduct`, and `InventoryPlusOrderTime`. `OrderTime` is assigned the average cycle time for the entity **Order**. `FinishedProduct` is assigned the average number of units idle of the resource **FinProduct**. `InventoryPlusOrderTime` is a linear combination of the two, and statistics are collected for that attribute. Thus, the Model Attribute `InventoryPlusOrderTime` is the value to minimize. Note that the System Attribute `Replication` is used to get the value from the appropriate replication of the simulation.

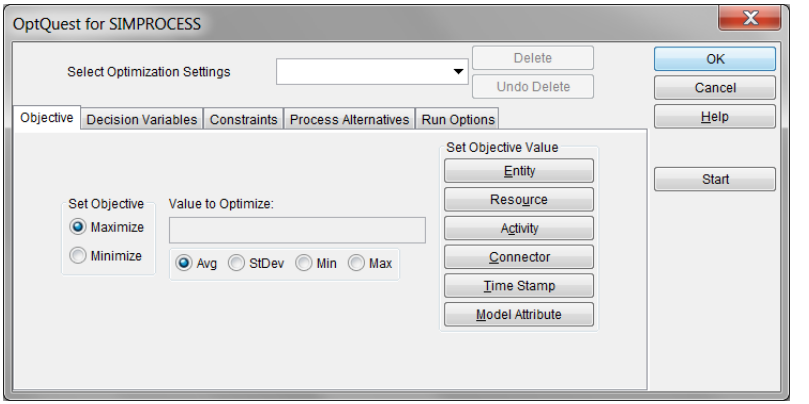
Standard or specialized statistics can also be used as constraints in OptQuest. Therefore, if performance measures need to be used as constraints, make sure the appropriate statistics are being collected. For example, the values being assigned to `OrderTime` and `FinishedProduct` are also constraints in the Inventory demonstration model.

The final steps in preparing a model for optimization apply whether or not OptQuest is intended to be used. Make sure the model is complete. That is, all required fields in properties dialogs have been defined (especially if portions of the model have come from other models), and all activities and processes are connected. Run the model for several replications to ensure there are no errors. Determine the appropriate number of replications needed to achieve statistical significance. Finally, and most importantly, make sure the model accurately reflects the system being modeled.

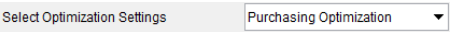
Defining An Optimization

Optimizations can only be defined for the currently active model in SIMPROCESS. Thus, if the model to optimize is not loaded, load it; or if already loaded, make sure the model to optimize is the active model.

Select **Tools/OptQuest**. (If **OptQuest** is not enabled, contact the SIMPROCESS Sales Manager to purchase a license for OptQuest for SIMPROCESS.) This brings up the OptQuest for SIMPROCESS setup dialog.



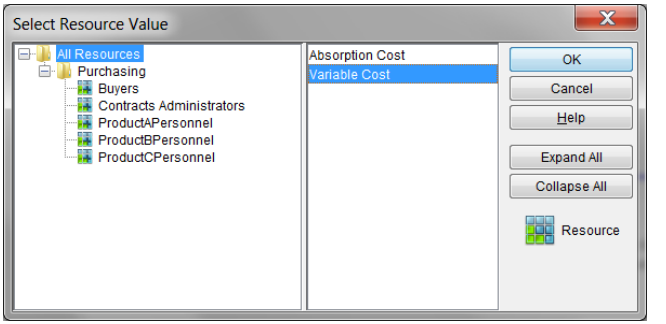
If no optimizations have been created for the model, **Select Optimization Settings** will be empty. The first step in defining an optimization is to give it a name. Enter a name in **Select Optimization Settings**. If there are already other optimization settings defined, entering a new name creates a new setting. To switch between settings, simply select the desired setting from **Select Optimization Settings**.



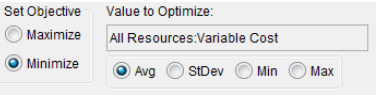
Once a name has been entered, continue defining the optimization. There are five tabs on the dialog: **Objective**, **Decision Variables**, **Constraints**, **Process Alternatives**, and **Run Options**. The tabs can be edited in any order.

Objective

The **Objective** tab sets whether to **Minimize** or **Maximize** and the value to optimize. The buttons on the right are used to select the value to optimize. **Entity**, **Resource**, **Activity**, **Connector**, **Time Stamp**, or **Model Attribute** values can be optimized. For the Purchasing demonstration model, the objective is to minimize the variable cost of all the resources. So **Minimize** was selected, then the **Resource** button was selected to choose the appropriate value to minimize.



On the left are all the resources defined in the model, plus an additional item named **All Resources**. When an item is selected on the left, the possible values are displayed on the right. In this example, **All Resources** is selected on the left and **Variable Cost** on the right. When **OK** is selected, **All Resources:Variable Cost** appears in **Value to Optimize**. (In the Inventory model, the **Value to Optimize** is **Model.InventoryPlusOrderTime:Value**.)



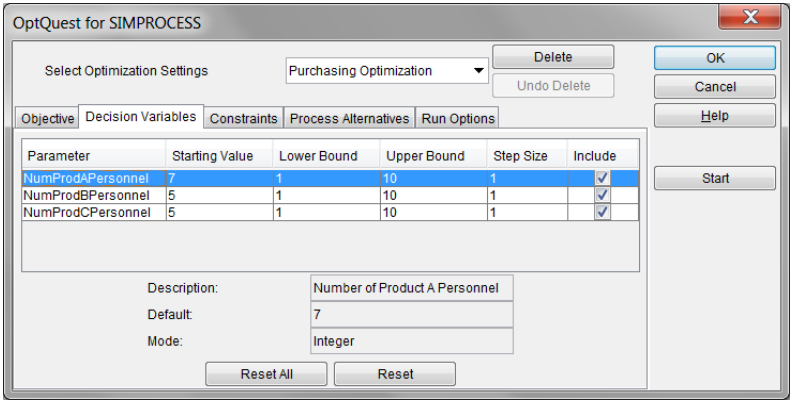
There are four options for the **Value to Optimize**: **Avg** (Average), **StDev** (Standard Deviation), **Min** (Minimum), and **Max** (Maximum). For this example, the average variable cost for all the resources is needed. Since stochastic (random) simulation models should be run for more than one replication, the variable cost of all resources for each replication is averaged.

Note about Min and Max: Min or Max should only be used when the constraints should bring convergence. For instance, if minimizing the maximum wait time was the goal, **Max** should be chosen for the **Value to Optimize**. Doing that, though, assumes there are constraints that would limit the maximum.

Decision Variables

The Decision Variables tab contains a table similar to the table on the Model Parameters dialog. This table lists all the model parameters in the model (no matter what **Mode**). However, only those model parameters whose **Mode** is Integer or Real are decision variables. The **Starting Value** is the value OptQuest will use on the first iteration. If new values are not entered, the **Starting Value** is the default value of the model parameter assigned at the time the attribute was defined. The last column of the table, **Include**, determines which decision variables OptQuest will modify during the optimization. The remaining columns, **Lower Bound**, **Upper Bound**, and **Step Size** apply only to model parameters whose **Mode** is Integer or Real. If the model parameter is not one of those modes, N/A will be entered in appropriate cell. Also, if the **Mode** is Real, N/A will be entered in the **Step Size** column of that row.

If **Lower Bound** is left blank, negative infinity is assumed. Likewise, if **Upper Bound** is left blank, infinity is assumed. The **Reset** and **Reset All** buttons work as on the Model Parameters dialog. These buttons only apply to the **Starting Value** column.



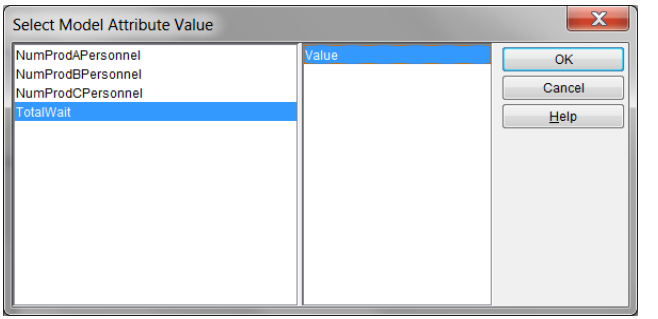
Constraints

The **Constraints** tab is where constraints are defined. (Constraints are not required for an optimization, but are usually used in an optimization.) Two types of constraints can be defined. The first type is a constraint on a performance measure of the model. The **Entity**, **Resource**, **Activity**, **Connector**, **Time Stamp**, and **Model Attribute** buttons work as on the **Objective** tab. These buttons allow performance measures to be selected as constraints. Unlike the **Objective**, though, which can only be one performance measure, more than one performance measure constraint can be added to the optimization. Also, on performance measure constraints, there is an **Upper Bound** and/or a **Lower Bound**. (Note: If there are no bounds for a performance measure, do not include it as a constraint.) In the Purchasing model there is a Model Attribute named `TotalWait`. This attribute represents the total amount of time all entity types (Product A, Product B, and Product C) wait for resources. The code below is from the **End Simulation** expression of the **Model Expressions** (**Define/Model Expressions**).

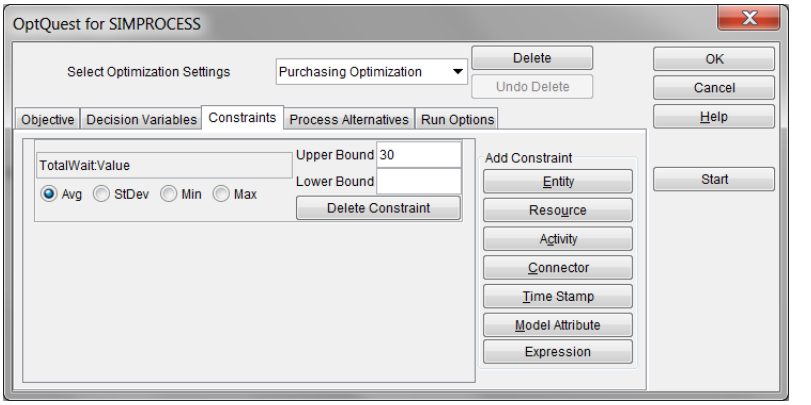
```
aWait, bWait, cWait : REAL;
aWait := GetEntityStatistic("Product A", "tokenwaitdelay", "Avg",
Replication);
bWait := GetEntityStatistic("Product B", "tokenwaitdelay", "Avg",
Replication);
cWait := GetEntityStatistic("Product C", "tokenwaitdelay", "Avg",
Replication);
Model.TotalWait := aWait + bWait + cWait;
```

The local variables `aWait`, `bWait`, and `cWait` contain the average wait time for Product A, Product

B, and Product C respectively. The Model Attribute `TotalWait` contains the sum of all the waits. Selecting the **Model Attribute** button brings up the list of the Model Attributes in the Purchasing model. `TotalWait` was selected, then `Value` was selected.



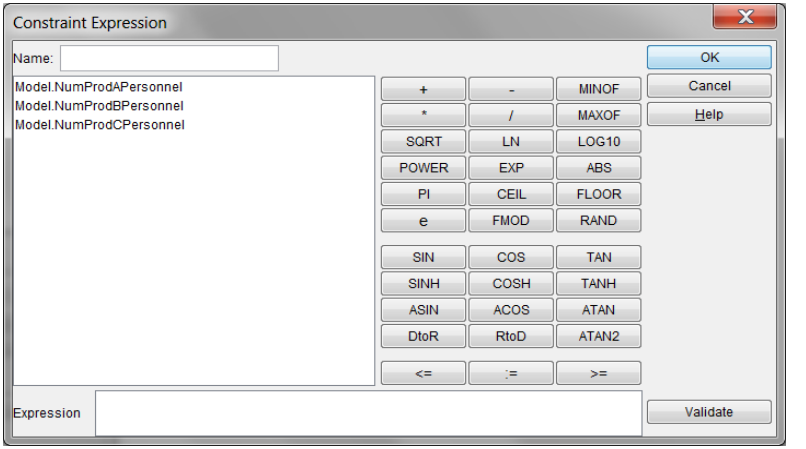
Performance measure constraints appear as the objective except an **Upper Bound** field, a **Lower Bound** field, and a **Delete Constraint** button are added.



As with the decision variables, a blank **Lower Bound** assumes negative infinity, and a blank **Upper Bound** assumes infinity. Once a performance measure constraint has been added, only the **Upper Bound**, **Lower Bound,** and type of statistic can be edited. If the type of value chosen needs to be changed, the constraint must be deleted and recreated.

The second type of constraint is an expression constraint. An expression constraint defines mathematical relationships between decision variables. The **Expression** button is used to create expression constraints. This button displays a dialog that lists the decision variables on the left and the allowed mathematical operators and functions on the right. The example below shows the Purchasing model. Note that the decision variable names must be the fully qualified SIMPROCESS

attribute name. That is, the type of the attribute (Model, Entity, etc.) must be included.



A name must be entered for the constraint in the **Name** field. Names for expression constraints within the same optimization setting must be unique. The expression can be typed directly into the **Expression** text area, or by clicking on the decision variable or mathematical operator or function. The syntax is the same as used in the SIMPROCESS Expression language. Some mathematical functions allowed by OptQuest are not allowed in the SIMPROCESS Expression language, therefore all of the functions allowed by OptQuest are listed in the following table along with descriptions.

Expression Constraint Functions

Function	Description
ABS(x)	Returns the absolute value of x.
ACOS(x)	Returns the arccosine of x in the range 0 to n radians. x is between -1 and 1.
ASIN(x)	Returns the arcsine of x in the range –n/2 to n/2 radians. x is between -1 and 1.
ATAN(x)	Returns the arctangent of x in the range of –p/2 to p/2 radians. If x is 0, atan returns 0.

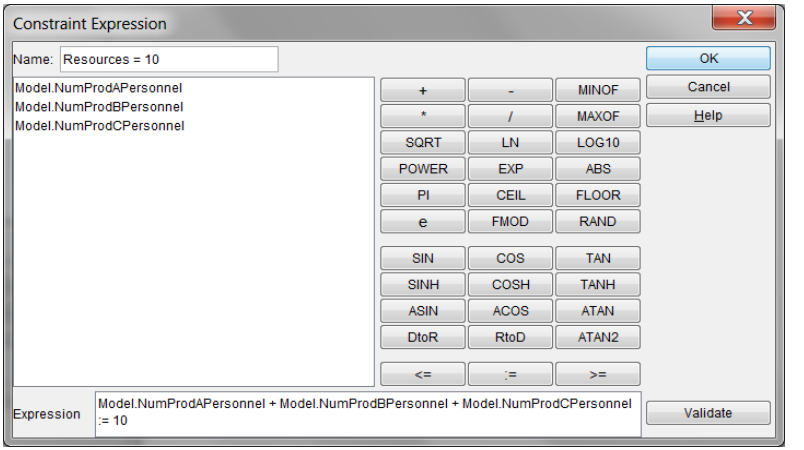
Expression Constraint Functions

Function	Description
ATAN2(x, y)	Returns the arctangent of y/x in the range –pi to pi radians. If both parameters of ATAN2 are 0, the function returns 0.
CEIL(x)	Returns the smallest whole number greater than or equal to x.
COS(x)	Returns the cosine of x, where x is an angle in radians.
COSH(x)	Returns the hyperbolic cosine of x, where x is an angle in radians.
DtoR(x)	Converts degrees to radians.
e	Mathematical constant e, approximately equal to 2.718.
EXP(x)	Returns *e* raised to the x power.
FLOOR(x)	Returns the largest whole number less than or equal to x.
FMOD(x, y)	Returns the remainder of x / y.
LN(x)	Returns the base *e* logarithm of x.
LOG10(x)	Returns the base 10 logarithm of x.
MAXOF(x, y)	Returns the larger of two numbers. (Note: Unlike the SIMPROCESS MAXOF, only two parameters are allowed.)

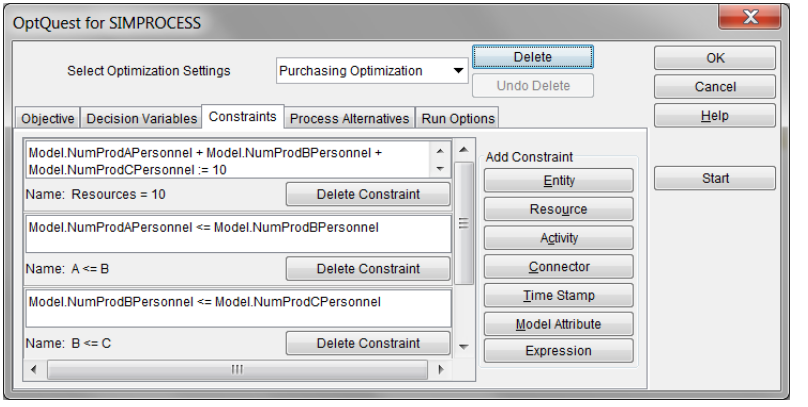
Expression Constraint Functions

Function	Description
MINOF(x, y)	Returns the smaller of two numbers. (Note: Unlike the SIMPROCESS MINOF, only two parameters are allowed.)
PI	Mathematical constant pi, approximately equal to 3.14159.
POWER(x, y)	Returns x raised to the y power.
RAND	Returns a random number between 0 and 1 inclusive.
RtoD(x)	Converts radians to degrees.
SIN(x)	Returns the sine of x, where x is an angle in radians.
SINH(x)	Returns the hyperbolic sine of x, where x is an angle in radians.
SQRT(x)	Returns the square root of x.
TAN(x)	Returns the tangent of x, where x is an angle in radians.
TANH(x)	Returns the hyperbolic tangent of x, an angle in radians.

The optimization setup for the Purchasing model is highly constrained. This is because of the three expression constraints. The first constraint is named **Resources = 10**. This constraint requires that the sum of the decision variables (model parameters) must equal 10. Since these decision variables set the number of units of each resource type, the name **Resources = 10** was assigned. The dialog has a **Validate** button. This button allows validation of the syntax of the expression constraint before selecting **OK**.



There are two other expression constraints in the Purchasing model, **A <= B** and **B <= C**. These constraints state that the number of units of the resource ProductAPersonnel must be less than or equal to the number of units of the resource ProductBPersonnel, and the number of units of the resource ProductBPersonnel must be less than or equal to the number of units of ProductCPersonnel. Scrolling down on the **Constraints** tab brings all three expression constraints into view.

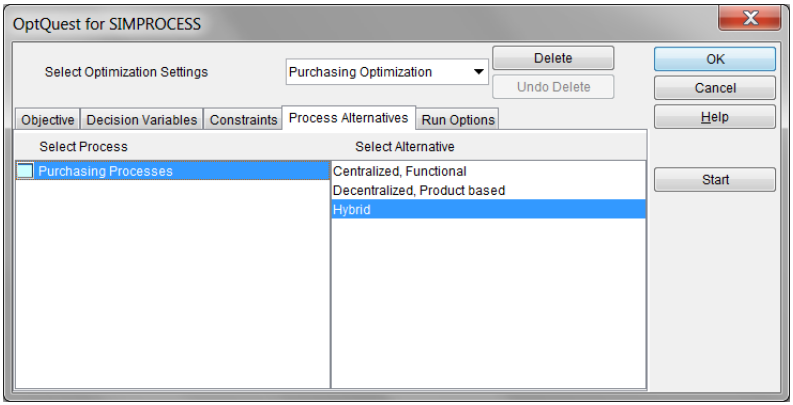


Expression constraints can be modified from the Constraints tab by clicking in the text area and changing as necessary. However, expression constraint validation is not available. To use validation, the constraint must be deleted and recreated.

Process Alternatives

The **Process Alternatives** tab displays all processes in the model that have more than one alternative.

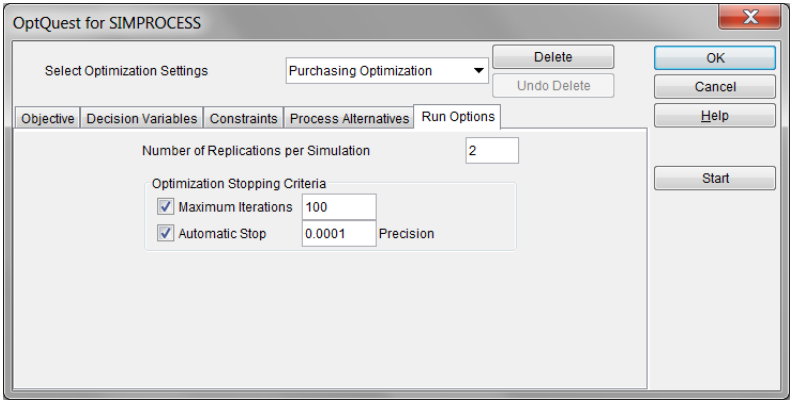
When a process is selected on the left, the alternatives for that process are displayed on the right. This allows the alternative to be selected that will be active for the optimization. If no alternative is selected for a process, the alternative that was active the last time the model was saved will be active for the optimization. If a Master Process (Master Processes) has multiple alternatives, the copies of the Master Process will not be displayed. The selected alternative for the Master Process will be applied to all copies. In the Purchasing model, the Purchasing process is the only process that has more than one alternative, thus, it is the only process listed on the left. The Purchasing process' three alternatives are listed on the right.



Run Options

The Run Options tab sets the number of replications for each simulation and the optimization stopping criteria. Unless the model is discrete, it is recommended that each simulation be run for more than one replication. The number of replications needed should be determined before setting up the optimization. If no value is entered, each simulation run will run with the number of replications last saved in the model. If a value is entered, that value will be reflected in the **Number of Replications** in the **Run Settings** of the model the next time the model is opened after the optimization. In the Purchasing model, the **Number of Replications per Simulation** is set to 2. This was set to a small number for demonstration purposes only. Only two replications and the tight constraints cause the Purchasing optimization to run quickly.

Apart from user intervention, the length of the optimization is controlled by two options: **Maximum Iterations** and **Automatic Stop**. At least one of the stopping criteria must be selected. Both options can be selected. If both are selected, the criteria that is satisfied first will stop the optimization. **Maximum Iterations** is the maximum number of iterations OptQuest will attempt before stopping the optimization. The value must be an integer greater than zero. When **Automatic Stop** is selected, the optimization stops when the value of the objective stops improving. **Precision** determines when two objective values are considered equal. The default is 0.0001.

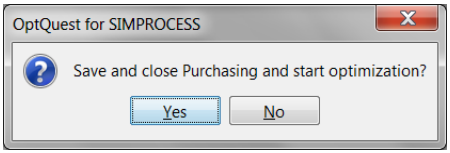


Saving Optimization Settings

Selecting **OK** will close the dialog and add the defined optimization settings to the model. The model must be saved for the optimization settings to persist. Selecting **Start** will close the dialog, add the defined optimization settings to the model, and save the model. Thus, performing another save is not necessary. **Cancel** closes the dialog and discards any changes.

Running an Optimization

To run an optimization, if the optimization settings dialog is not already displayed, select **OptQuest** from the **Tools** menu. Once the optimization settings dialog has appeared, either define an optimization or select a previously defined optimization. Select the **Start** button. At that point a dialog will appear asking for confimation before saving the model and starting the optimization. If **No** is selected, the operation is canceled. If **Yes** is selected, the dialog closes, the model is saved and closed, and the optimization starts.

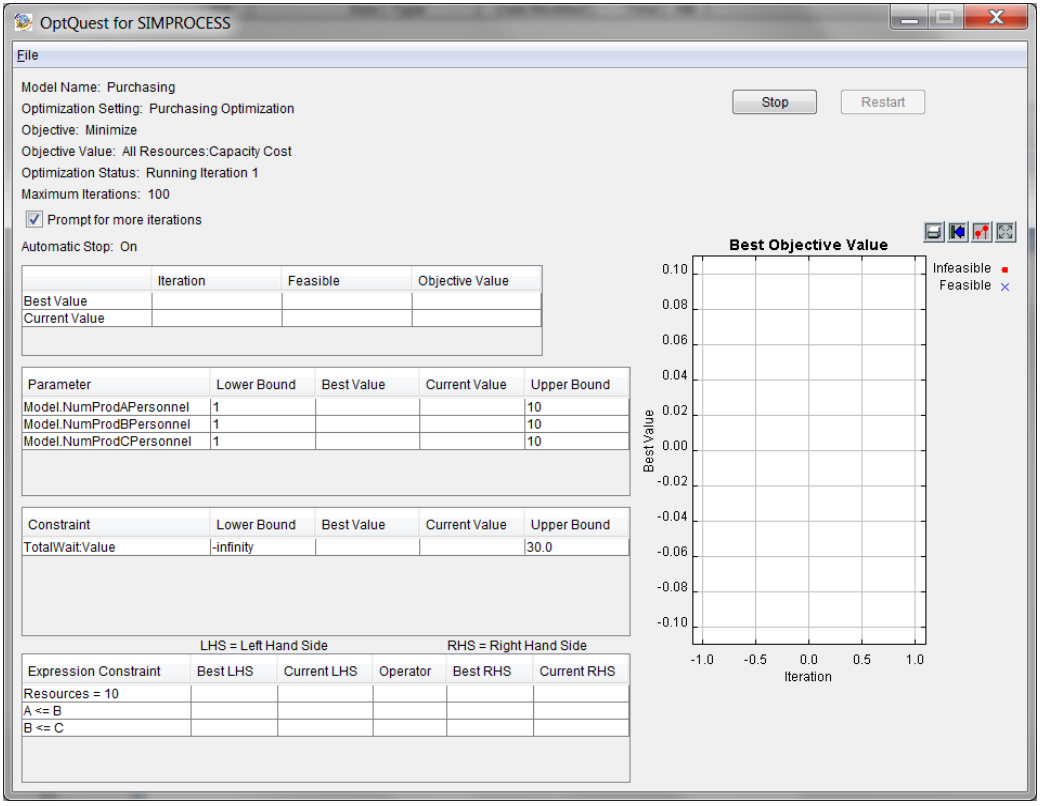


OptQuest for SIMPROCESS Interface

An OptQuest for SIMPROCESS optimization runs in a separate Java Virtual Machine (JVM) and uses a separate interface from SIMPROCESS. Therefore, once an optimization has started, SIMPROCESS can continue to be used without affecting the optimization. This means multiple optimizations can run simultaneously. However, that may not be practical, since running an optimization is very CPU intensive (since it is running a simulation, which monopolizes the CPU), and, depending on the size of the model, memory intensive. Also, it may not be practical to run a simulation with SIMPROCESS while an optimization is running for the same reasons. Running multiple optimizations and/or simulations will often perform better on non-Windows systems.

The OptQuest for SIMPROCESS interface primarily consists of text, tables, and a graph to monitor the optimization. There is a button to stop the optimization, and there are reporting options on the **File** menu once the optimization is complete.

When the interface appears, all status objects are empty except for the **Optimization Status** field, which displays **Initializing**. The tables show the status of the objective value, the decision variables, the performance measure constraints, and the expression constraints. The objective value table and the decision variables table will always appear. The performance measure constraints table and the expression constraints table will only appear if those types of constraints exist. The graph plots the best objective value based on whether or not the best is feasible. The example below is from the Purchasing model.



Status Text

The following status text is displayed:

- **Model Name** - name of the model being optimized
- **Optimization Setting** - the name of the optimization setting selected when the optimization was started
- **Objective** - Minimize or Maximize
- **Objective Value** - the performance measure selected for optimization
- **Optimization Status** - Initializing, Running Iteration n, Complete, Stopping Optimization, User Terminated, or Error
- **Maximum Iterations** - number of iterations or Off
- **Automatic Stop** - On or Off

Optimization Status descriptions:

- **Initializing** - Optimization is in setup stage
- **Running Iteration n** - Optimization is running the *n*th iteration
- **Optimal Solution Found** - Optimization completed normally with no errors
- **Stopping Optimization** - Optimization is in the process of stopping because user selected **Stop** button
- **User Terminated** - Optimization stopped by user
- **Error** - Optimization stopped due to error.

Status Tables

The objective value table lists the **Best Value** and the **Current Value** for the **Iteration**, whether the best and current values are **Feasible**, and the best and current **Objective Value**.

	Iteration	Feasible	Objective Value
Best Value	4	Yes	34375.58873
Current Value	7	No	33132.9172

The decision variables table lists the decision variables (**Parameters**) included in the optimization. The **Lower Bound**, **Best Value**, **Current Value**, and **Upper Bound** are displayed.

Parameter	Lower Bound	Best Value	Current Value	Upper Bound
Model.NumProdAPersonnel	1	2	2	10
Model.NumProdBPersonnel	1	2	3	10
Model.NumProdCPersonnel	1	6	5	10

The performance measure constraints table lists the performance measures identified as constraints for the optimization. The **Lower Bound**, **Best Value**, **Current Value**, and **Upper Bound** are displayed.

Constraint	Lower Bound	Best Value	Current Value	Upper Bound
TotalWait:Value	-infinity	5.73741	31.50034	30.0

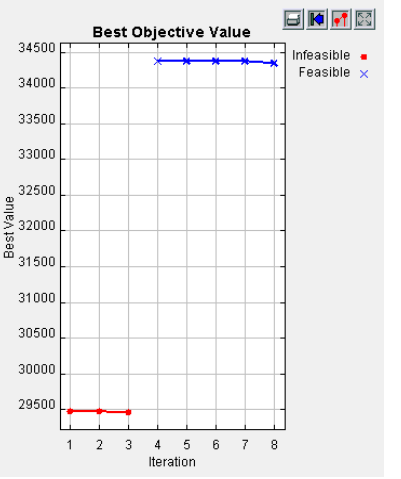
The expression constraint table lists the expression constraints defined for the optimization. The name, not the expression itself, is displayed in the **Expression Constraint** column. The other columns are **Best LHS**, **Current LHS**, **Operator**, **Best RHS**, and **Current RHS**. **LHS** stands for **Left Hand Side**, and **RHS** stands for **Right Hand Side**. These columns show the best and current values for both sides of each expression constraint. The **Operator** column shows the comparison operator between both sides of each expression constraint.

Expression Constraint	Best LHS	Current LHS	Operator	Best RHS	Current RHS
Resources = 10	10.0	10.0	=	10.0	10.0
A <= B	2.0	2.0	<=	2.0	3.0
B <= C	2.0	3.0	<=	6.0	5.0

Objective Graph

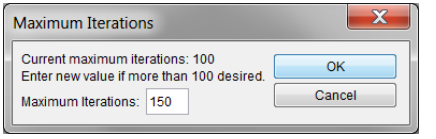
The graph of the objective value plots the best objective value for each iteration. Infeasible values are plotted with dots in red, and feasible values are plotted in blue with Xs. In the example below from the Purchasing model, the first five values were infeasible and the remaining feasible. Remember, once the best value is feasible, the best value will not be infeasible again. To zoom in on a portion of the graph, click and drag over the portion of the graph to enlarge. The buttons above the graph do the following:

- Print the plot.
- Reset the X and Y ranges to their original (empty plot starting) values.
- Set the plot format.
- Rescale the plot to fit the data. Used after zooming to return to full view of plot.



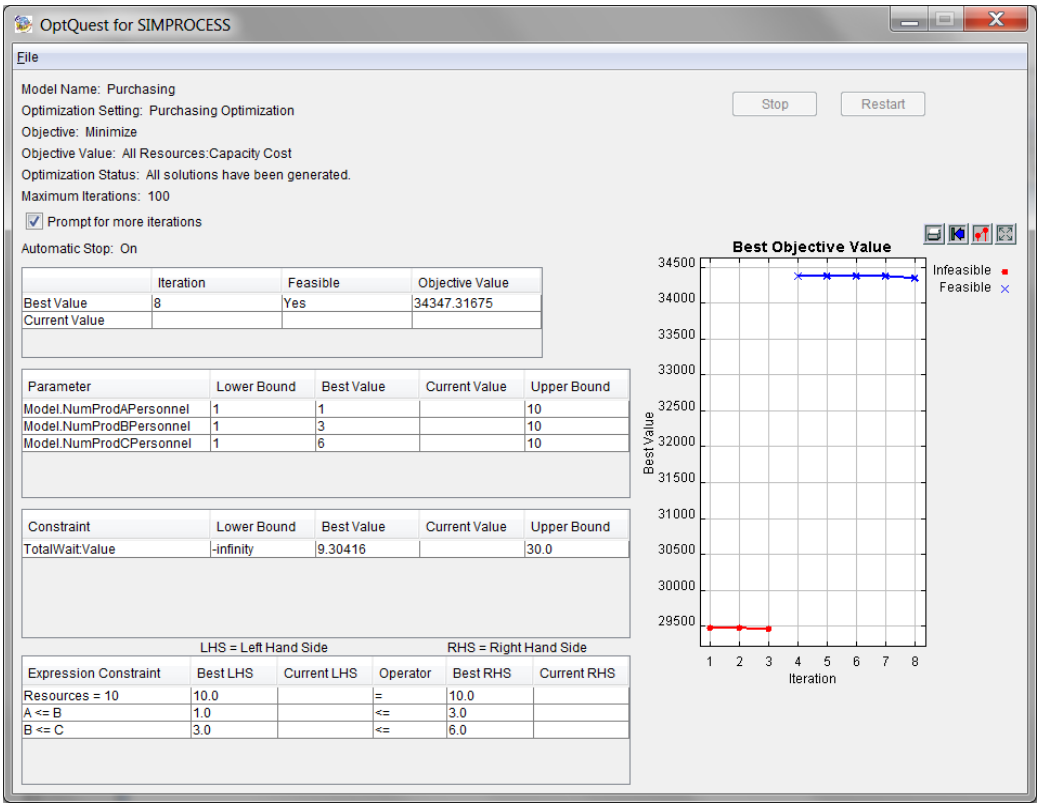
Maximum Iterations Prompt

When a maximum number of iterations has been set for the optimization, the checkbox **Prompt for more iterations** is enabled. This checkbox defaults to selected. When selected, the optimization prompts for more iterations at the end of the next to the last iteration. For instance, if the maximum iterations is set to 100, at the end of iteration 99 the optimization will prompt for a new maximum for the iterations. A new maximum must be entered, not the number of additional iterations. So, for this example, the number entered must be 101 or larger (150 entered). If **Prompt for more iterations** remains selected, the optimization will prompt again when the iteration is one less than the maximum (149 in this example). Deselect this option to have the optimization complete without interruption.

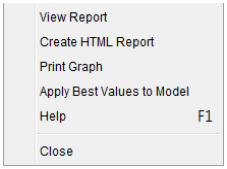


Optimization Solution Options

The OptQuest for SIMPROCESS interface shows the current status of the optimization while it is running. Once the optimization is complete, the interface only shows the best values. The example below is from the Purchasing model.

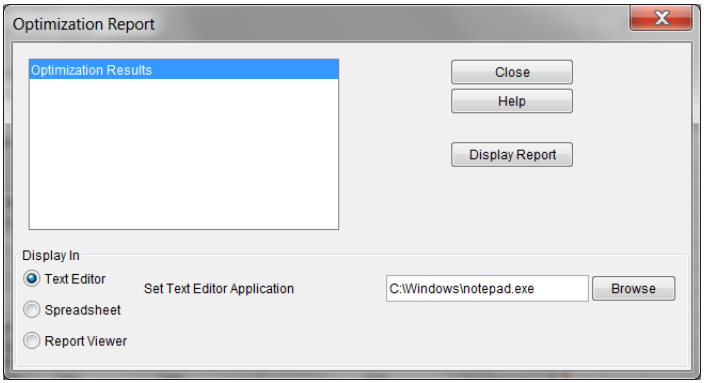


The **File** menu contains other options for examining solutions, and it has an option for applying the best solution to the model. **Help** is also accessed from the **File** menu.



View Report

View Report brings up a dialog similar to the **Standard Report** dialog. With this dialog the report can be displayed in a text editor, a spreadsheet, or the Report Viewer (see "Report Viewer" on page 120).



The report lists the name of the model, the start date/time and end date/time of the optimization, the optimization settings, and the best values for the objective, decision variables, and constraints.

Create HTML Report

This option creates the same report as View Report. However, since it is an HTML file, the objective graph is included as well.

Print Graph

This option sends the objective graph to the selected printer.

Apply Best Values to Model

Apply Best Values to Model assigns the best values from the decision variables to the model parameters in the model. Thus, when the model is reopened in SIMPROCESS, and run is selected, the run values for the model parameters reflect the best values from the optimization. Also, if **OptQuest** is selected again, the starting values for the decision variables are the best values from the optimization.

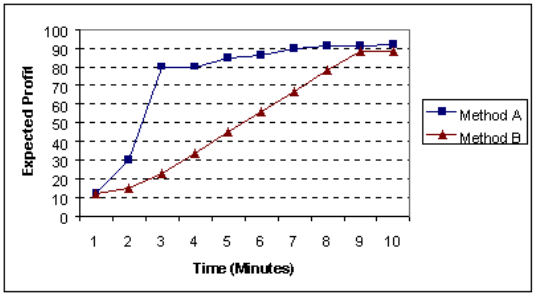
Tips and Suggestions

Search Methodology

There are many factors that influence the performance of OptQuest. For example, consider two optimization methods, A and B, applied to a problem with the objective of maximizing expected returns. When evaluating the performance of each method, consider which method satisfies the following criteria:

- Finds a solution with a larger expected return
- Jumps to the range of high-quality solutions faster

Below is the Performance Graph for the two hypothetical methods.



The figure above shows that, although both methods find solutions with a similar expected profit after 10 minutes of searching, method A jumps to the range of high-quality solutions faster than B. For the criteria listed above, method A performs better than method B.

OptQuest will obtain performance profiles similar to method A. OptQuest's search methodology is very aggressive and attempts to immediately find high-quality solutions, causing large improvements, (with respect to the initial solution), early in the search. This is critical when OptQuest can perform only a limited number of simulations.

However, several factors affect OptQuest's performance, and the importance of these factors varies from one situation to another. This section reviews these factors and offers tips and suggestions on how to achieve maximum performance.

Factors that affect search performance

Any heuristic method for solving problems cannot guarantee that it will find the optimal solution. It

might only find a solution that is close to the optimal solution, usually referred to as the best solution; this is why maximizing performance is critical.

The following is a list of relevant factors that directly affect search performance. Each factor is explained in sections after the list.

- Number of decision variables
- Initial values
- Decision variable bounds
- Complexity of the objective
- Constraints
- Feasibility
- Number of replications and simulations
- Simulation accuracy
- Simulation speed

Number of Decision Variables

The number of decision variables greatly affects OptQuest's performance. OptQuest has no physical limit on the number of decision variables that can be used in any given problem. However, the performance might deteriorate if more than 100 decision variables are used.

Also, as the number of decision variables increases, more simulations are needed to find high-quality solutions. General guidelines for the minimum number of simulations required for a given number of decision variables in a problem are:

Decision Variables	**Minimum number of simulations**
Less than 10	100
Between 10 and 20	500
Between 20 and 50	2,000
Between 50 and 100	5,000

For very large numbers of decision variables, try this procedure:

- Decrease the number of replication per simulation, at least initially
- Run the optimization to get an approximate solution
- Set the suggested values to the approximate solution
- Further restrict the bounds on the decision variables

- Increase the number of replications to increase accuracy
- Rerun the optimization

One option is to de-select certain decision variables and optimize the rest. If an optimization has already been run, there might be information available about which decision variables have the least effect on the objective function. When one or more decision variables is de-selected and the optimization is rerun, the search focuses on the remaining, more important, decision variables.

Initial Values

The initial values are the values listed as the **Starting Values** of the **Decision Variables** tab. Initial values are important because the closer they are to the optimal value, the faster OptQuest can find the optimal solution. If the initial values are constraint-infeasible, they are ignored.

For potentially large models with many decision variables, it might be helpful to first run a simplified version of the optimization to find initial values for the full-blown model. For example, expected values could be used for some of the random variables in the model.

Decision Variable Bounds

OptQuest's performance can be significantly improved by selecting meaningful bounds for the decision variables. Suppose, for example, that the bounds for three decision variables (X, Y, and Z) are:

```
0 <= X <= 100
0 <= Y <= 100
0 <= Z <= 100
```

And in addition to the bounds, there is the following constraint:

```
10*X + 12*Y + 20*Z <= 200
```

Although the optimization model is correct, the decision variables bounds are not meaningful. A better set of bounds for these decision variables would be:

```
0 <= X <= 20
0 <= Y <= 16.667
0 <= Z <= 10
```

These bounds take into consideration the values of the coefficients and the constraint limit to determine the maximum value for each decision variable. The new "tighter" bounds result in a more efficient search for the optimal values of the decision variables. However, this efficiency comes at the expense of missing the optimal solution if it lies outside the specified bounds.

Complexity of the Objective

A complex objective has a highly nonlinear surface with many local minimum and maximum points.

OptQuest is designed to find global solutions for all types of objectives, especially complex objectives. However, for more complex objectives, generally it is required to run more simulations to find high-quality global solutions.

Since only one value can be optimized, complex objectives must be defined within the model expressions. (See "Customizing a Model with Attributes and Expressions".)

Constraints

Constraints can be used to restrict the values of decision variables (model parameters) by defining relationships among the decision variables; constraints can also restrict the value of output variables (performance measures).

If a constraint is defined using only decision variables, OptQuest can eliminate sets of decision variables values that are constraint-infeasible before it runs the simulation. Limiting the optimization by defining constraints on decision variables is extremely time-effective.

If a constraint contains a performance measure, a simulation must be run to determine whether the suggested solution satisfies the constraint.

The search process benefits from the use of constraints on decision variables and tight bounds on decision variables. However, performance generally suffers when performance measure constraints are included in the optimization model for two reasons:

- Performance measure constraints are very time-consuming to evaluate, since OptQuest must run an entire simulation before determining whether the results are constraint-infeasible.
- To avoid running constraint-infeasible simulations, OptQuest must identify the characteristics of solutions likely to be constraint-feasible; this makes the search more complex and requires more time.

Even though performance measure constraints can greatly decrease the number of feasible simulations performed during an optimization, performance measure constraints can focus the search to effectively rule out undesirable solutions.

If there are lots of performance measure constraints that OptQuest can't easily satisfy, consider combining the output constraints into one multi-objective function.

Feasibility

OptQuest makes finding a feasible solution its highest priority. Once it finds a feasible solution, it then

concentrates on finding better solutions.

The fact that a particular solution may be infeasible does not imply that the problem itself is infeasible. However, infeasible problems do exist. For example, suppose that in a Job Shop problem a foreman insists on finding an optimal configuration with the following constraints:

```
drills + grinders <= 4
drills + grinders >= 5
```

Clearly, there is no combination that will satisfy both of these constraints.

Or, for this same example, suppose the bounds for a decision variable were:

```
3 <= saws <= 5
```

And a constraint was:

```
saws <= 2
```

This also results in an infeasible problem. OptQuest will display an error message and terminate infeasible optimizations.

Infeasible problems can be made feasible by fixing the inconsistencies of the relationships modeled by the constraints. OptQuest detects optimization models that are constraint-infeasible and reports them.

If a model is constraint-feasible, OptQuest will always find a feasible solution and search for the optimal solution (i.e., the best solution that satisfies all constraints).

Number of Replications and Simulations

When OptQuest runs an optimization, it runs a simulation to evaluate each set of decision variable (model parameter) values. Therefore, the quality of the optimization results depends on the number of simulations (iterations) and the number of replications per simulation.

For a set period of time, the number of replications per simulation is inversely related to the number of simulations; as one increases, the other decreases. Decreasing the number of replications can help increase the number of simulations.

The more simulations OptQuest can run, the more sets of values it can evaluate, and the more likely OptQuest is to find a solution close to the optimal solution.

Simulation Accuracy

There are two factors that affect simulation accuracy:

- Number of replications per simulation

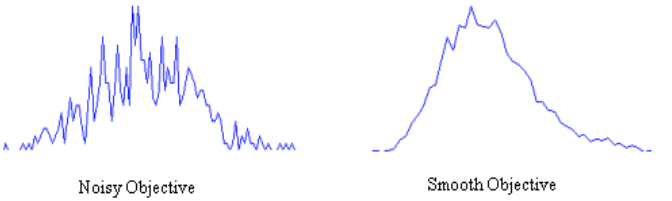
- Noisiness of the objective

Number of replications per simulation

For sufficient accuracy, the number of replications per simulation must be set to the minimum number necessary to obtain a reliable estimate of the objective function being optimized; this minimum number is typically found with empirical testing.

Objective noisiness

Noisiness can also affect the accuracy of OptQuest results.



The objective on the left has significant amounts of noise caused by the probability distributions used to model the problem's uncertainty. For these types of objectives, OptQuest might have trouble discerning the minimum or maximum value. Noisy functions can be detected by watching for best solutions that seem to "bounce around" from one set of values to completely different sets of values. To help solve this problem, increase the number of replications per simulation. On the right, the objective appears smooth due to the relative certainty in the model assumptions. In these cases, OptQuest should quickly converge to the best solution.

Simulation Speed

Some suggestions for increasing speed include:

- Reduce the size of the model (or the time horizon of the simulation).
- Increase the system's RAM memory.
- Reduce the number of uncertain elements in the simulation.
- Close other applications.

OptQuest Demonstration Models

Two of the demonstration models that come with SIMPROCESS have been set up for optimization: the Purchasing model (Purchasing.spm) and the Inventory model (Inventory.spm).

Purchasing Model

This model was used throughout this chapter to describe how to set up and run an OptQuest optimization. Thus, an extended description will not be given here. The model is described in Chapter 5 of the *SIMPROCESS Getting Started Manual* (Demonstration and Reference Models). The optimization set up for this model should not be considered a true optimization scenario. The constraints are very tight, and the number of replications per simulation is only two. This was done so the example would run within a minute or two. The value of this model is in learning how to create decision variables and constraints.

Inventory Model

This model was developed for the purpose of finding optimal values. The model is described in Chapter 5 of the *SIMPROCESS Getting Started Manual* (Demonstration and Reference Models). Briefly, the Inventory model demonstrates an Inventory Pull and Manufacturing system. The process is characterized by the Reorder Points and Reorder Quantities defined for each resource in the supply chain. There are four steps in the supply chain: Warehouse, Assembly, Component1 Vendor and Component2 Vendor, and the Raw Material Vendors. Inventory is pulled only when it is needed (there is insufficient stock to fill the order or the Reorder Point has been reached). There are 10 Model Attributes in the model that have been designated as model parameters.

- Comp1ReOrderPt - Component Vendor 1 (Comp1 resource) Reorder Point
- Comp1ReOrderQty - Component Vendor 1 (Comp1 resource) Reorder Quantity
- Comp2ReOrderPt - Component Vendor 2 (Comp2 resource) Reorder Point
- Comp2ReOrderQty - Component Vendor 2 (Comp2 resource) Reorder Quantity
- FinProdReOrderPt - Finished Product (FinProduct resource) Reorder Point
- FinProdReOrderQty - Finished Product (FinProduct resource) Reorder Quantity
- Raw1ReOrderPt - Raw Material Vendor 1 (Raw1 resource) Reorder Point
- Raw1ReOrderQty - Raw Material Vendor 1 (Raw1 resource) Reorder Quantity
- Raw2ReOrderPt - Raw Material Vendor 2 (Raw2 resource) Reorder Point
- Raw2ReOrderQty - Raw Material Vendor 2 (Raw2 resource) Reorder Quantity

The goal is to minimize the amount of inventory held without impacting the customer negatively

(increase order cycle time). This can be done by finding the optimal Reorder Points and Quantities for each node in the supply chain. Using OptQuest the optimal Reorder Points and Reorder Quantities can be found. Thus, the decision variables for the optimization are the model parameters, and the default values for the attributes are the **Starting Values** for the decision variables. (See "Decision Variables" on page 48.)

The objective is twofold: minimize inventory (FinProduct resource) and minimize the amount of time it takes to fill a customer order. See "Preparing for Optimization," beginning on page 45 for a discussion on how these dual objectives are combined into one objective.

The performance measure for the amount of inventory and the performance measure for the amount of time it takes to fill a customer order are used as performance measure constraints. These values are obtained using the `GetResourceStatistic` and `GetEntityStatistic` expression statements in the **End Simulation** expression for the model (**Define/Model Expressions**). These values are assigned to Model Attributes, which are used as constraints.

The number of replications per simulation is set to 1, the maximum number of iterations is 150, and the automatic stop feature is turned on with a precision of 0.1. Note that the number of replications is set to 1 to reduce the time required to run the optimization. In normal usage, the number of replications would be greater than one.

This optimization takes from 20 minutes to a few hours to run, depending on system memory and processor speed.

CHAPTER 5

SIMPROCESS Dashboards

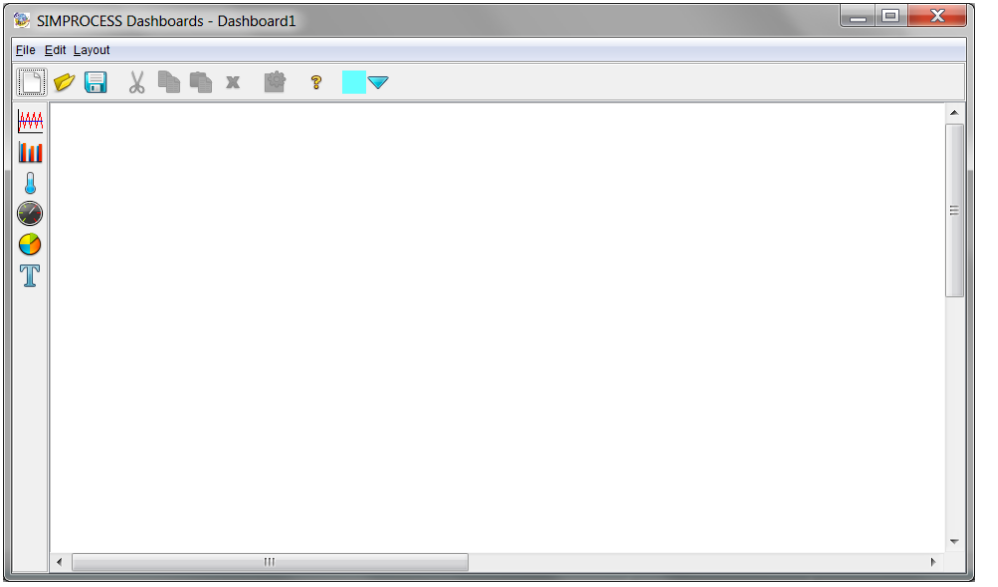
SIMPROCESS Dashboards are collections of dynamic graphs that can be displayed locally or remotely by a Dashboard Server. The graphs contained by Dashboards can be of the same type or of differing types.

Dashboards are defined independent of SIMPROCESS models. Thus, a Dashboard can be used with multiple models, or multiple models may use a single Dashboard. To use a Dashboard with a model, the Dashboard must be Assigned to a model. The Assign process links the Dashboard to the model, indicates the location of the Dashboard Server for display (host or IP address and port), and sets the values that will be displayed on the graphs defined for that Dashboard.

During a simulation, Dashboards can only be displayed by using a Dashboard Server. A copy of the Dashboard file must be located with the Dashboard Server.

Defining Dashboards

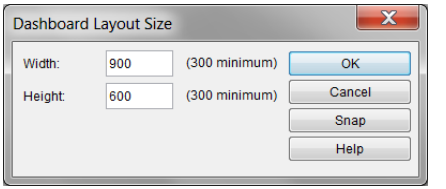
Dashboards are defined independent of SIMPROCESS models. Thus, no model needs to be open to define a Dashboard. The **Report** menu contains a **Dashboard** item with a submenu that has two items: **Define...** and **Assign...**. Selecting **Define...** brings up the SIMPROCESS Dashboard graphical user interface (GUI).



The Dashboard GUI is similar to the SIMPROCESS GUI. The primary differences are that only one Dashboard can be open at a time, and the Dashboard GUI initially opens with an empty Dashboard named **Dashboard1**.

Dashboard Layout

The Dashboard layout is similar to the layout of a SIMPROCESS model in that its size can be set (**Layout/Size** menu item), and its background color can be set (**Layout/Background Color** menu item). Selecting **Layout/Size** brings up the following dialog.



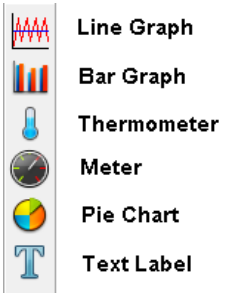
Each Dashboard has a size denoted by a gray line in the editor window. This size, measured in pixels, will be used by the Dashboard Server to build a window in which to display the Dashboard and its Graphs. The width and height values must be at least 300 pixels each but are otherwise unlimited in size. However, when determining the size, consideration should be given to the size of the display on which the server will run when a Dashboard is displayed. The **Snap** button will automatically adjust the width and height values to the smallest values required to contain all Graphs currently in the Dashboard. **Snap** will not resize the layout to smaller than 300 x 300 pixels.

Selecting **Layout/Background Color** sets the color of the layout to the color displayed on the color button on the System Toolbar. The arrow to the right of the color button brings up a color chooser to select the desired color.

Dashboard Graphs

The Graph Toolbar on the left contains the graph types that can be added to the layout. The types of graphs that can be displayed on a Dashboard are

- Line Graph
- Bar Graph (Horizontal or Vertical, 2D or 3D)
- Meter
- Thermometer
- Pie Chart (2D or 3D)
- Text Label



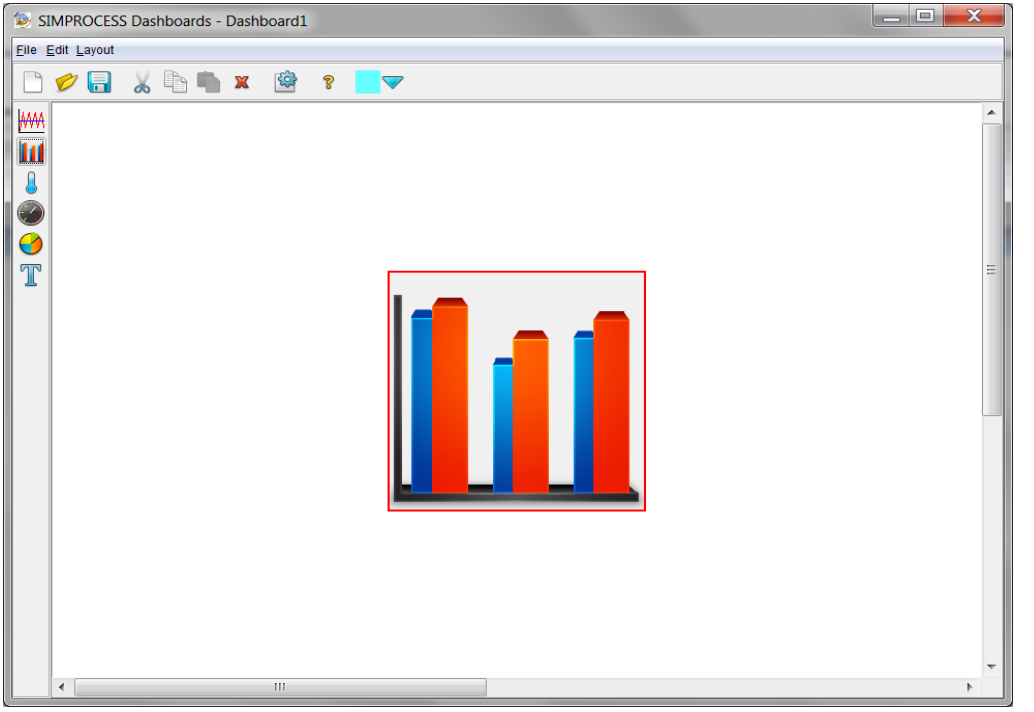
The Line Graph displays one or more X-Y series where the X axis represents simulation time. The Bar Graph displays one or more values. The Thermometer and Meter display one value each. The Pie Chart displays two or more values. The Text Label can be static, or it can display one value dynamically.

Note that the values to be displayed are not assigned while defining Dashboards. The properties that can be set while defining are appearance and

location related properties. The properties set while defining a Dashboard are considered to be the default properties of the Dashboard. Most appearance properties (not location) can be changed when using a Dashboard with a model.

Adding a Graph

Selecting a graph on the Graph Toolbar causes a new panel that represents a graph of that type to appear in the center of the layout. The graph itself does not appear on the layout. Once on the layout, the graph panel can be moved and resized. Also, actual properties of the graph represented by the graph panel can be set. The new graph panel is automatically selected. A graph that is selected has a red border around it. Multiple graphs can be selected by holding down the Control key when clicking on graph panels. Multiple graph panels can also be selected by clicking and dragging a selection rectangle around the desired panels.



Moving a Graph

To move a graph, click on the graph panel and drag while holding down the mouse button. Multiple graph panels can be moved simultaneously by selecting multiple graphs before moving. The same type grid that is available for creating SIMPROCESS models is available for Dashboards on the **Layout**

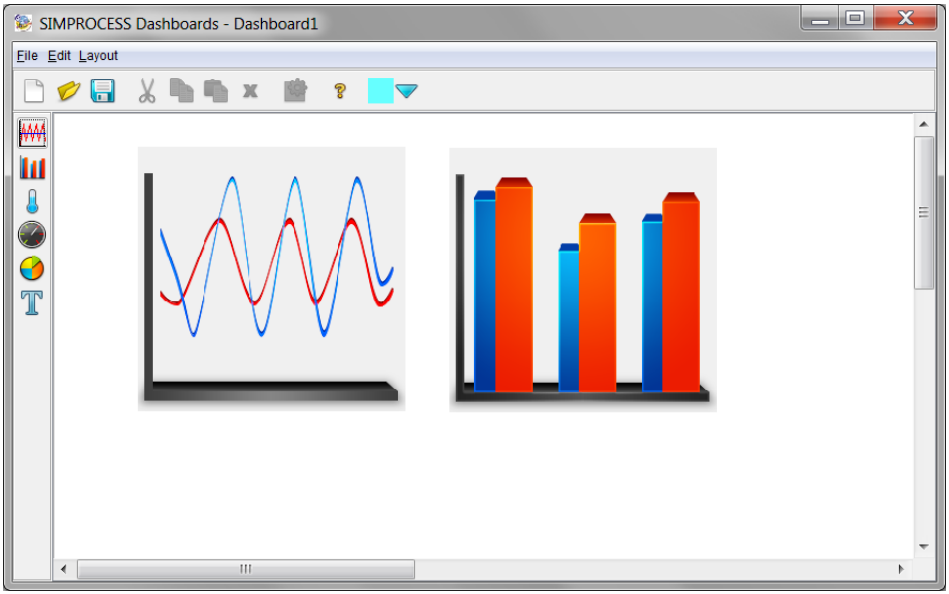
menu. However, due to the size of the graph panels, movement can be difficult with the grid turned on.

Resizing a Graph

The default size of a graph panel (other than Text Label, which is 250 x 100 pixels) is 250 x 250 pixels. A graph panel can be resized by clicking on the edge of a graph panel and dragging. Also, a specific size can be set by selecting **Edit/Resize** from the menu or by right mouse clicking on the graph panel and choosing **Edit/Resize**. Multiple graph panels can be resized simultaneously by selecting multiple graphs before resizing.

Aligning and Distributing Graphs

When two or more graph panels are selected, they can be aligned using the **Layout/Align** menu item or by right clicking on a selected graph panel and choosing the appropriate align item. When three or more graph panels are selected, they can be distributed horizontally, vertically, or circularly using the **Layout/Distribute** menu item or by right clicking on a selected graph panel and choosing the appropriate distribute item. The image below shows a Dashboard with a Line Graph and a Bar Graph that have been moved from the center, resized to 350 x 350 pixels, and aligned.



Editing Graph Properties

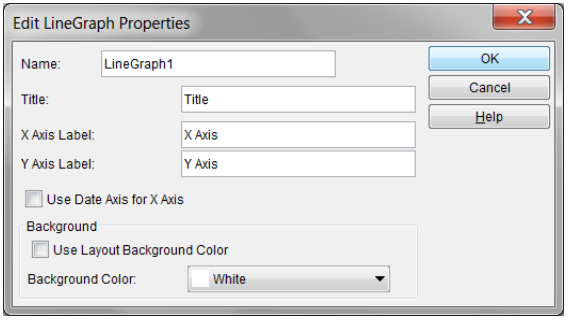
The default properties for graphs can be edited by double clicking a graph panel, selecting a panel and choosing **Edit/Properties** from the menu, selecting a panel and clicking the properties button on the

System Toolbar, or by right clicking and choosing **Properties**. Properties common to all graphs are **Name** and **Background**. The **Name** is used to identify the graph to the Dashboard and must be unique. **Background** sets the color of the background of the graph. A specific color can be selected from the **Background Color** list, or **Use Layout Background Color** can be selected. **Use Layout Background Color** sets the background color of the graph to background color of the layout. Note that **Name** and **Background** cannot be changed when a Dashboard is assigned to a model.

Line Graph Properties

Title, **X Axis**, **Y Axis**, and **Use Date Axis for X Axis** are additional properties for the Line Graph. These default properties can be overridden when the Dashboard is assigned to a model.

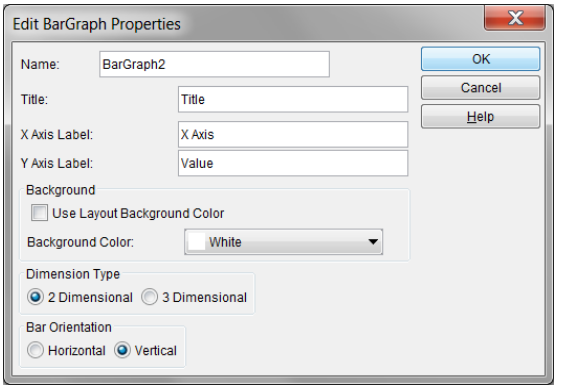
- **Title** - default title for the Line Graph
- **X Axis** - label for the x axis
- **Y Axis** - label for the y axis
- **Use Date Axis for X Axis** - if selected, the x axis displays times and dates instead of a number



Bar Graph Properties

Title, **X Axis**, **Y Axis**, **Dimension Type**, and **Bar Orientation** are additional properties for the Bar Graph. These default properties can be overridden when the Dashboard is assigned to a model.

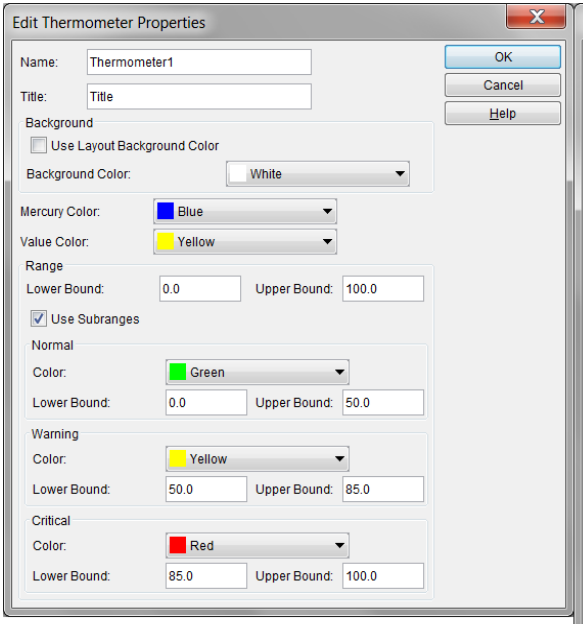
- **Title** - default title for the Bar Graph
- **X Axis** - label for the x axis
- **Y Axis** - label for the y axis
- **Dimension Type** - **2 Dimensional** or **3 Dimensional**
- **Bar Orientation** - **Horizontal** or **Vertical**

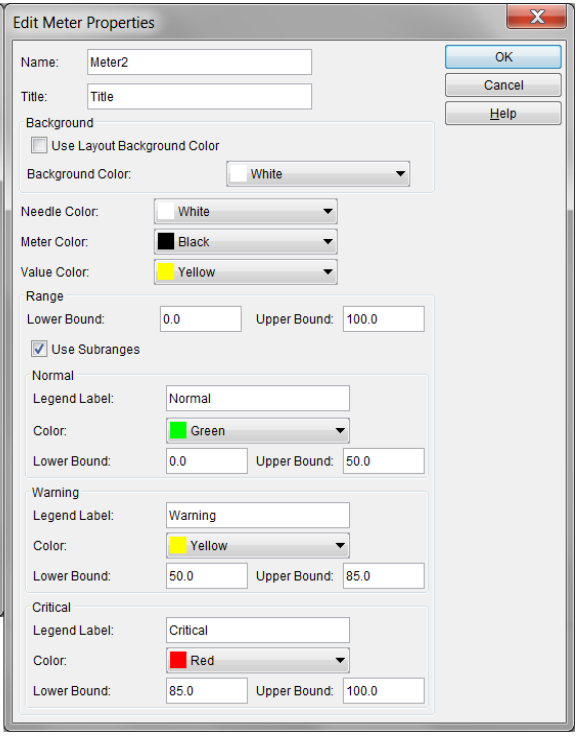


Meter and Thermometer Properties

The Meter and Thermometer have similar properties. Both have a **Title**, a **Range** with optional subranges, and a **Value Color**. The differences are in the other color options. The Meter has a **Needle Color** and a **Meter Color** option. The Thermometer has a **Mercury Color** option.

- **Title** - default title for the Meter or Thermometer
- **Mercury Color** - default color for the mercury in the Thermometer
- **Needle Color** - default color for the needle on the Meter
- **Meter Color** - default color for the Meter
- **Value Color** - default color for the current value of the Meter or Thermometer
- **Lower Bound** - default lower bound for the Meter or Thermometer
- **Upper Bound** - default upper bound for the Meter or Thermometer
- **Use Subranges** - if selected, lower bounds, upper bounds, and colors for three subranges (**Normal**, **Warning**, and **Critical**) can be set. If subranges are used for the Thermometer, the colors selected for the subranges override the **Mercury Color** setting.
- **Normal Lower Bound** - default lower bound for the **Normal** subrange
- **Normal Upper Bound** - default upper bound for the **Normal** subrange
- **Normal Color** - default color for the **Normal** subrange
- **Warning Lower Bound** - default lower bound for the **Warning** subrange
- **Warning Upper Bound** - default upper bound for the **Warning** subrange
- **Warning Color** - default color for the **Warning** subrange
- **Critical Lower Bound** - default lower bound for the **Critical** subrange
- **Critical Upper Bound** - default upper bound for the **Critical** subrange
- **Critical Color** - default color for the **Critical** subrange

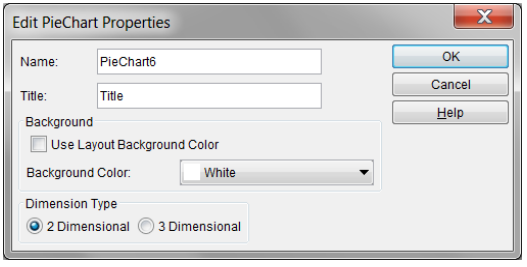




Pie Chart Properties

Title and **Dimension Type** are additional properties for the Pie Chart. These default properties can be overridden when the Dashboard is assigned to a model.

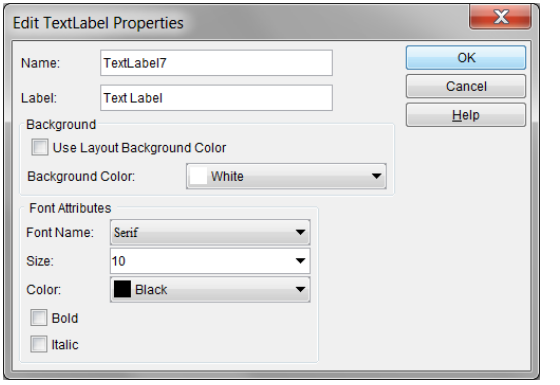
- **Title** - default title for the Pie Chart
- **Dimension Type** - **2 Dimensional** or **3 Dimensional**



Text Label Properties

Text Label properties include **Label** and **Font Attributes,** which include **Font Name**, **Size**, **Color**, **Bold**, and **Italic**.

- **Label** - default text that is displayed
- **Font Name** - default font for label
- **Size** - default font size for label
- **Color** - default font color for label
- **Bold** - default selection for bold
- **Italic** - default selection for italic



Dashboard Menus

File Menu

New places a new Dashboard in the work area displayed in the window. If the current Dashboard has been changed, SIMPROCESS will prompt to save its contents to a file before clearing it in favor of a new Dashboard in the work area.

Open prompts for the selection of a Dashboard file (having the extension ".spd"), which will be read and displayed in the work area. If the current Dashboard has been changed, SIMPROCESS will prompt to save its contents to a file before clearing it in favor of a new Dashboard created from the selected file.

Save saves the current work area into its Dashboard file. If no file has yet been associated with it, this behaves the same as **Save As**.

Save As saves the current Dashboard to a new file. If an existing file is selected, SIMPROCESS will

question whether to overwrite that file.

Preview will be enabled when the Dashboard in the work area contains one or more Graphs. It will open a separate window with the specified layout size (**Layout/Size**) containing actual sample graphs configured with the properties set in each graph panel. This is how the actual Dashboard will look when shown by a Dashboard Server.

Close closes the SIMPROCESS Dashboards window. If the current Dashboard has been changed, SIMPROCESS will prompt to save those changes.

Edit Menu

Cut is enabled only if one or more Graphs are selected (as indicated by a red border). **Cut** removes the selected Graphs from the layout and places them onto the clipboard, where they may later be pasted back into the current or a different Dashboard.

Copy is enabled only if one or more Graphs are selected (as indicated by a red border). **Copy** places a copy of the selected Graphs onto the clipboard, where they may later be pasted back into the current or a different Dashboard.

Paste is enabled only if one or more Graphs has been placed (via **Cut** or **Copy**) onto the clipboard. **Paste** places Graphs from the clipboard onto the layout. The clipboard's contents remain intact until changed by a subsequent **Cut** or **Copy** action.

Duplicate is enabled only if one or more Graphs are selected (as indicated by a red border). **Duplicate** acts as though **Copy** was selected followed by **Paste**. That is, it copies all selected Graphs onto the clipboard and then immediately pastes them back into the current layout.

Clear is enabled only if one or more Graphs are selected (as indicated by a red border). **Clear** removes the selected Graphs from the layout and does not place them onto the clipboard.

Select All is enabled only if one or more Graphs appear on the layout. Select All selects all Graphs on the layout and displays red borders to indicate their selected status.

Resize is enabled only if one or more Graphs are selected (as indicated by a red border). **Resize** presents a Resize dialog where the width and height of the selected Graphs are specified in pixels. If more than one Graph is selected, the size entered will be applied to all of them. Individual Graphs may also be resized by selecting them and moving the mouse cursor over a portion of the border, then dragging when the cursor changes to indicate the direction of the resizing operation.

Properties is enabled only if one or more Graphs are selected (as indicated by a red border). **Properties** presents a Graph Properties dialog for each selected Graph in turn. Properties can also be edited by double-clicking any Graph, clicking the properties button on the System Toolbar, or right clicking and choosing **Properties**.

Layout Menu

Size is always enabled since a Dashboard is considered to be open in the work area at all times. **Size** displays the Dashboard Layout Size dialog.

Background Color is always enabled since a Dashboard is considered to be open in the work area at all times. **Background Color** sets the background color of the Dashboard layout to the currently selected color on the Dashboard Color button on the System Toolbar.

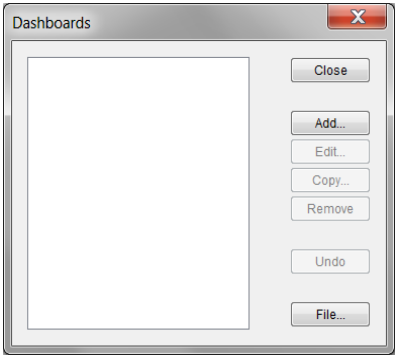
Align is enabled only if two or more Graphs are selected (as indicated by a red border). **Align** presents a dialog with **Top**, **Bottom**, **Left**, **Right**, **Center (Vertical)** and **Center (Horizontal)**. Selected Graphs will be repositioned so that they align as directed. Alignment can also be set by right clicking on a selected Graph.

Distribute is enabled only if three or more Graphs are selected (as indicated by a red border). **Distribute** presents a dialog with **Vertically**, **Horizontally** and **Circular**. Selected Graphs will be repositioned so that they are distributed within the available space (the layout size) as directed, much like the equivalent function in the SIMPROCESS model editor. Selected Graphs can also be distributed by right clicking on a selected Graph.

Snap to Grid, **Grid Lines**, **Grid Spacing**, and **Grid Color** work like their counterparts in the SIMPROCESS model editor. They allow the optional use of grid lines to help in placement of Graphs in a Dashboard. However, due to the size of graph panels, movement is difficult with grid lines turned on.

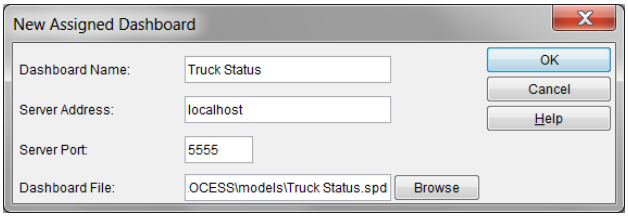
Assigning Dashboards

To assign a Dashboard to a model, select **Dashboard/Assign...** from the **Report** menu. For the **Assign...** menu item to be active, a SIMPROCESS model must be open, and the Dashboard GUI must be closed. **Assign...** brings up a list of the Dashboards assigned to the model. Use the **File...** button to update the Dashboard file's location if the Dashboard was developed on a different system or moved to a location other than where it was located when originally assigned.



Adding a Dashboard Assignment

Dashboards are assigned to a model by using the **Add...** button on the Dashboard list dialog. The New Assigned Dashboard Dialog has four fields: **Dashboard Name**, **Server Address**, **Server Port**, and **Dashboard File**. The **Dashboard Name** defaults to "Dashboard" plus an internally assigned number. This name can be changed to something more meaningful. The **Server Address** defaults to localhost (which refers to the same machine on which SIMPROCESS is running), and **Server Port** defaults to 5555. If not using localhost for the **Server Address**, the **Server Address** must be a fully qualified server name or an Internet Protocol (IP) address. Use the **Browse** button to locate the **Dashboard File** that is to be displayed. (**TIP:** Whenever practical, the file should be in the same directory with the SIMPROCESS model, or in a directory below the one where the SIMPROCESS model resides. This allows the stored reference to be relative, so that moving both the SIMPROCESS model and the Dashboard file to a new location where their relative positions are unchanged will still maintain a correct reference.) Note that **Add...** does not complete the assignment of a Dashboard to a model. The Graphs must be edited as described below to customize the way they appear and plot values to them.

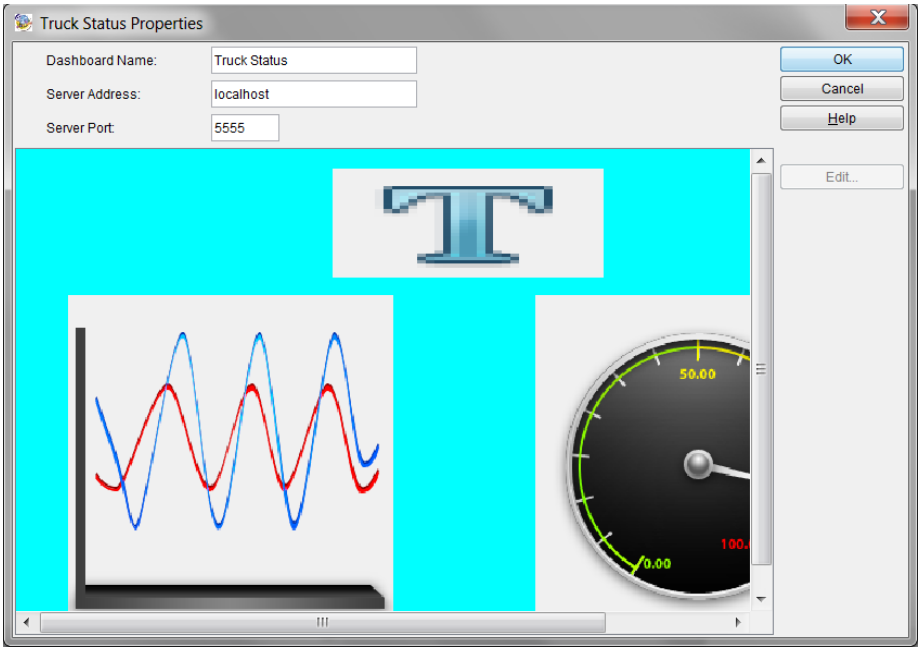


Editing and Removing a Dashboard Assignment

Selecting a Dashboard assignment in the list causes the **Edit** and **Remove** buttons to activate.

Remove removes the Dashboard assignment from the model. A Dashboard assignment that has been removed can be restored with the **Undo** button. **Edit** displays a Dashboard Edit dialog that allows customizing of the Dashboard for the model and assignment of plot values to any of the graphs on the Dashboard. The **File** button presents a dialog that allows selecting the Dashboard file for the selected Dashboard assignment and should normally only be needed if its location has changed.

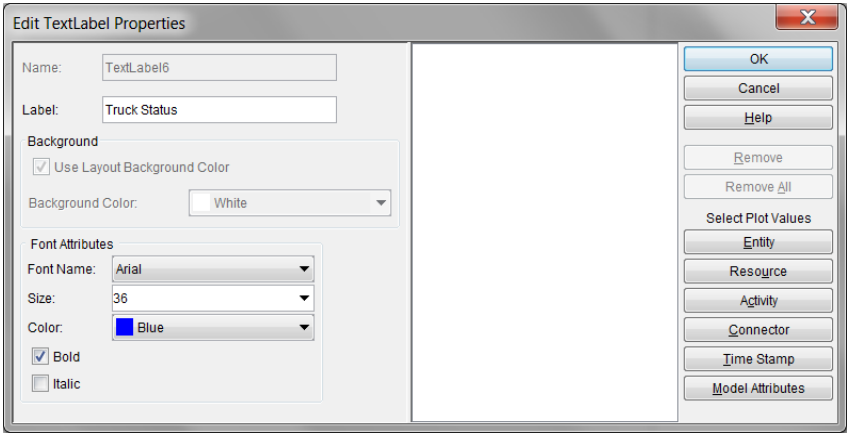
The properties dialog contains three fields that can be edited: **Dashboard Name**, **Server Address**, and **Server Port**. Below those fields is a view of the Dashboard that was assigned to the model. The example below contains a Text Label, Line Graph, and Meter.



Selecting a Graph causes the **Edit** button to activate. **Edit** displays a dialog with the default properties of the Graph. The Graph properties can also be accessed by double clicking the Graph. These properties (other than **Name** and **Background**) can be customized for the model. Also, the values to be plotted are added to the Graph.

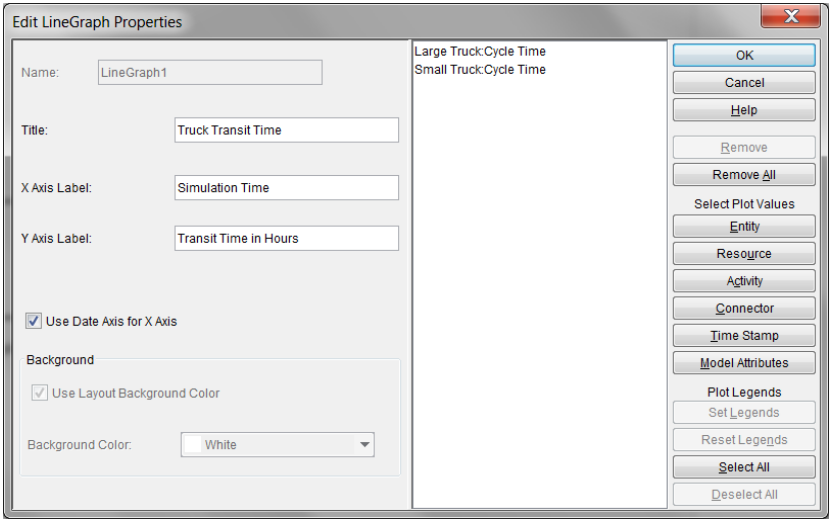
Customizing Properties

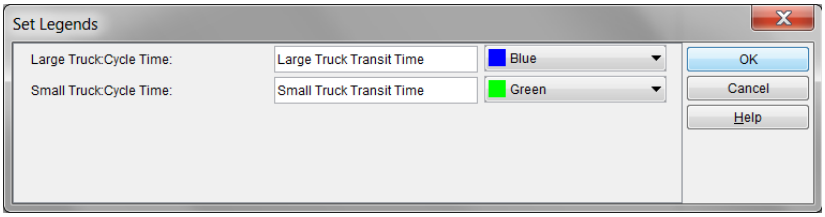
Editing the Text Label brings up the dialog below. The left side of the dialog contains the same properties that are editable when defining Dashboards. Note that **Name** and **Background** cannot be changed. Any changes made in this dialog will override the default properties in the Dashboard file when the Dashboard is displayed. The properties of the other Graphs are customized in the same manner. The right side of the dialog contains the value to display. Since the right side is empty, this Text Label is a static label. Although Text Labels can display one value dynamically, they are best used as static labels. (**Important Note:** If you are planning to have multiple models referring to the same Dashboard while it is displayed, you can prevent any Graphs your model does not use from receiving an initialization message at simulation time which could change their appearance at the Dashboard Server. For a Text Label, make sure the **Label** value is empty and do not select any plot values. For all other Graphs, do not select any plot values.) Adding values to Graphs is discussed below.



Adding Values to a Graph

Values are added to a graph using the buttons under the **Select Plot Values** heading (**Entity**, **Resource**, **Activity**, **Connector**, **Time Stamp**, and **Model Attribute**). The procedure for adding values is the same as for Custom Plots. See "Adding Values To Custom Plots" for detailed instructions on adding values. The Line Graph, Bar Graph, and Pie Chart can plot multiple values. The Meter, Thermometer, and Text Label can only plot one value. The Line Graph, Bar Graph, and Pie Chart also allow the text and color of the legend to be set. The Thermometer allows the text of the legend to be set, but not the color. This is because the color is set in the Thermometer properties. The procedure for setting legends is the same as for Custom Plots. See "Setting Legends" for detailed instructions on setting legends. The images below show the properties of the Line Graph with the legend setting below that.

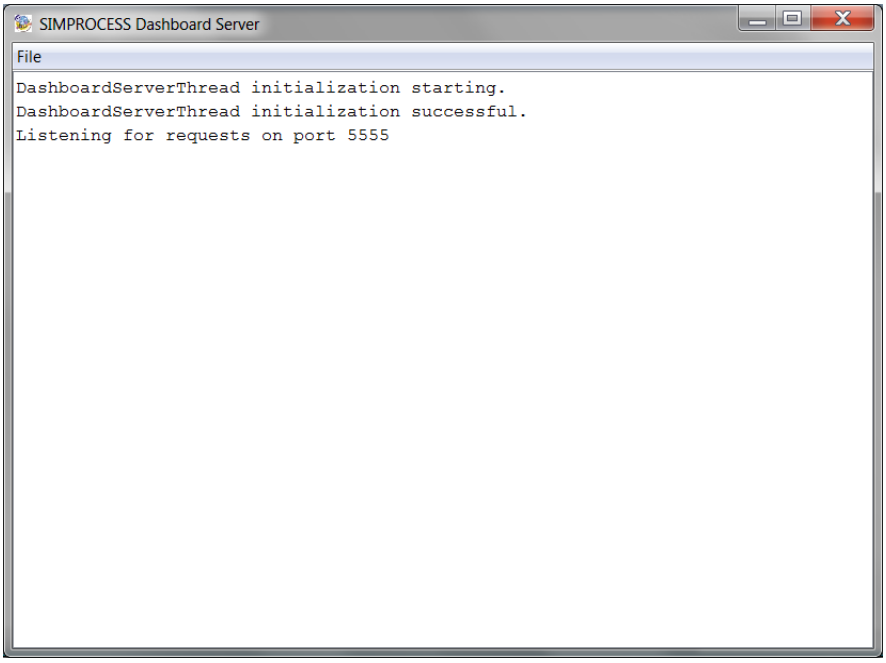

Edit LineGraph Properties
Name:
LineGraph1
Title:
Truck Transit Time
X Axis Label:
Simulation Time
Y Axis Label:
Transit Time in Hours
Use Date Axis for X Axis
Background
Use Layout Background Color
Background Color:
White
Large Truck:Cycle Time
Small Truck:Cycle Time
OK
Cancel
Help
Remove
Remove All
Select Plot Values
Entity
Resource
Activity
Connector
Time Stamp
Model Attributes
Plot Legends
Set Legends
Reset Legends
Select All
Deselect All


Set Legends
Large Truck:Cycle Time:
Large Truck Transit Time
Blue
Small Truck:Cycle Time:
Small Truck Transit Time
Green
OK
Cancel
Help

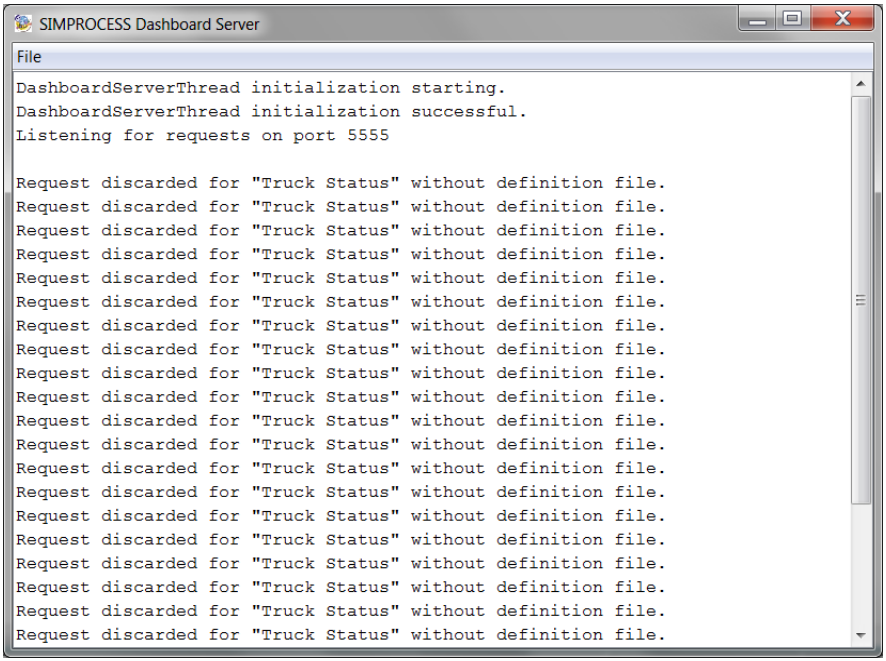
Displaying Dashboards

Dashboards are displayed by a Dashboard Server. A directory named `dashboardserver` is located in the SIMPROCESS directory. A copy of the contents of this directory must be located on each system that is to run a Dashboard Server. Within the `dashboardserver` directory is a directory named `dashboards`. A copy of the Dashboard files (`.spd` files) assigned to the model must be placed in the `dashboards` directory used by a Dashboard Server for it to be displayed. To use a directory other than `dashboards`, modify the `server.properties` file to reference the alternate directory. The port for the Dashboard Server can also be changed in this file. Multiple Dashboard Servers on the same system can use the same `dashboards` directory, but they must have different ports. **Note:** There is one exception to this last item. Multiple Dashboard Servers on one system may listen on the same port if subscribing to multicast groups, even if all subscribe to the same one.

If there are multiple Dashboard assignments in the model with different server addresses and/or ports, a Dashboard Server must be started on each **Server Address** and **Server Port** referenced (or subscribe to the multicast group). If the **Server Address** for any assigned Dashboards is `localhost`, a Dashboard Server can be started from SIMPROCESS by selecting **Tools/Remote/Start Local Dashboard Server**. On Windows, the `server.bat` file can be used to start a Dashboard Server. The `server.sh` file is for Linux systems, though it will also work on other Unix systems. Note that these files specify a path to the JRE installed with SIMPROCESS. If the system does not have a version 1.6 or later JRE, the JRE installed with SIMPROCESS (in the `jre` directory) can be copied if it has the same or a compatible operating system, or a JRE can be downloaded from java.sun.com for most platforms, or from the appropriate vendor for others. The Dashboard Server must be initialized before the simulation begins. When started, the Dashboard Server window appears.



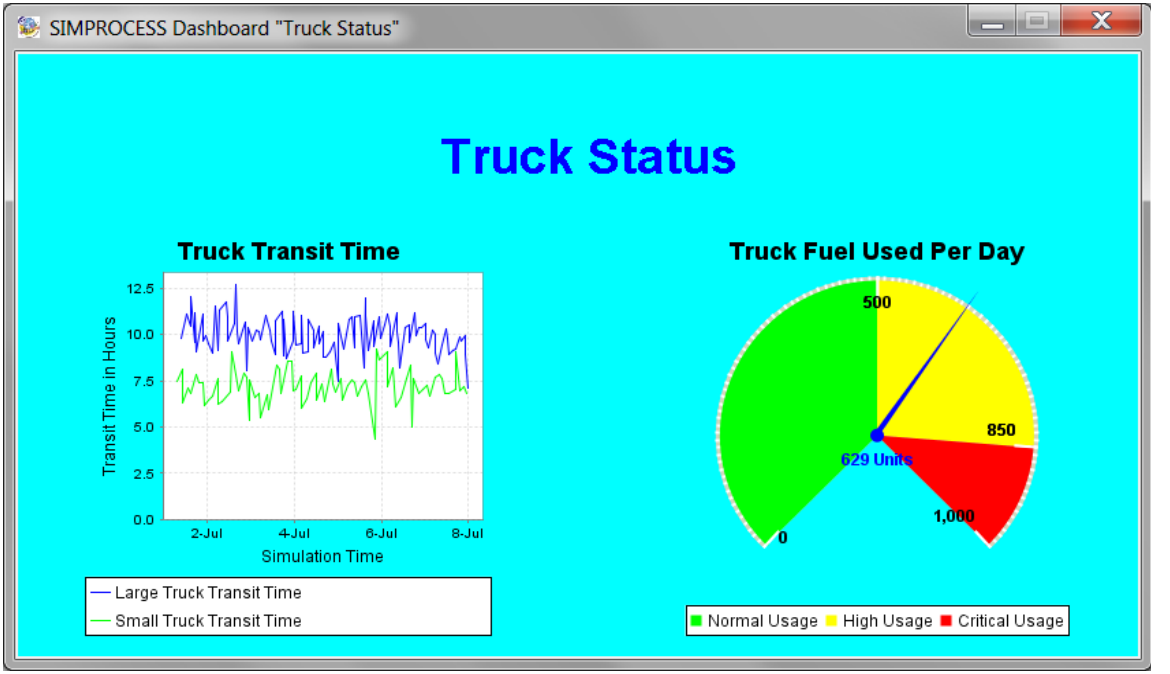
When the simulation is started, the Dashboards assigned to the model will initialize if there are any plot values assigned to any Graphs (for a Text Label, it must also have a **Label** value as noted above). Note that the Dashboard Server window may need to be the active (i.e., frontmost) window for the Dashboard windows to appear if running on the same Windows system as SIMPROCESS. If the Dashboard files referenced by the model are not located in the `dashboards` directory on the server, no Dashboard window will appear, and error messages will be displayed in the Dashboard Server window.



Note that it is possible for messages to not be received by the Dashboard Server, or to be received out of order. To minimize this risk (and improve performance), it is suggested that Dashboards not display a large (more than 10) number of values on a large number of Graphs (more than 4). For example, one Line Graph can plot multiple values (no limit as far as the Line Graph is concerned). However, this means that messages for 10 different values must be received by the Dashboard Server. Alternatively, there could be 10 Meters which each take one value. Again, 10 different values must be received by the Dashboard, and the Dashboard must update 10 different Graphs. If possible, keep the number of values to 10 or less on 4 or less Graphs. Also, it is suggested that the simulation be run with the animation turned on. This slows down the simulation. Simulations that are run with the animation off run extremely fast, so that messages sent to Dashboards could very well arrive faster than the Dashboard Server can handle them (resulting in packet losses). It's usually not necessary to turn on entity movement (although that helps). Turning on Activity Counts normally slows the simulation enough to minimize loss of messages to the Dashboard Server. Running a Dashboard Server across a network can sometimes impose sufficient delay to reduce the chances of message overload as well. This is the recommended way to use Dashboards. These suggestions may or may not be applicable to your environment, depending on a wide range of factors. You should experiment in your own environment as needed.

The image below shows the Truck Status Dashboard during a run. Note that a Dashboard window cannot be closed. Closing the Dashboard Server's main window closes all associated Dashboard

windows and shuts down the server.



CHAPTER 6

Model Bundles

A model bundle file contains a SIMPROCESS model along with the model's directory and any associated Metamodels (*SIMPROCESS Metadata Manual*), OrgModels (*SIMPROCESS OrgModel Manual*), and Dashboards (SIMPROCESS Dashboards). Model bundle files facilitate sharing SIMPROCESS models since the bundle, in most instances, will contain all the files needed to display and run the model.

The **Bundle** menu item on the **File** menu contains two options: **Create** and **Extract**. **Create** builds a model bundle file, and **Extract** extracts a model bundle file and opens the enclosed model.

Bundle File Contents

What's Included

A model bundle file contains a SIMPROCESS model along with its directory. The bundled model directory will contain the complete contents of the original model directory. In addition, the model bundle file may contain the following:

- Assigned Metamodels
- Assigned OrgModels
- Assigned Dashboards
- Files referenced by the model that were not in the original model directory

Metamodels are required to be in the `metamodel` directory. Therefore, since Metamodels cannot be in the model directory, a copy of any Metamodel assigned to the SIMPROCESS model being bundled will be included in the model bundle file.

OrgModels are required to be in the `orgmodel` directory. Therefore, since OrgModels cannot be in the model directory, a copy of any OrgModel assigned to the SIMPROCESS model being bundled will be included in the model bundle file.

Dashboard files are not required to be in a particular location. (However, a copy of each Dashboard file must be in the Dashboard Server's `dashboards` directory when it runs.) Typically, Dashboard files are located in the same directory as the model and not the model directory. Again, if not already in the model directory, a copy of each Dashboard file referenced by the model will be included in the model bundle file.

The model directory is automatically created when a new SIMPROCESS model is saved for the first time or when a model is opened and the model directory does not exist. The purpose of the directory is to store files needed by the model and provide a default location for the model to create files as necessary (for example, plots and the Standard Report). Although placing files required by the model in the model directory is strongly encouraged, it is not required. Thus, during the bundling process, the model is examined to determine if files are referenced that are not in the model directory. The following items are examined for external (non-model directory) file references:

- File schedule event files (See "Defining Specific Entity Generation Events in a File" for more information on event files.)
- Spreadsheet schedule files (See "Adding a Spreadsheet Schedule" for more information on Spreadsheet schedules.)
- File entries (**URL/File** field) on the **Documentation** tab of Activity properties dialog (See "Common Activity Input Fields" for more information.)

- External expression files (See "Expression Editor" for more information on external expression files.)
- Files referenced in the `OpenFile`, `OpenSpreadsheet`, and `OpenDatabase` expression statements. (See "Methods OpenFile, CloseFile, ReadFromFile, and WriteToFile" for more information on `OpenFile`. See "Spreadsheet System Methods" for more information on `OpenSpreadsheet`. See "Interfacing With A Database" for more information on `OpenDatabase`.)

If any of the above reference files are not located in the model directory, copies of the files are placed in the model directory of the model bundle file, and the location references of the files in the model are changed to the model directory. This is done since it cannot be guaranteed that the computer where the model bundle file is extracted will have the same directory structure. The original model file is not modified. Only the copy of the model file that will go in the bundle file is changed. Note that only files opened for `Input` will be copied to the model directory. Files opened for `Output` will only have the file references changed to the model directory.

What's Excluded

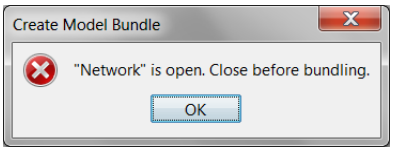
Certain model unique items are not automatically included in the model bundle file. User imported icons and background images are not included in the bundle unless the images were imported for the model. Images can either be imported for SIMPROCESS (which means the images are available for any model), or they can be imported for a specific model. The model directory will contain the file `ModelImages.jar` if images have been imported for the model. See "Importing Graphics Image Files" for a detailed discussion on importing graphics.

External Java class files or jar files used with the `ExternalCall` expression statement (see "Method ExternalCall") should be located in the `classes` directory, which is in the model directory (see "Model-Specific Java Additions"). If so, these files will be automatically bundled. Files of this type located in the `ext` directory will not be bundled (see "ext Directory").

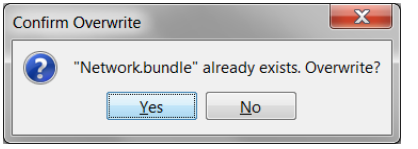
The `OpenDatabase` expression statement creates a connection to a database (see "Interfacing With A Database"). The first (and usually only) parameter for `OpenDatabase` is a properties file that contains the database connection information. Even though the properties file will be contained in the model bundle file, no other database files will be included in the model bundle file unless these files are located in the model directory. Note that if JDBC driver jar files are required for database connection, they must be in the `jre/lib/ext` directory. Thus, they will be excluded since they are not in the model directory, and the names of these files are unknown to the model.

Creating Bundle Files

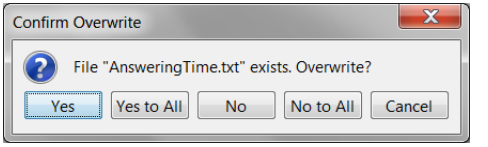
Selecting **File/Bundle/Create** displays a dialog for selecting the model to bundle. The selected model must not be open. If it is, an error message will display and the bundling process will stop.



The bundle file will be created in the same location as the model selected for bundling. The name of the bundle file will be the name of the selected model plus the extension `.bundle`. If a bundle file with the same name already exists in the same location, permission must be given to overwrite the existing bundle file. If permission is not granted, the bundling process ends.



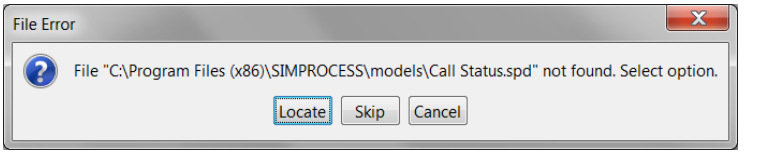
Files not already in the model directory (for example, Metamodels, OrgModels, Dashboards, and files with a specific path) are copied to a temporary location before creating the bundle file. If there is already a file with the same name in the temporary location, a dialog will appear asking permission to overwrite the file. Referencing the same file more than once in the model can cause this to happen.



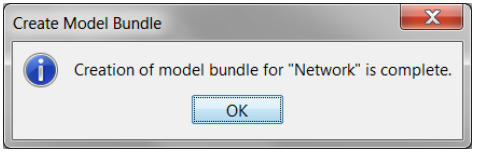
Select **Yes to All** if file overwrite should always happen or **No to All** if file overwrite should never happen. **Cancel** causes the bundling process to stop.

If a file that should be copied is not found at the location specified in the model, a dialog will appear that offers the option to **Locate** the file, **Skip** the file, or **Cancel** the bundling process. If **Locate** is selected, a dialog is displayed for finding the file. Note that this scenario indicates a problem with the model being bundled that should be fixed. Model bundling does not correct invalid file references in the original model. If **Locate** is selected and the file is found, the copy of the model included in the bundle will have the correct file reference. Thus, one way of correcting the original model would be to delete

the original model and its directory and extract the bundle to the original model's location.



A notification appears when the bundling process completes successfully.



Extracting Bundle Files

Select **File/Bundle/Extract** to extract a model bundle file. The extraction process begins once a bundle file has been selected. If there is a model open from the same location as the bundle file and with the same name as the bundled model, the extraction process will stop. Also, before any files are extracted from the bundle file, the version of the included model is checked to make sure that the model is compatible with the version of SIMPROCESS that is extracting the bundle. The extraction process stops if the bundled model is not compatible.

The bundled model and its directory are extracted in the same location as the bundle file. If same named files are found in the location, the extraction process requests permission to overwrite. The same options are offered as with the bundle creation process: **Yes**, **Yes to All**, **No**, **No to All**, **Cancel**. **Cancel** will stop the extraction process. Metamodel files are placed in the `metamodel` directory, OrgModel files are placed in the `orgmodel` directory, and Dashboard files are placed in the same directory as the bundle file.

The model is opened in SIMPROCESS when the extraction process successfully completes. Note that the model opening is not a guarantee that the model will run. If there are items required by the model that were not included in the bundle file, the model will not run correctly. Typically, these items would be database related.

A sample model bundle, `InputSource.bundle`, is in the Demos directory. The bundle contains an Input Sources demonstration model and supporting files. See "Input Sources" for more information.

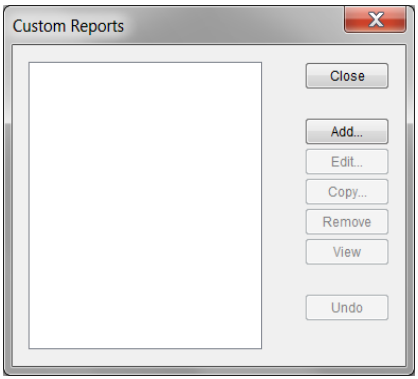
CHAPTER 7

Custom Reports

Custom reports are user-defined reports. They are defined for specific purposes and can contain statistical and non-statistical information. JasperReports is used to view custom reports. Custom reports are typically defined and viewed once all modeling, simulation, and analysis is complete. Multiple custom reports can be defined for a single model. Each report displays in a report viewer. From that viewer reports can be printed or saved in various file formats.

Defining a Custom Report

Select **Custom Reports...** from the **Report** menu to define a new custom report or edit an existing custom report. This displays a dialog that lists the reports that have been defined.



The **Custom Report** dialog is used to define new reports, modify existing report definitions, copy report definitions, delete reports, and view reports. The **Add** function creates a new report definition. The **Edit, Copy**, **Remove**, and **View** functions are only active if there are existing report definitions. These functions operate on the report selected in the list:

Edit modifies an existing report definition.

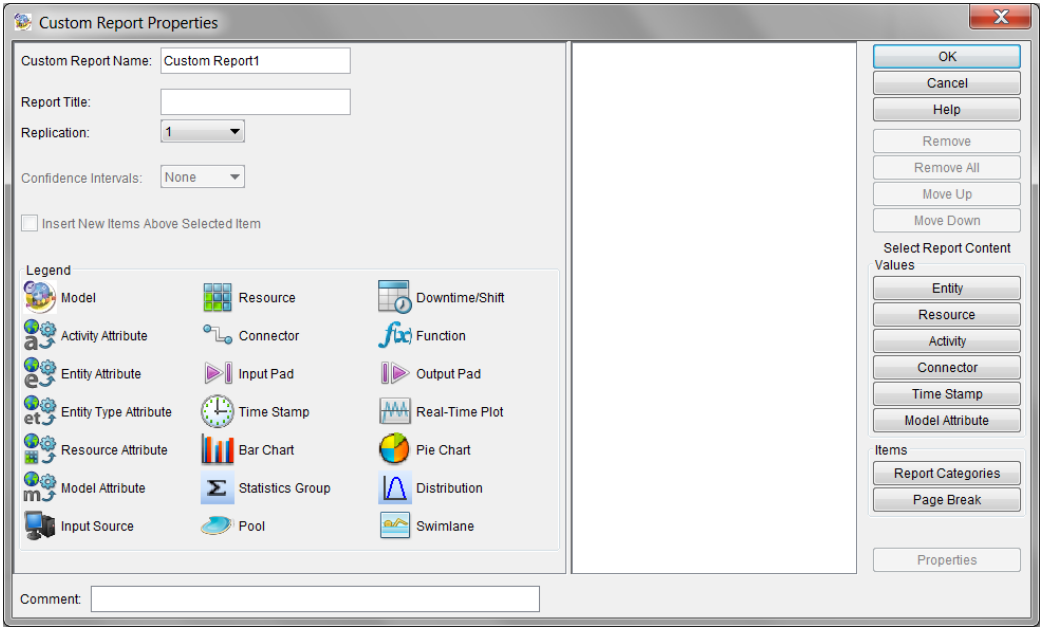
Copy creates a new report based on the definition of an existing one.

Remove deletes a report definition.

View displays a report in the report viewer.

Undo restores a report that has been removed.

Selecting **Add...** displays the **Custom Report Properties** dialog. The dialog displays with a default name and with the list of items to report empty.



Custom Report Name is the name of the report definition. This name displays in the custom report list and must be unique.

Report Title is the title of the report. This field must be populated.

Replication is the simulation replication results to include in the report. (See "Number of Replications" for information on setting the number of simulation replications.) If the model is only set to run one replication, then 1 will be the only choice in the selection list. If multiple replications are set, then each replication will be listed along with Sum of Replications and Average of Replications. **Replication** only applies if simulation results are included in the report.

Confidence Interval is only enabled when Average of Replications is selected for **Replication**. The choices are None, 90%, 95%, and 99%. As with **Replication**, **Confidence Interval** only applies if simulation results are included in the report.

Insert New Items Above Selected Item is enabled when an item in the report item list on the right is selected. When **Insert New Items Above Selected Item** is selected, new items added to the report item list are added above the selected item instead of to the end of the report item list.

The **Legend** helps to identify the items in the report item list.

Comment is an optional field for entering a brief description of the report.

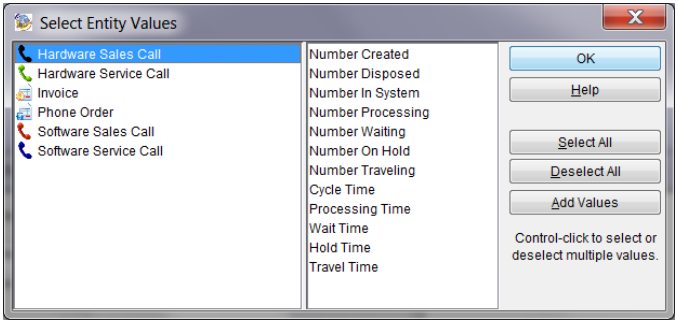
The **Remove**, **Remove All**, **Move Up**, and **Move Down** buttons on the right are used for managing report items already in the report item list. **Remove All** is enabled when there is at least one item in the report item list. This button empties the report item list. **Remove** is enabled when one or more items are selected in the report item list. It removes the selected items from the report item list. **Move Up** and **Move Down** are enabled when only one item is selected in the report item list. These buttons adjust the placement in the list of the selected item.

Adding Items to the Report Item List

The buttons under **Select Report Content** are used to add items to the report item list. There are two groups of buttons: **Values** and **Items**. The **Values** buttons (**Entity**, **Resource**, **Activity**, **Connector**, **Time Stamp**, and **Model Attribute**) allow individual statistical values to be added to the report Item list. The **Items** buttons (**Report Categories** and **Page Break**) allow the addition of statistical and non-statistical information. There are no restrictions on the report item list. Items can be listed in any order, and items can be duplicated.

Values Buttons

The buttons in the **Values** group offer the same values as the same-named buttons on the **Custom Plot Properties** dialog. (See "Adding Values To Custom Plots" for a listing of these values.) Also, values are added to the report item list in the same way as values are added to the plot list. If **Entity** is selected the following displays.



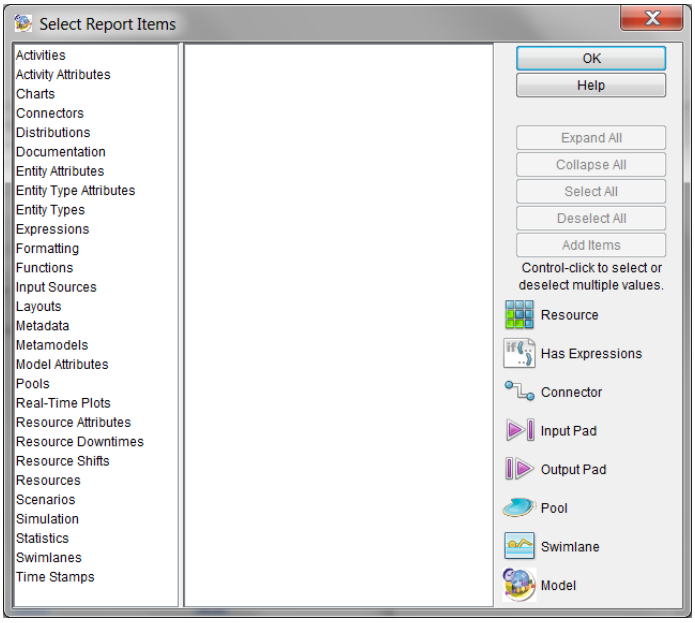
Once an Entity is selected on the left, statistical values for that entity can be selected on the right. However, the values are not added to the report Item list unless **Add Values** is clicked. This allows values from multiple Entities to be added before the dialog is closed.

Items Buttons

As the name suggests, the **Page Break** button simply adds a page break to the report Item list. Page

breaks can also be added through the **Report Categories** button. Note that page breaks are not required but are useful for separating different categories of information.

The **Report Categories** button displays the following dialog.

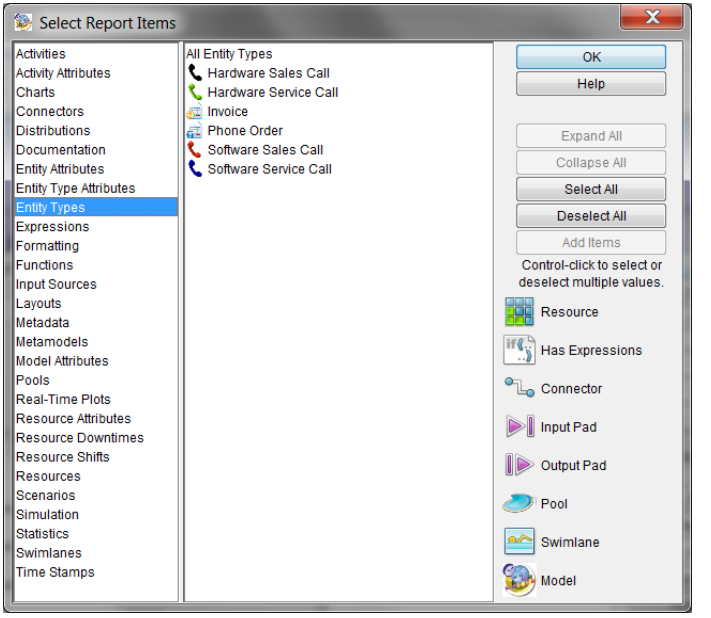


The list on the left contains the categories of items that can be added to the report item list. When a category is selected on the left, the right side displays the items that can be added to the report item list. All categories display a list of items on the right except for **Activities**, **Connectors**, and **Resources**. The **Activities** and **Connectors** categories display a tree view of the model, and the **Resources** category displays a tree view of the model Resources. If there are no items for a particular category, the right side remains empty. Items are added to the report item list by selecting items on the right hand side and clicking **Add Items**. The categories available are

- Activities - displays a tree view of the Processes and Activities
- Activity Attributes - displays a list of the global and local Activity Attributes
- Charts - displays the Bar Chart and Pie Chart items
- Connectors - displays a tree view of the Processes and Connectors
- Distributions - displays the user-defined Standard, Tabular, and Auto-Fit distributions
- Documentation - displays a list of model components that have Documentation
- Entity Attributes - displays a list of global and local Entity Attributes
- Entity Type Attributes - displays a list global and local Entity Type Attributes

- Entity Types - displays a list of Entity Types
- Expressions - displays a list of model components that have Expressions
- Formatting - displays the items Page Break, Header, Line, and Standard Report Header
- Functions - displays a list of user-defined Functions
- Input Sources - displays a list of File, Spreadsheet, and Database Input Sources
- Layouts - displays a list of all model layouts
- Metadata - displays a list of all model components that have Metadata assigned
- Metamodels - displays a list of all Metamodels in the `metamodel` directory
- Model Attributes - displays a list of all Model Attributes
- Pools - displays a list of all pools
- Real-Time Plots - displays a list of all Real-Time Plots and Custom Plots
- Resource Attributes - displays a list of all global and local Resource Attributes
- Resource Downtimes - displays a list of all global Resource Downtimes
- Resource Shifts - displays a list of all Resource Shifts
- Resources - displays a tree view of all Resources
- Scenarios - displays a list of all saved Scenario results
- Simulation - displays the items Run Settings, Cost Periods, Global Statistics Collection, Active Alternatives, and Model Parameters
- Statistics - displays a list of statistical groups
- Swimlanes - displays a list of all swimlanes
- Time Stamps - displays a list of Time Stamps

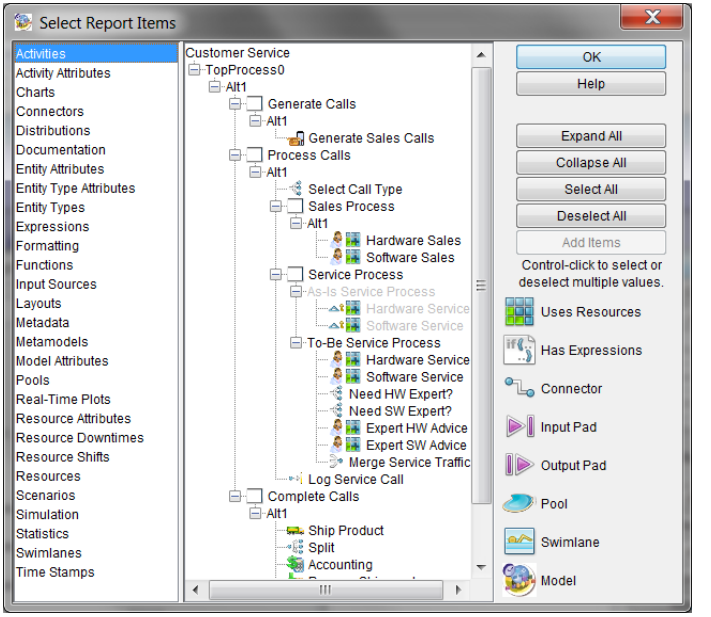
Most categories offer an “All” option. For example, if the **Entity Types** category is selected, the first item in the list is **All Entity Types** followed by each Entity Type defined in the model.



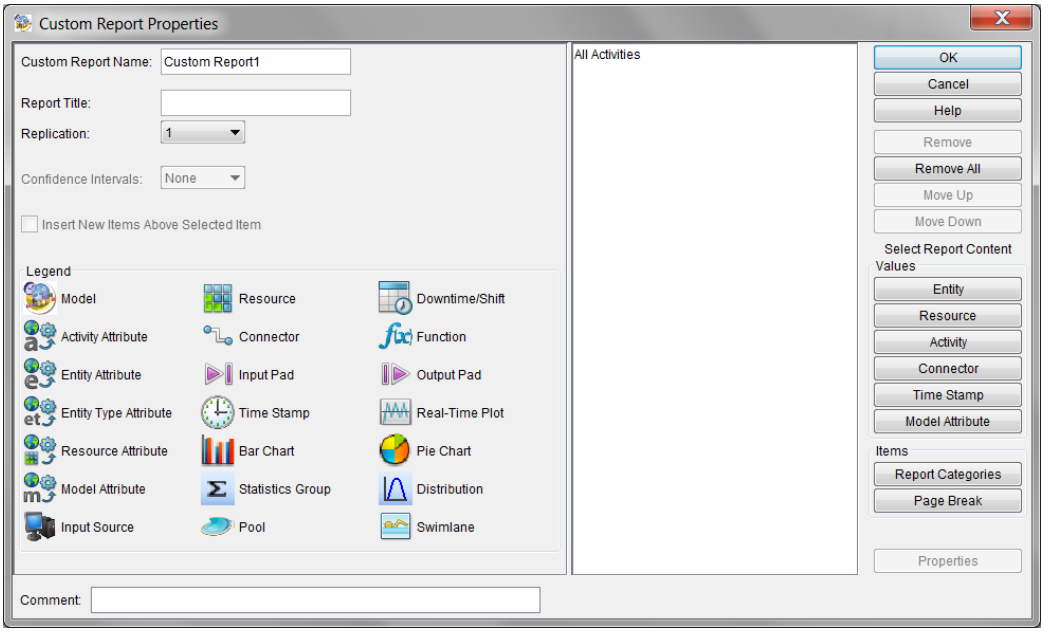
In the example above, all of the Entity Types in the model can be included in the report by either adding **All Entity Types** to the report list or adding each Entity Type individually to the report list. (Since there are no restrictions as to what can be in the report list, and even though it would lead to duplication of information, **All Entity Types** could be included in the report list along with individual Entity Types.) Each "All" item offers a default header for that category. Thus, if the default header is desired, then the "All" item should be added to the report list. If the default header is not desired, then individual items of that category should be added to the report list. The default header for **All Entity Types** is shown below. Note that the default header includes the name of the model followed by the category.

Customer Service Entity Types

The **Activities** and **Connectors** categories do not list **All Activities** or **All Connectors** options. However, those options do exist. The top level of the tree is the "All" option. For instance, as shown below, selecting **Customer Service** and clicking **Add Items** adds **All Activities** to the report item list.



Note that when a tree is displayed the **Expand All** and **Collapse All** buttons are enabled. The tree displays the TopProcess and each process alternative. These items can be selected, but they will not be added to the report Item list when **Add Items** is selected. When the Connector tree is displayed, the top item represents **All Connectors**. Also, Processes and alternatives display in the tree, but only selected Connectors will be added to the report item list when **Add Items** is clicked.



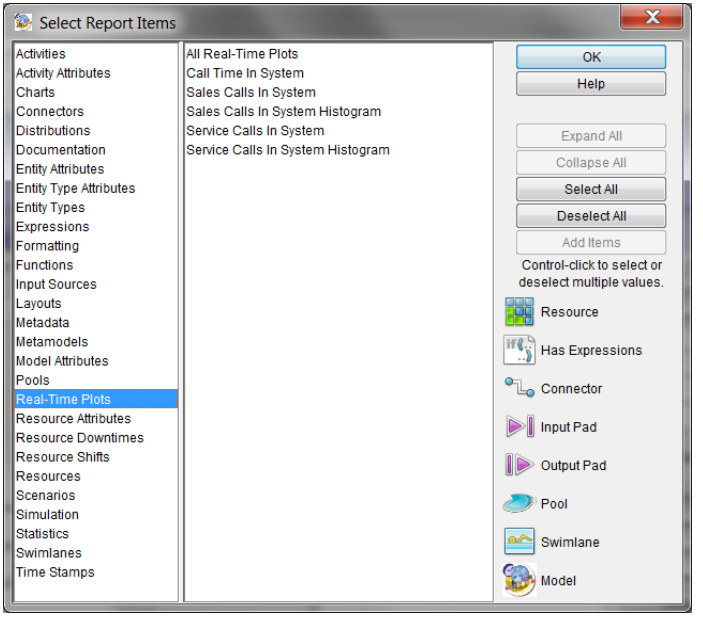
For some of the categories, the "All" option should be used judiciously. For example, for a large model, including all Activities would create a very long report with much information that has limited usefulness.

Statistical Categories

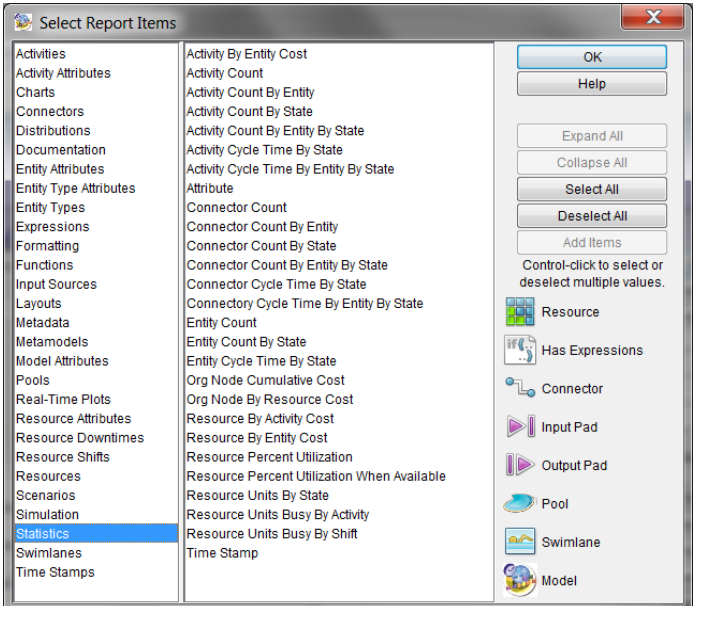
Charts, **Real-Time Plots**, **Scenarios**, and **Statistics** are the only categories that offer statistical results. If items from **Charts**, **Real-Time Plots**, or **Statistics** are added to the report Item list, the simulation must be run before the report is viewed. **Scenarios** provides statistical results from previously run Scenarios; thus, the model does not need to be run before the report is viewed. All the other categories provide properties or other information. For example, selecting the **Resources** category displays on the right a tree of the Resources defined in the model. If one of those Resources is added to the report Item list, the properties of that Resource are included in the report, not any statistical results from that Resource.

The **Charts** category offers two items: **Bar Chart** and **Pie Chart**. (The values that populate a chart are added after the chart has been added to the report item list. See "Configuring Report Items" on page 110.)

The **Real-Time Plots** category lists an **All Real-Time Plots** item and all real-time and custom plots that are defined in the model.



The **Statistics** category contains statistics groups.

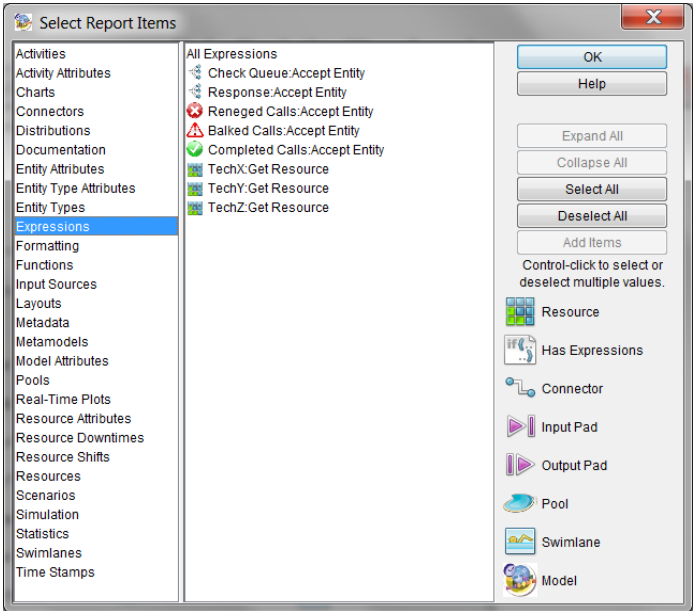


These groups represent the same statistics displayed in the Standard Report and described in Chapter 8 of Part A (see "Statistical Output Reports"). For example, the statistics group **Entity Count** displays Total Generated, Remaining In System, and Total Disposed for all Entity Types which had statistics collected. Thus, individual statistical values cannot be added by selecting the **Report Categories** button on the Custom Report dialog. Individual statistical values can only be added to the report item list by one of the buttons in the **Values** group (see "Values Buttons" on page 100).

The **Scenarios** category lists the scenarios that have saved statistical results. A Scenario displayed in a Custom Report will show the Run Settings, Model Parameters selections, and Process alternative selections along with all the statistical results for the selected **Replication**. Note that a Scenario may or may not have the requested **Replication** since a Custom Report's available replication values are based on the model's current **Number of Replications** value in the Run Settings. The selected Scenario may have been run with a different number of replications. (See "Scenarios" for more information on Scenarios and Scenario Reports.)

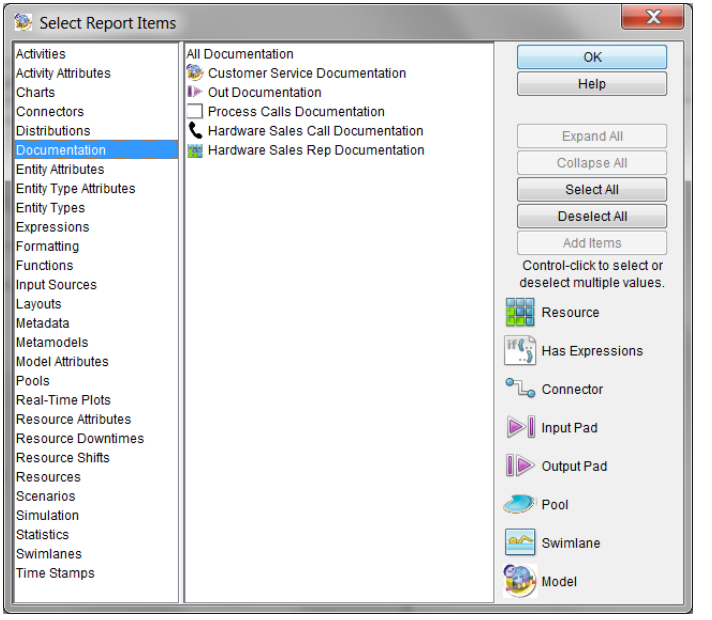
Non-Statistical Categories

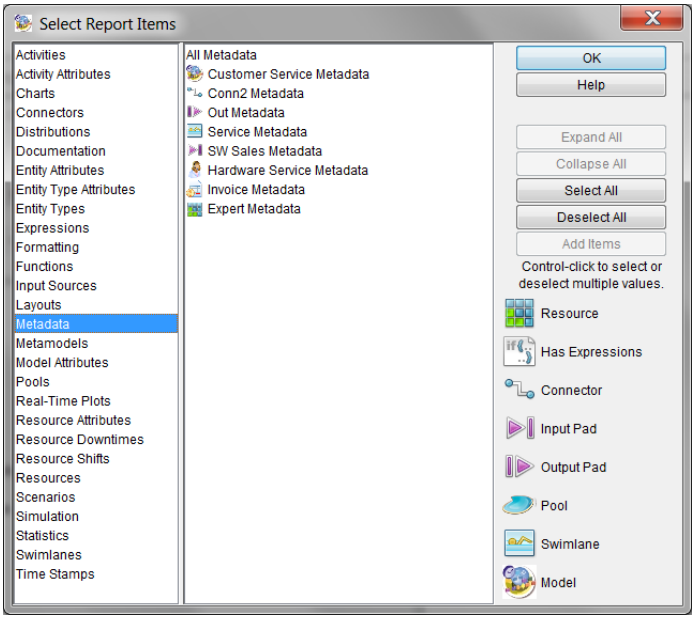
All of the non-statistical categories (with the exception of **Formatting**) offer report items dealing with model/simulation defintion or documentation. For example, the **Expressions** category lists all expressions in the model.



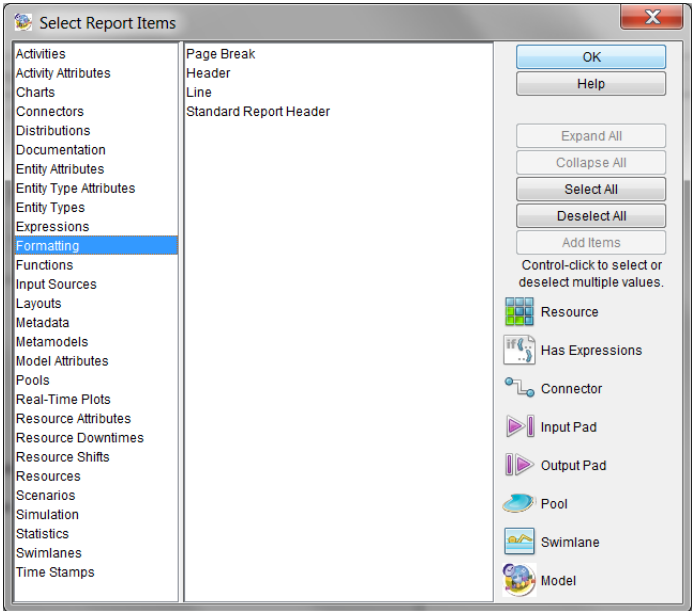
Listed below the **All Expressions** item are each Expression found in the model. The Expression item

consists of the name of the Expression owner followed by the name of the Expression. This naming convention follows for each type of item that is specific to a SIMPROCESS component. Local Attribute items for the categories **Activity Attributes**, **Entity Attributes**, **Entity Type Attributes**, and **Resource Attributes** are listed with the name of the Attribute owner followed by the name of the Attribute. (Global Attributes are listed by name only since there are not specific owners for global Attributes.) The **Documentation** category lists the name of the documentation owner followed by the word **Documentation**, and the **Metadata** category lists the name of the metadata owner followed by the word **Metadata**.





The **Formatting** category contains items for formatting a report.

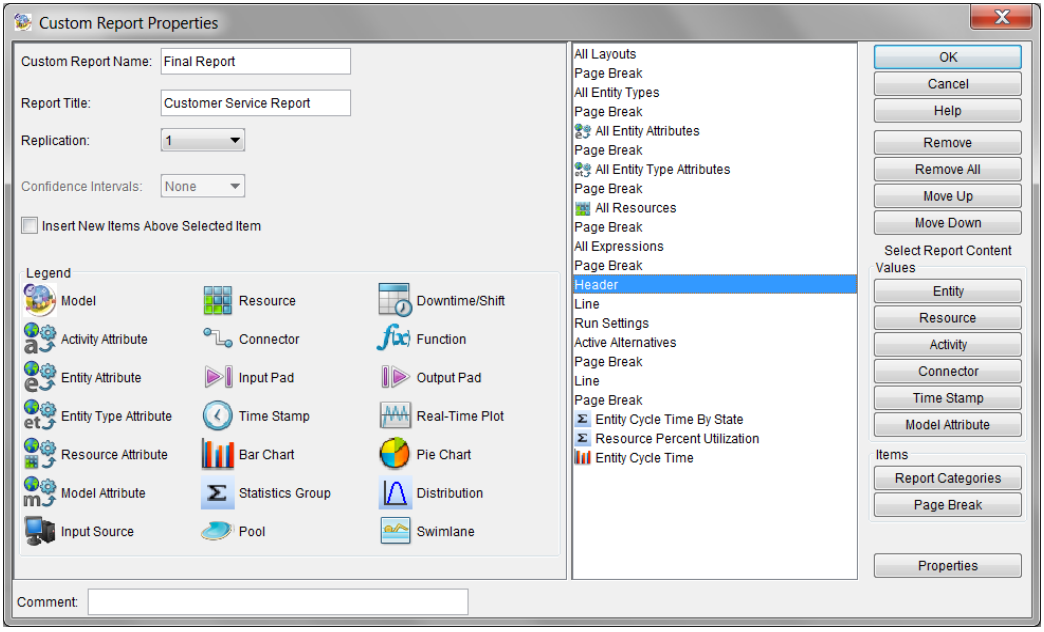


The **Page Break** item adds a page break to the report item list (as does the **Page Break** button on the Custom Report dialog). **Header** allows for user-defined headers, and **Line** is simply a horizontal line. (The properties for **Header** and **Line** are set from the report item list.) The **Standard Report Header** displays the same simulation information as the Standard Report. It is not configurable.

The **Simulation** category lists items related to simulation settings (**Run Settings**, **Cost Periods**, **Global Statistics Collection**, **Active Alternatives**, and **Model Parameters**). The **Run Settings** item adds the settings of the Run Settings dialog (Run Settings) to a report. The **Cost Periods** item lists the cost periods (Setting Up Cost Periods) defined on the Run Settings dialog. **Global Statistics Collection** lists the settings defined for global statistics collection (Default Performance Measures). **Active Alternatives** adds to the report a list of which Process Alternatives are active in the model. This only lists Alternatives from Processes that have more than one Alternative (Alternative Sub-Processes). Model Parameters list all the global Attributes designated as Model Parameters (Running a Simulation with Model Parameters).

Configuring Report Items

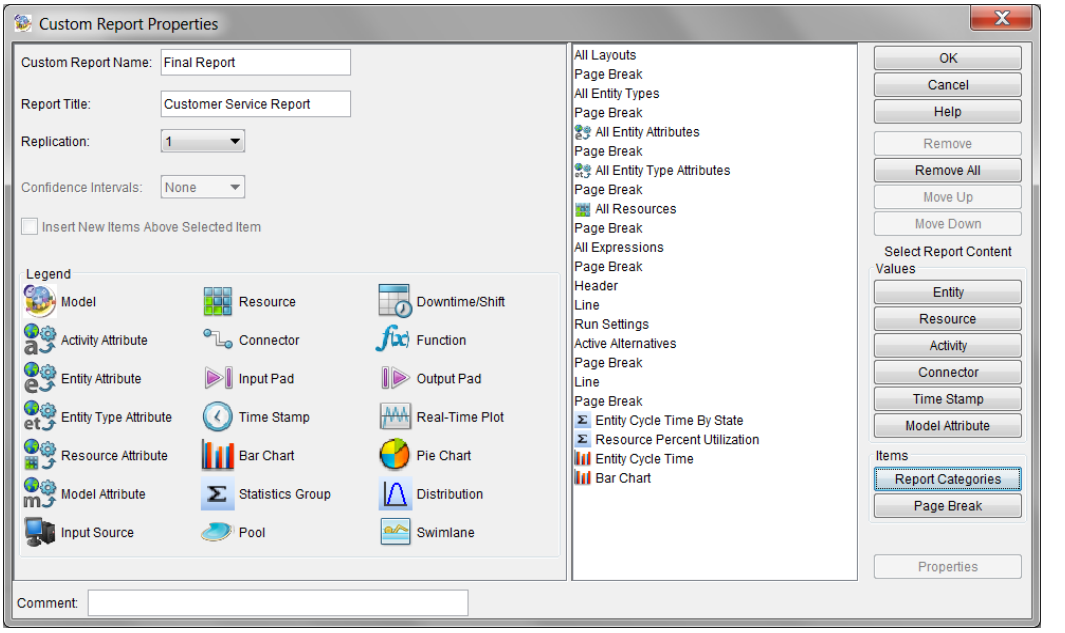
The following report items are configurable: **Bar Chart**, **Pie Chart**, **Header**, and **Line**. When one of these items are selected in the Custom Report dialog, the **Properties** button enables.



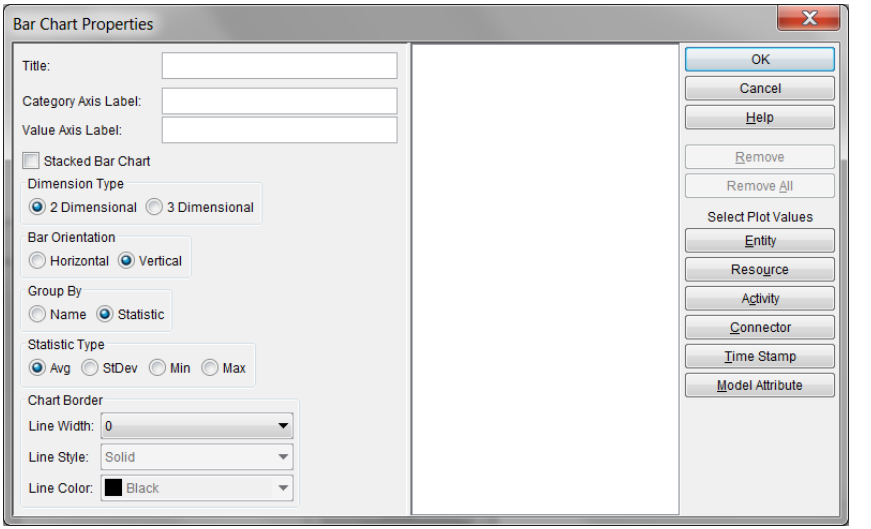
The properties of the selected item can be accessed by clicking the **Properties** button or by double-clicking the selected item. If the selected item is not configurable, the **Properties** button will not enable and double-clicking results in no action.

Bar Chart Properties

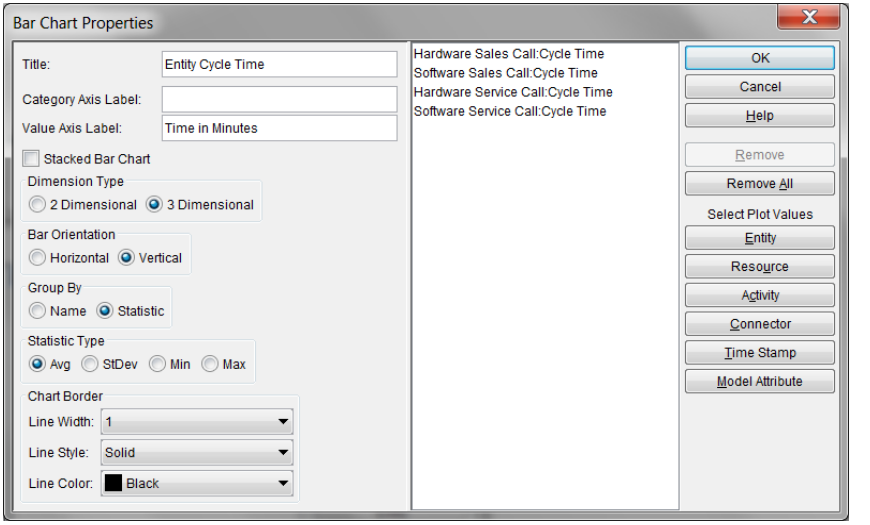
When a Bar Chart is added to the report item list, it displays as **Bar Chart**. Once the properties have been set, the report item displays as the name of the chart. In the example below the last two items are Bar Charts. The first one has its properties set and the second one doesn't.



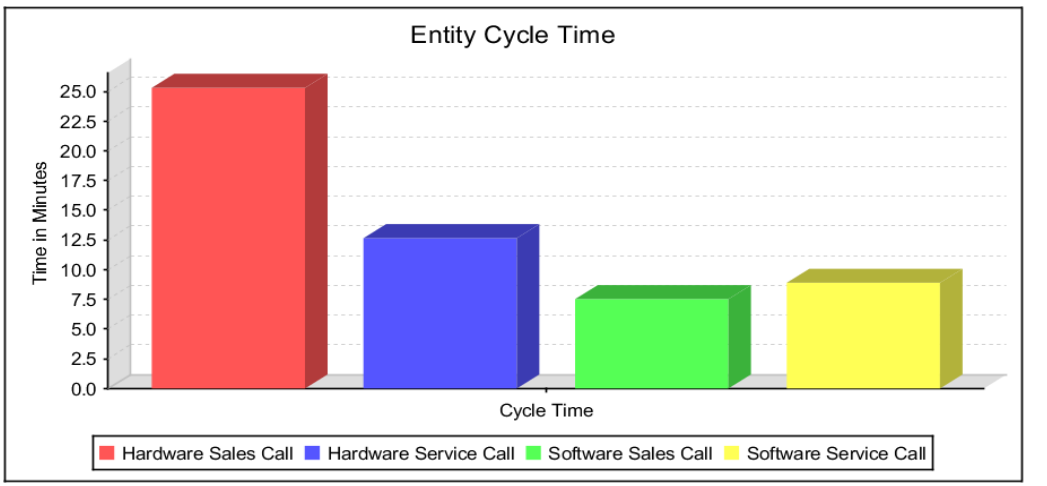
Editing a **Bar Chart** report item displays the following.



The **Title** is the title of the chart and replaces **Bar Chart** in the report item list. This field must be populated. **Category Axis Label** (horizontal axis) and **Value Axis Label** (vertical axis) are optional. When selected, **Stacked Bar Chart** creates stacked bars. **Dimension Type** sets the display of the bars as 2 **Dimensional** or **3 Dimensional**, and **Bar Orientation** sets the display of the bars as **Vertical** (from horizontal axis) or **Horizontal** (from vertical axis). **Group By** determines how the selected values are grouped into bars. **Statistic Type** determines the type of the value to display. Values are added to the chart just as values are added to the report list or a custom plot (see "Values Buttons" on page 100). **Chart Border** sets the border properties for the chart. A **Line Width** of 0 indicates that the chart will not have a border. If a value other than 0 is selected for **Line Width**, **Line Style** and **Line Color** can also be specified. Note the example below.

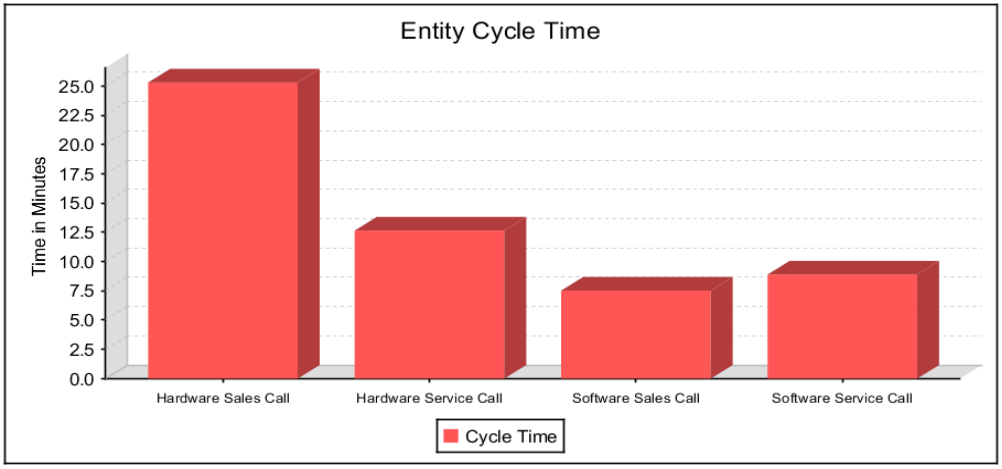


There are four Entity values to display. Since **Avg** is selected as the **Statistic Type**, the average Cycle Time will be displayed. Since **Statistic** is the **Group By** selection, the bars are grouped by Cycle Time. That is, there is one category, which is Cycle Time, and the legend shows that there are 4 different values for that category.



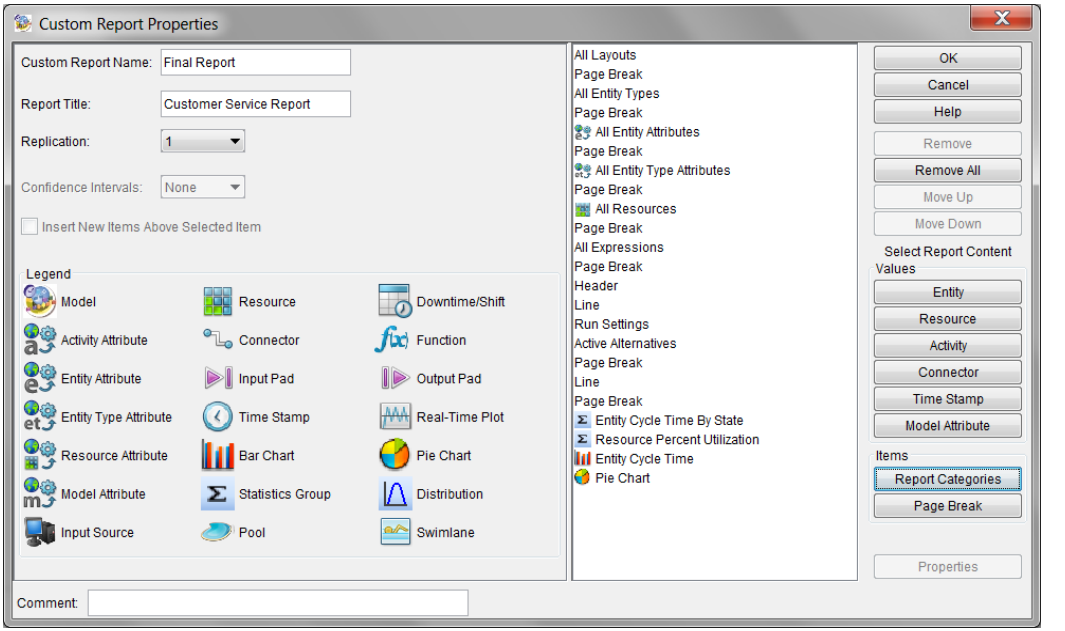
As shown below, if the **Group By** setting is changed to **Name**, then there are 4 categories since there are 4 different Entity names and only one value type. The legend shows that there is one value for each category. Thus, the **Group By** setting to use depends upon the types of values to be displayed. Using

Name for the **Group By** is useful when multiple value types from the same item (thus there is only one name category) are displayed in a chart.

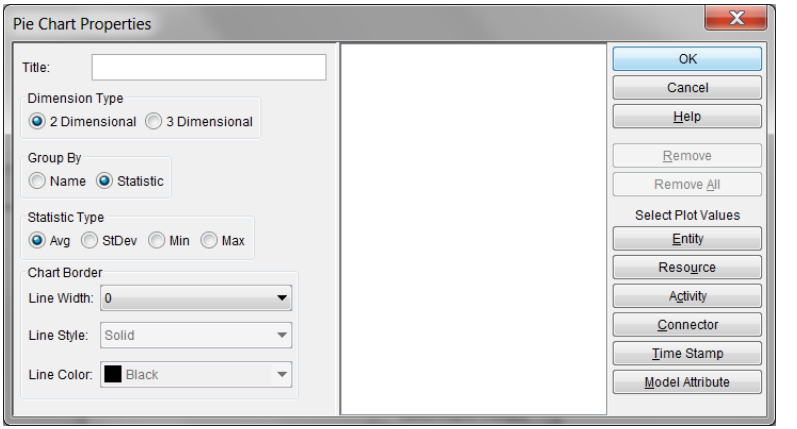


Pie Chart Properties

When a Pie Chart is added to the report item list, it displays as **Pie Chart**. Once the properties have been set, the report item displays as the name of the chart. The example below repeats the Bar Chart example except the last two items are Pie Charts. The first one has its properties set and the second one doesn't.

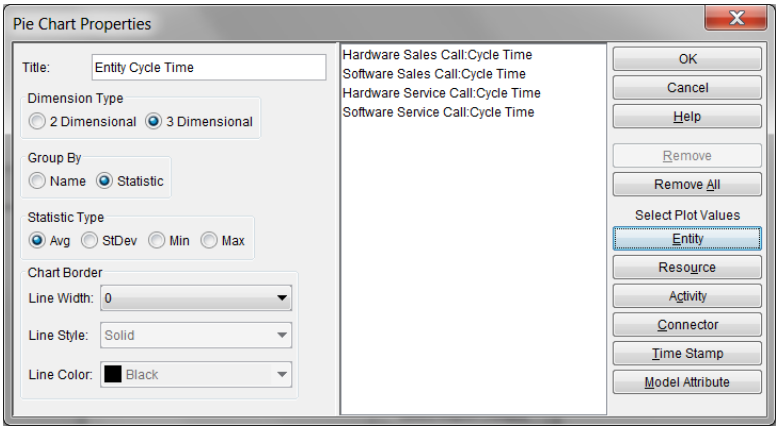


Editing a **Pie Chart** report item displays the following.

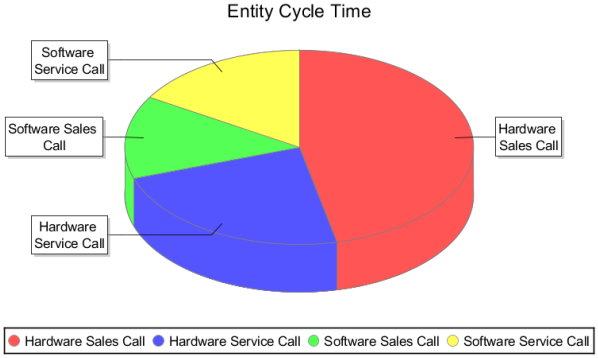


The **Title** is the title of the chart and replaces **Pie Chart** in the report item list. This field must be populated. **Dimension Type** sets the display of the pie chart as 2 **Dimensional** or **3 Dimensional**. Just as with a bar chart, **Group By** determines how the selected values are grouped. **Statistic Type** determines the type of the value to display. **Chart Border** sets the border properties for the chart. A **Line Width** of

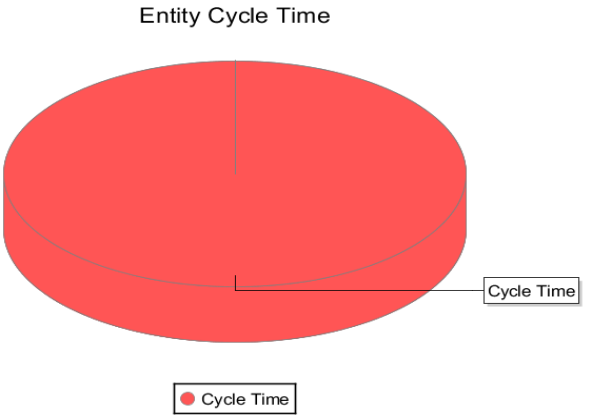
0 indicates that the chart will not have a border. If a value other than 0 is selected for **Line Width**, **Line Style** and **Line Color** can also be specified. Values are added to the chart just as values are added to the report list or a custom plot (see "Values Buttons" on page 100). The example below contains the same values to plot as the bar chart example.



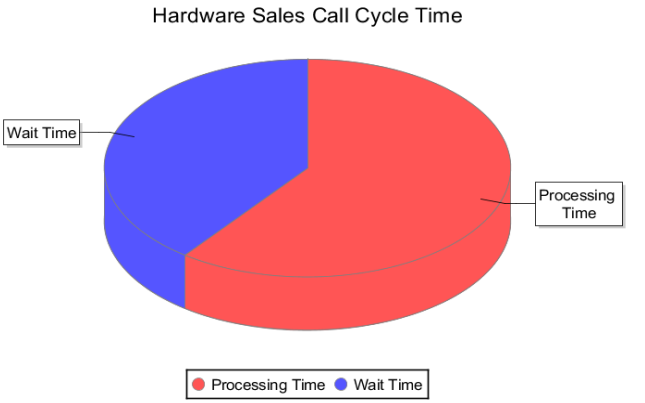
Again, there are four values to display, each from a different Entity. Since **Avg** is selected as the **Statistic Type**, the average Cycle Time will be displayed. Since **Statistic** is the **Group By** selection, the pie chart shows 4 values for the one group Cycle Time. That is, there is one statistic, which is Cycle Time, and the legend shows that there are 4 different values for that statistic.



As shown below, if the **Group By** setting is changed to **Name**, then there are 4 groups since there are 4 different Entity names and only one value type. The legend shows that there is one value for each category. However, this chart is not useful since there is only one value for each group, thus, each group has the same color.

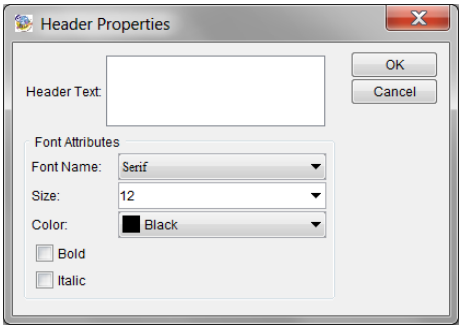


The next chart uses **Name** for the **Group By** but displays different value types for the same Entity (thus there is only one name).



Header Properties

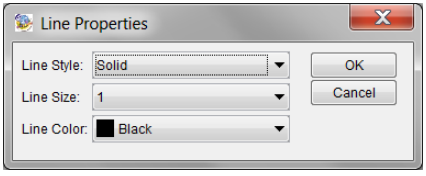
When a **Header** report item is edited, the Header Properties dialog displays.



The text entered in the **Header Text** field will display centered in the report using the settings in **Font Attributes**. **Font Name** lists any found found on the system, and the header will display in the report with the selected font. However, if the report is saved to PDF, it is possible that some of the system unique fonts may not be compatible with PDF. If so, another font will be automatically substituted. Note that this is not an issue when a report is saved to RTF.

Line Properties

When a Line report item is edited, the Line Properties dialog displays.



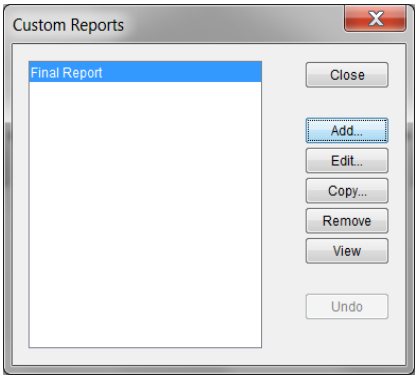
The **Line Style** options are **Solid**, **Dashed**, **Dotted**, and **Double**. **Line Size** can range from **1** (thinnest) to **10** (thickest). **Line Color** can be any of the 65 predefined colors offered by SIMPROCESS (see "SIMPROCESS Color Table"). Lines display the full width of the report page.

Below is a **Header** and a **Line** from a report. For the **Header**, the **Font Attribute** settings were **Font Name**: Times New Roman, **Size**: 36, and **Color**: Blue with **Bold** selected. The **Line** properties were **Line Style**: Solid, **Line Size**: 4, and **Line Color**: Blue.

Simulation Settings

Viewing a Custom Report

Custom Reports are viewed from the Custom Report dialog. When a custom report in the list is selected the **View** button enables.



Clicking View causes SIMPROCESS to create the report then display the report in the Report Viewer. Note that there will be a delay before the report displays of a few seconds up to a couple of minutes. The length of the delay depends on the size of the report being generated.

Report Characteristics

The first page of each report is the title page. The title of the report is the title entered in the **Report Title** field of the Custom Report properties dialog. In addition to the title, the title page list the SIMPROCESS model which created the report and the date and time of report creation.

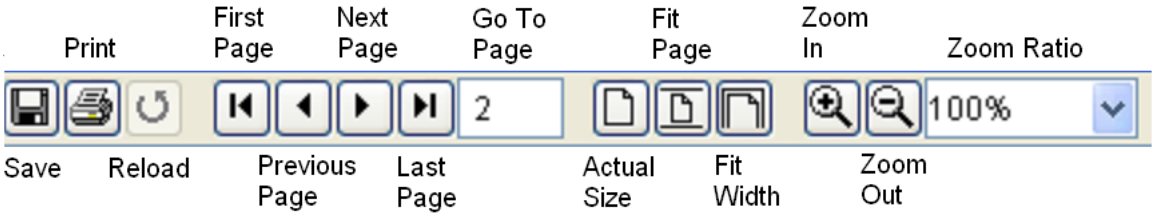
Customer Service Final Report

SIMPROCESS Model: Customer Service
Report Created: Mon Sep 29 13:15:30 EDT 2008

Each page except the first page has a header and a footer. The header is the title of the report displayed at the center of each page. The footer is the page number in the format Page x of y (for example Page 3 of 14).

Report Viewer

The Report Viewer is a separate window within SIMPROCESS for viewing a custom report. It allows navigation, zooming, printing, and saving of the report. The Report Viewer functions are located on the Report Viewer toolbar, which is shown below.



The Save button defaults to saving the report to the model's directory. Reports can be saved in various formats.

- JasperReports (*.jrprint)
- Portable Document Format (*.pdf)
- Rich Text Format (*.rtf)
- Open Document Format (*.odt)
- Microsoft Word Open XML (*.docx)
- Hypertext Markup Language (*.html)
- Single Sheet XLS (*.xls)
- Multiple Sheet XLS (*.xls)
- Comma Separated Values (*.csv)
- Extensible Markup Language (*.jrpxml, *.xml)
- Embedded Images XML (*.jrpxml, *.xml)

CHAPTER 8

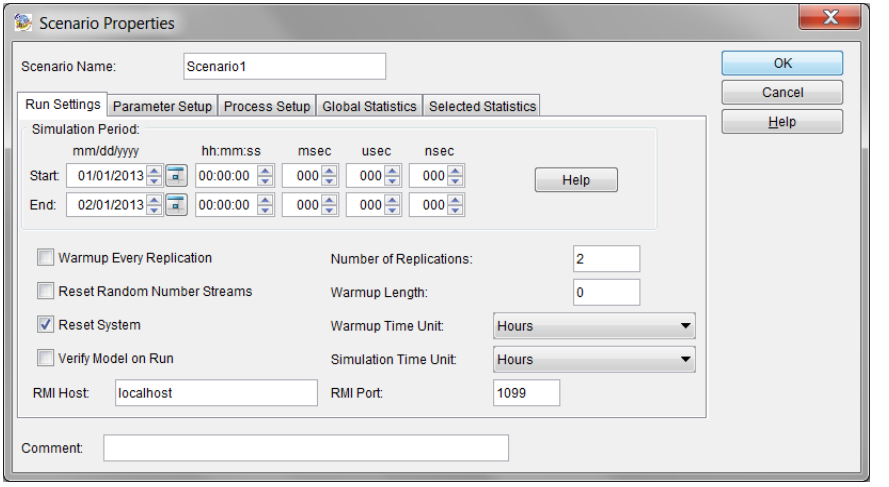
Scenarios

Scenarios are used to save simulation results for comparison purposes and are specific to a model. Thus, Scenario results from different models cannot be compared directly in SIMPROCESS. A Scenario is defined, and then the model is run with the Scenario selected. When the simulation completes the results from the Scenario are saved into an external file (`Scenarios.xml`) for later use. Scenario Reports can be created to compare results from different Scenarios or within the same Scenario.

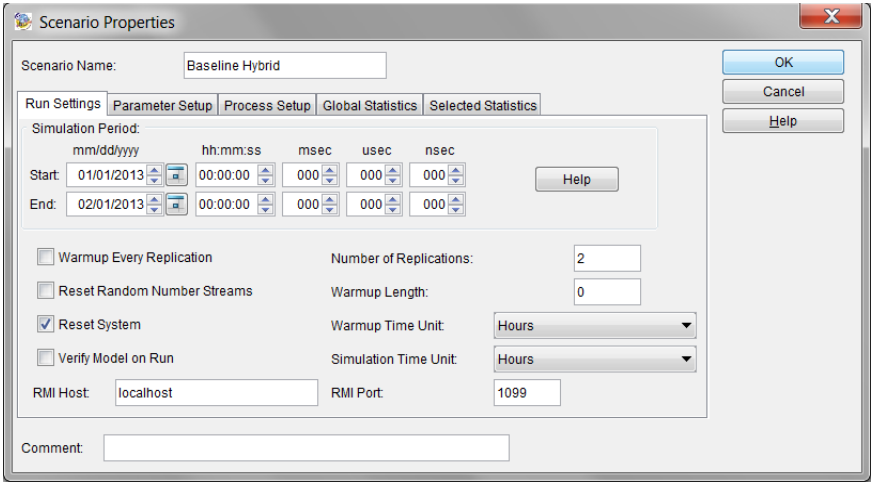
Defining Scenarios

To define a scenario, select **Scenarios...** from the **Simulate** menu. This displays a dialog that has the options to **Add...**, **Edit...**, **Copy...**, or **Remove** a Scenario. There is also an option to **Undo** the removal of a Scenario. Some of these buttons also either directly or indirectly manage Scenario results, which are stored in a file separate from the model.

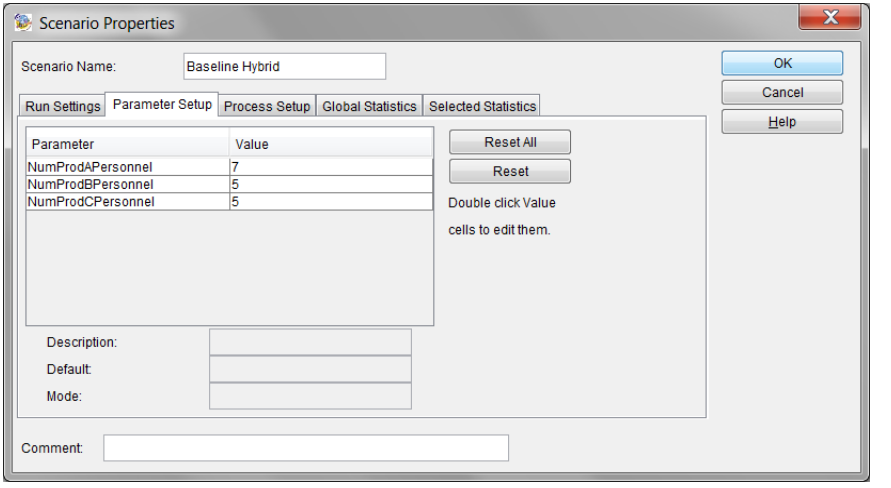
Selecting **Add...** brings up a dialog similar to the one below.



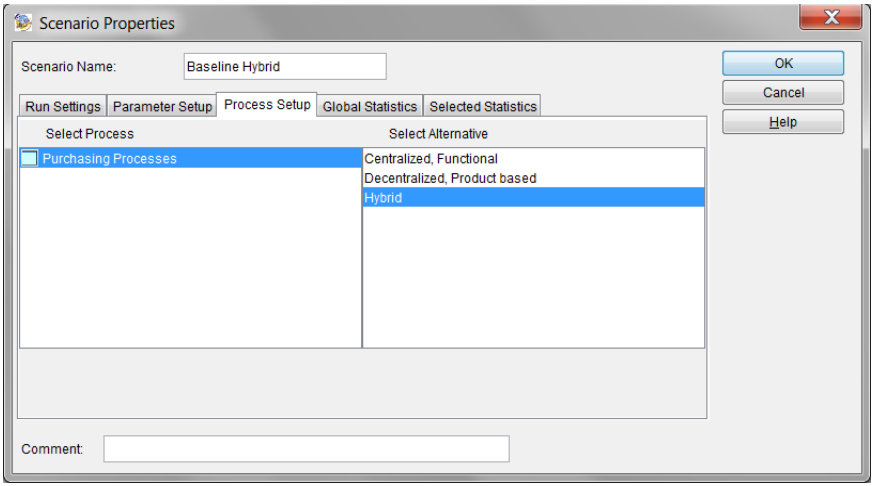
A default name is given for the **Scenario Name**. The **Scenario Name** specified must be unique among all defined Scenarios. A Scenario consists of model run settings (**Run Settings** tab), Model Parameter settings (**Parameter Setup** tab), process alternative selections (**Process Setup** tab), global statistics selections (**Global Statistics** tab), and selected individual statistics (**Selected Statistics** tab). Below is a Scenario defined for the `Purchasing.spm` demonstration model.



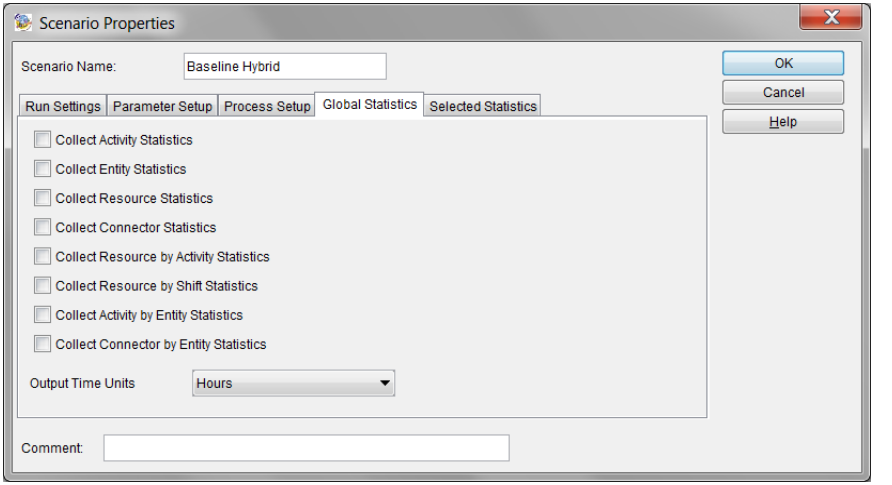
The **Run Settings** tab, **Parameter Setup** tab, and **Global Statistics** tab default to the last saved values in the model for each. After specifying Run Settings, set the Model Parameter values.



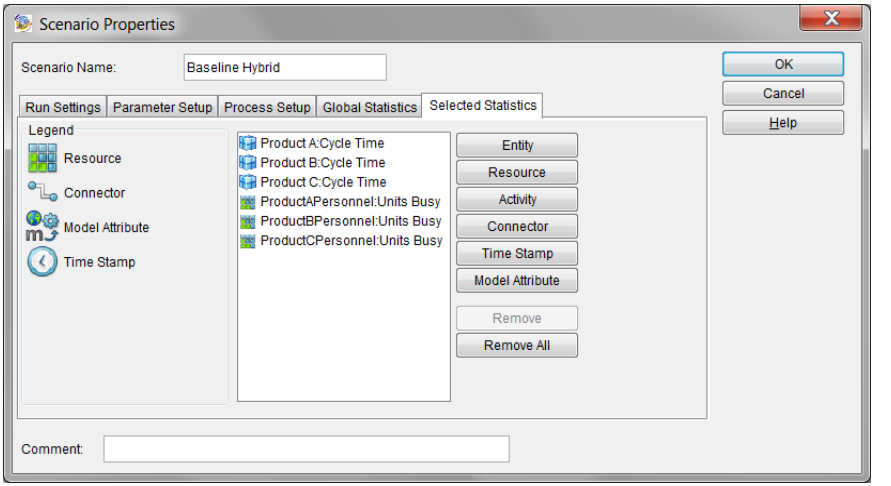
The **Parameter Setup** tab is used to set the value for each Model Parameter defined in the model. (See "Running a Simulation with Model Parameters" for more information on Model Parameters.)



The **Process Setup** tab lists all the hierarchical processes in the model that have more than one alternative. (See "Alternative Sub-Processes" for more information on Process alternatives.) Selecting a process on the left displays the available alternatives on the right. Note that if no alternative is selected, when run, the Scenario will use the last active alternative for a process. If a Master Process (Master Processes) has multiple alternatives, the copies of the Master Process will not be displayed. The selected alternative for the Master Process will be applied to all copies.

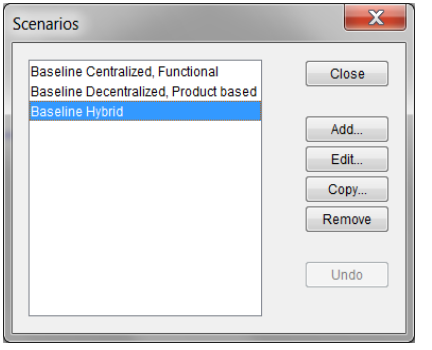


The **Global Statistics** tab displays the same statistics options as **Report/Define Global Statistics Collection** (Default Performance Measures). However, the buttons to clear individual stats are not available.



The **Selected Statistics** tab is used to add individual statistics to the Scenario. Note that for this example no global statistics were selected, only individual statistics. Individual statistics are selected in the same manner as for Custom Reports (see "Adding Items to the Report Item List") or Custom Plots (see "Adding Values To Custom Plots").

Once the scenario definition is complete, select **OK** and the scenario will appear in the scenario list.



There are three scenarios defined that are based on the three alternatives found in the Purchasing model.

The **Edit...** button allows changes to be made to the Scenario. Note that, other than the Scenario name, these changes only apply to the Scenario definition, not the Scenario results. Thus, a change in the Scenario name is reflected in `Scenarios.xml`, but, if there are other changes, the model must be run again with the changed Scenario selected for the results in `Scenarios.xml` to be updated.

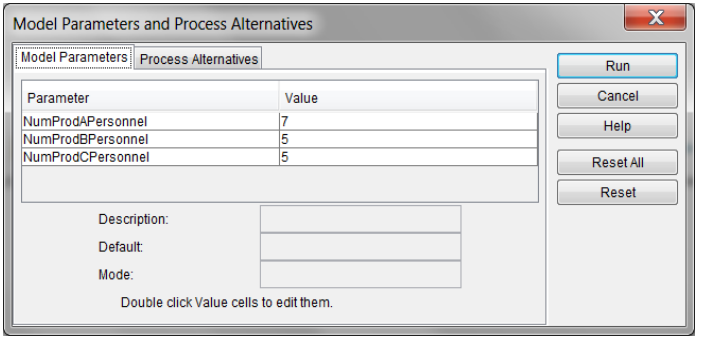
The **Copy...** button copies the selected Scenario into a new Scenario. If there are results in `Scenarios.xml` for the Scenario that is being copied, those results are not copied. The model must be run with the new Scenario selected to place results in `Scenarios.xml`.

The **Remove** button removes a Scenario definition from the model and, if there are associated results in `Scenarios.xml`, those results are removed as well.

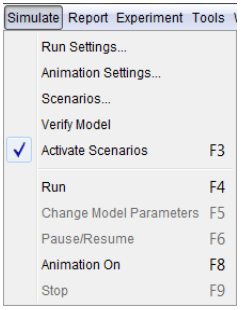
When a Scenario definition has been removed, the **Undo** button enables. When selected, the last removed Scenario definition is restored. However, any associated results are not restored. The model must be run again with the restored Scenario selected to restore Scenario results in `Scenarios.xml`.

Running Scenarios

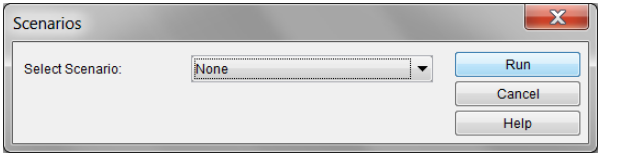
Once a Scenario has been defined it can be run and the results stored for future analysis. When a model has Model Parameters and **Run** is selected, the Model Parameters dialog appears.



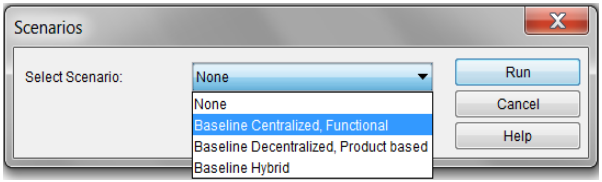
This dialog allows the value of the Model Parameters to be changed before the simulation begins. Scenarios cannot be used in conjunction with this dialog since the values of Model Parameters are set in the Scenario. Thus, to use Scenarios, **Activate Scenarios** must be selected on the **Simulate** menu.



When **Activate Scenarios** is checked and there are Scenarios defined, the Scenarios dialog appears when **Run** is selected. (Note that if the model is a new model that has never been saved, an error message will display when attempting to run a Scenario. The model must first be saved.)



The original default selection is **None**. That is, if **None** is the selected Scenario, no changes are made to the model and no results are stored in `Scenarios.xml`. However, once a Scenario is selected from the list, it remains the default Scenario until the Scenario is deleted from the model or another Scenario is selected.

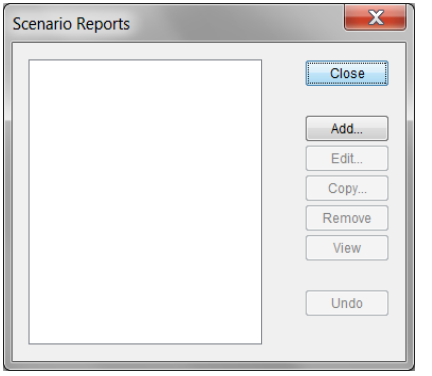


When **Run** is selected the model is updated with the information stored in the Scenario. Thus, after the simulation if **Simulate/Run Settings** is selected, the properties there will match the **Run Settings** of the Scenario that just completed. All Model Parameters keep the values set from the Scenario. If **Report/ Define Global Statistics Collection** is selected, the properties there will match the **Global Statistics** selections of the Scenario. Also, any Process alternative selections from the Scenario will be in the model. However, individual statistics selections on the **Selected Statistics** tab are not saved in the model.

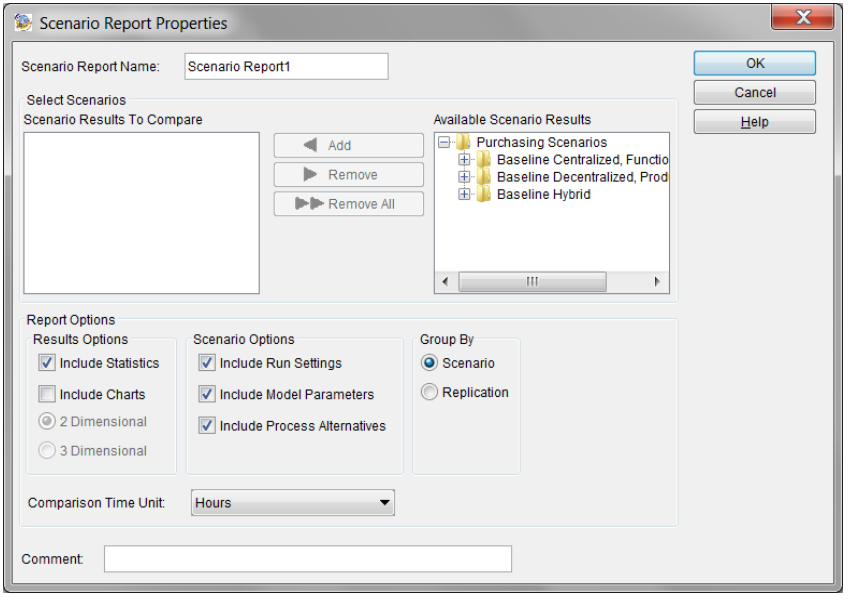
When the simulation is complete, the statistical results of the simulation are stored in the file `Scenarios.xml`, which is located in the model's directory. The results are stored based on the Scenario name. The results are only removed from `Scenarios.xml` if the scenario is removed from the model (see "Defining Scenarios" on page 122).

Comparing Scenarios

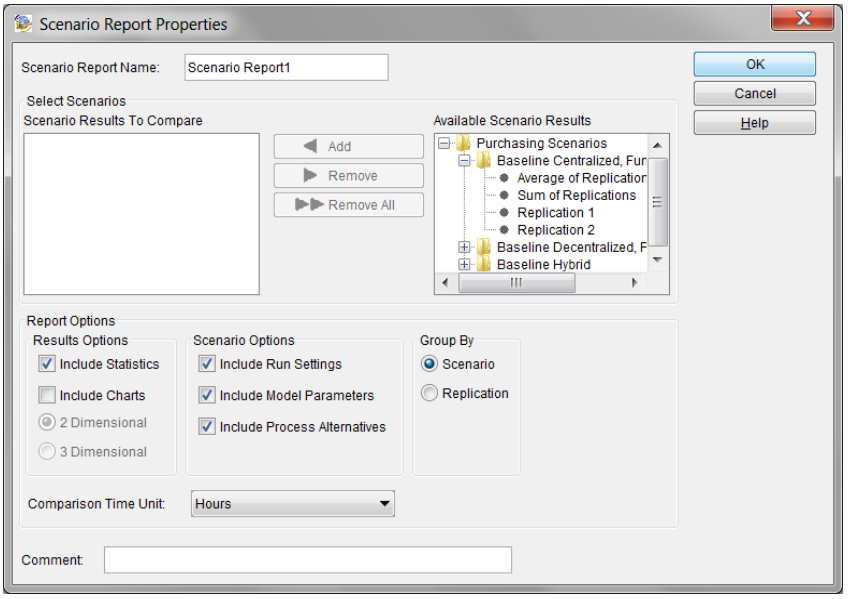
Scenario Reports are used to compare Scenario results. Selecting **Report/Scenario Reports...** displays the dialog for creating, managing, and viewing Scenario Reports. (Note that if the model is a new model that has never been saved, an error message will display. Scenario Reports cannot be defined until a new model has been saved.)



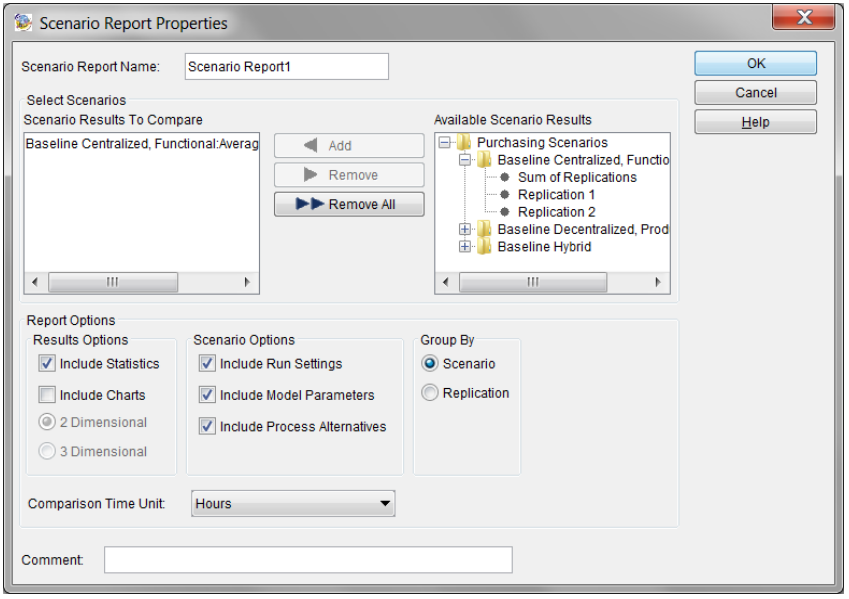
Selecting **Add...** displays the dialog for defining a Scenario Report.



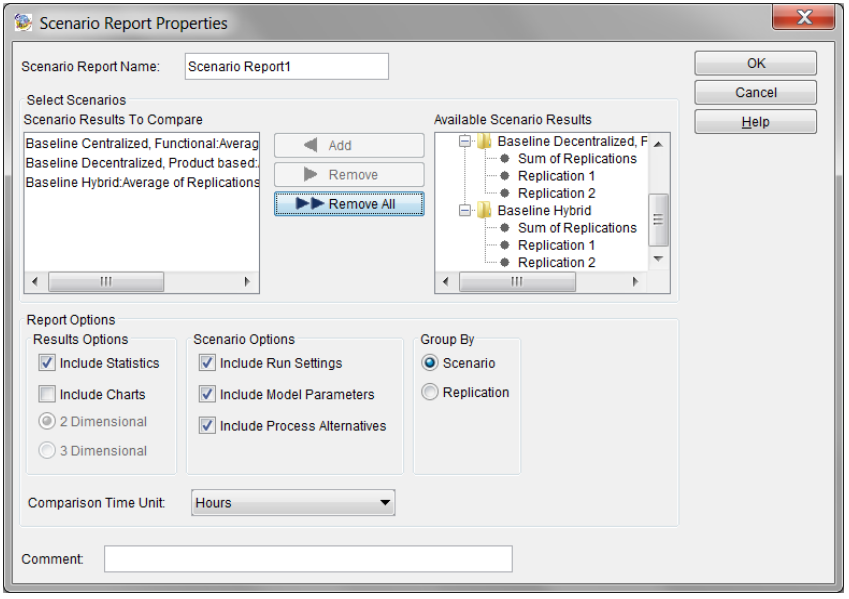
The properties default to a **Scenario Report Name**. This name can be changed but the name given to the report must be unique among all Scenario Reports. On the right are the **Available Scenario Results**. This displays all Scenario results that are stored in `Scenarios.xml`. Specific results from the right are added to **Scenario Results To Compare**. Expanding a Scenario shows the specific results for the scenario.



Selecting a Scenario item causes the **Add** button to enable. Multiple items can be selected by using the Control key when selecting. Clicking **Add** moves the selected items from **Available Scenario Results** to **Scenario Results To Compare**.



For this example, the **Average of Replications** will be compared for all scenarios.



There must be at least two items under **Scenario Results To Compare**. If not, an error will display when **OK** is selected.

Once the Scenario items to compare have been determined **Report Options** can be selected. There are four categories of **Report Options**: **Results Options**, **Scenario Options**, **Group By**, and **Comparison Time Unit**. At least one of the options within **Results Options** or **Scenario Options** must be selected. If not, an error will display when **OK** is clicked since the report would be empty.

Results Options are **Include Statistics** and **Include Charts**. When **Include Statistics** is selected the report will display a table of the statistics selected in the Scenarios. When **Include Charts** is selected the report will include bar charts displaying the statistics selected in the Scenarios. If both are selected, each section of statistics is displayed in a table followed by the accompanying bar chart. The bar charts can be **2 Dimensional** or **3 Dimensional**. These options are enabled when **Include Charts** is checked. Note that charts are most useful when the number of statistics being compared is small. A large number of statistics will cause the X axis labels to be unreadable.

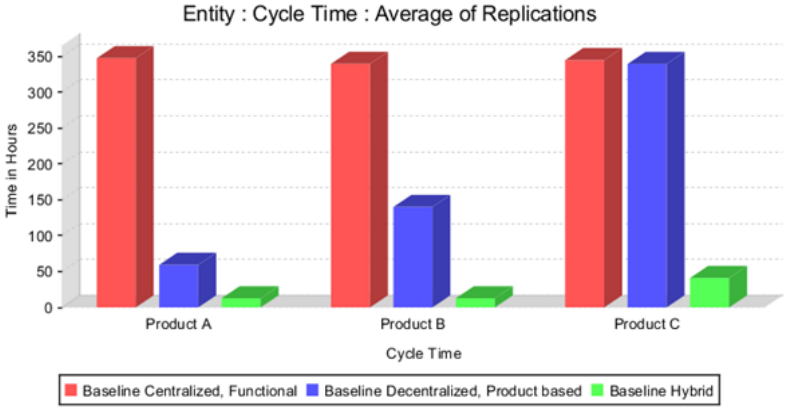
The **Scenario Options** relate to the scenario definition. **Include Run Settings** adds the Run Settings specified for the Scenario. **Include Model Parameters** adds the Model Parameter settings, and **Include Process Alternatives** lists the active alternatives for each Process with multiple alternatives (even if not specified in the Scenario definition). Any selected **Scenario Options** are displayed in the report before statistics or charts. Thus, the Scenario information from each scenario represented under **Scenario Results To Compare** are displayed by scenario at the beginning of the report.

The **Comparison Time Unit** sets the time unit to use for time related statistics. Thus, if one Scenario's results are stored in Minutes and another Scenario's results are stored in Hours, the **Comparison Time Unit** ensures the Scenario Report uses a common time unit.

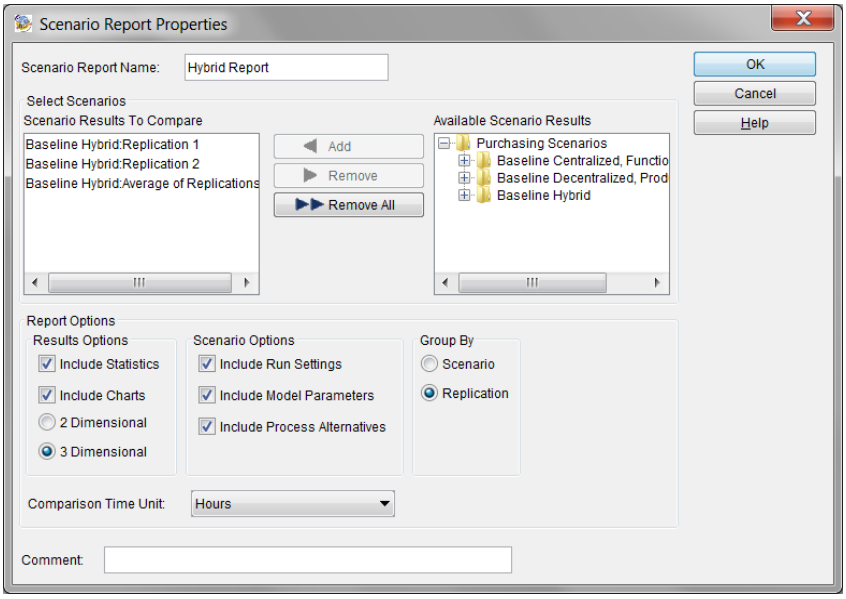
Below is an example of a table of statistics followed by a bar chart.

Entity : Cycle Time in Hours : Average of Replications

Name	Avg	St Dev	Min	Max
Baseline Centralized, Functional				
Product A	347.26	10.25	334.31	354.81
Product B	339.32	12.59	327.70	352.88
Product C	344.56	7.59	337.20	352.38
Baseline Decentralized, Product based				
Product A	59.27	0.03	59.24	59.30
Product B	140.03	9.52	130.85	149.88
Product C	339.19	0.66	338.53	339.84
Baseline Hybrid				
Product A	12.66	0.19	12.45	12.84
Product B	12.66	0.04	12.63	12.70
Product C	40.98	1.20	39.78	42.19



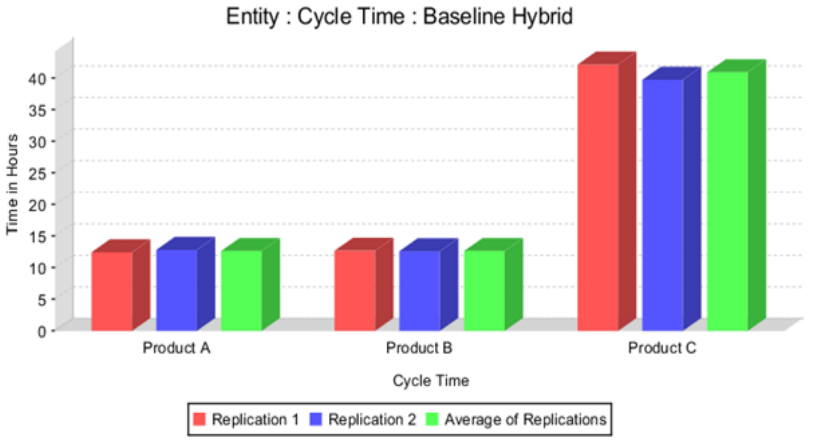
Note that the table is grouped by Scenario. This is because the **Group By** selection for this report was **Scenario**. This is the **Group By** option that should be used when comparing results across Scenarios. **Replication** should be selected when comparing results within a Scenario. In the following report definition the items being compared are only from the **Baseline Hybrid** scenario.



Below is a section of the report. Note that the titles of the table and the chart ends with the Scenario name instead of the replication, and the table is grouped by **Replication** instead of **Scenario**.

Entity : Cycle Time in Hours : Baseline Hybrid

Name	Avg	St Dev	Min	Max
Replication 1				
Product A	12.45	2.46	0.00	17.02
Product B	12.70	2.07	0.00	18.15
Product C	42.19	12.18	0.00	67.38
Replication 2				
Product A	12.84	2.14	0.00	16.95
Product B	12.63	2.01	0.00	17.02
Product C	39.78	11.69	0.00	65.69
Average of Replications				
Product A	12.66	0.19	12.45	12.84
Product B	12.66	0.04	12.63	12.70
Product C	40.98	1.20	39.78	42.19



Scenario Reports display in the Report Viewer (see "Report Viewer"). Thus, reports can be saved in any format supported by the Report Viewer.

The complete results of a Scenario can be viewed in Custom Reports (see "Statistical Categories").